

Updates to P4080 QorIQ Multicore Communication Processor Reference Manual, Rev. 2, as of 2015-09-25

This section provides updates to the *P4080 QorIQ Multicore Communication Processor Reference Manual*, Rev 2. We are providing known corrections, but do not guarantee that the list is exhaustive. For convenience, the section number and page number of the item in the reference manual are provided.

Note: This PDF file contains updates embedded as inline sticky notes; use the provided links and scroll to the inline location. Future versions of the document will incorporate the sticky notes into the source text of the document. Freescale recommends viewing this file with Adobe Acrobat Reader.

Section, Page No.	Changes
3.6.16, 183	In the “Signals_B1TTLCRn0 field descriptions” table, changed the last protocol setting for FLT_SEL to “Recommended setting (all protocols): 11.”
3.6.20, 193	In the “Signals_B2TTLCRn0 field descriptions” table, changed the last protocol setting for FLT_SEL to “Recommended setting (all protocols): 11.”
3.6.24, 203	In the “Signals_B3TTLCRn0 field descriptions” table, changed the last protocol setting for FLT_SEL to “Recommended setting (all protocols): 11.”
Chapter 6, 273-332	Attached P4080_secure_boot_post_taug_replacement.pdf as a replacement for Chapter 6.
10.6.1.1, 474	In “Setting Up PAMU,” removed the bullet point stating: “The PAMU fetch attributes register is programmed with the necessary information.”
21.2.8, 1643	In the exceptions and special behaviors list of “Special Values and Exceptions,” replaced the third bullet point with the following bullet point: <ul style="list-style-type: none"> – FDs, compound frame, or multi-buffer frame SG entries with ADDR/BPID/LENGTH=0 encoding indicate that the FD or SG entry does not convey buffer information, that is, the SG entry is unused and is, therefore, skipped during input and/or output processing.
21.2.9, 1643	In “Releasing Buffers to the BMan,” removed the first bullet point and added the second bullet point as plain text to the end of the first paragraph.
22.1.1, 1651	In “MPIC features summary,” updated to “IRQ[0:11].”
22.1.6, 1657	In “Interrupt Sources,” updated to “IRQ[0:11].”
22.2.1, 1660	In the first paragraph of “MPIC signal descriptions,” added that there are “12” distinct external interrupt request input signals.
	In Table 22-3 , “External interrupt signals—detailed signal descriptions,” updated the first signal name to “IRQ[0:11].”
22.3, 1668	Attached P4080_external_interrupts_memory_map.pdf as a replacement for the registers at offset 5_0000, 5_0010, and 5_0018.

[22.3.70-72, 1742](#) Attached P4080_external_interrupt_registers.pdf, which contains missing Sections 22.3.70, 22.3.71, and 22.3.72.

[25.3.35, 1836](#) In the first paragraph of “I/O Voltage Selection Status Register (DCFG_IOVSELSR),” added a note to see Section 3.3.1, “I/O Voltage Select,” for more information.

P4080 QorIQ Multicore Communication Processor Reference Manual

Also supports P4040 and P4081

Document Number: P4080RM
Rev 2, 05/2014



Contents

Section number	Title	Page
	Chapter 1 Overview	
1.1	Introduction.....	69
1.2	P4080 Features Summary.....	69
1.3	P4080 Block Diagram.....	70
1.4	P4080 Application Examples.....	71
1.4.1	Multicore Processing Scenarios.....	72
1.4.2	P4080 Applications.....	73
1.4.2.1	Virtual Private Network (VPN)/ IP Services Router.....	74
1.4.2.2	Security Services Blade for Switch or Server.....	74
1.4.2.3	Wireless Infrastructure/Radio Node Controller.....	75
1.4.2.4	High-Performance Compute Blade.....	75
1.5	Subsystem Features	76
1.5.1	e500 Core and Cache Memory Complex.....	76
1.5.2	CoreNet Fabric and Address Map.....	78
1.5.3	Memory Complex.....	78
1.5.3.1	DDR Memory Controllers.....	79
1.5.3.2	PreBoot Loader and Nonvolatile Memory Interfaces.....	79
1.5.3.2.1	Enhanced Local Bus Controller	80
1.5.3.2.2	Serial Memory Controllers.....	80
1.5.4	Universal Serial Bus (USB) 2.0.....	80
1.5.5	High-Speed Peripheral Interface Complex.....	81
1.5.5.1	PCI Express Controllers.....	81
1.5.5.2	Serial RapidIO	82
1.5.6	Datapath Acceleration Architecture (DPAA).....	82
1.5.6.1	Datapath Acceleration Architecture Programming Model.....	84
1.5.6.2	DPAA Definitions.....	85

Section number	Title	Page
1.5.7	Major DPAA Components.....	89
1.5.7.1	Frame Manager.....	89
1.5.7.1.1	Network Interfaces.....	90
1.5.7.1.2	Parse Function	90
1.5.7.1.3	Distribution and Policing.....	91
1.5.7.2	Queue Manager.....	92
1.5.7.3	Buffer Manager.....	93
1.5.7.4	Security Engine (SEC 4.0).....	93
1.5.7.5	Pattern Matching Engine (PME 2.0).....	94
1.6	Resource Partitioning and QorIQ Trust Architecture.....	96
1.6.1	e500mc MMU and Embedded Hypervisor.....	96
1.6.2	Peripheral Access Management Unit (PAMU).....	97
1.6.3	Secure Boot and Sensitive Data Protection.....	97
1.7	Advanced Power Management.....	98
1.8	Debug support.....	100

Chapter 2 Memory Map

2.1	Memory Map Overview.....	101
2.2	Global Source and Target IDs.....	102
2.3	Local Access Windows (LAWs).....	105
2.3.1	Precedence of Local Access Windows.....	105
2.3.2	Configuring Local Access Windows.....	106
2.3.3	Distinguishing Local Access Windows from Other Mapping Functions.....	106
2.3.4	SRAM Windows.....	107
2.3.5	Local Address Map Example.....	107
2.4	Local Access Window (LAW) Memory Map.....	109
2.4.1	LAWn base address register high (LAW_LAWBARH _n).....	112
2.4.2	LAWn base address register low (LAW_LAWBARL _n).....	112
2.4.3	LAWn attribute register (LAW_LAWAR _n).....	113

Section number	Title	Page
2.5	Address Translation and Mapping Units.....	114
2.5.1	Address Translation	114
2.5.2	Outbound ATMUs.....	115
2.5.3	Inbound ATMUs.....	115
	2.5.3.1 Illegal Interaction Between Inbound ATMUs and LAWs.....	116
2.6	Configuration, Control, and Status Register (CCSR) Space.....	116
2.6.1	Accessing CCSR Memory from the Local Processor.....	116
2.6.2	Accessing CCSR Memory from External Masters.....	117
2.6.3	Accessing Reserved Registers and Bits.....	117
2.6.4	Organization of CCSR Memory.....	118
2.6.5	CCSR Address Map.....	118

Chapter 3 Signal Descriptions

3.1	Signals Introduction.....	123
3.2	Signals Overview.....	123
3.3	Dedicated Configuration Signals.....	134
3.3.1	I/O Voltage Select.....	134
3.4	Configuration Signals Sampled at Reset	135
3.5	Signal Multiplexing Details.....	136
3.5.1	IEEE 1588 and GPIO Signal Multiplexing.....	137
3.5.2	Frame Manager 1 dTSEC1 and USB1 Signal Multiplexing.....	137
3.5.3	Frame Manager 2 dTSEC1, Frame Manager 1 dTSEC2, and USB2 Signal Multiplexing.....	138
3.5.4	UART and GPIO Signal Multiplexing.....	139
3.5.5	I2C3, eSDHC, and GPIO Signal Multiplexing.....	140
3.5.6	I2C4 and Debug Signal Multiplexing.....	141
3.5.7	MPIC and GPIO Signal Multiplexing.....	141
3.5.8	MPIC and Debug Signal Multiplexing.....	141
3.5.9	eSPI and eSDHC Signal Multiplexing.....	142
3.5.10	DMA1, Debug, and GPIO Signal Multiplexing.....	142

Section number	Title	Page
3.5.11	DMA2, Debug, and GPIO Signal Multiplexing.....	142
3.5.12	SerDes Lane Assignments and Multiplexing.....	143
3.6	SerDes Control Memory Map.....	145
3.6.1	SerDes Bank n Reset Control Register (Signals_SRDSBnRSTCTL).....	151
3.6.2	SerDes Bank n PLL Control Register 0 (Signals_SRDSBnPLLCR0).....	154
3.6.3	SerDes Bank1 PLL Control Register 1 (Signals_SRDSBnPLLCR1).....	155
3.6.4	SerDes Transmit Calibration Control Register (Signals_SRDSTCALCR).....	157
3.6.5	SerDes Receive Calibration Control Register (Signals_SRDSRCALCR).....	158
3.6.6	SerDes General Register 0 (Signals_SRDSGR0).....	159
3.6.7	SerDes Protocol Converter Configuration Register 0 (Signals_SRDSPCCR0).....	160
3.6.8	SerDes Protocol Converter Configuration Register 1 (Signals_SRDSPCCR1).....	162
3.6.9	SerDes Protocol Converter Configuration Register 2 (Signals_SRDSPCCR2).....	164
3.6.10	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRn0).....	167
3.6.11	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRn1).....	170
3.6.12	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRn0).....	173
3.6.13	Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRn0).....	176
3.6.14	Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRn0).....	178
3.6.15	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRn0).....	180
3.6.16	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRn0).....	182
3.6.17	Bank 2 General Control Register 0 - Lane x (Signals_B2GCRn0).....	184
3.6.18	Bank 2 General Control Register 1 - Lane x (Signals_B2GCRn1).....	187
3.6.19	Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRn0).....	190
3.6.20	Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRn0).....	192
3.6.21	Bank 3 General Control Register 0 - Lane x (Signals_B3GCRn0).....	194
3.6.22	Bank 3 General Control Register 1 - Lane x (Signals_B3GCRn1).....	197
3.6.23	Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRn0).....	200
3.6.24	Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRn0).....	202
3.7	SerDes PLL reset and reconfiguration.....	204
3.8	Output Signal States During Reset.....	204

Section number	Title	Page
	Chapter 4 Reset, Clocking, and Initialization	
4.1	Reset, clocking, and initialization overview.....	205
4.2	External Signal Descriptions.....	205
4.2.1	System Control Signals.....	206
4.2.2	External Clock Signals.....	207
4.3	Local Configuration Control Registers.....	208
4.3.1	Accessing Configuration, Control, and Status Registers.....	208
4.3.1.1	Updating CCSRBARs.....	208
4.3.2	Accessing Alternate Configuration Space.....	209
4.3.3	Boot Space Translation.....	210
4.4	Local Configuration Control Memory Map.....	210
4.4.1	Configuration, control, and status registers base address register high (LCC_CCSRBARH).....	211
4.4.2	Configuration, control, and status registers base address register low (LCC_CCSRBARL).....	211
4.4.3	Configuration, control, and status registers attribute register (LCC_CCSRAR).....	212
4.4.4	Alternate configuration base address register high (LCC_ALTCBARH).....	212
4.4.5	Alternate configuration base address register low (LCC_ALTCBARL).....	213
4.4.6	Alternate configuration attribute register (LCC_ALTCAR).....	213
4.4.7	Boot space translation register high (LCC_BSTRH).....	214
4.4.8	Boot space translation register low (LCC_BSTRL).....	214
4.4.9	Boot space translation attribute register (LCC_BSTAR).....	214
4.5	Clocking Memory Map/Register Definition.....	215
4.5.1	Core n clock control/status register (Clocking_CLKCnCSR).....	216
4.5.2	PLL cluster n general status register (Clocking_PLLnCGSR).....	218
4.5.3	Platform clock domain control/status register. (Clocking_CLKPCSR).....	219
4.5.4	Platform PLL general status register. (Clocking_PLLPGSR).....	221
4.5.5	DDR PLL general status register (Clocking_PLLDGSR).....	223
4.6	Functional Description.....	224
4.6.1	Power-On Reset Sequence.....	224

Section number	Title	Page
4.6.2	Hard Reset Sequence.....	227
4.6.3	Power-On Reset Configuration.....	228
4.6.3.1	Reset Configuration Word Source.....	228
4.6.3.2	General-Purpose Input.....	229
4.6.3.3	eLBC FCM ECC Control.....	229
4.6.3.4	DRAM type select.....	230
4.6.4	Reset Configuration Word (RCW).....	230
4.6.4.1	RCW Field Definitions.....	231
4.6.4.1.1	Hard Coded RCW Options.....	244
4.6.5	Clocking.....	245
4.6.5.1	IP Logic Clock Distribution and Configuration.....	246
4.6.5.2	CLK_OUT Configuration.....	249
4.6.5.3	Reference Clocks for SerDes Protocols.....	249

Chapter 5 Pre-Boot Loader (PBL)

5.1	PBL Overview.....	251
5.2	PBL Features Summary.....	252
5.3	PBL Modes of Operation.....	252
5.4	PBL Functional Description.....	253
5.4.1	Configuration of Device via Reset Configuration Word (RCW).....	253
5.4.2	Device Initialization by PBL.....	253
5.4.2.1	CCSR Registers Blocked from PBL During Secure Boot.....	254
5.4.3	Required Format of Data Structure Consumed by PBL.....	254
5.4.4	RCW Loading by PBL.....	256
5.4.5	Pre-Boot Initialization Command Loading by PBL.....	257
5.4.6	Reserved Address Space Used as Internal PBL Commands.....	257
5.4.7	PBL Error Codes.....	258
5.5	Registers Written by PBL During RCW and PBI Phases.....	268
5.5.1	I2C Registers.....	268

Section number	Title	Page
5.5.2	eSPI Registers.....	268
5.5.3	eLBC Registers.....	268
5.5.4	eSDHC Registers.....	269
5.6	Addressing Multiple I2C EEPROMs.....	269
5.7	eSPI Mode Assumptions.....	270
5.8	PBL Initialization/Application Information.....	270
5.8.1	Starting Addresses.....	270
5.8.2	Software Restrictions.....	271
5.8.3	Software Recommendations.....	271

Chapter 6 **Secure Boot and Trust Architecture 1.0**

6.1	Trust Architecture overview.....	273
6.1.1	Objectives of trust architecture 1.0.....	273
6.1.2	Trust Architecture as implemented on the chip.....	274
6.1.2.1	Power Architecture core	275
6.1.2.2	No execute bit (UX and SX bits).....	275
6.1.2.3	Hypervisor.....	276
6.1.2.4	PAMUs.....	276
6.1.2.5	Secure debug controller.....	277
6.1.2.6	SEC 4.0.....	278
6.1.2.7	Internal boot ROM and ISBC.....	279
6.1.2.8	External tamper-detection.....	279
6.1.2.9	Pre-boot loader (PBL).....	279
6.1.2.10	Security fuse processor.....	280
6.1.2.11	Security monitor.....	281
6.1.3	Code signing.....	281
6.1.4	Secure boot sequence.....	283
6.1.4.1	Pre-boot phase.....	283
6.1.4.2	ISBC phase.....	284

Section number	Title	Page
	6.1.4.3 ESBC phase.....	284
	6.1.4.3.1 ESBC U-Boot and boot script.....	285
6.2	Security fuse processor (SFP).....	285
6.2.1	Fuse programming.....	285
6.2.2	Fuse read errors.....	286
6.2.3	Security fuse processor (SFP) memory map.....	286
6.2.3.1	Instruction Register (SFP_INGR).....	288
6.2.3.2	Debug Error Syndrome Status Register (SFP_DESSR).....	289
6.2.3.3	OEM Security Policy Register (SFP_OSPR).....	290
6.2.3.4	Freescale Section Write Protect Register (SFP_FSWPR).....	291
6.2.3.5	Debug Permissions Register (SFP_DPR).....	292
6.2.3.6	Debug Challenge Value Register n (SFP_DCVRn).....	293
6.2.3.7	Debug Response Value Register n (SFP_DRVRn).....	294
6.2.3.8	One Time Programmable Master Key n (SFP OTPMKRn).....	296
6.2.3.9	Super Root Key Hash n (SFP_SRKHRn).....	299
6.2.3.10	OEM Unique ID Register (SFP_OUIDR).....	300
6.2.3.11	OEM Scratch Pad Fuse Register (SFP OSPFR).....	301
6.2.3.12	OEM Scratch Pad/Security Control Register (SFP_ONSEC).....	302
6.2.3.13	Freescale Unique ID Register (SFP_FUIDR).....	303
6.2.3.14	Freescale Scratch Pad Fuse Register n (SFP_FSPFRn).....	303
6.3	Security monitor.....	304
6.3.1	Security monitor features summary	305
6.3.2	Operational states.....	306
6.3.3	Signals.....	307
6.3.4	Security Monitor memory map/register definition.....	307
6.3.4.1	SM_HP Lock Register (SECMON_HPLR).....	309
6.3.4.2	SM_HP Command Register (SECMON_HPCOMR).....	311
6.3.4.3	SM_HP Security Interrupt Control Register (SECMON_HPSICR).....	314
6.3.4.4	SM_HP Security Violation Control Register (SECMON_HPSVCR).....	316

Section number	Title	Page
6.3.4.5	SM_HP Status Register (SECMON_HPSR).....	318
6.3.4.6	SM_HP Security Violation Status Register (SECMON_HPSVSR).....	321
6.3.4.7	SM_HP High Assurance Counter Initial Value Register (SECMON_PHACIVR).....	322
6.3.4.8	SM_HP High Assurance Counter Register (SECMON_PHACR).....	323
6.3.4.9	SM_HP Version ID Register 1 (SECMON_HPVIDR1).....	323
6.3.4.10	SM_HP Version ID Register 2 (SECMON_HPVIDR2).....	324
6.3.5	Security Monitor functional description.....	324
6.3.5.1	SM_HP description.....	325
6.3.5.1.1	Security state machine.....	325
6.3.5.1.2	State definitions.....	326
6.3.5.2	HP security violation policy.....	329
6.3.5.3	Master key checking and control.....	330
6.3.6	Initialization guidelines.....	330

Chapter 7 **e500mc Core Integration**

7.1	Introduction.....	333
7.2	e500mc Core Overview.....	333
7.3	P4080 -Specific Core Implementation Details.....	336
7.3.1	Eight e500mc Cores.....	336
7.3.2	CoreNet Coherency Fabric.....	336
7.3.3	Reset and Clocking.....	336
7.3.4	Register Model Implementation Details.....	337
7.3.4.1	Processor Version Register (PVR).....	337
7.3.4.2	System Version Register (SVR)	337
7.3.4.3	(Guest) Processor ID register (PIR/GPIR).....	338
7.3.4.4	Timer Control Register (TCR).....	338
7.3.5	Cache Model Implementation Details.....	339
7.3.6	Interrupt Model Implementation Details.....	339
7.3.6.1	Watchdog Timer Expiration Out to Platform.....	340

Section number	Title	Page
7.3.7	Performance Monitor Details.....	340
7.3.8	Decorated Load and Store Operations	341
7.3.9	Debug Features.....	342
7.3.10	Power Management.....	342

Chapter 8 CoreNet Platform Cache (CPC)

8.1	Introduction.....	343
8.1.1	CPC Overview.....	343
8.1.2	Features.....	343
8.2	CoreNet Platform Cache (CPC) Memory Map.....	344
8.2.1	CPC configuration and status register 0 (CPC _x _CPCCSR0).....	349
8.2.2	CPC configuration register 0 (CPC _x _CPCCFG0).....	352
8.2.3	CPC external write control register n (CPC _x _CPCEWCR _n).....	354
8.2.4	CPC external write base address register n (CPC _x _CPCEWBAR _n).....	355
8.2.5	CPC SRAM control register 1 (CPC _x _CPCSRCR1).....	356
8.2.6	CPC SRAM control register 0 (CPC _x _CPCSRCR0).....	356
8.2.7	CPC partition ID register 0 (CPC _x _CPCPIR0).....	357
8.2.8	CPC partition allocation register 0 (CPC _x _CPCPAR0).....	358
8.2.9	CPC partition way register 0 (CPC _x _CPCPWR0).....	359
8.2.10	CPC partition ID register n (CPC _x _CPCPIR _n).....	360
8.2.11	CPC partition allocation register n (CPC _x _CPCPAR _n).....	360
8.2.12	CPC partition way register 0 (CPC _x _CPCPWR _n).....	362
8.2.13	CPC error injection high register (CPC _x _CPCERRINJHI).....	362
8.2.14	CPC error injection low register (CPC _x _CPCERRINJLO).....	362
8.2.15	CPC error injection control register (CPC _x _CPCERRINJCTL).....	363
8.2.16	CPC capture data high register (CPC _x _CPCCAPTDATAHI).....	364
8.2.17	CPC capture data low register (CPC _x _CPCCAPTDATALO).....	364
8.2.18	CPC capture ECC register (CPC _x _CPCCAPTECC).....	364
8.2.19	CPC error detect register (CPC _x _CPCERRDET).....	366

Section number	Title	Page
8.2.20	CPC error disable register (CPC _x _CPCERRDIS).....	368
8.2.21	CPC error interrupt enable register (CPC _x _CPCERRINTEN).....	369
8.2.22	CPC error extended address register (CPC _x _CPCERREADDR).....	370
8.2.23	CPC error address register (CPC _x _CPCERRADDR).....	370
8.2.24	CPC error control register (CPC _x _CPCERRCTL).....	371
8.2.25	CPC hardware debug control register 0 (CPC _x _CPCHDBCR0).....	371
8.3	CPC Functional Description.....	372
8.3.1	Register Sections.....	372
8.3.1.1	SRAM Mode Registers.....	372
8.3.1.2	Partitioning Control Registers.....	372
8.3.1.3	Error Registers.....	373
8.3.1.3.1	Error Detection and Reporting.....	373
8.3.1.3.2	Error Capture.....	373
8.3.1.3.3	Error Injection.....	374
8.3.2	Line Locking.....	374
8.3.3	Cache Operation Instructions and Transactions.....	375
8.3.4	Decorated Storage.....	375
8.3.4.1	Decorated Load Operations.....	376
8.3.4.2	Decorated Store Operations.....	376
8.3.4.3	Notify Operations.....	379
8.3.5	Cache/SRAM Data Clear.....	379
8.4	Initialization/Application Information.....	380
8.4.1	Supported SRAM Mode Configurations.....	380
8.4.2	Programming Examples.....	380
8.4.2.1	Modifying CPC Control and Status Registers.....	380
8.4.2.2	Enabling the CPC after Power-On Reset.....	381
8.4.2.3	Changing the Configuration of an Enabled CPC.....	381
8.4.2.4	Disabling the CPC.....	382

Section number	Title	Page
	Chapter 9 CoreNet Coherency Fabric (CCF)	
9.1	CCF Introduction.....	383
9.1.1	CCF Features Summary.....	384
9.2	CoreNet Coherency Fabric (CCF) Memory Map.....	385
9.2.1	Snoop ID 0 Port Mapping Register (CCF_SIDMR0).....	388
9.2.2	Snoop ID n Port Mapping Register (CCF_SIDMR n).....	389
9.2.3	CSDID 0 Port Mapping Register (CCF_CIDMR0).....	390
9.2.4	CSDID n Port Mapping Register (CCF_CIDMR n).....	391
9.2.5	CCF Error Detect Register (CCF_CEDR).....	392
9.2.6	CCF Error Enable Register (CCF_CEER).....	393
9.2.7	CCF Error Capture Attribute Register (CCF_CECAR).....	394
9.2.8	CCF Error Capture Address Register High (CCF_CECADR H).....	394
9.2.9	CCF Error Capture Address Register Low (CCF_CECADRL).....	395
9.2.10	CCF Error Capture Attribute Register 2 (CCF_CECAR2).....	395
	Chapter 10 Peripheral Access Management Unit (PAMU)	
10.1	PAMU Introduction.....	397
10.1.1	PAMU Overview.....	397
10.1.2	PAMU Features Summary.....	398
10.2	Data Structures Used by PAMU.....	399
10.2.1	Overview of Data Structures.....	399
10.2.1.1	Peripheral Access Authorization and Control Tables (PAACTs).....	400
10.2.1.2	Operation Mapping Table.....	401
10.2.1.3	Access Capabilities Across LIODs.....	401
10.2.2	Structure and Contents of PAACTs.....	401
10.2.3	Peripheral Access Authorization and Control Entry (PAACE).....	403
10.2.3.1	PAACE Offset 0x00.....	404

Section number	Title	Page
	10.2.3.2 PAACE Offset 0x08.....	406
	10.2.3.2.1 PAACE Domain Attributes	407
	10.2.3.3 PAACE Offset 0x10.....	408
	10.2.3.4 PAACE Offset 0x18.....	409
10.3	PAMU Memory Map/Register Definitions.....	410
10.4	PAMU Memory Map.....	410
10.4.1	Primary PAACT Base Address High register (PAMU _x _PPBAH).....	430
10.4.2	Primary PAACT Base Address Low register (PAMU _x _PPBAL).....	431
10.4.3	Primary PAACT Limit Address High register (PAMU _x _PPLAH).....	431
10.4.4	Primary PAACT Limit Address Low register (PAMU _x _PPLAL).....	432
10.4.5	Secondary PAACT Base Address High register (PAMU _x _SPBAH).....	432
10.4.6	Secondary PAACT Base Address Low register (PAMU _x _SPBAL).....	433
10.4.7	Secondary PAACT Limit Address High register (PAMU _x _SPLAH).....	433
10.4.8	Secondary PAACT Limit Address Low register (PAMU _x _SPLAL).....	433
10.4.9	OMT Base Address High register (PAMU _x _OBAH).....	434
10.4.10	OMT Base Address Low register (PAMU _x _OBAL).....	434
10.4.11	OMT Limit Address High register (PAMU _x _OLAH).....	435
10.4.12	OMT Limit Address Low register (PAMU _x _OLAL).....	435
10.4.13	PAMU Address Capabilities Register 1 (PAMU _x _PAC1).....	436
10.4.14	PAMU Address Capabilities Register 2 (PAMU _x _PAC2).....	436
10.4.15	PAMU Operation Error Status register 1 (PAMU _x _POES1).....	437
10.4.16	PAMU Operation Error Status register 2 (PAMU _x _POES2).....	438
10.4.17	PAMU Operation Error Address High register (PAMU _x _POEAH).....	438
10.4.18	PAMU Operation Error Address Low register (PAMU _x _POEAL).....	439
10.4.19	Access Violation Status register 1 (PAMU _x _AVS1).....	440
10.4.20	Access Violation Status register 2 (PAMU _x _AVS2).....	442
10.4.21	Access Violation Address High register (PAMU _x _AVAH).....	443
10.4.22	Access Violation Address Low register (PAMU _x _AVAL).....	443
10.4.23	ECC Error Control Register (PAMU _x _EECTL).....	443

Section number	Title	Page
10.4.24	ECC Error Interrupt Enable Register (PAMU _x _EEINTEN).....	444
10.4.25	ECC Error Detect Register (PAMU _x _EEDET).....	445
10.4.26	ECC Error Attributes Register (PAMU _x _EEATTR).....	446
10.4.27	ECC Error Address High (PAMU _x _EEAHI).....	447
10.4.28	ECC Error Address Low (PAMU _x _EEALO).....	447
10.4.29	ECC Error Data High (PAMU _x _EEDHI).....	448
10.4.30	ECC Error Data Low (PAMU _x _EEDLO).....	448
10.4.31	Unauthorized device access detection register (PAMU _x _UDAD).....	449
10.4.32	PAMU Revision register 1 (PAMU _x _PR1).....	449
10.4.33	PAMU Revision register 2 (PAMU _x _PR2).....	450
10.4.34	PAMU Capabilities register 1 (PAMU _x _PC1).....	450
10.4.35	PAMU Capabilities register 2 (PAMU _x _PC2).....	451
10.4.36	PAMU Capabilities register 3 (PAMU _x _PC3).....	452
10.4.37	PAMU Capabilities register 4 (PAMU _x _PC4).....	454
10.4.38	PAMU Control register (PAMU _x _PC).....	455
10.4.39	PAMU Interrupt Control and Status register (PAMU _x _PICS).....	456
10.5	PAMU Functional Description.....	457
10.5.1	System Set-Up for PAMU Operation.....	457
10.5.2	Steps in Processing of DSA Operations by PAMU.....	458
10.5.3	Detailed Description of PAMU Actions.....	460
10.5.3.1	PAMU Gate Closed and PAMU Enable Check.....	460
10.5.3.2	PPAACT Request Range Check.....	461
10.5.3.3	PPAACT Cache Lookup.....	462
10.5.3.4	PAMU Fetch Request.....	462
10.5.3.5	Primary PAACE Processing.....	462
10.5.3.6	SPAACE Access and Processing.....	464
10.5.3.6.1	SPAACT Request Range Check.....	464
10.5.3.6.2	SPAACT Cache Lookup Request.....	465
10.5.3.6.3	Secondary PAACE Processing.....	465

Section number	Title	Page
	10.5.3.7 Address Translation Service.....	466
	10.5.3.7.1 No Address Translation Mode.....	466
	10.5.3.7.2 Window Address Translation Mode.....	467
	10.5.3.8 OMT Access and Service.....	469
	10.5.3.8.1 No Operation Translation.....	469
	10.5.3.8.2 Operation Type Translation.....	470
	10.5.3.8.3 OMT Cache Access.....	472
	10.5.3.9 Access Violation.....	473
10.6	PAMU Initialization/Application Information.....	473
	10.6.1 System Set-Up.....	473
	10.6.1.1 Setting Up PAMU.....	473
	10.6.1.2 Power-On Reset.....	474
	10.6.2 System with Multiple PAMUs	474
	10.6.2.1 PAACT Locations.....	474
	10.6.2.2 OMT Locations.....	475
	10.6.2.3 Location of PAACT and OMT Data Structures.....	475
	10.6.3 Peer-to-Peer I/O Operations.....	475
	10.6.4 PAMU Cache Coherency.....	476
	10.6.5 Quiescing I/O Devices.....	476
	10.6.5.1 Quiescing I/O Devices for Table/Entry Updates.....	477
	10.6.5.2 Quiescing I/O Devices for Enabling PAMU and its Caches.....	477
	10.6.6 Locality of References.....	477
	10.6.6.1 Spatial Locality.....	477
	10.6.6.2 Temporal Locality.....	478
	10.6.7 Recovering Address Space.....	478
	10.6.7.1 Primary PAACT Address Window.....	478
	10.6.7.2 Secondary PAACT Address Window.....	478
	10.6.7.3 Secondary Sub-Windows and PAACEs.....	478
	10.6.8 Data Structure Size and Alignment.....	479

Section number	Title	Page
10.7	PAMU Setup.....	479
10.7.1	PAMU Operation Encoding.....	479
10.7.1.1	Ingress Operation Encoding (IOE).....	480
10.7.1.2	Egress Operation Encodings.....	480
10.7.1.3	IOE to EOE Translations.....	482
10.7.1.3.1	PAMU Bypass Mode and No Operation Translation Mode.....	482
10.7.1.3.2	Immediate and Indexed Operation Translation Modes.....	483
10.7.2	Domain Attributes.....	483
10.7.2.1	Constrained Domain Attribute.....	483
10.7.3	Implementation Attributes.....	484
10.7.4	Operation Mapping Table (OMT).....	484

Chapter 11 DDR Memory Controller

11.1	DDR Introduction.....	485
11.2	DDR Features Summary.....	486
11.2.1	DDR Modes of Operation.....	487
11.3	DDR External Signal Descriptions.....	487
11.3.1	DDR Signals Overview.....	487
11.3.2	DDR Detailed Signal Descriptions.....	488
11.3.2.1	Memory Interface Signals.....	488
11.3.2.2	Clock Interface Signals.....	493
11.4	DDR Memory Controller Memory Map.....	494
11.4.1	Chip select n memory bounds (DDR _x _CS _n _BNDS).....	499
11.4.2	Chip select n configuration (DDR _x _CS _n _CONFIG).....	500
11.4.3	Chip select n configuration 2 (DDR _x _CS _n _CONFIG_2).....	502
11.4.4	DDR SDRAM timing configuration 3 (DDR _x _TIMING_CFG_3).....	503
11.4.5	DDR SDRAM timing configuration 0 (DDR _x _TIMING_CFG_0).....	505
11.4.6	DDR SDRAM timing configuration 1 (DDR _x _TIMING_CFG_1).....	508
11.4.7	DDR SDRAM timing configuration 2 (DDR _x _TIMING_CFG_2).....	512

Section number	Title	Page
11.4.8	DDR SDRAM control configuration (DDR _x _DDR_SDRAM_CFG).....	516
11.4.9	DDR SDRAM control configuration 2 (DDR _x _DDR_SDRAM_CFG_2).....	519
11.4.10	DDR SDRAM mode configuration (DDR _x _DDR_SDRAM_MODE).....	522
11.4.11	DDR SDRAM mode configuration 2 (DDR _x _DDR_SDRAM_MODE_2).....	522
11.4.12	DDR SDRAM mode control (DDR _x _DDR_SDRAM_MD_CNTL).....	523
11.4.13	DDR SDRAM interval configuration (DDR _x _DDR_SDRAM_INTERVAL).....	526
11.4.14	DDR SDRAM data initialization (DDR _x _DDR_DATA_INIT).....	527
11.4.15	DDR SDRAM clock control (DDR _x _DDR_SDRAM_CLK_CNTL).....	527
11.4.16	DDR training initialization address (DDR _x _DDR_INIT_ADDR).....	528
11.4.17	DDR training initialization extended address (DDR _x _DDR_INIT_EXT_ADDRESS).....	529
11.4.18	DDR SDRAM timing configuration 4 (DDR _x _TIMING_CFG_4).....	530
11.4.19	DDR SDRAM timing configuration 5 (DDR _x _TIMING_CFG_5).....	533
11.4.20	DDR ZQ calibration control (DDR _x _DDR_ZQ_CNTL).....	535
11.4.21	DDR write leveling control (DDR _x _DDR_WRLVL_CNTL).....	537
11.4.22	DDR Self Refresh Counter (DDR _x _DDR_SR_CNTR).....	540
11.4.23	DDR Register Control Words 1 (DDR _x _DDR_SDRAM_RCW_1).....	541
11.4.24	DDR Register Control Words 2 (DDR _x _DDR_SDRAM_RCW_2).....	542
11.4.25	DDR write leveling control 2 (DDR _x _DDR_WRLVL_CNTL_2).....	543
11.4.26	DDR write leveling control 3 (DDR _x _DDR_WRLVL_CNTL_3).....	546
11.4.27	DDR SDRAM mode configuration 3 (DDR _x _DDR_SDRAM_MODE_3).....	548
11.4.28	DDR SDRAM mode configuration 4 (DDR _x _DDR_SDRAM_MODE_4).....	549
11.4.29	DDR SDRAM mode configuration 5 (DDR _x _DDR_SDRAM_MODE_5).....	550
11.4.30	DDR SDRAM mode configuration 6 (DDR _x _DDR_SDRAM_MODE_6).....	551
11.4.31	DDR SDRAM mode configuration 7 (DDR _x _DDR_SDRAM_MODE_7).....	551
11.4.32	DDR SDRAM mode configuration 8 (DDR _x _DDR_SDRAM_MODE_8).....	552
11.4.33	DDR Debug Status Register 1 (DDR _x _DDRDSR_1).....	553
11.4.34	DDR Debug Status Register 2 (DDR _x _DDRDSR_2).....	554
11.4.35	DDR Control Driver Register 1 (DDR _x _DDRCDR_1).....	555
11.4.36	DDR Control Driver Register 2 (DDR _x _DDRCDR_2).....	558

Section number	Title	Page
11.4.37	DDR IP block revision 1 (DDR _x _DDR_IP_REV1).....	559
11.4.38	DDR IP block revision 2 (DDR _x _DDR_IP_REV2).....	560
11.4.39	DDR Memory Test Control Register (DDR _x _DDR_MTCR).....	561
11.4.40	DDR Memory Test Pattern n Register (DDR _x _DDR_MTPn).....	562
11.4.41	Memory data path error injection mask high (DDR _x _DATA_ERR_INJECT_HI).....	563
11.4.42	Memory data path error injection mask low (DDR _x _DATA_ERR_INJECT_LO).....	563
11.4.43	Memory data path error injection mask ECC (DDR _x _ECC_ERR_INJECT).....	564
11.4.44	Memory data path read capture high (DDR _x _CAPTURE_DATA_HI).....	565
11.4.45	Memory data path read capture low (DDR _x _CAPTURE_DATA_LO).....	565
11.4.46	Memory data path read capture ECC (DDR _x _CAPTURE_ECC).....	566
11.4.47	Memory error detect (DDR _x _ERR_DETECT).....	567
11.4.48	Memory error disable (DDR _x _ERR_DISABLE).....	569
11.4.49	Memory error interrupt enable (DDR _x _ERR_INT_EN).....	571
11.4.50	Memory error attributes capture (DDR _x _CAPTURE_ATTRIBUTES).....	573
11.4.51	Memory error address capture (DDR _x _CAPTURE_ADDRESS).....	574
11.4.52	Memory error extended address capture (DDR _x _CAPTURE_EXT_ADDRESS).....	575
11.4.53	Single-Bit ECC memory error management (DDR _x _ERR_SBE).....	575
11.5	DDR Functional Description.....	576
11.5.1	DDR SDRAM Interface Operation.....	581
11.5.1.1	Supported DDR SDRAM Organizations.....	582
11.5.2	DDR SDRAM Address Multiplexing.....	583
11.5.3	JEDEC Standard DDR SDRAM Interface Commands.....	589
11.5.4	DDR SDRAM Interface Timing.....	591
11.5.5	DDR SDRAM Mode-Set Command Timing.....	595
11.5.6	DDR SDRAM Registered DIMM Mode.....	595
11.5.7	DDR SDRAM Write Timing Adjustments.....	596
11.5.8	DDR SDRAM Refresh.....	597
11.5.8.1	DDR SDRAM Refresh Timing.....	598

Section number	Title	Page
	11.5.8.2 DDR SDRAM Refresh and Power-Saving Modes.....	599
	11.5.8.2.1 Self-Refresh in Sleep Mode.....	600
11.5.9	DDR Data Beat Ordering.....	602
11.5.10	Page Mode and Logical Bank Retention.....	603
11.5.11	Error Checking and Correcting (ECC).....	603
11.5.12	Error Management.....	605
11.5.13	DDR Rapid Clear of Memory.....	606
11.6	DDR Initialization/Application Information.....	606
11.6.1	Programming Differences Between Memory Types.....	610
11.6.2	Supported DDR Interleaving Configurations.....	615
11.6.3	DDR SDRAM Initialization Sequence.....	617
11.6.4	Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System.....	617
	11.6.4.1 Hardware Based Self-Refresh Scheme.....	618
	11.6.4.2 Software Based Self-Refresh Scheme.....	618
	11.6.4.3 Bypassing Re-initialization During Battery-Backed Operation	618
	Chapter 12	
	I2C Modules	
12.1	I2C Overview.....	621
12.2	Introduction to I2C.....	621
12.2.1	Definition: I2C Module.....	621
12.2.2	Advantages of the I2C Bus.....	622
12.2.3	I2C Module Block Diagram.....	622
12.2.4	I2C Features Summary.....	622
12.2.5	I2C Modes of Operation.....	623
12.2.6	Definition: I2C Conditions.....	623
12.3	I2C Signal Descriptions.....	624
12.3.1	Signal Overview.....	624
12.3.2	I2C Detailed Signal Descriptions.....	624

Section number	Title	Page
12.4	I2C Controller Memory Map.....	625
12.4.1	I2C address register (I2Cx_I2CADR).....	627
12.4.2	I2C frequency divider register (I2Cx_I2CFDR).....	627
12.4.3	I2C control register (I2Cx_I2CCR).....	629
12.4.4	I2C status register (I2Cx_I2CSR).....	631
12.4.5	I2C data register (I2Cx_I2CDR).....	632
12.4.6	I2C digital filter sampling rate register (I2Cx_I2CDFSRR).....	633
12.5	I2C Functional Description.....	633
12.5.1	Notes About Module Operation.....	634
12.5.2	Transactions.....	634
12.5.2.1	Protocol Overview.....	634
12.5.2.2	Definitions.....	635
12.5.2.3	I2C Calling Address Requirements.....	635
12.5.2.4	High-Level Protocol Steps.....	636
12.5.2.5	START Condition.....	636
12.5.2.6	Slave Address Transmission.....	636
12.5.2.7	General Call (Broadcast) Addressing.....	637
12.5.2.8	Data Transmission.....	637
12.5.2.9	STOP Condition.....	638
12.5.2.10	Repeated START Condition.....	638
12.5.3	Protocol Implementation Details.....	638
12.5.3.1	Transaction Monitoring.....	638
12.5.3.2	Control Transfer.....	639
12.5.4	Bus Arbitration.....	639
12.5.4.1	Bus Arbitration Overview.....	640
12.5.4.2	Loss of Arbitration.....	640
12.5.4.3	Module Startup During a Data Transfer.....	640
12.5.5	Clock Behavior.....	640
12.5.5.1	SCL Synchronization.....	641

Section number	Title	Page
12.5.5.2	Clock Stretching.....	641
12.5.5.3	Handshaking.....	641
12.5.6	Filtering of SCL and SDA Lines.....	641
12.5.6.1	Filtering of SCL and SDA Lines-Overview.....	642
12.5.6.2	Sample Rate Control.....	642
12.6	I2C Initialization/Application Information.....	642
12.6.1	Recommended Interrupt Service Flow.....	642
12.6.2	General Programming Guidelines (for Both Master and Slave Mode).....	644
12.6.2.1	Initializing the Module.....	644
12.6.2.2	Software Response After a Transfer.....	644
12.6.2.3	Generating SCL when SDA is Low.....	645
12.6.3	Programming Guidelines Specific to Master Mode.....	646
12.6.3.1	Generating START.....	646
12.6.3.2	Generating STOP.....	646
12.6.3.3	Generating Repeated START.....	647
12.6.3.4	Loss of Arbitration and Forcing Slave Mode.....	647
12.6.4	Programming Guidelines Specific to Slave Mode.....	647
12.6.4.1	Slave Mode Interrupt Service Routine.....	647
12.6.4.2	Acknowledge Receipt During Transmission.....	648

Chapter 13 **Enhanced Local Bus Controller**

13.1	eLBC introduction.....	649
13.1.1	Overview.....	650
13.1.2	Features.....	651
13.1.3	Modes of operation.....	652
13.1.3.1	eLBC bus clock and clock ratios.....	652
13.2	eLBC external signal descriptions.....	653
13.3	Enhanced Local Bus Controller (eLBC) Memory Map.....	656
13.3.1	Base register 0 (eLBC_BR0).....	659

Section number	Title	Page
13.3.2	Options register 0 layout for GPCM Mode (eLBC_ORg0).....	661
13.3.3	Options register 0 layout for FCM Mode (eLBC_ORf0).....	664
13.3.4	Options register 0 layout for UPM Mode (eLBC_ORu0).....	668
13.3.5	Base register n (eLBC_BRn).....	672
13.3.6	Options register n layout for GPCM Mode (eLBC_ORgn).....	673
13.3.7	Options register n layout for FCM Mode (eLBC_ORfn).....	677
13.3.8	Options register n layout for UPM Mode (eLBC_ORun).....	681
13.3.9	UPM address register (eLBC_MAR).....	683
13.3.10	UPMn mode register (eLBC_MnMR).....	684
13.3.11	Memory refresh timer prescaler register (eLBC_MRTPR).....	687
13.3.12	UPM data register (eLBC_MDRu).....	687
13.3.13	FCM data register (eLBC_MDRf).....	688
13.3.14	Special operation initiation register (eLBC_LSOR).....	688
13.3.15	UPM refresh timer (eLBC_LURT).....	689
13.3.16	Transfer error status register (eLBC_LTESR).....	690
13.3.17	Transfer error disable register (eLBC_LTEDR).....	692
13.3.18	Transfer error interrupt register (eLBC_LTEIR).....	693
13.3.19	Transfer error attributes register (eLBC_LTEATR).....	694
13.3.20	Transfer error address register (eLBC_LTEAR).....	695
13.3.21	Transfer error ECC register (eLBC_LTECCR).....	696
13.3.22	Configuration register (eLBC_LBCR).....	697
13.3.23	Clock ratio register (eLBC_LCRR).....	698
13.3.24	Flash mode register (eLBC_FMR).....	700
13.3.25	Flash instruction register (eLBC_FIR).....	702
13.3.26	Flash command register (eLBC_FCR).....	704
13.3.27	Flash block address register (eLBC_FBAR).....	704
13.3.28	Flash page address register [Large Page Device (ORx[PGS] = 1)] (eLBC_FPARI).....	705
13.3.29	Flash page address register [Small Page Device (ORx[PGS] = 0)] (eLBC_FPARs).....	706
13.3.30	Flash byte count register (eLBC_FBCR).....	707

Section number	Title	Page
13.3.31	Flash ECC block n registers (eLBC_FECCn).....	708
13.4	eLBC functional description.....	708
13.4.1	Basic architecture.....	710
13.4.1.1	Address and address space checking.....	710
13.4.1.2	External address latch enable signal (LALE).....	710
13.4.1.3	Data transfer acknowledge (TA).....	711
13.4.1.4	Data buffer control (LBCTL).....	712
13.4.1.5	Parity generation and checking (LDP).....	713
13.4.1.6	Bus monitor.....	713
13.4.1.7	PLL Bypass mode.....	714
13.4.2	General-purpose chip-select machine (GPCM).....	714
13.4.2.1	GPCM read signal timing.....	715
13.4.2.2	GPCM write signal timing.....	717
13.4.2.3	Chip-select assertion timing.....	719
13.4.2.3.1	Programmable wait state configuration.....	719
13.4.2.3.2	Chip-select and write enable negation timing.....	720
13.4.2.3.3	Relaxed timing.....	721
13.4.2.3.4	Output enable (LOE_B) timing.....	724
13.4.2.3.5	Extended hold time on read accesses-GPCM.....	724
13.4.2.4	External access termination (LGTA_B).....	725
13.4.2.5	GPCM boot chip-select operation.....	726
13.4.3	Flash control machine (FCM).....	727
13.4.3.1	FCM buffer RAM	729
13.4.3.1.1	Buffer layout and page mapping for small-page NAND flash devices	730
13.4.3.1.2	Buffer layout and page mapping for large-page NAND flash devices	731
13.4.3.1.3	Error correcting codes and the spare region	732
13.4.3.2	Programming FCM.....	734
13.4.3.2.1	FCM command instructions	735
13.4.3.2.2	FCM no-operation instruction	736

Section number	Title	Page
	13.4.3.2.3 FCM address instructions	736
	13.4.3.2.4 FCM data read instructions	737
	13.4.3.2.5 FCM data write instructions	737
13.4.3.3	FCM signal timing.....	738
	13.4.3.3.1 FCM chip-select timing	738
	13.4.3.3.2 FCM command, address, and write data timing	738
	13.4.3.3.3 FCM ready/busy timing	740
	13.4.3.3.4 FCM read data timing	741
	13.4.3.3.5 FCM extended read hold timing	742
13.4.3.4	FCM boot chip-select operation	743
	13.4.3.4.1 FCM bank 0 reset initialization	744
	13.4.3.4.2 Boot block loading into the FCM buffer RAM	744
13.4.4	User-programmable machines (UPMs).....	746
13.4.4.1	UPM requests.....	747
	13.4.4.1.1 Memory access requests.....	748
	13.4.4.1.2 UPM refresh timer requests.....	749
	13.4.4.1.3 Software requests-RUN command.....	749
	13.4.4.1.4 Exception requests.....	750
13.4.4.2	Programming the UPMs.....	750
	13.4.4.2.1 UPM programming example (two sequential writes to the RAM array).....	751
	13.4.4.2.2 UPM programming example (two sequential reads from the RAM array).....	752
13.4.4.3	UPM signal timing.....	753
13.4.4.4	RAM array.....	753
	13.4.4.4.1 RAM words.....	754
	13.4.4.4.2 Chip-select signal timing (CSTn).....	758
	13.4.4.4.3 Byte select signal timing (BSTn).....	758
	13.4.4.4.4 General-purpose signals (GnTn, GOn).....	759
	13.4.4.4.5 Loop control (LOOP).....	760
	13.4.4.4.6 Repeat execution of current RAM word (REDO).....	760

Section number	Title	Page
	13.4.4.4.7 Address multiplexing (AMX).....	761
	13.4.4.4.8 Data valid and data sample control (UTA).....	762
	13.4.4.4.9 LGPL[0:5] signal negation (LAST).....	763
	13.4.4.4.10 Wait mechanism (WAEN).....	763
13.4.4.5	Extended hold time on read accesses-UPM.....	765
13.5	eLBC initialization/application information.....	766
13.5.1	Interfacing to peripherals in different address modes.....	766
13.5.1.1	Multiplexed address/data bus for 32-bit addressing.....	766
13.5.1.2	Non-multiplexed address and data buses.....	767
13.5.1.3	Peripheral hierarchy on the local bus for high bus speeds.....	767
13.5.1.4	GPCM timings.....	768
13.5.2	Bus turnaround.....	769
13.5.2.1	Address phase after previous read.....	770
13.5.2.2	Read data phase after address phase.....	770
13.5.2.3	Read-modify-write cycle for parity protected memory banks.....	770
13.5.2.4	UPM cycles with additional address phases.....	771
13.5.3	Interface to different port-size devices.....	771
13.5.4	Command sequence examples for NAND flash EEPROM.....	772
13.5.4.1	NAND flash soft reset command sequence example.....	772
13.5.4.2	NAND flash read status command sequence example.....	773
13.5.4.3	NAND flash read identification command sequence example.....	774
13.5.4.4	NAND flash page read command sequence example.....	774
13.5.4.5	NAND flash block erase command sequence example.....	775
13.5.4.6	NAND flash program command sequence example.....	776
13.5.5	Interfacing to fast-page mode DRAM using UPM.....	777
13.5.6	Interfacing to ZBT SRAM using UPM.....	786

Section number	Title	Page
	Chapter 14 Enhanced Serial Peripheral Interface	
14.1	Introduction.....	789
14.1.1	Features.....	790
14.1.2	eSPI transmission and reception process.....	791
14.1.3	Modes of operation.....	791
14.2	External signal descriptions.....	792
14.2.1	eSPI detailed signal descriptions	793
14.3	Enhanced serial peripheral interface (eSPI) memory map.....	794
14.3.1	eSPI mode register (ESPI_SPMODE).....	795
14.3.2	eSPI event register (ESPI_SPIE).....	797
14.3.3	eSPI mask register (ESPI_SPIM).....	799
14.3.4	eSPI command register (ESPI_SPCOM).....	801
14.3.5	eSPI transmit FIFO access register (ESPI_SPITF).....	803
14.3.6	eSPI receive FIFO access register (ESPI_SPIRF).....	805
14.3.7	eSPI CS0 mode register (ESPI_SPMODE0).....	806
14.3.8	eSPI CS1 mode register (ESPI_SPMODE1).....	808
14.3.9	eSPI CS2 mode register (ESPI_SPMODE2).....	810
14.3.10	eSPI CS3 mode register (ESPI_SPMODE3).....	812
14.4	eSPI transfer formats.....	813
14.5	CI and CP values for various eSPI devices.....	814
14.6	eSPI programming examples.....	815
14.6.1	24-bit address example.....	815
14.6.2	16-bit address example.....	815

Chapter 15 Enhanced Secure Digital Host Controller

15.1	eSDHC overview.....	817
15.2	eSDHC features summary.....	819
15.2.1	Data transfer modes.....	820

Section number	Title	Page
15.3	eSDHC external signal description.....	820
15.4	Enhanced Secure Digital Host Controller (eSDHC) Memory Map.....	821
15.4.1	DMA system address (eSDHC_DSADDR).....	823
15.4.2	Block attributes (eSDHC_BLKATTR).....	823
15.4.3	Command argument (eSDHC_CMDARG).....	824
15.4.4	Command transfer type (eSDHC_XFERTYP).....	825
15.4.5	Command response n (eSDHC_CMDRSPn).....	828
15.4.6	Data buffer access port (eSDHC_DATPORT).....	829
15.4.7	Present state (eSDHC_PRSSTAT).....	830
15.4.8	Protocol control (eSDHC_PROCTL).....	835
15.4.9	System control (eSDHC_SYSCTL).....	838
15.4.10	Interrupt status (eSDHC_IRQSTAT).....	841
15.4.11	Interrupt status enable (eSDHC_IRQSTATEN).....	846
15.4.12	Interrupt signal enable (eSDHC IRQSIGEN).....	849
15.4.13	Auto CMD12 status (eSDHC_AUTOC12ERR).....	851
15.4.14	Host controller capabilities (eSDHC_HOSTCAPBLT).....	854
15.4.15	Watermark level (eSDHC_WML).....	856
15.4.16	Force event (eSDHC_FEVT).....	857
15.4.17	Host controller version (eSDHC_HOSTVER).....	859
15.4.18	DMA control register (eSDHC_DCR).....	860
15.5	eSDHC functional description.....	861
15.5.1	Data buffer.....	861
15.5.1.1	Write operation sequence.....	862
15.5.1.2	Read operation sequence.....	863
15.5.1.3	Data buffer size.....	864
15.5.2	DMA interface.....	864
15.5.2.1	Internal DMA request.....	865
15.5.2.2	DMA burst length.....	865
15.5.2.3	Master interface.....	865

Section number	Title	Page
15.5.3	SD protocol unit.....	866
15.5.3.1	SD transceiver.....	866
15.5.3.2	SD clock and monitor.....	866
15.5.3.3	Command agent.....	867
15.5.3.4	Data agent.....	867
15.5.4	Clock and reset manager.....	868
15.5.5	Clock generator.....	868
15.5.6	Card insertion and removal detection.....	869
15.5.7	Power management and wake-up events.....	869
15.5.7.1	Setting wake-up events.....	870
15.6	Initialization/application information.....	870
15.6.1	Command send and response receive basic operation.....	871
15.6.2	Card identification mode.....	871
15.6.2.1	Card detect.....	871
15.6.2.2	Reset.....	872
15.6.2.3	Voltage validation.....	873
15.6.2.4	Card registry.....	874
15.6.3	Card access.....	876
15.6.3.1	Block write.....	876
15.6.3.1.1	Normal write.....	876
15.6.3.1.2	Write with pause.....	877
15.6.3.2	Block read.....	878
15.6.3.2.1	Normal read.....	878
15.6.3.2.2	Read with pause.....	879
15.6.3.3	Transfer error.....	880
15.6.3.3.1	CRC error.....	880
15.6.3.3.2	Internal DMA error.....	880
15.6.3.3.3	Auto CMD12 error.....	881
15.6.3.4	Card interrupt.....	881

Section number	Title	Page
15.6.4	Switch function.....	881
15.6.4.1	Query, enable and disable SD high speed mode.....	882
15.6.4.2	Query, enable and disable MMC high speed mode.....	882
15.6.4.3	Set MMC bus width.....	883
15.6.5	Commands for MMC/SD.....	883
15.6.6	Software restrictions.....	889

Chapter 16 Universal Serial Bus Interface

16.1	Overview.....	891
16.1.1	USB features summary.....	892
16.1.2	Modes of operation.....	892
16.2	USB External Signals.....	893
16.2.1	ULPI.....	893
16.2.2	PHY clocks.....	894
16.3	USB memory map/register definition.....	895
16.3.1	Identification register (USB_USB_ID).....	897
16.3.2	General purpose timer load n (USB_GPTIMERnLD).....	898
16.3.3	General purpose timer control n (USB_GPTIMERnCTRL).....	899
16.3.4	Capability Register Length (USB_CAPLENGTH).....	900
16.3.5	Host Controller Interface Version Number (USB_HCIVERSION).....	900
16.3.6	Host Controller Structural Parameters (USB_HCSPARAMS).....	901
16.3.7	Host Controller Capability Parameters (USB_HCCPARAMS).....	903
16.3.8	Device Controller Interface Version Number (USB_DCIVERSION).....	905
16.3.9	Device Controller Capability Parameters (USB_DCCPARAMS).....	906
16.3.10	USB Command (USB_USBCMD).....	906
16.3.11	USB Status (USB_USBSTS).....	911
16.3.12	USB Interrupt Enable (USB_USBINTR).....	915
16.3.13	USB Frame Index (USB_FRINDEX).....	917
16.3.14	Periodic Frame List Base Address [host mode] (USB_PERIODICLISTBASE).....	918

Section number	Title	Page
16.3.15	USB Device Address [device mode] (USB_DEVICEADDR).....	919
16.3.16	Next Asynchronous List Addr [host mode] (USB_ASYNCLISTADDR).....	919
16.3.17	Address at Endpoint List [device mode] (USB_ENDPOINTLISTADDR).....	920
16.3.18	Master Interface Data Burst Size (USB_BURSTSIZE).....	920
16.3.19	Transmit FIFO Tuning Controls (USB_TXFILLTUNING).....	921
16.3.20	ULPI Register Access (USB_ULPI_VIEWPORT).....	923
16.3.21	Endpoint NAK Indicaton Register (USB_ENDPTNAK).....	926
16.3.22	Endpoint NAK Indication Enable Register (USB_ENDPTNAKEN).....	926
16.3.23	Configured Flag Register (USB_CONFIGFLAG).....	927
16.3.24	Port Status/Control (USB_PORTSC).....	928
16.3.25	USB Device Mode (USB_USBMODE).....	935
16.3.26	Endpoint Setup Status (USB_ENDPTSETUPSTAT).....	936
16.3.27	Endpoint Initialization (USB_ENDPOINTPRIME).....	937
16.3.28	Endpoint Flush (USB_ENDPTFLUSH).....	938
16.3.29	Endpoint Status (USB_ENDPTSTATUS).....	938
16.3.30	Endpoint Complete (USB_ENDPTCOMPLETE).....	939
16.3.31	Endpoint Control 0 (USB_ENDPTCTRL0).....	941
16.3.32	Endpoint Control n (USB_ENDPTCTRLn).....	943
16.3.33	Snoop n (USB_SNOOPn).....	945
16.3.34	Age Count Threshold (USB_AGE_CNT_THRESH).....	946
16.3.35	Priority Control (USB_PRI_CTRL).....	947
16.3.36	System Interface Control (USB_SI_CTRL).....	948
16.3.37	Control (USB_CONTROL).....	949
16.4	Functional description.....	951
16.4.1	System interface.....	951
16.4.2	DMA engine.....	951
16.4.3	FIFO RAM controller.....	952
16.4.4	PHY interface.....	952

Section number	Title	Page
16.5	Host data structures.....	953
16.5.1	Periodic frame list.....	953
16.5.2	Asynchronous list queue head pointer.....	955
16.5.3	Isochronous (high-speed) transfer descriptor (iTД).....	956
16.5.3.1	Next link pointer-iTD.....	956
16.5.3.2	iTД transaction status and control list.....	957
16.5.3.3	iTД buffer page pointer list (plus).....	958
16.5.4	Split transaction isochronous transfer descriptor (siTD).....	960
16.5.4.1	Next link pointer-siTD.....	960
16.5.4.2	siTD endpoint capabilities/characteristics.....	961
16.5.4.3	siTD transfer state.....	962
16.5.4.4	siTD buffer pointer list (plus).....	963
16.5.4.5	siTD back link pointer.....	964
16.5.5	Queue element transfer descriptor (qTD).....	964
16.5.5.1	Next qTD pointer.....	965
16.5.5.2	Alternate next qTD pointer.....	965
16.5.5.3	qTD token.....	966
16.5.5.4	qTD buffer page pointer list.....	968
16.5.6	Queue head.....	969
16.5.6.1	Queue head horizontal link pointer.....	970
16.5.6.2	Endpoint capabilities/characteristics	970
16.5.6.3	Transfer overlay.....	972
16.5.7	Periodic frame span traversal node (FSTN).....	974
16.5.7.1	FSTN normal path pointer.....	974
16.5.7.2	FSTN back path link pointer.....	975
16.6	Host operations.....	976
16.6.1	Host controller initialization.....	976
16.6.2	Power port.....	977
16.6.3	Reporting over-current.....	977

Section number	Title	Page
16.6.4	Suspend/resume	978
16.6.4.1	Port suspend/resume.....	978
16.6.5	Schedule traversal rules.....	980
16.6.6	Periodic schedule frame boundaries vs. bus frame boundaries.....	982
16.6.7	Periodic schedule.....	985
16.6.8	Managing isochronous transfers using iTDs.....	986
16.6.8.1	Host controller operational model for iTDs.....	987
16.6.8.2	Software operational model for iTDs.....	989
16.6.8.2.1	Periodic scheduling threshold.....	991
16.6.9	Asynchronous schedule.....	992
16.6.9.1	Adding queue heads to asynchronous schedule.....	994
16.6.9.2	Removing queue heads from asynchronous schedule.....	994
16.6.9.3	Empty asynchronous schedule detection	997
16.6.9.4	Asynchronous schedule traversal: Start event.....	998
16.6.9.5	Reclamation status bit (USBSTS Register).....	998
16.6.10	Managing control/bulk/interrupt transfers via queue heads.....	999
16.6.10.1	Buffer pointer list use for data streaming with qTDs	1000
16.6.10.2	Adding interrupt queue heads to the periodic schedule	1002
16.6.10.3	Managing transfer complete interrupts from queue heads	1002
16.6.11	Ping control.....	1003
16.6.12	Split transactions.....	1004
16.6.12.1	Split transactions for asynchronous transfers	1005
16.6.12.1.1	Asynchronous-do-start-split.....	1005
16.6.12.1.2	Asynchronous-do-complete-split	1006
16.6.12.2	Split transaction interrupt	1007
16.6.12.2.1	Split transaction scheduling mechanisms for interrupt	1008
16.6.12.2.2	Host controller operational model for FSTNs	1011
16.6.12.2.3	Software operational model for FSTNs.....	1014
16.6.12.2.4	Tracking split transaction progress for interrupt transfers	1015

Section number	Title	Page
16.6.12.2.5	Split transaction execution state machine for interrupt	1016
16.6.12.2.6	Periodic interrupt-do-start-split	1017
16.6.12.2.7	Periodic interrupt-do-complete-split	1018
16.6.12.2.8	Managing the QH[FrameTag] field	1022
16.6.12.2.9	Rebalancing the periodic schedule	1023
16.6.12.3	Split transaction isochronous	1023
16.6.12.3.1	Split transaction scheduling mechanisms for isochronous	1024
16.6.12.3.2	Tracking split transaction progress for isochronous transfers	1029
16.6.12.3.3	Split transaction execution state machine for isochronous	1031
16.6.12.3.4	Periodic isochronous-do-start-split	1032
16.6.12.3.5	Periodic isochronous-do complete split.....	1034
16.6.12.3.6	Complete-split for scheduling boundary cases 2a, 2b.....	1037
16.6.12.3.7	Split transaction for isochronous-processing example.....	1039
16.6.13	Port test modes.....	1041
16.6.14	Interrupts.....	1041
16.6.14.1	Transfer/transaction based interrupts.....	1042
16.6.14.1.1	Transaction error.....	1043
16.6.14.1.2	Serial bus babble.....	1043
16.6.14.1.3	Data buffer error	1044
16.6.14.1.4	USB interrupt (interrupt on completion (IOC)).....	1045
16.6.14.1.5	Short packet.....	1045
16.6.14.2	Host controller event interrupts.....	1046
16.6.14.2.1	Port change events.....	1046
16.6.14.2.2	Frame list rollover.....	1046
16.6.14.2.3	Interrupt on async advance.....	1046
16.6.14.2.4	Host system error.....	1047
16.7	Device data structures.....	1047
16.7.1	Endpoint queue head.....	1048
16.7.1.1	Endpoint capabilities/characteristics	1049

Section number	Title	Page
16.7.1.2	Transfer overlay	1050
16.7.1.3	Current dTD pointer.....	1050
16.7.1.4	Setup buffer.....	1051
16.7.2	Endpoint transfer descriptor (dTD).....	1051
16.8	Device operational model.....	1054
16.8.1	Device controller initialization.....	1054
16.8.2	Port state and control.....	1055
16.8.2.1	Bus reset.....	1057
16.8.2.2	Suspend/resume	1058
16.8.2.2.1	Suspend description.....	1058
16.8.2.2.2	Suspend operational model.....	1058
16.8.2.2.3	Resume.....	1059
16.8.3	Managing endpoints.....	1059
16.8.3.1	Endpoint initialization.....	1060
16.8.3.1.1	Stalling.....	1060
16.8.3.2	Data toggle.....	1061
16.8.3.2.1	Data toggle reset.....	1061
16.8.3.2.2	Data toggle inhibit.....	1062
16.8.3.3	Device operational model for packet transfers.....	1062
16.8.3.3.1	Priming transmit endpoints.....	1063
16.8.3.3.2	Priming receive endpoints.....	1063
16.8.3.4	Interrupt/bulk endpoint operational model.....	1063
16.8.3.4.1	Interrupt/bulk endpoint bus response matrix.....	1065
16.8.3.5	Control endpoint operation model.....	1066
16.8.3.5.1	Setup phase.....	1066
16.8.3.5.2	Data phase.....	1067
16.8.3.5.3	Status phase.....	1067
16.8.3.5.4	Control endpoint bus response matrix.....	1068

Section number	Title	Page
16.8.3.6	Isochronous endpoint operational model.....	1068
16.8.3.6.1	Isochronous pipe synchronization.....	1070
16.8.3.6.2	Isochronous endpoint bus response matrix.....	1070
16.8.4	Managing queue heads.....	1071
16.8.4.1	Queue head initialization.....	1072
16.8.4.2	Operational model for setup transfers.....	1072
16.8.5	Managing transfers with transfer descriptors.....	1073
16.8.5.1	Software link pointers.....	1073
16.8.5.2	Building a transfer descriptor.....	1074
16.8.5.3	Executing a transfer descriptor.....	1074
16.8.5.4	Transfer completion.....	1075
16.8.5.5	Flushing/depriming an endpoint.....	1076
16.8.5.6	Device error matrix.....	1077
16.8.6	Servicing interrupts.....	1077
16.8.6.1	High-frequency interrupts.....	1077
16.8.6.2	Low-frequency interrupts.....	1078
16.8.6.3	Error interrupts.....	1078
16.9	Deviations from the EHCI specifications.....	1079
16.9.1	Embedded transaction translator function.....	1079
16.9.1.1	Capability registers.....	1079
16.9.1.2	Operational registers.....	1080
16.9.1.3	Discovery differences	1080
16.9.1.4	Data structures.....	1081
16.9.1.5	Operational model.....	1082
16.9.1.5.1	Microframe pipeline.....	1082
16.9.1.5.2	Split state machines.....	1082
16.9.1.5.3	Asynchronous transaction scheduling and buffer management.....	1083
16.9.1.5.4	Periodic transaction scheduling and buffer management.....	1083
16.9.1.5.5	Multiple transaction translators.....	1084

Section number	Title	Page
16.9.2	Device operation.....	1084
16.9.3	Non-zero fields the register file.....	1084
16.9.4	SOF interrupt.....	1084
16.9.5	Embedded design.....	1085
16.9.5.1	Frame adjust register.....	1085
16.9.6	Miscellaneous variations from EHCI.....	1085
16.9.6.1	Discovery.....	1085
16.9.6.1.1	Port reset.....	1085
16.9.6.1.2	Port speed detection.....	1086

Chapter 17 **DUART**

17.1	DUART Overview.....	1087
17.2	DUART Features Summary.....	1088
17.3	DUART Modes of Operation.....	1089
17.3.1	DUART Signal Mode Selection.....	1089
17.4	DUART External Signal Descriptions.....	1090
17.5	DUART Memory Map/Register Definition.....	1091
17.5.1	UART receiver buffer register (DUART _x _URBR _n).....	1094
17.5.2	UART transmitter holding register (DUART _x _UTHR _n).....	1095
17.5.3	UART divisor least significant byte register (DUART _x _UDLB _n).....	1095
17.5.4	UART divisor most significant byte register (DUART _x _UDMB _n).....	1097
17.5.5	UART interrupt enable register (DUART _x _UIER _n).....	1097
17.5.6	UART interrupt ID register (DUART _x _UIIR _n).....	1098
17.5.7	UART FIFO control register (DUART _x _UFCR _n).....	1100
17.5.8	UART alternate function register (DUART _x _UAFR _n).....	1101
17.5.9	UART line control register (DUART _x _ULCR _n).....	1101
17.5.10	UART modem control register (DUART _x _UMCR _n).....	1103
17.5.11	UART line status register (DUART _x _ULSR _n).....	1104
17.5.12	UART modem status register (DUART _x _UMSR _n).....	1106

Section number	Title	Page
17.5.13	UART scratch register (DUART _x _USCR _n).....	1106
17.5.14	UART DMA status register (DUART _x _UDSR _n).....	1107
17.6	DUART Functional Description.....	1108
17.6.1	Serial Interface.....	1109
17.6.1.1	START Bit.....	1109
17.6.1.2	Data Transfer.....	1110
17.6.1.3	Parity Bit.....	1110
17.6.1.4	STOP Bit.....	1110
17.6.2	Baud-Rate Generator Logic.....	1110
17.6.3	Local Loopback Mode.....	1111
17.6.4	Errors.....	1111
17.6.4.1	Framing Error.....	1111
17.6.4.2	Parity Error.....	1112
17.6.4.3	Overrun Error.....	1112
17.6.5	FIFO Mode.....	1112
17.6.5.1	FIFO Interrupts.....	1112
17.6.5.2	DMA Mode Select.....	1113
17.6.5.3	Interrupt Control Logic.....	1113
17.7	DUART Initialization/Application Information.....	1114

Chapter 18 PCI Express Interface Controller

18.1	Introduction.....	1115
18.1.1	Outbound transactions.....	1117
18.1.2	Inbound transactions.....	1118
18.2	PCI Express features summary.....	1119
18.3	PCI Express modes of operation.....	1120
18.3.1	Root complex/endpoint modes.....	1120
18.3.2	Link width.....	1120
18.3.3	Link speed.....	1120

Section number	Title	Page
18.4	PCI Express signal descriptions.....	1121
18.5	Memory map/register overview.....	1122
18.6	PCI Express memory-mapped registers.....	1122
18.6.1	PCI Express configuration address register (PEX _x _PEX_CONFIG_ADDR).....	1130
18.6.2	PCI Express configuration data register (PEX _x _PEX_CONFIG_DATA).....	1131
18.6.3	PCI Express outbound completion timeout register (PEX _x _PEX_OTB_CPL_TOR).....	1132
18.6.4	PCI Express configuration retry timeout register (PEX _x _PEX_CONF_RTY_TOR).....	1133
18.6.5	PCI Express configuration register (PEX _x _PEX_CONFIG).....	1134
18.6.6	PCI Express PME & message detect register (PEX _x _PEX_PME_MES_DR).....	1136
18.6.7	PCI Express PME & message disable register (PEX _x _PEX_PME_MES_DISR).....	1138
18.6.8	PCI Express PME & message interrupt enable register (PEX _x _PEX_PME_MES_IER).....	1140
18.6.9	PCI Express power management command register (PEX _x _PEX_PMCR).....	1142
18.6.10	IP block revision register 1 (PEX _x _PEX_IP_BLK_REV1).....	1142
18.6.11	IP block revision register 2 (PEX _x _PEX_IP_BLK_REV2).....	1143
18.6.12	PCI Express outbound translation address register (PEX _x _PEXOTAR _n).....	1143
18.6.13	PCI Express outbound translation extended address register n (PEX _x _PEXOTEAR _n).....	1144
18.6.14	PCI Express outbound window attributes register n (PEX _x _PEXOWAR0).....	1145
18.6.15	PCI Express outbound window base address register n (PEX _x _PEXOWBAR _n).....	1147
18.6.16	PCI Express outbound window attributes register n (PEX _x _PEXOWAR _n).....	1148
18.6.17	PCI Express outbound window attributes register 3 (PEX _x _PEXOWAR3).....	1151
18.6.18	PCI Express outbound window attributes register 4 (PEX _x _PEXOWAR4).....	1154
18.6.19	PCI Express MSI inbound translation address register (PEX _x _PEXMSIITAR).....	1156
18.6.20	PCI Express MSI inbound window base address register (PEX _x _PEXMSIIWBAR).....	1157
18.6.21	PCI Express MSI inbound window base extended address register (PEX _x _PEXMSIIBEAR).....	1157
18.6.22	PCI Express MSI inbound window attributes register (PEX _x _PEXMSIIWAR).....	1158
18.6.23	PCI Express inbound translation address register n (PEX _x _PEXITAR _n).....	1160
18.6.24	PCI Express inbound window base address register n (PEX _x _PEXIWBAR _n).....	1161
18.6.25	PCI Express inbound window base extended address register n (PEX _x _PEXIIBEAR _n).....	1161
18.6.26	PCI Express inbound window attributes register n (PEX _x _PEXIWAR _n).....	1162

Section number	Title	Page
18.6.27	PCI Express error detect register (PEX _x _PEX_ERR_DR).....	1165
18.6.28	PCI Express error interrupt enable register (PEX _x _PEX_ERR_EN).....	1168
18.6.29	PCI Express error disable register (PEX _x _PEX_ERR_DISR).....	1171
18.6.30	PCI Express error capture status register (PEX _x _PEX_ERR_CAP_STAT).....	1173
18.6.31	PCI Express error capture register n (PEX _x _PEX_ERR_CAP_R _n).....	1174
18.7	PCI Express configuration-space registers.....	1175
18.7.1	PCI compatible configuration headers.....	1175
18.8	Type 0 configuration header registers.....	1177
18.8.1	PCI Express Vendor ID Register (Vendor_ID_Register).....	1178
18.8.2	PCI Express Device ID Register (Device_ID_Register).....	1178
18.8.3	PCI Express Command Register (Command_Register).....	1179
18.8.4	PCI Express Status Register (Status_Register).....	1181
18.8.5	PCI Express Revision ID Register (Revision_ID_Register).....	1182
18.8.6	PCI Express Class Code Register (Class_Code_Register).....	1182
18.8.7	PCI Express Cache Line Size Register (Cache_Line_Size_Register).....	1183
18.8.8	PCI Express Latency Timer Register (Latency_Timer_Register).....	1184
18.8.9	PCI Express Header Type Register (Header_Type_Register).....	1184
18.8.10	PCI Express Base Address Register 0 (PEXCSRBAR).....	1185
18.8.11	PCI Express Base Address Register 1 (BAR1).....	1186
18.8.12	PCI Express Base Address Register 2,4 (BAR _n).....	1187
18.8.13	PCI Express Base Address Register 3,5 (BAR _n).....	1188
18.8.14	PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register).....	1189
18.8.15	PCI Express Subsystem ID Register (Subsystem_ID_Register).....	1190
18.8.16	Capabilities Pointer Register (Capabilities_Pointer_Register).....	1190
18.8.17	PCI Express Interrupt Line Register (Interrupt_Line_Register).....	1191
18.8.18	PCI Express Interrupt Pin Register (Interrupt_Pin_Register).....	1191
18.8.19	PCI Express Minimum Grant Register (Minimum_Grant_Register).....	1192
18.8.20	PCI Express Maximum Latency Register (Maximum_Latency_Register).....	1192

Section number	Title	Page
18.9	Type 1 configuration header registers.....	1192
18.9.1	PCI Express Base Address Register 0 (PEXCSRBAR).....	1194
18.9.2	PCI Express Primary Bus Number Register (Primary_Bus_Number_Register).....	1195
18.9.3	PCI Express Secondary Bus Number Register (Secondary_Bus_Number_Register).....	1195
18.9.4	PCI Express Subordinate Bus Number Register (Subordinate_Bus_Number_Register).....	1195
18.9.5	PCI Express I/O Base Register (IO_Base_Register).....	1196
18.9.6	PCI Express I/O Limit Register (IO_Limit_Register).....	1196
18.9.7	PCI Express Secondary Status Register (Secondary_Status_Register).....	1197
18.9.8	PCI Express Memory Base Register (Memory_Base_Register).....	1198
18.9.9	PCI Express Memory Limit Register (Memory_Limit_Register).....	1198
18.9.10	PCI Express Prefetchable Memory Base Register (Prefetchable_Memory_Base_Register).....	1199
18.9.11	PCI Express Prefetchable Memory Limit Register (Prefetchable_Memory_Limit_Register).....	1199
18.9.12	PCI Express Prefetchable Base Upper 32 Bits Register (Prefetchable_Base_Upper_32_Bits_Register)...	1200
18.9.13	PCI Express Prefetchable Limit Upper 32 Bits Register (Prefetchable_Limit_Upper_32_Bits_Register).	1200
18.9.14	PCI Express I/O Base Upper 16 Bits Register (IO_Base_Upper_16_Bits_Register).....	1201
18.9.15	PCI Express I/O Limit Upper 16 Bits Register (IO_Limit_Upper_16_Bits_Register).....	1201
18.9.16	Capabilities Pointer Register (Capabilities_Pointer_Register).....	1202
18.9.17	PCI Express Interrupt Line Register (Interrupt_Line_Register).....	1202
18.9.18	PCI Express Interrupt Pin Register (Interrupt_Pin_Register).....	1202
18.9.19	PCI Express Bridge Control Register (Bridge_Control_Register).....	1203
18.10	PCI compatible device-specific configuration space.....	1205
18.10.1	PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register).....	1206
18.10.2	PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)....	1206
18.10.3	PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register).....	1207
18.10.4	PCI Express Power Management Data Register (Power_Management_Data_Register).....	1208
18.10.5	PCI Express Capability ID Register (Capability_ID_Register).....	1208
18.10.6	PCI Express Capabilities Register (Capabilities_Register).....	1208

Section number	Title	Page
18.10.7	PCI Express Device Capabilities Register (Device_Capabilities_Register).....	1209
18.10.8	PCI Express Device Control Register (Device_Control_Register).....	1210
18.10.9	PCI Express Device Status Register (Device_Status_Register).....	1211
18.10.10	PCI Express Link Capabilities Register (Link_Capabilities_Register).....	1212
18.10.11	PCI Express Link Control Register (Link_Control_Register).....	1214
18.10.12	PCI Express Link Status Register (Link_Status_Register).....	1215
18.10.13	PCI Express Slot Capabilities Register (Slot_Capabilities_Register).....	1216
18.10.14	PCI Express Slot Control Register (Slot_Control_Register).....	1217
18.10.15	PCI Express Slot Status Register (Slot_Status_Register).....	1219
18.10.16	PCI Express Root Control Register (Root_Control_Register).....	1220
18.10.17	PCI Express Root Status Register (Root_Status_Register).....	1221
18.10.18	PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register).....	1222
18.10.19	PCI Express Device Control 2 Register (Device_Control_2_Register).....	1222
18.10.20	PCI Express Link Control 2 Register (Link_Control_2_Register).....	1223
18.10.21	PCI Express Link Status 2 Register (Link_Status_2_Register).....	1224
18.10.22	PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register).....	1224
18.10.23	PCI Express MSI Message Control Register (MSI_Message_Control_Register).....	1225
18.10.24	PCI Express MSI Message Address Register (MSI_Message_Address_Register).....	1225
18.10.25	PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register).....	1226
18.10.26	PCI Express MSI Message Data Register (MSI_Message_Data_Register).....	1226
18.11	PCI Express extended configuration space.....	1227
18.11.1	PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reportng_Capability_ID_Register).....	1228
18.11.2	PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register).....	1228
18.11.3	PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register).....	1230
18.11.4	PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register).....	1231
18.11.5	PCI Express Correctable Error Status Register (Correctable_Error_Status_Register).....	1232
18.11.6	PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register).....	1233

Section number	Title	Page
18.11.7	PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register).....	1235
18.11.8	PCI Express Header Log Register 1 (Header_Log_Register_DWORD1).....	1236
18.11.9	PCI Express Header Log Register 2 (Header_Log_Register_DWORD2).....	1237
18.11.10	PCI Express Header Log Register 3 (Header_Log_Register_DWORD3).....	1237
18.11.11	PCI Express Header Log Register 4 (Header_Log_Register_DWORD4).....	1238
18.11.12	PCI Express Root Error Command Register (Root_Error_Command_Register).....	1239
18.11.13	PCI Express Root Error Status Register (Root_Error_Status_Register).....	1240
18.11.14	PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register).....	1241
18.11.15	PCI Express Error Source ID Register (Error_Source_ID_Register).....	1241
18.11.16	LTSSM State Status Register (LTSSM_State_Status_Register).....	1242
18.11.17	N_FTS Control Register (PEX_N_FTS_CTL).....	1244
18.11.18	ACK Replay Time-Out Register (PEX_ACK_REPLAY_TIMEOUT).....	1245
18.11.19	PCI Express Controller Core Clock Ratio Register (Controller_Core_Clock_Ratio_Register).....	1245
18.11.20	PCI Express Power Management Timer Register (Power_Management_Timer_Register).....	1246
18.11.21	PCI Express PME Time-Out Register (PME_Time_Out_Register).....	1247
18.11.22	PCI Express Subsystem Vendor ID Update Register (Subsystem_Vendor_ID_Update_Register).....	1247
18.11.23	Configuration Ready Register (Configuration_Ready_Register).....	1248
18.11.24	Flow Control Update Timeout Register (Flow_Control_Update_Timeout_Register).....	1249
18.11.25	Secondary Status Interrupt Mask Register (Secondary_Status_Interrupt_Mask_Register).....	1250
18.12	Functional description.....	1251
18.12.1	Architecture.....	1252
18.12.1.1	PCI Express transactions.....	1252
18.12.1.2	Byte ordering.....	1253
18.12.1.2.1	Address invariance.....	1254
18.12.1.2.2	Data invariance.....	1255
18.12.1.2.3	Byte order for configuration transactions.....	1257
18.12.1.3	Lane reversal.....	1258
18.12.1.4	Transaction ordering rules.....	1259

Section number	Title	Page
18.12.1.5	PCI Express outbound ATMUs.....	1261
18.12.1.6	PCI Express inbound ATMUs.....	1261
18.12.1.6.1	EP inbound ATMU implementation.....	1262
18.12.1.6.2	RC inbound ATMU implementation.....	1262
18.12.1.7	Memory space addressing.....	1263
18.12.1.8	I/O space addressing.....	1263
18.12.1.9	Configuration space addressing.....	1264
18.12.1.10	PCI Express configuration space access.....	1264
18.12.1.10.1	RC configuration register access.....	1264
	18.12.1.10.1.1 PCI Express configuration access register mechanism.....	1265
	18.12.1.10.1.2 Outbound ATMU configuration mechanism (RC-only).....	1265
18.12.1.10.2	EP configuration register access.....	1266
18.12.1.11	Serialization of configuration and I/O writes.....	1267
18.12.1.12	Messages.....	1267
18.12.1.12.1	Outbound ATMU message generation.....	1267
18.12.1.12.2	Inbound messages.....	1269
18.12.1.13	Error handling.....	1270
18.12.1.13.1	PCI Express error logging and signaling.....	1271
18.12.1.13.2	PCI Express controller internal interrupt sources.....	1272
18.12.1.13.3	Error conditions	1273
18.12.1.13.4	Error capture registers.....	1276
	18.12.1.13.4.1 Error capture registers (outbound error).....	1276
	18.12.1.13.4.2 Error capture registers (inbound error).....	1277
18.12.2	Interrupts.....	1280
18.12.2.1	EP interrupt generation.....	1281
18.12.2.1.1	Hardware INTx message generation.....	1281
18.12.2.1.2	Hardware MSI generation.....	1281
18.12.2.1.3	Software INTx message generation.....	1281
18.12.2.1.4	Software MSI generation.....	1281

Section number	Title	Page
	18.12.2.2 RC handling of INTx message and MSI interrupts.....	1282
	18.12.2.2.1 INTx message handling.....	1282
	18.12.2.2.2 MSI handling.....	1282
18.12.3	Initial credit advertisement.....	1282
18.12.4	Power management.....	1283
	18.12.4.1 L2/L3 ready link state.....	1284
18.12.5	Hot reset.....	1284
18.12.6	Link down.....	1285
18.13	Initialization/application information.....	1285
18.13.1	EP Boot mode and inbound configuration transactions.....	1285
18.13.2	Automatic link retraining during initialization.....	1286
18.13.3	Configuration accesses and inbound writes to CCSR space.....	1286
18.13.4	Configuring ACK replay time-out when ASPM is enabled.....	1287

Chapter 19 Serial RapidIO Interface

19.1	Serial RapidIO overview.....	1289
19.2	Serial RapidIO features summary.....	1290
19.3	Serial RapidIO modes of operation.....	1291
	19.3.1 RapidIO port.....	1291
	19.3.2 Message unit.....	1291
19.4	LP-serial signal descriptions.....	1292
	19.4.1 Serial Rapid I/O overview.....	1292
	19.4.2 Serial Rapid I/O detailed signal descriptions.....	1292
	19.4.2.1 SD_TXn/SD_TX n_B-outputs.....	1292
	19.4.2.2 SD_RXn/SD_RX n_B-inputs.....	1293
19.5	Serial RapidIO Memory Map/Register Definition.....	1293
	19.5.1 Device identity capability register (SRIO_DIDCAR).....	1306
	19.5.2 Device information capability register (SRIO_DICAR).....	1307
	19.5.3 Assembly identity capability register (SRIO_AIDCAR).....	1307

Section number	Title	Page
19.5.4	Assembly information capability register (SRIO_AICAR).....	1308
19.5.5	Processing element features capability register (SRIO_PEFCAR).....	1309
19.5.6	Source operations capability register (SRIO_SOCAR).....	1311
19.5.7	Destination operations capability register (SRIO_DOCAR).....	1314
19.5.8	Mailbox command and status register (SRIO_MCSR).....	1317
19.5.9	Port -Write and doorbell command and status register (SRIO_PWDCSR).....	1319
19.5.10	Processing element logical layer control command and status register (SRIO_PELLCCSR).....	1321
19.5.11	Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR).....	1322
19.5.12	Base device ID command and status register (SRIO_BDIDCSR).....	1323
19.5.13	Host base device ID lock command and status register (SRIO_HBDIDLCSR).....	1324
19.5.14	Component tag command and status register (SRIO_CTCCSR).....	1324
19.5.15	Port maintenance block header 0 (SRIO_PMBH0).....	1325
19.5.16	Port link time-out control command and status register (SRIO_PLTOCCSR).....	1325
19.5.17	Port response time-out control command and status register (SRIO_PRTOCCSR).....	1326
19.5.18	Port General control command and status register (SRIO_PGCCSR).....	1327
19.5.19	Port 1 Link maintenance request command and status register (SRIO_P1LMREQCSR).....	1328
19.5.20	Port 1 Link maintenance response command and status register (SRIO_P1LMRESPCSR).....	1328
19.5.21	Port 1 Local ackID status command and status register (SRIO_P1LASCSR).....	1329
19.5.22	Port 1 Error and status command and status register (SRIO_P1ESCSR).....	1331
19.5.23	Port 1 Control command and status register (SRIO_P1CCSR).....	1334
19.5.24	Port 2 Link maintenance request command and status register (SRIO_P2LMREQCSR).....	1336
19.5.25	Port 2 Link maintenance response command and status register (SRIO_P2LMRESPCSR).....	1336
19.5.26	Port 2 Local ackID status command and status register (SRIO_P2LASCSR).....	1337
19.5.27	Port 2 Error and status command and status register (SRIO_P2ESCSR).....	1339
19.5.28	Port 2 Control command and status register (SRIO_P2CCSR).....	1342
19.5.29	Error reporting block header (SRIO_ERBH).....	1344
19.5.30	Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR).....	1344
19.5.31	Logical/Transport layer error enable command and status register (SRIO_LTLEECSR).....	1347
19.5.32	Logical/Transport layer address capture command and status register (SRIO_LTLACCSR).....	1348

Section number	Title	Page
19.5.33	Logical/Transport layer device ID capture command and status register (SRIO_LTLDIDCCSR).....	1349
19.5.34	Logical/Transport layer control capture command and status register (SRIO_LTLCCCSR).....	1350
19.5.35	Port 1 Error detect command and status register (SRIO_P1EDCSR).....	1351
19.5.36	Port 1 Error rate enable command and status register (SRIO_P1ERECSR).....	1352
19.5.37	Port 1 Error capture attributes command and status register (SRIO_P1ECACSR).....	1354
19.5.38	Port 1 Packet/control symbol error capture command and status register 0 (SRIO_P1PCSECCSR0).....	1355
19.5.39	Port 1 Packet error capture command and status register 1 (SRIO_P1PECCSR1).....	1356
19.5.40	Port 1 Packet error capture command and status register 2 (SRIO_P1PECCSR2).....	1356
19.5.41	Port 1 Packet error capture command and status register 3 (SRIO_P1PECCSR3).....	1357
19.5.42	Port 1 Error rate command and status register (SRIO_P1ERCSR).....	1357
19.5.43	Port 1 Error rate threshold command and status register (SRIO_P1ERTCSR).....	1358
19.5.44	Port 2 Error detect command and status register (SRIO_P2EDCSR).....	1359
19.5.45	Port 2 Error rate enable command and status register (SRIO_P2ERECSR).....	1361
19.5.46	Port 2 Error capture attributes command and status register (SRIO_P2ECACSR).....	1362
19.5.47	Port 2 Packet/control symbol error capture command and status register 0 (SRIO_P2PCSECCSR0).....	1363
19.5.48	Port 2 Packet error capture command and status register 1 (SRIO_P2PECCSR1).....	1364
19.5.49	Port 2 Packet error capture command and status register 2 (SRIO_P2PECCSR2).....	1364
19.5.50	Port 2 Packet error capture command and status register 3 (SRIO_P2PECCSR3).....	1365
19.5.51	Port 2 Error rate command and status register (SRIO_P2ERCSR).....	1365
19.5.52	Port 2 Error rate threshold command and status register (SRIO_P2ERTCSR).....	1366
19.5.53	Logical layer configuration register (SRIO_LLCSR).....	1367
19.5.54	Error / port-write interrupt status register (SRIO_EPWISR).....	1369
19.5.55	Logical retry error threshold configuration register (SRIO_LRETCR).....	1370
19.5.56	Physical retry error threshold configuration register (SRIO_PRETCR).....	1370
19.5.57	Port 1 Alternate device ID command and status register (SRIO_P1ADIDCSR).....	1371
19.5.58	Port 1 accept-all configuration register (SRIO_P1AACR).....	1372
19.5.59	Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR).....	1372
19.5.60	Port 1 Implementation error command and status register (SRIO_P1IECSR).....	1374
19.5.61	Port 1 Physical configuration register (SRIO_P1PCR).....	1375

Section number	Title	Page
19.5.62	Port 1 Serial link command and status register (SRIO_P1SLCSR).....	1376
19.5.63	Port 1 Serial link error injection configuration register (SRIO_P1SLEICR).....	1377
19.5.64	Port 2 Alternate device ID command and status register (SRIO_P2ADIDCSR).....	1378
19.5.65	Port 2 accept-all configuration register (SRIO_P2AACR).....	1379
19.5.66	Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR).....	1379
19.5.67	Port 2 Implementation error command and status register (SRIO_P2IECSR).....	1380
19.5.68	Port 2 Physical configuration register (SRIO_P2PCR).....	1381
19.5.69	Port 2 Serial link command and status register (SRIO_P2SLCSR).....	1382
19.5.70	Port 2 Serial link error injection configuration register (SRIO_P2SLEICR).....	1383
19.5.71	IP Block Revision Register 1 (SRIO_IPBRR1).....	1384
19.5.72	IP Block Revision Register 2 (SRIO_IPBRR2).....	1384
19.5.73	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR0).....	1385
19.5.74	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEARn).....	1386
19.5.75	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWARN).....	1387
19.5.76	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTARn).....	1390
19.5.77	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBARn).....	1391
19.5.78	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1Rn).....	1392
19.5.79	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2Rn).....	1394
19.5.80	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3Rn).....	1396
19.5.81	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTARn).....	1398
19.5.82	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBARn).....	1399
19.5.83	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWARn).....	1400
19.5.84	Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0).....	1402
19.5.85	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR0).....	1404
19.5.86	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEARn).....	1406
19.5.87	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWARN).....	1407
19.5.88	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTARn).....	1410
19.5.89	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBARn).....	1411
19.5.90	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1Rn).....	1412

Section number	Title	Page
19.5.91	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2Rn).....	1414
19.5.92	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3Rn).....	1416
19.5.93	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTARn).....	1418
19.5.94	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBARn).....	1419
19.5.95	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWARn).....	1420
19.5.96	Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0).....	1422
19.5.97	Outbound message n mode register (SRIO_OMnMR).....	1424
19.5.98	Outbound message n status register (SRIO_OMnSR).....	1427
19.5.99	Extended outbound message n descriptor queue dequeue pointer address register (SRIO_EOMnDQDPAR).....	1429
19.5.100	Outbound message n descriptor queue dequeue pointer address register (SRIO_OMnDQDPAR).....	1430
19.5.101	Extended outbound message n source address register (SRIO_EOMnSAR).....	1431
19.5.102	Outbound message n source address register (SRIO_OMnSAR).....	1431
19.5.103	Outbound message n destination port register (SRIO_OMnDPR).....	1432
19.5.104	Outbound message n destination attributes Register (SRIO_OMnDATR).....	1433
19.5.105	Outbound message n double-word count register (SRIO_OMnDCR).....	1434
19.5.106	Extended outbound message n descriptor queue enqueue pointer address register (SRIO_EOMnDQEPAR).....	1435
19.5.107	Outbound message n descriptor queue enqueue pointer address register (SRIO_OMnDQEPAR).....	1435
19.5.108	Outbound message n retry error threshold configuration register (SRIO_OMnRETCR).....	1436
19.5.109	Outbound message n multicast group register (SRIO_OMnMGR).....	1437
19.5.110	Outbound message n multicast list register (SRIO_OMnMLR).....	1438
19.5.111	Inbound message n mode register (SRIO_IMnMR).....	1439
19.5.112	Inbound message n status register (SRIO_IMnSR).....	1442
19.5.113	Extended inbound message n frame queue dequeue pointer address register (SRIO_EIMnFQDPAR).....	1444
19.5.114	Inbound message n frame queue dequeue pointer address register (SRIO_IMnFQDPAR).....	1444
19.5.115	Extended inbound message n frame queue enqueue pointer address register (SRIO_EIMnFQEPAR).....	1445
19.5.116	Inbound message n frame queue enqueue pointer address register (SRIO_IMnFQEPAR).....	1446
19.5.117	Inbound message n maximum interrupt report interval register (SRIO_IMnMIRIR).....	1447

Section number	Title	Page
19.5.118	Outbound doorbell mode register (SRIO_ODMR).....	1447
19.5.119	Outbound doorbell status register (SRIO_ODSR).....	1449
19.5.120	Outbound doorbell n destination port register (SRIO_ODDPR).....	1451
19.5.121	Outbound doorbell n destination attributes register (SRIO_ODDATR).....	1452
19.5.122	Outbound doorbell n retry error threshold configuration register (SRIO_ODRETCR).....	1453
19.5.123	Inbound doorbell n mode register (SRIO_IDMR).....	1454
19.5.124	Inbound doorbell n status register (SRIO_IDS).....	1457
19.5.125	Extended inbound doorbell n queue dequeue pointer address register (SRIO_EIDQDPAR).....	1458
19.5.126	Inbound doorbell n queue dequeue Pointer address register (SRIO_IDQDPAR).....	1459
19.5.127	Extended inbound doorbell n queue enqueue pointer address register (SRIO_EIDQEPR).....	1460
19.5.128	Inbound doorbell n Queue enqueue pointer address register (SRIO_IDQEPR).....	1460
19.5.129	Inbound doorbell n maximum interrupt report interval register (SRIO_IDMIRIR).....	1461
19.5.130	Inbound port-write n mode register (SRIO_IPWMR).....	1462
19.5.131	Inbound port-write n status register (SRIO_IPWSR).....	1463
19.5.132	Extended inbound port-write n queue base address register (SRIO_EIPWQBAR).....	1464
19.5.133	Inbound port-write n queue base address register (SRIO_IPWQBAR).....	1464
19.6	Serial RapidIO functional description.....	1465
19.6.1	Segmented Outbound Window Description.....	1465
19.6.2	RapidIO transaction summary.....	1467
19.6.3	RapidIO packet format summary.....	1469
19.6.4	RapidIO control symbol summary.....	1470
19.6.5	Accessing configuration registers using rapidIO packets.....	1472
19.6.5.1	Guidelines.....	1473
19.6.5.2	Outbound maintenance accesses.....	1473
19.6.6	RapidIO outbound ATMU.....	1473
19.6.6.1	Valid hits to multiple ATMU windows.....	1473
19.6.6.2	Outbound window boundary crossing errors.....	1474
19.6.7	RapidIO inbound ATMU.....	1476
19.6.7.1	Hits to multiple ATMU windows.....	1476

Section number	Title	Page
19.6.7.2	Inbound window boundary crossing errors.....	1476
19.6.8	Generating link-request/reset-device.....	1477
19.6.9	Outbound drain mode.....	1477
19.6.10	Input port disable mode.....	1478
19.6.11	Software assisted error recovery register support.....	1478
19.6.12	Hot-swap support.....	1479
19.6.12.1	Method 1.....	1480
19.6.12.1.1	Extraction-method 1.....	1480
19.6.12.1.2	Insertion-method 1.....	1481
19.6.12.2	Method 2 with RapidIO port hot-swapped.....	1481
19.6.12.2.1	Extraction-method 2.....	1481
19.6.12.2.2	Insertion-method 2.....	1482
19.6.13	Software re-training.....	1482
19.6.14	Errors and error handling.....	1483
19.6.14.1	RapidIO error description.....	1483
19.6.14.2	Physical layer RapidIO errors detected.....	1484
19.6.14.3	Logical layer errors and error handling.....	1488
19.6.14.3.1	Logical layer RapidIO errors detected.....	1488
19.7	RapidIO message unit	1520
19.7.1	Message unit overview.....	1520
19.7.2	Message unit features.....	1521
19.7.3	Outbound modes of operation.....	1522
19.7.4	Outbound message controller operation.....	1522
19.7.4.1	Direct mode operation.....	1522
19.7.4.1.1	Direct mode interrupts.....	1524
19.7.4.1.2	Direct mode message error response errors.....	1524
19.7.4.1.3	Direct mode packet response time-out errors.....	1524
19.7.4.1.4	Direct mode retry error threshold exceeded errors.....	1525
19.7.4.1.5	Direct mode transaction errors.....	1525

Section number	Title	Page
19.7.4.1.6	Direct mode error handling.....	1525
19.7.4.1.7	Disabling and enabling the message controller.....	1526
19.7.4.1.8	Direct mode hardware error handling.....	1526
19.7.4.1.9	Direct mode programming errors.....	1529
19.7.4.2	Chaining mode.....	1530
19.7.4.2.1	Message controller initialization.....	1531
19.7.4.2.2	Chaining mode operation.....	1532
19.7.4.2.3	Changing descriptor queues in chaining mode.....	1533
19.7.4.2.4	Preventing queue overflow in chaining mode.....	1534
19.7.4.2.5	Switching between direct and chaining modes.....	1534
19.7.4.2.6	Chaining mode descriptor format.....	1534
19.7.4.2.7	Chaining mode controller interrupts.....	1535
19.7.4.2.8	Chaining mode message error response errors.....	1536
19.7.4.2.9	Chaining mode packet response time-out errors.....	1536
19.7.4.2.10	Chaining mode retry error threshold exceeded errors.....	1537
19.7.4.2.11	Chaining mode transaction errors.....	1537
19.7.4.2.12	Chaining mode error handling.....	1537
19.7.4.2.13	Chaining mode hardware error handling.....	1538
19.7.4.2.14	Chaining mode programming errors.....	1538
19.7.4.3	Message controller arbitration.....	1539
19.7.5	Inbound message controller operation.....	1540
19.7.5.1	Inbound message controller initialization.....	1541
19.7.5.2	Inbound controller operation.....	1542
19.7.5.3	Message steering.....	1543
19.7.5.4	Inbound message controller retry response conditions.....	1543
19.7.5.5	Inbound message controller interrupts.....	1544
19.7.5.5.1	Message request time-out errors.....	1544
19.7.5.5.2	Inbound message controller transaction errors.....	1545
19.7.5.5.3	Inbound message controller error handling.....	1545

Section number	Title	Page
	19.7.5.5.4 Inbound message controller hardware error handling.....	1545
	19.7.5.5.5 Programming errors.....	1550
	19.7.5.5.6 Disabling and enabling the inbound message controller.....	1551
19.7.6	Message unit message passing logical specification registers.....	1551
19.8	RapidIO doorbell and port-write unit.....	1552
19.8.1	Doorbell and port-write unit features.....	1552
19.8.2	Doorbell controller.....	1552
19.8.2.1	Outbound doorbell controller.....	1553
19.8.2.1.1	Outbound doorbell controller interrupts.....	1554
19.8.2.1.2	Error response errors.....	1555
19.8.2.1.3	Outbound doorbell controller packet response time-out errors.....	1555
19.8.2.1.4	Outbound doorbell controller retry error threshold exceeded errors.....	1555
19.8.2.1.5	Error handling.....	1555
19.8.2.1.6	Disabling and enabling the doorbell controller-outbound.....	1556
19.8.2.1.7	Outbound doorbell controller hardware error handling.....	1556
19.8.2.1.8	Outbound doorbell controller programming errors.....	1559
19.8.2.2	Inbound doorbell controller.....	1559
19.8.2.2.1	Inbound doorbell controller initialization.....	1559
19.8.2.2.2	Inbound doorbell controller operation.....	1560
19.8.2.2.3	Inbound doorbell queue entry format.....	1560
19.8.2.2.4	Inbound doorbell controller retry response conditions.....	1561
19.8.2.2.5	Doorbell controller interrupts.....	1562
19.8.2.2.6	Doorbell controller transaction errors.....	1562
19.8.2.2.7	Doorbell controller error handling.....	1563
19.8.2.2.8	Inbound doorbell controller hardware error handling.....	1563
19.8.2.2.9	Inbound doorbell controller programming errors.....	1565
19.8.2.2.10	Disabling and enabling the doorbell controller-inbound.....	1566
19.8.2.3	Doorbell message passing logical specification registers.....	1566

Section number	Title	Page
19.8.3	Port-write controller.....	1567
19.8.3.1	Port-write controller initialization.....	1568
19.8.3.2	Port-write controller operation.....	1568
19.8.3.3	Port-write controller interrupt.....	1569
19.8.3.4	Discarding port-writes.....	1569
19.8.3.5	Transaction errors.....	1570
19.8.3.5.1	Port-write controller error handling.....	1570
19.8.3.6	Port-write controller hardware error handling.....	1570
19.8.3.6.1	Port-write controller programming errors.....	1573
19.8.3.7	Disabling and enabling the port-write controller.....	1574
19.8.3.8	Port-write controller message passing logical specification registers.....	1574

Chapter 20 DMA Controller

20.1	DMA overview.....	1577
20.1.1	DMA features summary.....	1578
20.1.2	DMA modes of operation.....	1578
20.2	DMA external signal description.....	1581
20.2.1	Signal overview.....	1581
20.2.2	DMA signal descriptions.....	1582
20.3	DMA controller memory map.....	1583
20.3.1	DMA mode register (DMA _x _MR _n).....	1589
20.3.2	DMA status register (DMA _x _SR _n).....	1593
20.3.3	DMA current link descriptor extended address register (DMA _x _ECLNDAR _n).....	1594
20.3.4	DMA current link descriptor address register (DMA _x _CLNDAR _n).....	1595
20.3.5	DMA source attributes register (DMA _x _SATR _n).....	1597
20.3.6	DMA source address register (DMA _x _SAR _n).....	1598
20.3.7	DMA destination attributes register (DMA _x _DATR _n).....	1599
20.3.8	DMA destination address register (DMA _x _DAR _n).....	1600
20.3.9	DMA byte count register (DMA _x _BCR _n).....	1601

Section number	Title	Page
20.3.10	DMA extended next link descriptor address register (DMA x _ENLNDAR n).....	1601
20.3.11	DMA next link descriptor address register (DMA x _NLNDAR n).....	1602
20.3.12	DMA extended current list descriptor address register (DMA x _ECLSDAR n).....	1603
20.3.13	DMA current list descriptor address register (DMA x _CLSDAR n).....	1604
20.3.14	DMA extended next list descriptor address register (DMA x _ENLSDAR n).....	1605
20.3.15	DMA next list descriptor address register (DMA x _NLSDAR n).....	1605
20.3.16	DMA source stride register (DMA x _SSR n).....	1606
20.3.17	DMA destination stride register (DMA x _DSR n).....	1607
20.3.18	DMA general status register (DMA x _DGSR).....	1608
20.4	DMA functional description.....	1611
20.4.1	DMA channel operation.....	1611
20.4.1.1	Source/destination transaction size calculations.....	1612
20.4.1.2	Basic DMA mode transfer.....	1614
20.4.1.2.1	Basic direct mode.....	1614
20.4.1.2.2	Basic, direct, single-write start mode.....	1614
20.4.1.2.3	Basic chaining mode.....	1615
20.4.1.2.4	Basic chaining, single-write start mode.....	1616
20.4.1.3	Extended DMA mode transfer.....	1617
20.4.1.3.1	Extended direct mode	1617
20.4.1.3.2	Extended direct, single-write start mode.....	1617
20.4.1.3.3	Extended chaining mode.....	1617
20.4.1.3.4	Extended chaining, single-write start mode.....	1618
20.4.1.4	External control mode transfer.....	1619
20.4.1.5	Channel continue mode for cascading transfer chains.....	1620
20.4.1.5.1	Basic mode.....	1621
20.4.1.5.2	Extended mode.....	1621
20.4.1.6	Channel abort.....	1621
20.4.1.7	Bandwidth control.....	1621
20.4.1.8	Channel state.....	1622

Section number	Title	Page
20.4.1.9	Illustration of stride size and stride distance.....	1622
20.4.2	DMA transfer interfaces.....	1623
20.4.3	DMA errors.....	1623
20.4.4	DMA descriptors.....	1623
20.4.5	DMA controller limitations and restrictions.....	1626
20.5	DMA system considerations.....	1627
20.5.1	Unusual DMA scenarios.....	1627
20.5.1.1	DMA to core.....	1627
20.5.1.2	DMA to configuration, control, and status registers (CCSR).....	1628
20.5.1.3	DMA to I2C.....	1628
20.5.1.4	DMA to DUART.....	1628
20.5.1.5	DMA in multi-partition systems.....	1628

Chapter 21

Data Path Acceleration Architecture (DPAA) Overview and P4080 DPAA Implementation

21.1	DPAA Introduction and Terms.....	1631
21.2	Data Formats Used in the DPAA.....	1633
21.2.1	Frame Descriptor (FD).....	1633
21.2.1.1	FD Format.....	1634
21.2.1.2	FD Considerations.....	1635
21.2.2	Multi-Buffer Frames.....	1636
21.2.2.1	Scatter/Gather Entry Format.....	1636
21.2.2.2	Multi-Buffer Frame Considerations.....	1637
21.2.2.3	Situations Where Multi-Frame Buffering Stops	1638
21.2.3	Single-Buffer Frames.....	1638
21.2.4	Compound Frames.....	1639
21.2.4.1	When to Use Compound Frames.....	1639
21.2.4.2	Compound Frame Considerations.....	1640
21.2.5	Simple Frames.....	1641
21.2.6	Frame Format Codes.....	1641

Section number	Title	Page
21.2.7	Frame Formats Supported by Accelerators.....	1642
21.2.8	Special Values and Exceptions	1642
21.2.9	Releasing Buffers to the BMan.....	1643
21.3	Accessing Memory Using Logical IO Device Numbers (LIODNs).....	1643
21.3.1	Role of the PAMU in the DPAA.....	1643
21.3.2	Using Multiple LIODNs.....	1644
21.3.2.1	Benefit of Using Multiple LIODNs.....	1644
21.3.2.2	LIODN Requirements.....	1644
21.4	Packet Walk-Through Example.....	1645
21.5	P4080-Specific DPAA Implementation Details.....	1647
21.5.1	P4080 Queue Manager (QMan) Implementation.....	1647
21.5.2	P4080 Buffer Manager (BMan) Implementation.....	1647
21.5.3	P4080 Frame Manager (FMan) Implementation.....	1647
21.5.4	P4080 Datapath Three Speed Ethernet Controller (dTSEC) Implementation.....	1648
21.5.5	P4080 Security and Encryption Engine (SEC) Implementation.....	1649
21.5.6	P4080 Pattern Matching Engine (PME) Implementation.....	1649

Chapter 22 Multicore Programmable Interrupt Controller (MPIC)

22.1	MPIC overview.....	1651
22.1.1	MPIC features summary.....	1651
22.1.2	MPIC block diagram.....	1653
22.1.3	The MPIC in Multiple- and Single-Core Implementations.....	1654
22.1.4	MPIC modes of operation.....	1655
22.1.4.1	External Proxy Facility Mode (GCR[M] = 11).....	1655
22.1.4.2	Mixed Mode (GCR[M] = 01).....	1655
22.1.4.3	Pass-Through Mode (GCR[M] = 00).....	1655
22.1.5	Interrupts to the Processor Core.....	1656
22.1.6	Interrupt Sources.....	1657
22.1.6.1	Interrupt Destinations.....	1658

Section number	Title	Page
	22.1.6.2 Interrupt Routing-Mixed Mode.....	1659
22.2	MPIC external signal descriptions.....	1660
	22.2.1 MPIC signal descriptions.....	1660
22.3	MPIC Memory Map.....	1661
	22.3.1 Block revision register 1 (MPIC_BRR1).....	1689
	22.3.2 Block revision register 2 (MPIC_BRR2).....	1689
	22.3.3 Interprocessor interrupt dispatch register n (MPIC_IPIDR n).....	1690
	22.3.4 Current task priority register (MPIC_CTPR).....	1691
	22.3.5 Who am I register (MPIC_WHOAMI).....	1692
	22.3.6 Interrupt acknowledge register (MPIC_IACK).....	1693
	22.3.7 End of interrupt register (MPIC_EOI).....	1694
	22.3.8 Feature reporting register (MPIC_FRR).....	1695
	22.3.9 Global configuration register (MPIC_GCR).....	1696
	22.3.10 Vendor identification register (MPIC_VIR).....	1696
	22.3.11 Processor initialization register (MPIC_PIR).....	1697
	22.3.12 Processor NMI register (MPIC_PNMIR).....	1698
	22.3.13 IPI vector/priority register n (MPIC_IPIVPR n).....	1699
	22.3.14 Spurious vector register (MPIC_SVR).....	1700
	22.3.15 Timer frequency reporting register [Group A] (MPIC_TFRRA).....	1701
	22.3.16 Global timer current count register [Group A] n (MPIC_GTCCRAn).....	1701
	22.3.17 Global timer base count register [Group A] n (MPIC_GTBCCRAn).....	1702
	22.3.18 Global timer vector/priority register [Group A] n (MPIC_GTVPRAn).....	1703
	22.3.19 Global timer destination register [Group A] n (MPIC_GTDRA n).....	1704
	22.3.20 Timer control register [Group A] (MPIC_TCRA).....	1705
	22.3.21 Message register [Group A] n (MPIC_MSGRA n).....	1707
	22.3.22 Message enable register [Group A] (MPIC_MERA).....	1707
	22.3.23 Message status register [Group A] (MPIC_MSRA).....	1708
	22.3.24 Shared message signaled interrupt register [Bank A] n (MPIC_MSIRAn).....	1709
	22.3.25 Shared message signaled interrupt status register [Bank A] (MPIC_MSISRA).....	1711

Section number	Title	Page
22.3.26	Shared message signaled interrupt index register [Bank A] (MPIC_MSIIRA).....	1711
22.3.27	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB n).....	1712
22.3.28	Shared message signaled interrupt status register [Bank B] (MPIC_MSISRB).....	1714
22.3.29	Shared message signaled interrupt index register [Bank B] (MPIC_MSIIRB).....	1715
22.3.30	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC n).....	1716
22.3.31	Shared message signaled interrupt status register [Bank C] (MPIC_MSISRC).....	1718
22.3.32	Shared message signaled interrupt index register [Bank C] (MPIC_MSIIRC).....	1718
22.3.33	Timer frequency reporting register [Group B] (MPIC_TFRRB).....	1719
22.3.34	Global timer current count register [Group B] n (MPIC_GTCCR Bn).....	1720
22.3.35	Global timer base count register [Group B] n (MPIC_GTB $CRBn$).....	1721
22.3.36	Global timer vector/priority register [Group B] n (MPIC_GTVPRB n).....	1722
22.3.37	Global timer destination register [Group B] n (MPIC_GTD RBn).....	1723
22.3.38	Timer control register [Group B] (MPIC_TCRB).....	1724
22.3.39	Message register [Group B] n (MPIC_MSGRB n).....	1726
22.3.40	Message enable register [Group B] (MPIC_MERB).....	1726
22.3.41	Message status register [Group B] (MPIC_MSRB).....	1727
22.3.42	Performance monitor n mask register 0 (MPIC_PM n MR0).....	1727
22.3.43	Performance monitor n mask register 1 (MPIC_PM n MR1).....	1728
22.3.44	Performance monitor n mask register 2 (MPIC_PM n MR2).....	1729
22.3.45	Performance monitor n mask register 3 (MPIC_PM n MR3).....	1729
22.3.46	Performance monitor n mask register 4 (MPIC_PM n MR4).....	1729
22.3.47	Performance monitor n mask register 5 (MPIC_PM n MR5).....	1730
22.3.48	External interrupt summary register (MPIC_ERQSR).....	1730
22.3.49	Error interrupt summary register 0 (MPIC_EISR0).....	1731
22.3.50	Error interrupt mask register 0 (MPIC_EIMR0).....	1731
22.3.51	Watchdog status register summary register 0 (MPIC_WSR $SR0$).....	1732
22.3.52	Critical interrupt summary register 0 (MPIC_CISR0).....	1733
22.3.53	Critical interrupt summary register 1 (MPIC_CISR1).....	1733
22.3.54	Critical interrupt summary register 2 (MPIC_CISR2).....	1734

Section number	Title	Page
22.3.55	Critical interrupt summary register 3 (MPIC_CISR3).....	1734
22.3.56	Critical interrupt summary register 4 (MPIC_CISR4).....	1734
22.3.57	Machine check summary register 0 (MPIC_MCSR0).....	1735
22.3.58	Machine check summary register 1 (MPIC_MCSR1).....	1735
22.3.59	Machine check summary register 2 (MPIC_MCSR2).....	1736
22.3.60	Machine check summary register 3 (MPIC_MCSR3).....	1736
22.3.61	Machine check summary register 4 (MPIC_MCSR4).....	1737
22.3.62	IRQ_OUT/Soc Interrupt Event summary register 0 (MPIC IRQSIESR0).....	1737
22.3.63	IRQ_OUT/Soc Interrupt Event summary register 1 (MPIC IRQSIESR1).....	1738
22.3.64	IRQ_OUT/Soc Interrupt Event summary register 2 (MPIC IRQSIESR2).....	1738
22.3.65	IRQ_OUT/Soc Interrupt Event summary register 3 (MPIC IRQSIESR3).....	1739
22.3.66	IRQ_OUT/Soc Interrupt Event summary register 4 (MPIC IRQSIESR4).....	1739
22.3.67	Shared message signaled interrupt index register [Bank A] (alias) (MPIC_MSIIRA_alias).....	1740
22.3.68	Shared message signaled interrupt index register [Bank B] (alias) (MPIC_MSIIRB_alias).....	1740
22.3.69	Shared message signaled interrupt index register [Bank C] (alias) (MPIC_MSIIRC_alias).....	1741
22.3.70	External interrupt n (IRQn) vector/priority register (MPIC_EIVPRn).....	0
22.3.71	External interrupt n (IRQn) destination register (MPIC_EIDRn).....	0
22.3.72	External interrupt n (IRQn) level register (MPIC_EILRn).....	0
22.3.73	Internal interrupt n vector/priority register (MPIC_IIVPRn).....	1742
22.3.74	Internal interrupt n destination register (MPIC_IIDRn).....	1743
22.3.75	Internal interrupt n level register (MPIC_IILRn).....	1744
22.3.76	Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA n).....	1745
22.3.77	Messaging interrupt destination register [Group A] n (MPIC_MIDRA n).....	1746
22.3.78	Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRB n).....	1747
22.3.79	Messaging interrupt destination register [Group B] n (MPIC_MIDRB n).....	1748
22.3.80	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA n).....	1750
22.3.81	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA n).....	1751
22.3.82	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB n).....	1752
22.3.83	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB n).....	1753

Section number	Title	Page
22.3.84	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC n).....	1754
22.3.85	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC n).....	1755
22.3.86	Interprocessor interrupt dispatch register 0 (MPIC_CPU n _IPIDR0).....	1757
22.3.87	Interprocessor interrupt dispatch register 1 (MPIC_CPU n _IPIDR1).....	1758
22.3.88	Interprocessor interrupt dispatch register 2 (MPIC_CPU n _IPIDR2).....	1760
22.3.89	Interprocessor interrupt dispatch register 3 (MPIC_CPU n _IPIDR3).....	1761
22.3.90	Current task priority register (MPIC_CPU n _CTPR).....	1762
22.3.91	Who am I register (MPIC_CPU n _WHOAMI).....	1763
22.3.92	Interrupt acknowledge register (MPIC_CPU n _IACK).....	1764
22.3.93	End of interrupt register (MPIC_CPU n _EOI).....	1765
22.4	MPIC functional description.....	1766
22.4.1	Flow of interrupt control.....	1766
22.4.1.1	Interrupts Routed to cint, mcp, sie, or IRQ_OUT_B.....	1767
22.4.1.2	Interrupts routed to int.....	1768
22.4.1.2.1	Nesting of Interrupts	1770
22.4.1.2.2	Interrupt Source Priority.....	1771
22.4.1.2.3	Interrupt Acknowledge	1771
22.4.1.2.4	Spurious Vector Generation.....	1772
22.4.2	Global- and private-access per-CPU registers.....	1773
22.4.3	Interprocessor Interrupts.....	1774
22.4.4	Messaging Interrupts.....	1774
22.4.4.1	Message Interrupts.....	1774
22.4.4.2	Shared Message Signaled Interrupts.....	1775
22.4.5	PCI Express INTx.....	1775
22.4.6	Global Timers.....	1775
22.4.7	Resets.....	1776
22.4.7.1	Resetting the MPIC.....	1776
22.4.7.2	Processor Core Initialization.....	1777
22.4.8	Processor Core NMI.....	1777

Section number	Title	Page
22.5	Initialization/Application Information.....	1777
22.5.1	Programming Guidelines.....	1777
22.5.1.1	MPIC Registers.....	1777
22.5.1.2	Changing Interrupt Source Configuration.....	1779
22.5.1.3	Requirement.....	1779

Chapter 23 Interrupt Assignments

23.1	Introduction.....	1781
23.2	Internal Interrupt Sources.....	1781
23.3	PCI Express INTx/External Interrupt IRQn Sharing.....	1785
23.4	SoC Interrupt Event Routing.....	1786

Chapter 24 General Purpose I/O (GPIO)

24.1	GPIO overview.....	1787
24.1.1	The GPIO module as implemented on the chip.....	1787
24.2	GPIO features summary.....	1788
24.3	GPIO signal descriptions.....	1788
24.4	GPIO Memory Map/Register Definition.....	1789
24.4.1	GPIO direction register (GPIO_GPDIR).....	1789
24.4.2	GPIO open drain register (GPIO_GPODR).....	1790
24.4.3	GPIO data register (GPIO_GPDAT).....	1790
24.4.4	GPIO interrupt event register (GPIO_GPIER).....	1791
24.4.5	GPIO interrupt mask register (GPIO_GPIMR).....	1791
24.4.6	GPIO interrupt control register (GPIO_GPICR).....	1792

Chapter 25 Device Configuration and Pin Control

25.1	Device Configuration and Pin Control Introduction.....	1793
25.2	Device Configuration and Pin Control Features Summary.....	1793
25.3	Device Configuration and Pin Control Memory Map/Register Definition.....	1794
25.3.1	POR status register 1 (DCFG_PORSR1).....	1798

Section number	Title	Page
25.3.2	POR status register 2 (DCFG_PORSR2).....	1799
25.3.3	General-purpose POR configuration register (DCFG_GPPORCR1).....	1800
25.3.4	Device disable register 1 (DCFG_DEVDISR1).....	1800
25.3.5	Device disable register 2 (DCFG_DEVDISR2).....	1804
25.3.6	Core disable register (DCFG_COREDISR).....	1806
25.3.7	Processor version register (DCFG_PVR).....	1808
25.3.8	System version register (DCFG_SVR).....	1809
25.3.9	Reset control register (DCFG_RSTCR).....	1810
25.3.10	Reset request preboot loader status register (DCFG_RSTRQPBLSR).....	1811
25.3.11	Reset request mask register (DCFG_RSTRQMR1).....	1812
25.3.12	Reset request status register (DCFG_RSTRQSR1).....	1813
25.3.13	Reset request WDT mask register (DCFG_RSTRQWDTMR).....	1815
25.3.14	Reset request WDT status register (DCFG_RSTRQWDTSR).....	1817
25.3.15	Boot release register (DCFG_BRR).....	1818
25.3.16	Reset control word status registers 1-16 (DCFG_RCWSR n).....	1820
25.3.17	Scratch read/write registers 1-4 (DCFG_SCRATCHRW n).....	1820
25.3.18	Scratch write-once registers 1-4 (DCFG_SCRATCHW1R n).....	1821
25.3.19	Core reset status register n (DCFG_CRSTS n).....	1821
25.3.20	PCI Express n LIODN register (DCFG_PEX n LIODNR).....	1823
25.3.21	RIO n LIODN register (DCFG_RIO n LIODNR).....	1824
25.3.22	USB n LIODN register (DCFG_USB n LIODNR).....	1824
25.3.23	eSDHC LIODN register (DCFG_eSDHCLIODNR).....	1825
25.3.24	RIO message unit LIODN register (DCFG_RMULIODNR).....	1825
25.3.25	DMA n LIODN register (DCFG_DMA n LIODNR).....	1826
25.3.26	PAMU bypass enable register (DCFG_PAMUBYPENR).....	1826
25.3.27	DMA control register (DCFG_DMACR1).....	1827
25.3.28	Pin Multiplexing Control Register (DCFG_PMUXCR).....	1828
25.3.29	Debug Source ID 1A (DCFG_DSRCID1A).....	1830
25.3.30	Debug Source ID 1B (DCFG_DSRCID1B).....	1831

Section number	Title	Page
25.3.31	Debug Source ID 1C (DCFG_DSRCID1C).....	1832
25.3.32	Debug Source ID 1D (DCFG_DSRCID1D).....	1833
25.3.33	Debug Source ID 2A (DCFG_DSRCID2A).....	1834
25.3.34	Debug Source ID 2B (DCFG_DSRCID2B).....	1835
25.3.35	I/O Voltage Selection Status Register (DCFG_IOVSELSR).....	1836
25.3.36	DDR Clock Disable Register (DCFG_DDRCLKDR).....	1836
25.3.37	eLBC Clock Disable Register (DCFG_ELBCCLKDR).....	1838
25.3.38	eSDHC Polarity Configuration Register (DCFG_SDHCPCCR).....	1839

Chapter 26 **Run Control/Power Management (RCPM)**

26.1	RCPM overview.....	1841
26.1.1	Power Management Features.....	1841
26.1.2	Modes of Operation.....	1842
26.1.2.1	Mode Summary-Core and Device Levels.....	1843
26.1.2.2	Modes Entry and Exit.....	1843
26.1.2.3	Reset Modes.....	1845
26.1.2.3.1	Power-On Reset State.....	1846
26.1.2.3.2	Hard Reset State.....	1846
26.1.2.4	Power Management Modes.....	1846
26.1.2.4.1	Core Dozing Mode.....	1846
26.1.2.4.2	Core Napping Mode.....	1846
26.1.2.4.3	Device Sleeping Mode.....	1847
26.2	RCPM Signal Descriptions.....	1847
26.3	RCPM Memory Map/Register Definition.....	1848
26.3.1	Core doze status register (RCPM_CDOZSR).....	1849
26.3.2	Core doze control register (RCPM_CDOZCR).....	1850
26.3.3	Core nap status register (RCPM_CNAPSR).....	1850
26.3.4	Core nap control register (RCPM_CNAPCR).....	1851
26.3.5	Core doze previous status register (RCPM_CDOZPSR).....	1852

Section number	Title	Page
26.3.6	Core nap previous status register (RCPM_CNAPPSR).....	1852
26.3.7	Core wait status register (RCPM_CWAITSR).....	1853
26.3.8	Core watchdog detect status register (RCPM_CWDTDSR).....	1854
26.3.9	Power management control and status register (RCPM_POWMGTCSR).....	1855
26.3.10	IP powerdown exception control register (RCPM_IPPDEXPCR).....	1856
26.3.11	Core Power Management Interrupt Mask Register (RCPM_CPMIMR).....	1859
26.3.12	Core Power Management Critical Interrupt Mask Register (RCPM_CPMCIMR).....	1859
26.3.13	Core Power Management Machine Check Mask Register (RCPM_CPMMMCMR).....	1860
26.3.14	Core Power Management NMI Mask Register (RCPM_CPMNMIMR).....	1861
26.3.15	Core timebase enable register (RCPM_CTBENR).....	1861
26.3.16	Core timebase clock select register (RCPM_CTBACKSELR).....	1862
26.3.17	Core timebase halt control register (RCPM_CTBHLTCR).....	1862
26.3.18	Core machine check mask control register (RCPM_CMCPMASKCR).....	1863
26.4	RCPM Functional Description	1863
26.4.1	Core Power Management.....	1863
26.4.1.1	Core Doze Operation.....	1864
26.4.1.1.1	Core Doze Operation Entry.....	1864
26.4.1.1.2	Core Doze Operation Exit.....	1864
26.4.1.1.3	Core Doze Operation Sequence.....	1864
26.4.1.2	Core Nap Operation.....	1865
26.4.1.2.1	Core Nap Operation Entry.....	1865
26.4.1.2.2	Core Nap Operation Exit.....	1865
26.4.1.2.3	Core Nap Operation Sequence.....	1865
26.4.2	Device Power Management.....	1866
26.4.2.1	Device Sleep Operation.....	1866
26.4.2.1.1	Device Sleep Operation Entry.....	1866
26.4.2.1.2	Device Sleep Operation Exit.....	1867
26.4.2.1.3	Device Sleep Operation Sequence.....	1867

Section number	Title	Page
26.4.3	Core Timebase and Decrementer Operation.....	1868
26.4.3.1	Timebase Operation in Normal and Debug Device Modes.....	1868
26.5	Initialization Information.....	1868
26.6	Application Information.....	1869
26.6.1	Magic Packet Wakeup.....	1869
26.6.2	Proper Use of Core Timebase.....	1869

Chapter 1 Overview

1.1 Introduction

The P4080 QorIQ communications processor combines eight Power Architecture® processor cores with high-performance Data Path Acceleration Architecture (DPAA), CoreNet fabric infrastructure, and network and peripheral bus interfaces required for networking, telecom/datacom, wireless infrastructure, and mil/aerospace applications.

The P4080 can be used for combined control, datapath, and application layer processing in routers, switches, base station controllers, and general-purpose embedded computing systems. Its high level of integration offers significant performance benefits compared to multiple discrete devices, while also greatly simplifying board design.

This manual also describes the P4040 QorIQ processor. The P4040 combines four Power Architecture e500mc cores with the same high-performance datapath acceleration, networking, and peripheral bus interfaces. Note that throughout this manual, the P4080 processor refers to eight cores (cores 0-7), but the P4040 processor is restricted to four cores (cores 0-3).

This section provides an overview of the P4080's features, with expanded explanations of multicore and datapath areas of innovation and power management.

1.2 P4080 Features Summary

The P4080 SoC includes the following functions and features:

- Eight e500mc cores built on Power Architecture technology, each with a private 128-Kbyte L2 cache
 - Three levels of instructions: user, supervisor, and hypervisor
 - Independent boot and reset
 - Secure boot capability
- 2-Mbyte shared L3 CoreNet platform cache (CPC)

- Hierarchical interconnect fabric
 - CoreNet fabric supporting coherent and non-coherent transactions with prioritization and bandwidth allocation amongst CoreNet end-points
 - 0.8 Tbps coherent read bandwidth
 - Queue manager fabric supporting packet-level queue management and quality of service scheduling
- Two 64-bit DDR2/DDR3 SDRAM memory controllers with ECC and interleaving support
- Data Path Acceleration Architecture incorporating acceleration for the following functions:
 - Packet parsing, classification, and distribution (FMan)
 - Queue management for scheduling, packet sequencing, and congestion management (QMan)
 - Hardware buffer management for buffer allocation and de-allocation (BMan)
 - Encryption/decryption (SEC 4.0)
 - RegEx pattern matching (PME 2.0)
- Ethernet Interfaces
 - Two 10 Gbps Ethernet (XAUI) controllers
 - Eight 1 Gbps Ethernet controllers
- High Speed Peripheral Interfaces
 - Three PCI Express 2.0 controllers/ports running at up to 5 GHz
 - Two serial RapidIO® 1.2 controllers/ports running at up to 3.125 GHz
- Additional peripheral interfaces
 - Two USB controllers with ULPI interface to external PHY
 - SD/MMC
 - SPI controller
 - Four I²C controllers
 - Two dual UARTs (DUART)
 - Enhanced local bus controller (eLBC)
- Multicore programmable interrupt controller (MPIC)
- Two 4-channel DMA engines

1.3 P4080 Block Diagram

Figure 1-1 shows the major functional units within the P4080.

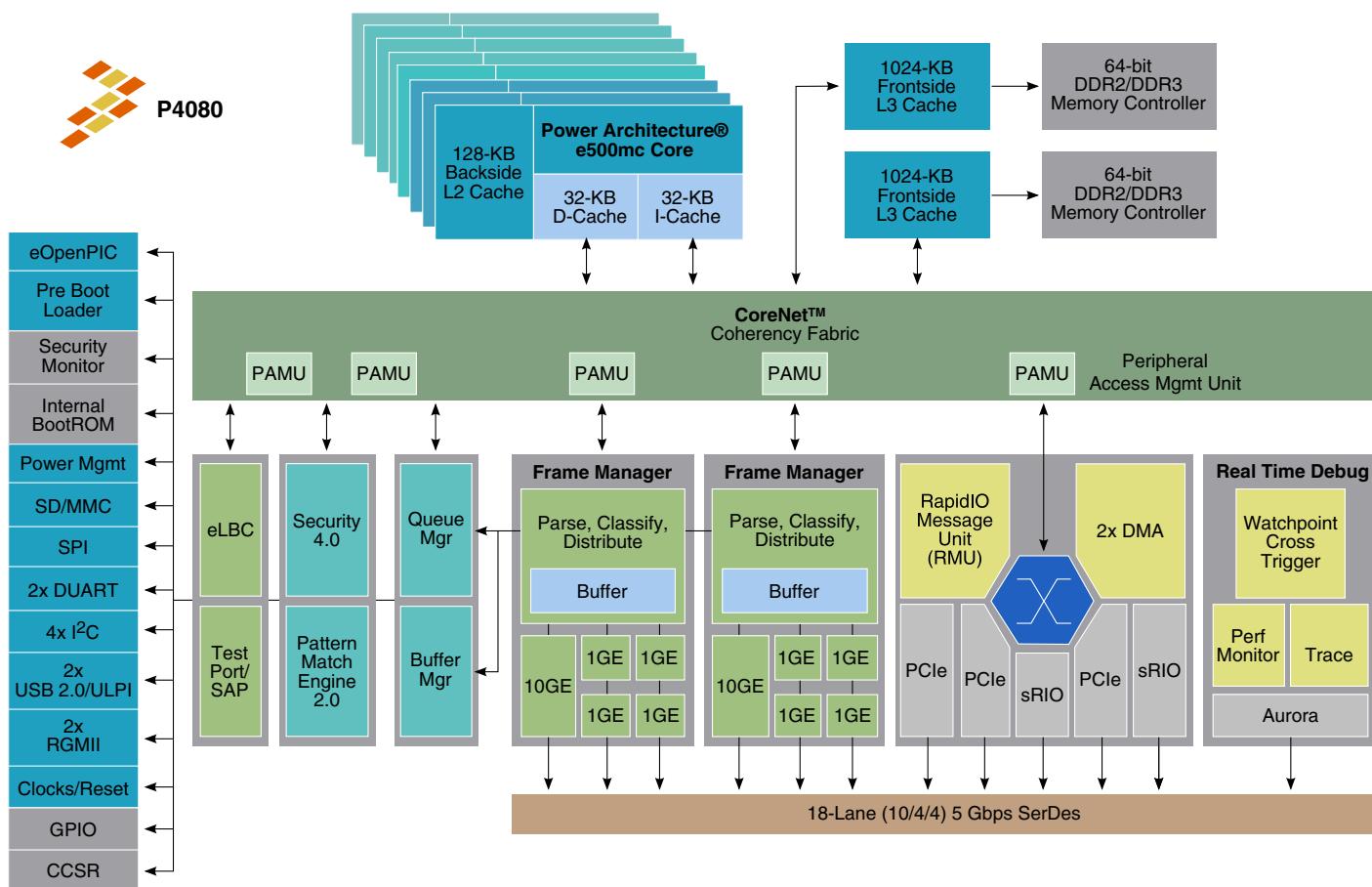


Figure 1-1. P4080 Preliminary Block Diagram

1.4 P4080 Application Examples

The P4080 is a flexible device that can be configured to meet many system application needs. The P4080's e500mc cores can be combined as a fully-symmetric, multi-processing, system-on-a-chip, or they can be operated with varying degrees of independence to perform asymmetric multi-processing. Full processor independence, including the ability to independently boot and reset each e500mc core, is a defining characteristic of the P4080. The ability of the cores to run different operating systems, or run OS-less, provides the user with significant flexibility in partitioning between control, datapath, and applications processing. This ability also simplifies consolidation of functions previously spread across multiple discrete processors onto a single device.

While the eight Power Architecture cores offer a major leap in available processor performance, in many throughput-intensive, packet-processing networking applications, raw processing power is not enough to achieve multi-Gbps data rates. To address this, the P4080 introduces Freescale's Data Path Acceleration Architecture (DPAA), which

significantly reduces data plane instructions per packet, enabling more CPU cycles to work on value-added services rather than repetitive low-level tasks. Combined with specialized accelerators for cryptography and pattern matching, the P4080 allows the user's software to perform complex packet processing at high data rates.

1.4.1 Multicore Processing Scenarios

There are several ways to map operating systems to the eight P4080 cores, as follows:>

- Eight-core asymmetric multi-processing (AMP)
 - Eight copies of the same uni-processor OS
 - Up to eight different uni-processor OSes
- Eight-core symmetric multi-processing (SMP)
- Mixed symmetric and asymmetric multi-processing
 - N cores running in SMP mode, while the remainder of the cores operate asymmetrically with up to 8-N different OSes.

It is also possible for one or more cores to run OS-less, using a simple scheduler. This is a likely scenario when cores are performing datapath operations with bounded real-time requirements. This use case is greatly enhanced by the provisioning of a 128-Kbyte private back-side L2 cache for each e500mc core. These caches can operate as a traditional unified cache, or be set to operate as instruction only, data only, or even locked and used as memory-mapped SRAM.

CPU cores operating asymmetrically can be run at asynchronous clock rates. Each processor can source its input clock from one of the multiple PLLs inside the P4080. This allows each core to operate at the minimum frequency required to perform its assigned function, saving power. The cores are also capable of running at half and quarter ratios of their input PLL frequency and can switch between PLLs and ratios nearly instantaneously. This allows lightly utilized CPUs to be slowed (under software control) for power savings, rather than performing more complex task migration operations.

[Figure 1-2](#) shows several CPU usage scenarios, along with potential interaction with the DPAA. Possible scenarios are as follows:

- In scenario A, all CPUs are running a single operating system, with any specialization of CPU function occurring through OS techniques such as Task Affinity. The I/Os and acceleration hardware are under the control of the SMP OS. Typically all CPUs operate at the same frequency.
- In scenario B, some number of the cores are operated as an SMP cluster, most likely running high complexity control plane operations. The control plane configures and manages the remaining processors, which are running individual copies of an RTOS or scheduler to perform dataplane operations. In this scenario, the SMP CPUs

typically operate at the same frequency, the remaining CPUs can run at a different frequency from the SMP CPUs, and even from each other.

- In scenario C, a single CPU is used as the control processor, configuring and managing the other seven processors, which are running individual copies of an RTOS or scheduler, as in B. As in B, CPU operating frequencies are an independent parameter.
- In scenario D, all CPUs are used for datapath operations, here shown as two sets of pipelined functions, each interacting independently with the I/Os and accelerators. Operating frequencies for each CPU in the pipeline can be set independently, and the provision of a 128-Kbyte back-side L2 provides significant flexibility in partitioning and rebalancing the pipeline as processing requirements change.

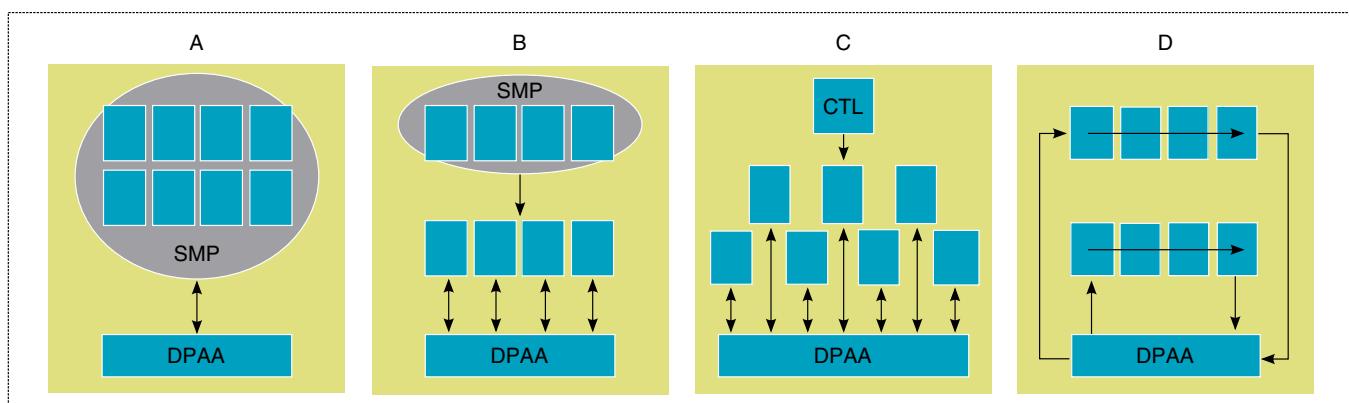


Figure 1-2. Multicore Processing Scenarios

A fifth scenario, shown in [Figure 1-3](#), involves the use of one of the CPUs as an I/O Processor. The DPAA (described in more detail later) can greatly simplify and accelerate processing for packets entering the system by means of the Ethernet interfaces. For systems requiring external ASICs or legacy network interface cards in the high performance datapath, system developers can allocate a CPU to help interwork between the native data buffers used by PCIe- or serial RapidIO-based network interfaces and the data buffers used by the DPAA hardware.

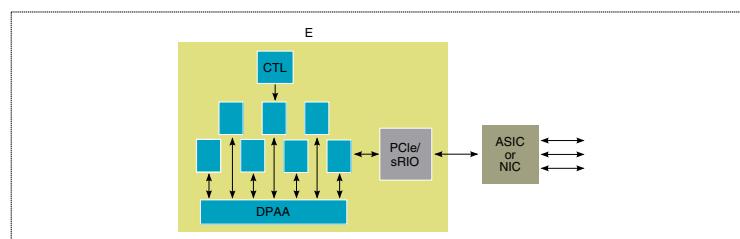


Figure 1-3. IO Processor Managing PCIe/Serial RapidIO-Based Network Interfaces

1.4.2 P4080 Applications

The P4080 is well-suited for applications that are highly compute-intensive, I/O intensive, or both. Some examples of each are shown in subsequent sections.

1.4.2.1 Virtual Private Network (VPN)/ IP Services Router

Figure 1-4 shows a virtual private network (VPN)/IP Services router enabled through PCIe and Ethernet. The QorIQ DPAA accelerates packet classification, filtering, and packet queuing, while the crypto accelerator (SEC 4.0) and pattern matching engine (PME 2.0) perform high throughput encryption/decryption and regex payload scanning security under control of stacks running on the CPUs. Session establishment, policy enforcement, and potentially application processing, are executed by the control processor(s). Security appliances would use the P4080 in a similar manner, however without direct connections to the WAN and DMZ server.

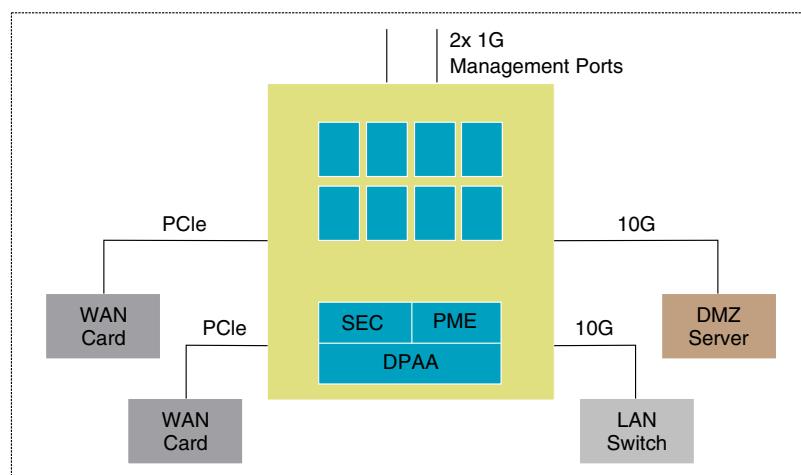


Figure 1-4. VPN/IP Services Router

1.4.2.2 Security Services Blade for Switch or Server

Figure 1-5 shows a P4080 operating as a packet processing engine in a security services blade. Due to the presence of an external control processor, all of the P4080's processors are dedicated to complex content-oriented packet processing, assisting the DPAA. Functions performed might include security protocol termination (MACSec, IPSec, SSL/TLS) and content inspection.

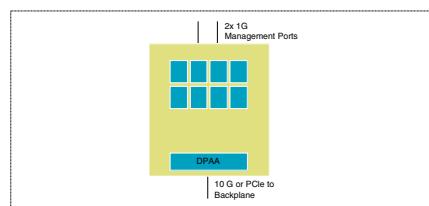


Figure 1-5. Security Services Blade for Switch, Router, or Server

1.4.2.3 Wireless Infrastructure/Radio Node Controller

Some of the more demanding packet-processing applications are found in the realm of wireless infrastructure. Systems have to interwork between wireless link layer protocols and IP networking protocols. Wireless link layer termination typically involves decryption, using algorithms such as Kasumi which are very specific to cellular wireless access networks. Connecting to the IP network offers wireless infrastructure tremendous cost savings, but introduces all the security threats found in the IP world. The P4080's network and peripheral interfaces provide it with the flexibility to connect to DSPs, and wireless link layer framing ASICs/FPGAs. Multiple processors may be dedicated to data path processing in each direction, while the data path acceleration architecture offers encryption acceleration for both wireless and IP networking protocols, plus packet filtering capability on the IP networking side. A representative wireless infrastructure application is shown in [Figure 1-6](#).

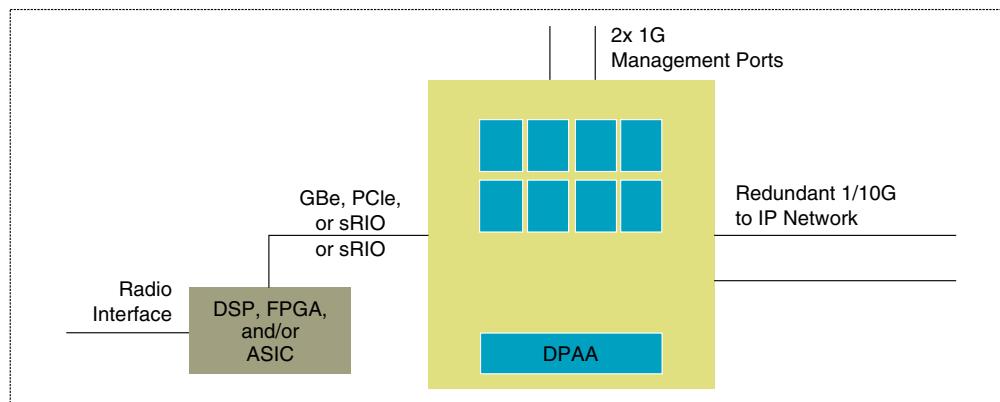


Figure 1-6. Wireless Infrastructure/Radio Node Controller

1.4.2.4 High-Performance Compute Blade

[Figure 1-7](#) shows likely scenarios for eight core symmetric multi-processing on the P4080. The eight processors may provide all the computing power needed for a stand-alone application, or multiple P4080s may be placed on a single board to create an extremely high-density computing platform suitable for military/aerospace, high-

performance test and measurement, and other compute-intensive applications. In a grid computing application, data shared among multiple computing platforms may use network transport protocols as the interconnect, and depending on the grid elements' physical proximity and security requirements, shared application data may be encrypted or at least cryptographically authenticated. In such a case, the DPAA assists by performing TCP or even SSL termination.

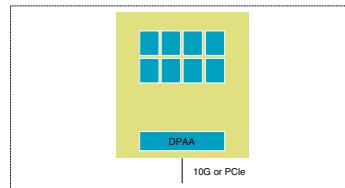


Figure 1-7. High-Performance Compute Blade

Figure 1-8 shows the potential scaling for very high performance compute blade.

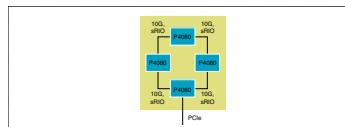


Figure 1-8. Potential Scaling for Very High-Performance Compute Blade

1.5 Subsystem Features

This section highlights the subsystems or complexes of the P4080 SoC and describes their features.

1.5.1 e500 Core and Cache Memory Complex

The P4080 offers eight high-performance 32-bit Power Architecture Book E-compliant e500mc cores. Like previous e500 cores, each e500mc is a superscalar dual issue processor, supporting out-of-order execution and in-order completion, which allows the Power Architecture e500mc to perform more instructions per clock than other RISC and CISC architectures.

Some of the important features of the e500mc are listed below:

- Up to 1.5 GHz at 1.0 V
- 36-bit physical addressing
- 64 TLB Variable-Size Pages
- 512-entry, 4-Kbyte pages
- Three integer units: two simple, one complex (integer multiply and divide)

- 64-byte cache line size
- L1 caches running at same frequency of CPU
 - 32-Kbyte Instruction, 8-way pseudo-LRU replacement policy
 - 32-Kbyte Data, 8-way pseudo-LRU replacement policy
 - Both with data and tag parity protection
- Supports Datapath Acceleration Architecture (DPAA) data and context stashing into L1 data cache and backside L2 cache
- User, Supervisor, and Hypervisor instruction level privileges
- New processor facilities
 - Hypervisor auxiliary processing unit (APU)
 - Classic double-precision, floating-point unit
 - Uses 32 64-bit floating-point registers (FPRs) for scalar single- and double-precision floating-point arithmetic
 - Replaces the embedded floating-point facility (SPE) implemented on the e500v1 and e500v2
 - "Decorated Storage" APU for improved statistics support
 - Provides additional atomic operations, including a "fire-and-forget" atomic update of up to two 64-bit quantities by a single access
 - Expanded interrupt model
 - Improved programmable interrupt controller (PIC) automatically ACKs interrupts
 - Implements message send and receive functions for interprocessor communication, including receive filtering
 - External PID load and store facility
 - Provides system software with an efficient means to move data and perform cache operations between two disjoint address spaces
 - Eliminates the need to copy data from a source context into a kernel context, change to destination address space, then copy the data to the destination address space or alternatively to map the user space into the kernel address space

Each e500mc core features a 128-Kbyte private L2 cache running at the same frequency of CPU. The L2 caches support:

- Write Back, pseudo LRU replacement algorithm
- Tag parity and ECC data protection
- 8-way, with arbitrary partitioning between instruction and data. For example, 3-way instruction, 5-way data, and so
- Supports direct stashing of datapath architecture data into L2

The P4080 also contains 2-Mbyte of shared L3 CoreNet platform cache (CPC), with the following features:

- Configurable as Write-Back or Write-Through
- ECC protection for both tags and data
- 64-byte coherency granule
- Two cache line reads (1024 bits) per cycle at 800 MHz, 0.8 terabits/sec read bandwidth
- Pseudo LRU replacement algorithm
- 32-way cache array configurable to any of several modes on a per-way basis
 - Unified cache, I-only, D-only
 - I/O stash (configurable portion of each packet copied to CPC on write to main memory)
 - Stashing of all transactions and sizes supported
 - Explicit (CoreNet signalled) and implicit (address range based) stash allocation
 - Addressable SRAM (32-Kbyte granularity)

1.5.2 CoreNet Fabric and Address Map

The CoreNet fabric is Freescale's next generation Front-side Interconnect Standard for multicore products. CoreNet is a highly concurrent, fully cache coherent, multi-ported fabric. CoreNet's point-to-point connectivity with flexible protocol architecture allows for pipelined interconnection between CPUs, platform caches, memory controllers, and I/O and accelerators at up to 800 MHz.

CoreNet has been designed to overcome bottlenecks associated with shared bus architectures, particularly address issue and data bandwidth limitations. The P4080's multiple, parallel address paths allow for high address bandwidth, which is a key performance indicator for large coherent multicore processors.

CoreNet also eliminates address retries triggered by CPUs being unable to snoop within the narrow snooping window of a shared bus. This results in the P4080 having lower average memory latency.

The flexible P4080 36-bit physical address map consists of local space and external address space. For the local address map, thirty-two local access windows define mapping within the local 36-bit (64-Gbyte) address space. Inbound and outbound translation windows can map the P4080 into a larger system address space such as the RapidIO or PCI-Express 64-bit address environment. This functionality is included in the address translation and mapping units (ATMUs).

1.5.3 Memory Complex

The P4080 memory complex consists of the two DDR controllers for main memory, and the memory controllers associated with the enhanced local bus controller (eLBC).

1.5.3.1 DDR Memory Controllers

The two DDR memory controllers support DDR2 and DDR3 SDRAM. The memory interface controls main memory accesses and together the two controllers support a maximum of 64 Gbyte of main memory. The P4080 also supports chip-select interleaving within a controller as well as interleaving across controllers on bank, page, or cache line boundaries.

The P4080 can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 64 simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, page mode can save up to 10 memory clock cycles for subsequent burst accesses that hit in an active page.

Using ECC, the P4080 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

Upon detection of a loss of power signal from external logic, the DDR controllers can put compliant DDR SDRAM DIMMs into self-refresh mode, allowing systems to implement battery-backed main memory protection. In addition, the DDR controllers offer an initialization bypass feature for use by system designers to prevent re-initialization of main memory during system power-on after an abnormal shutdown. The DDR controllers also support active zeroization of system memory upon detection of a user-defined security violation.

1.5.3.2 PreBoot Loader and Nonvolatile Memory Interfaces

The PreBoot Loader (PBL) is a new logic block that operates similarly to an I²C boot sequencer, but on behalf of a larger number of interfaces. The PBL exists to simplify boot operations, replacing pin strapping resistors with configuration data loaded from nonvolatile memory. The PBL uses the configuration data to initialize other system logic and to copy data from low speed memory interfaces (I²C, eLBC, SPI, and SD/MMC) into fully initialized DDR or the 2-Mbyte front-side cache. The PBL then releases CPU 0 from reset, allowing the boot processes to begin from fast system memory.

The nonvolatile memory interfaces accessible by the PBL are described in the subsequent sections. Note that these interfaces may be accessed by software running on the CPUs following boot; they are not dedicated to the PBL. Also note that the eLBC can be used for both volatile (SRAM) and nonvolatile memory, as well as a control and low performance data port for external memory-mapped devices.

1.5.3.2.1 Enhanced Local Bus Controller

The enhanced local bus controller (eLBC) port connects to a variety of external memories, DSPs, and ASICs. Three separate state machines share the same external pins and can be programmed separately to access different types of devices. The general-purpose chip select machine (GPM) controls accesses to asynchronous devices using a simple handshake protocol. The user-programmable machine (UPM) can be programmed to interface to synchronous devices or custom ASIC interfaces. The NAND flash control machine (FCM) further extends interface options. Each chip select can be configured so that the associated chip interface is controlled by the GPM, UPM, or FCM controller. All controllers can be enabled simultaneously. The eLBC internally arbitrates among the controllers, allowing each to read or write a limited amount of data before allowing another controller to use the bus.

Features of the local bus controller are as follows:

- Multiplexed 32-bit address and 16-bit data bus operating at up to 83 MHz
- Eight chip selects for eight external slaves
- Up to eight-beat burst transfers
- 8- and 16-bit port sizes controlled by an internal memory controller
- Three protocol engines on a per-chip-select basis
- Parity support
- Default boot ROM chip select with configurable bus width (8- or 16-bit)
- Support for parallel NAND and NOR flash

1.5.3.2.2 Serial Memory Controllers

In addition to the parallel NAND and NOR flash supported by means of the eLBC, the P4080 supports serial flash using SPI and SD/MMC card interfaces. The SD/MMC controller includes a DMA engine, allowing it to move data from serial flash to external or internal memory following initiation by software.

1.5.4 Universal Serial Bus (USB) 2.0

The two USB 2.0 controllers provide point-to-point connectivity complying with the USB specification, Rev. 2.0. Each of the USB controllers can be configured to operate as a stand-alone host, and one of the controllers (USB #2) can be configured as a stand-alone device, or with both host and device functions operating simultaneously.

USB 2.0 controller highlights:

- Complies with USB specification, Rev. 2.0
- Supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operations
- Supports external PHY with UTMI+ low-pin interface (ULPI)
 - ULPI interfaces are muxed with RGMII. Requires system configuration choice between use of USB and use of 10/100/1000 Ethernet MAC (dTSEC) by means of RGMII.
- Both controllers support operation as a stand-alone USB host controller
 - Supports USB root hub with one downstream-facing port
 - Enhanced host controller interface (EHCI)-compatible
- One controller supports operation as a stand-alone USB device
 - Supports one upstream-facing port
 - Supports six programmable USB endpoints

The host and device functions are both configured to support all four USB transfer types: bulk, control, interrupt, and isochronous.

1.5.5 High-Speed Peripheral Interface Complex

All high-speed peripheral interfaces connect to a common crossbar switch referred to as Ocean (OCN). Two high-speed I/O interface standards are supported: PCI Express (PCIe) and serial RapidIO (sRIO). The P4080 integrates three PCIe controllers and two serial RapidIO controllers plus a RapidIO Messaging Unit (eRMU). See [Reference Clocks for SerDes Protocols](#), for more information on SerDes usage for Ethernet and high-speed peripheral interfaces.

The features of each controller are described in the subsequent sections.

1.5.5.1 PCI Express Controllers

Each of the three PCIe interfaces is compliant with the *PCI Express Base Specification Revision 2.0*. Power-on reset configuration options allow root complex or endpoint functionality. The physical layer operates at 2.5 or 5 Gbaud data rate per lane. Receive and transmit ports operate independently, with an aggregate theoretical bandwidth of 32 Gbps. Other features of the PCIe interfaces include the following:

- x8, x4, x2, and x1 link widths supported
- Both 32- and 64-bit addressing and 256-byte maximum payload size
- Full 64-bit decode with 36-bit wide windows
- Inbound INTx transactions
- Message Signaled Interrupt (MSI) transactions

1.5.5.2 Serial RapidIO

The serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 1.2*. RapidIO is a high-performance, point-to-point, low-pin-count, packet-switched system-level interconnect that can be used in a variety of applications as an open standard. The rich feature set includes high data bandwidth, low-latency capability, and support for high-performance I/O devices as well as message-passing and software-managed programming models. Receive and transmit ports operate independently, and with 2 x4 serial RapidIO controllers, the aggregate theoretical bandwidth is 16 Gbps. Key features of the serial RapidIO interface unit include:

- Support for *RapidIO Interconnect Specification, Revision 1.2* (all transaction flows and priorities)
- Both 1x and 4x LP-serial link interfaces, with transmission rates of 2.5 or 3.125 Gbaud (data rates of 2.0 or 2.5 Gbps) per lane
- Auto-detection of 1x or 4x mode operation during port initialization
- 34-bit addressing and up to 256-byte data payload
- Receiver-controlled flow control
- RapidIO error injection

The RapidIO message unit (RMU) manages two inbox/outbox mailboxes (queues) for data and one doorbell message structure. The outbox operates in both chaining and direct modes, and messages can hold up to 16 packets of 256 bytes, or a total of 4 Kbyte. The RMU can also multi-cast a single-segment 156-byte message to up to 32 different destination DevIDs. The RMU supports Type11 message formats.

1.5.6 Datapath Acceleration Architecture (DPAA)

The P4080 includes the first implementation of the datapath acceleration architecture (DPAA). This architecture provides the infrastructure to support simplified sharing of networking interfaces and accelerators by multiple CPU cores. These resources are abstracted into enqueue/dequeue operations by means of a common DPAA Queue Manager Driver. Beyond enabling multicore sharing of resources, the DPAA significantly reduces software overheads associated with high-touch packet-forwarding operations. Examples of the types of packet-processing services this architecture is optimized to support include traditional routing and bridging, firewall, VPN termination for both IPsec and SSL VPNs, Intrusion Detection/Prevention (IDS/IPS), and network anti-virus (AV).

The DPAA generally leaves software in control of protocol processing while reducing CPU overheads through off-load functions, which fall into the following broad categories:

1. Packet distribution and queue/congestion management. Off-load functions in this category include:
 - Data buffer management: supports allocation and deallocation of buffers belonging to pools originally created by software with configurable depletion thresholds. Implemented in a block called the Buffer Manager (BMan).
 - Queue management: supports queuing and quality-of-service scheduling of frames to CPUs, network interfaces and DPAA logic blocks, maintains packet ordering within flows. Implemented in a block called the queue manager (QMan). The QMan, besides providing flow-level queuing, is also responsible for congestion management functions such as RED/WRED, Congestion Notifications and Tail Discards.
 - Packet distribution: supports in-line packet parsing and general classification to enable policing and QoS-based packet distribution to the CPUs for further processing of the packets. This function is implemented in the block called frame manager (FMan).
 - Policing: supports in-line rate-limiting by means of Two-Rate-Three Color Marking (RFC2698). Up to 256 policing profiles are supported. This function is also implemented in the Fman.
2. Content processing acceleration. Properly implemented acceleration logic can provide significant performance advantages over the most optimized software, with acceleration factors on the order of 10-100x. Accelerators in this category typically touch most of the bytes of a packet, not just headers. To avoid consuming CPU cycles to move data to the accelerators, these engines include well-pipelined DMAs. Specific content processing accelerators in the P4080 include the following:

- SEC 4.0: Crypto-Acceleration for protocols such as IPsec, SSL, and 802.16
- PME 2.0: Regex style pattern matching for unanchored searches, including cross-packet stateful patterns

Prior versions of the SEC and PME are integrated into multiple members of Freescale communications processor families. Both of these engines are enhanced to work within the DPAA and are also upgraded in both features and performance.

1.5.6.1 Datapath Acceleration Architecture Programming Model

The DPAA assumes the existence of network flows, each of which is a series of packets that have the same packet processing and ordering requirements. Software has full flexibility in how each flow is defined, from a broad grouping of packets down to an individual session that is more synonymous with the standard networking definition of a flow. In general, packets arriving from the network are categorized into broader flows to be distributed to the appropriate cores as defined by packet parsing and classification rules (for example, control plane traffic), while packets being sent to the hardware accelerators are categorized into more granular flows (for example, a specific IPsec tunnel). Control software would set up these flows through the SoC by updating or creating data structures describing how packets in the flow should be treated. The DPAA prescribes specific data structures to be created by the control processor at flow establishment time, and also defines how datapath processors should interact with these data structures in order to move packets through the off-load engines and outbound network interfaces with the least CPU overhead. The DPAA also provides for a uniform programming interface for the hardware accelerators and network interfaces. Rather than porting multiple distinct device drivers to control and datapath software, the user ports a unified queue manager driver.

It is important to note that the DPAA is not an all-or-nothing programming model. It is possible to use legacy implementations of packet classification and buffer management and still take advantage of the Queue Interface Driver's simplified, software-friendly interface. Operations are encoded in architected messages which are placed on queues, and responses (normal and error) flow back to software using the same queue structures.

[Figure 1-9](#) shows the queuing messages to channels.

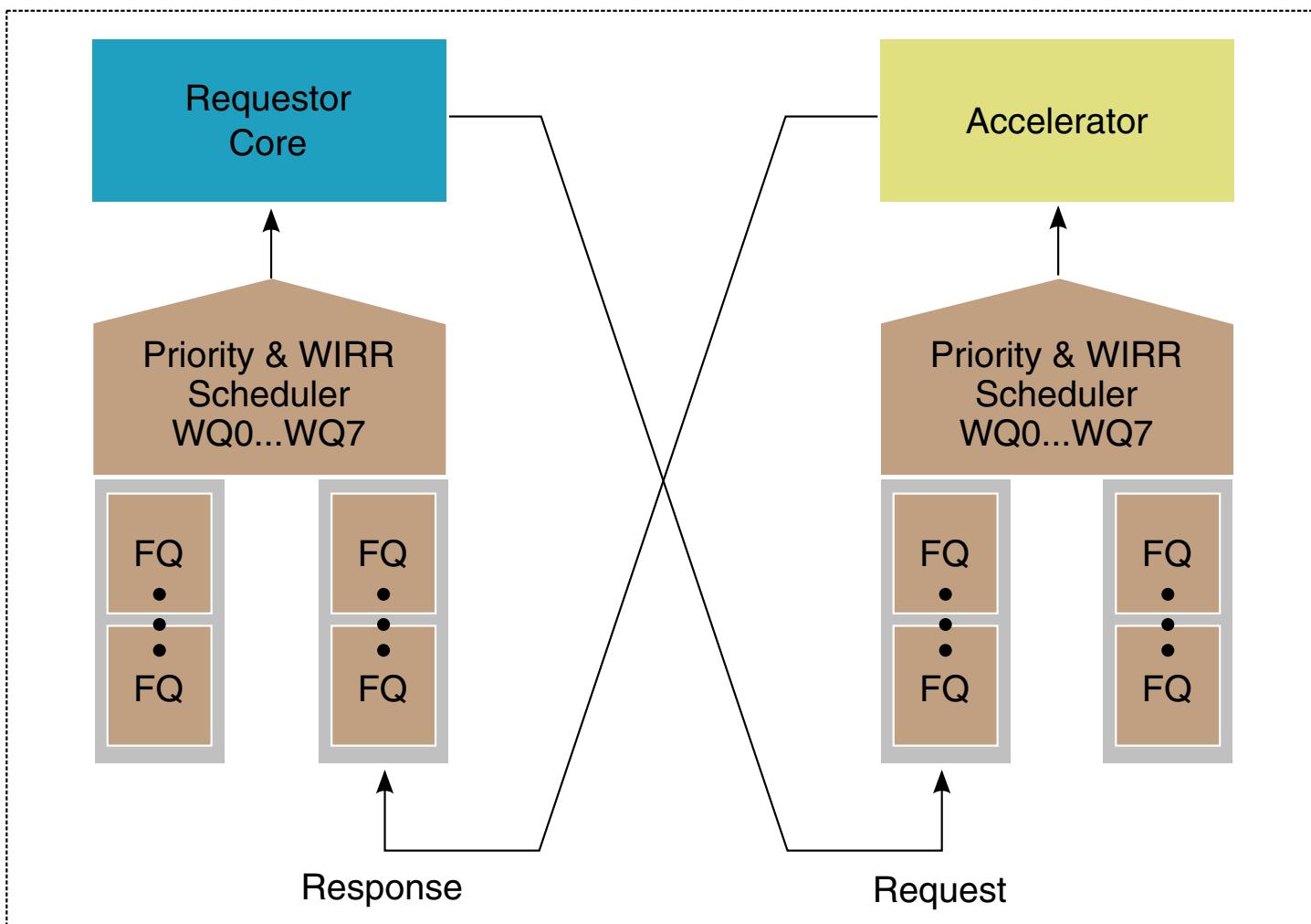


Figure 1-9. Queuing Messages to Channels

1.5.6.2 DPAA Definitions

Buffer

Region of contiguous memory, allocated by software, managed by the DPAA Buffer Manager.

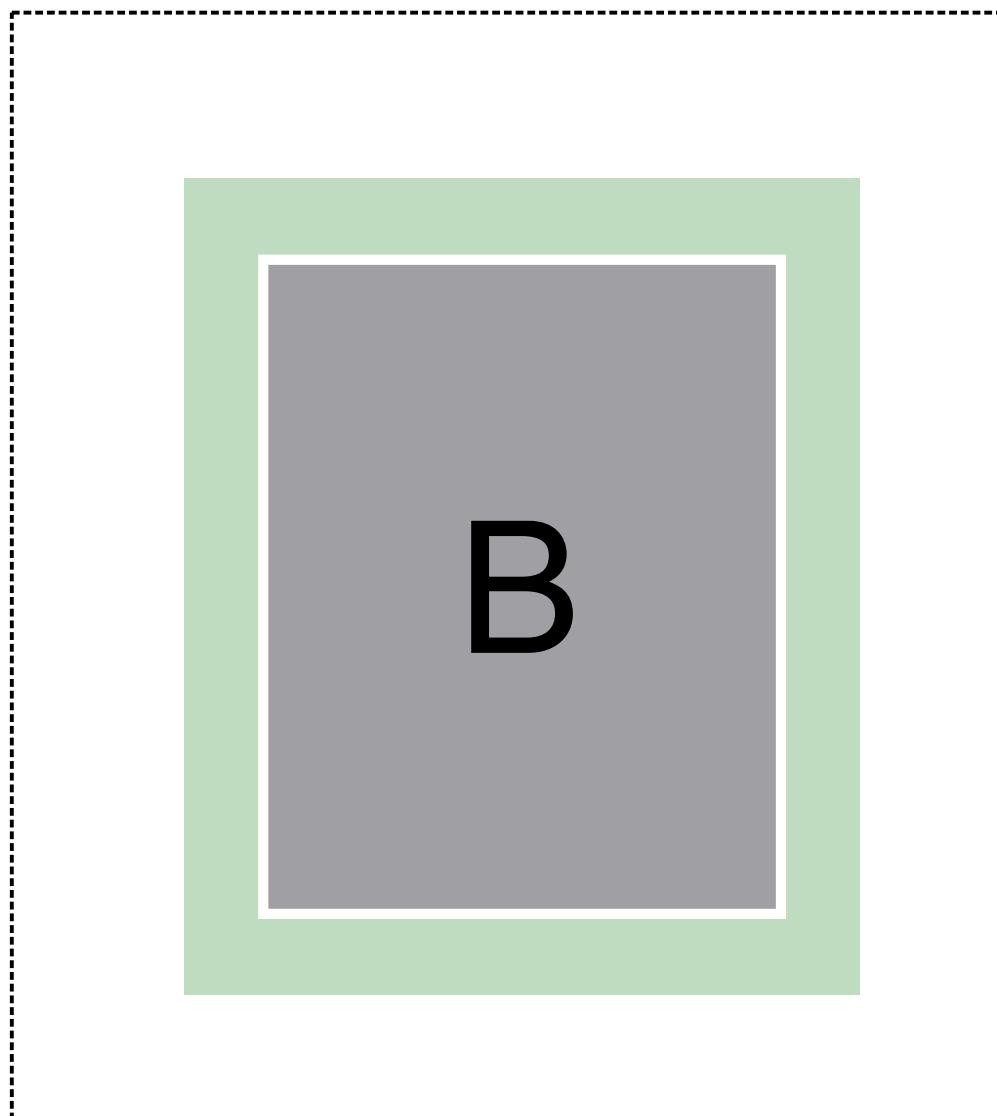


Figure 1-10. Buffer

Buffer Pool

Set of buffers with common characteristics (mainly size, alignment, access control)

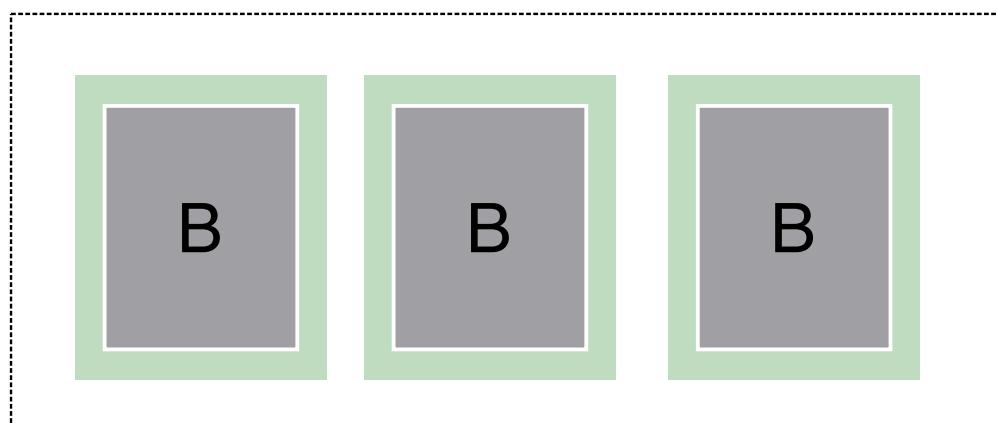


Figure 1-11. Buffer Pool

Frame

Contents of a single buffer or list of buffers that hold data. For example, packet payload, header, and other control information.

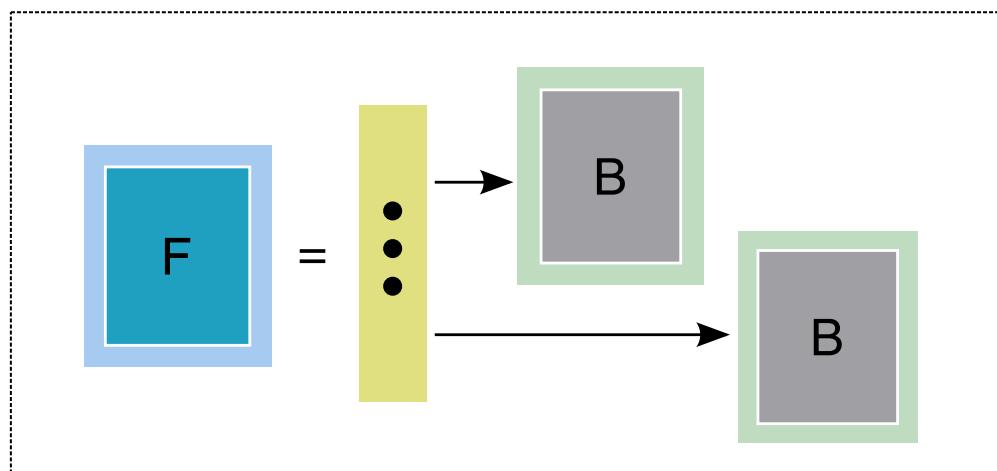


Figure 1-12. Frame

Frame Queue

FIFO of Frames

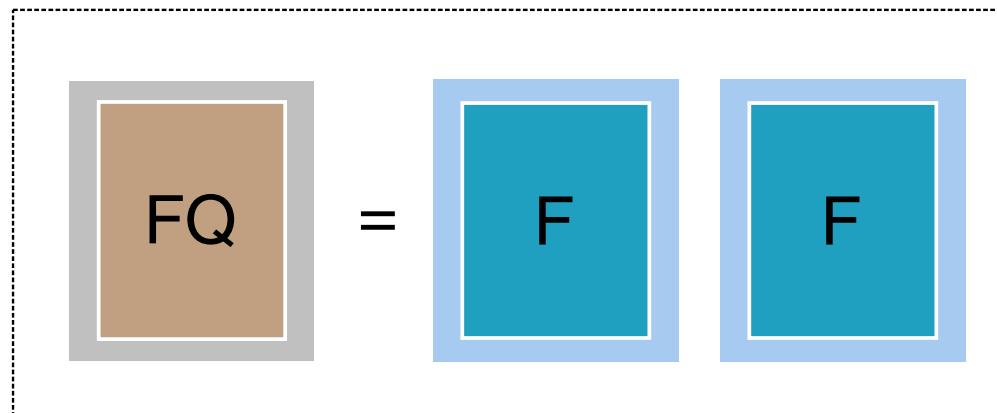


Figure 1-13. Frame Queue

Work Queue

FIFO of Frame Queues

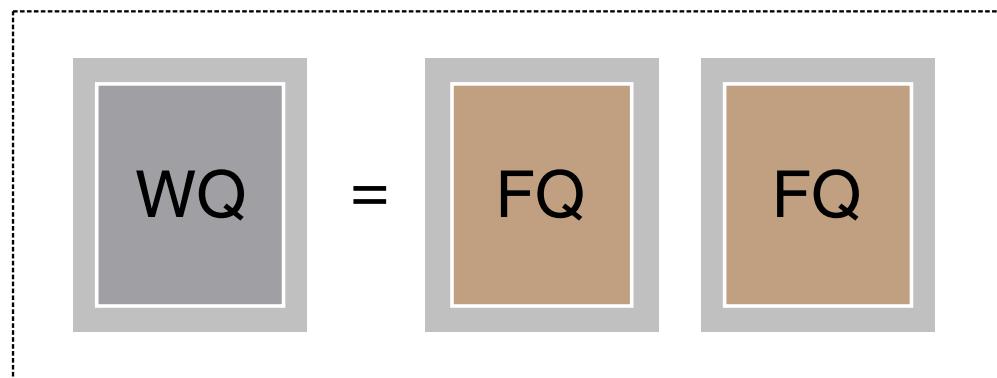


Figure 1-14. Work Queue

Channel

Set of eight Work Queues, with hardware provided prioritized access

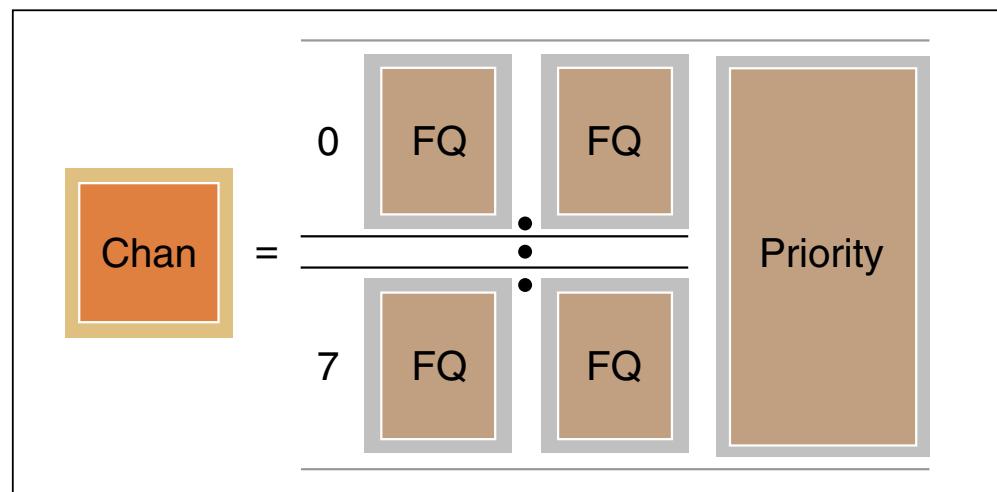


Figure 1-15. Channel

Dedicated Channel

Channel statically assigned to a particular end point, from which that end point can dequeue frames. End point may be a CPU, FMan, PME, or SEC.

Pool Channel

A channel statically assigned to a group of end points, from which any of the end points may dequeue frames.

1.5.7 Major DPAA Components

The QorIQ Data Path Acceleration Architecture (DPAA), shown in [Figure 1-16](#), includes the following major components:

- Frame Manager (FMan)
- Queue Manager (QMan)
- Buffer Manager (BMan)
- SEC 4.0
- PME 2.0

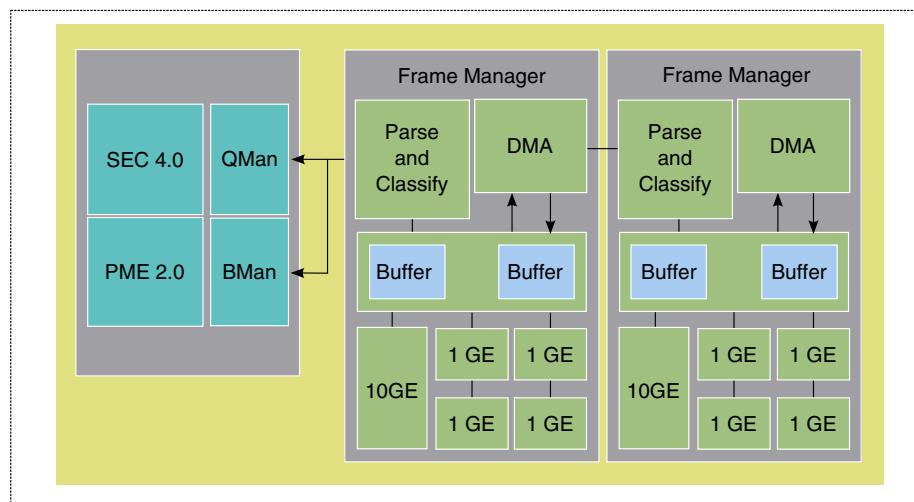


Figure 1-16. QorIQ Data Path Acceleration Architecture (DPAA)

1.5.7.1 Frame Manager

The P4080 is the first product in the QorIQ family to incorporate a Frame Manager (FMan). An FMan is a functional unit that combines the Ethernet network interfaces with packet distribution logic to provide intelligent distribution and queuing decisions for incoming traffic at line rate (18 Mpps). This integration allows the FMan to perform configurable parsing and classification of the incoming frame with the purpose of selecting the appropriate input frame queue (FQ) for expedited processing by a CPU or pool of CPUs.

1.5.7.1.1 Network Interfaces

Each of the two FMan in the P4080 integrates 4 datapath three-speed Ethernet controllers (dTSECs) and one 10-Gbit Ethernet controller (10GEC). The more basic parsing and filing capability found in prior eTSECs is removed from the MACs themselves and aggregated in the more flexible and robust parsing and classification logic described in [Parse Function](#).

The Ethernet controllers support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. They are designed to comply with IEEE Std 802.3®, IEEE 802.3u, IEEE 802.3x, IEEE 802.3z, IEEE 802.3ac, IEEE 802.3ab, and IEEE-1588 v2 (clock synchronization over Ethernet).

NOTE

P4080, Rev 3 supports 1588 time-stamping in SGMII 10/100 mode; P4080, Rev 2, does not support 1588 time-stamping in SGMII 10/100 mode.

The dTSECS are capable of full- and half-duplex Ethernet support (1000 Mbps supports only full duplex); the 10-Gbit MACs are single-speed full duplex. Both support IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software-programmed PAUSE frame generation and recognition).

SerDes flexibility makes it possible to enable up to 14 Gbps of Ethernet traffic on a single FMan; however, each FMan can support line rate parsing and classification at 12 Gbps. In this situation, it may be desirable to configure one FMan to process frames from the 10 Gbps interface, and the second FMan to process frames from the four individual 1Gbps interfaces.

See [Reference Clocks for SerDes Protocols](#), for more information on SerDes usage for Ethernet and high speed peripheral interfaces.

It is possible to configure two dTSECs to utilize RGMII physical interfaces in case all SerDes lanes are otherwise allocated.

1.5.7.1.2 Parse Function

The primary function of the Packet Parse logic is to identify the incoming frame for the purpose of determining the desired treatment to apply. This Parse function can parse many standard protocols (including options and tunnels) and supports a generic configurable capability to allow proprietary or future protocols to be parsed.

Proprietary headers can be defined as being self-describing or non-self-describing. Self-describing headers are announced by proprietary values of Ethertype, Protocol Identifier, Next Header, and other standard fields. They are self-describing in that the frame contains information that describes the presence of the proprietary header. In contrast, non-self-describing proprietary headers do not contain any information that indicates the presence of the header. For example, a frame that always contains a proprietary header before the Ethernet header would be non-self-describing. Both self-describing and non-self-describing headers are supported by means of parsing rules in the FMan.

The underlying notion is that different frames may require different treatment and only through detailed parsing of the frame can proper treatment be determined.

Parse results can optionally also be passed to software.

1.5.7.1.3 Distribution and Policing

When parsing is complete the treatment can be identified. That treatment can be to hash selected fields in the frame as part of a spreading mechanism or that treatment can be to look up certain fields in the frame to determine subsequent action to take, including policing.

The hash capability allows a hash key to be built from many different fields in the frame to provide differentiation between flows. Default values can be used if fields are not present in the frame. Alternatively, a key can be built exclusively from fields present in the frame based on what the Parse Function has detected. The result of the hash is a specific frame queue (FQ) identifier. To support added control this FQID can be indexed by values found in the frame such as TOS or p-bits or any other desired field(s). This is useful when it is required to spread traffic while obeying QoS constraints.

In some situations, a hash distribution based on parse results may be insufficient and a more detailed examination of the frame is required. Instead of or prior to the hash, the FMan supports a coarse classification capability to look up fields in the frame to determine the action to take. The FMan contains internal memory that holds small tables for this purpose.

Classification lookups are performed based on the combination of user configuration and what fields the parser actually encountered. That is, the user configures the sets of lookups to perform and the parse results dictate which one of those sets to use. Lookups can be chained together such that a successful lookup can provide key information for a subsequent lookup. After all the lookups are complete, the final classification result provides either a hash key to use for spreading or a FQ ID directly.

In addition to selecting the FQ ID, classification can determine whether policing is required and the policing context to use.

This choice of FQ could depend on the flow being either directed to a particular CPU, quality of service consideration, control plane/data plane traffic, etc. For example, the most obvious scenario is the distribution of flows on FQs based on their DSCP or IP precedence bits. Thus, up to eight different FQs would be created. CPUs which service those FQs can then preferentially schedule these queues or let the QMan perform preferential scheduling for the CPUs. Because the FMan has up to 256 policing profiles, any FQ or group of FQs can be policed to either drop or mark packets if the flow exceeds a preconfigured rate. The policing function in conjunction with classification can be used for mitigating Distributed Denial of Service (DDoS) attacks.

The policing is based on the Two-Rate-Three-Color Marking algorithm (RFC 2698). The sustained and peak rates as well as the burst sizes are user-configurable. Hence, the policing function can rate-limit traffic to conform to the rate to which the flow is mapped at flow setup time. By prioritizing and policing traffic prior to software processing, CPU cycles can be focused on the important and urgent traffic above other traffic.

1.5.7.2 Queue Manager

The Queue Manager (QMan) is the component in the DPAA that allows for simplified sharing of network interfaces and hardware accelerators by multiple CPU cores. It also provides a simple and consistent message and data passing mechanism for dividing processing tasks among multiple CPU cores.

The QMan offers the following features:

- Common interface between software and all hardware
 - controls the prioritized queuing of data between multiple processor cores, network interfaces, and hardware accelerators
 - supports both dedicated and pool channels, allowing both push and pull models of multicore load spreading
- Atomic access to common queues without software locking overhead
- Mechanisms to guarantee order preservation with atomicity and order restoration following parallel processing on multiple CPUs
- Two level queuing hierarchy with one or more channels per endpoint, eight Work Queues per channel, and numerous FQs per work queue
- Priority and work conserving fair scheduling between the work queues and the FQs
- Loss-less flow control for ingress network interfaces
- Congestion avoidance (RED/WRED) and congestion management with tail discard and up to 256 congestion groups. Each congestion group is composed of a user-configured number of FQs. Congestion notification is sent to an endpoint when the sum of the buffer occupancies of the group's FQs reaches a pre-determined congestion threshold. Congestion group status has hysteresis built in.

1.5.7.3 Buffer Manager

The buffer manager (BMan) manages pools of buffers on behalf of software for both hardware (accelerators and network interfaces) and software use.

The BMan offers the following features:

- Common interface for software and hardware
- Guarantees atomic access to shared buffer pools
- Supports 64 buffer pools. Software and hardware buffer consumers can request different size buffers and buffers in different memory partitions
- Supports depletion thresholds with congestion notifications
- On-chip per pool buffer stockpile to minimize access to memory for buffer pool management
- LIFO (last in first out) buffer allocation policy
 - Optimizes cache usage and allocation
 - A released buffer is immediately used for receiving new data

1.5.7.4 Security Engine (SEC 4.0)

The SEC 4.0 is the fourth generation crypto-acceleration engine in Freescale communications processors. [Figure 1-17](#) shows the SEC 4.0 block diagram.

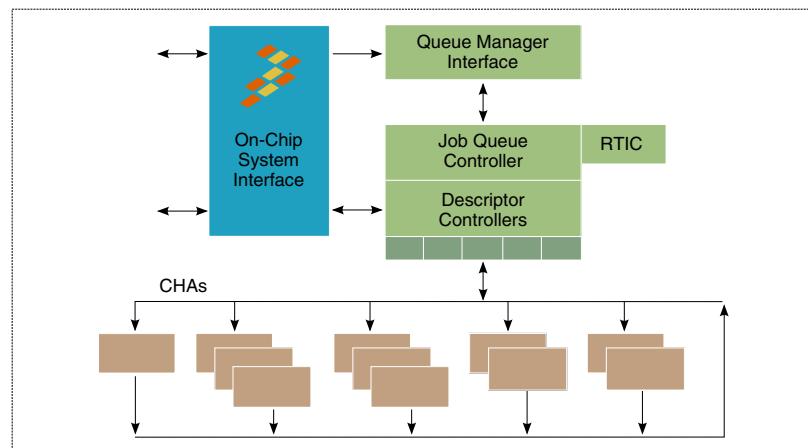


Figure 1-17. SEC 4.0 Block Diagram

In addition to off-loading cryptographic algorithms, the SEC 4.0 offers header and trailer processing for several established security protocols. The SEC 4.0 includes five descriptor controllers (DECOs), which are updated versions of the previous SEC crypto-

channels. DECOs are responsible for header and trailer processing as well as managing context and data flow into the crypto hardware accelerators (CHAs) assigned to the DECO for the length of an operation.

The DECOs can perform header and trailer processing as well as single pass encryption/integrity checking for the following security protocols:

- IPsec
- SSL/TLS
- SRTP
- 802.1AE MACSec
- 802.16e WiMax MAC layer
- 3GPP RLC encryption/decryption

In prior versions of the SEC, the individual algorithm accelerators were referred to as execution units (EUs). In the SEC 4.0, these are referred to as crypto hardware accelerators (CHAs) to distinguish them from prior implementations. Specific CHAs available to the DECOs are as follows:

- Advanced encryption standard accelerator (AES-A)
- "Alleged" RC4 hardware accelerator (AFHA)
- Cyclic redundancy check accelerator (CRCA)
- Data encryption standard accelerator (DESA)
- Kasumi f8/f9 hardware accelerator (KFHA)
- SNOW 3G hardware accelerator (STHA)
- Message digest hardware accelerator (MDHA)
- Public key hardware accelerator (PKHA)
- Random number generator version B (RNGB)

Depending on the security protocol and specific algorithms, the SEC 4.0's aggregate symmetric encryption/integrity performance is 10 Gbps, while asymmetric encryption (RSA public key) performance is ~10,000 1024b RSA operations per second.

The SEC 4.0 is also part of the QorIQ Trust Architecture, which gives the P4080 the ability to perform secure boot, runtime code integrity protection, and session key protection.

1.5.7.5 Pattern Matching Engine (PME 2.0)

The pattern matching engine (PME) is a self-contained hardware block capable of autonomously scanning data from streams for patterns that match a specification in a database dedicated to it. The PME 2.0 is an updated version of the PME used in previous Freescale products. Specific updates include a Queue Manager Interface, supporting the

DPAA queue interface driver, and a 2x increase in the number of patterns (16K → 32K) and stateful rules (8K → 32K) supported. Raw scanning performance has also increased 4x to ~10 Gbps. [Figure 1-18](#) shows the PME 2.0 block diagram.

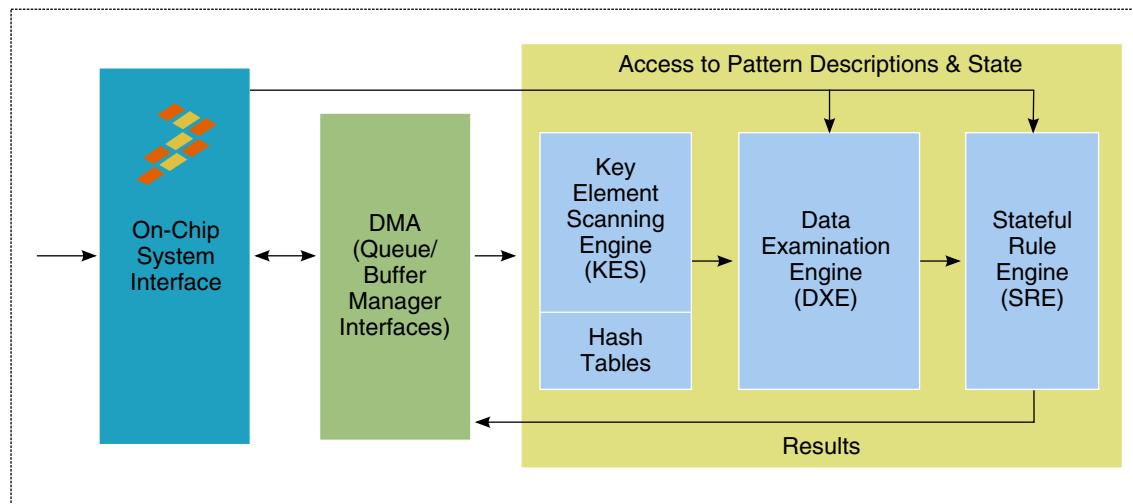


Figure 1-18. PME 2.0 Block Diagram

Patterns that can be recognized (matched) by the PME are of two general forms: byte patterns and event patterns. Byte patterns are simple matches such as 'abcd123' existing in both the data being scanned and in the pattern specification database. Event patterns are a sequence of multiple byte patterns. In the PME, event patterns are defined by stateful rules.

The PME specifies patterns of bytes as regular expressions (Regex). The P4080 (by means of an on-line or off-line process) converts regex patterns into the PME's pattern specification database. Generally there is a one-to-one mapping between a regular expression and a PME byte pattern. The PME's use of regular expression pattern matching offers built-in case-insensitivity and wildcard support with no pattern explosion. The PME's non-deterministic, finite automaton (NFA) style architecture offers fast pattern database compilation and fast incremental updates. Up to 32,000 regular expression patterns are supported, each up to 128 bytes long. The 32,000 regular expression patterns can be combined by means of stateful rules to detect a far larger set of event patterns. Comparative compilations against deterministic, finite automaton (DFA) style regular expression engines have shown that 300,000 DFA pattern equivalents can be achieved with ~8000 PME regular expressions with stateful rules.

Within the PME, match detection proceeds in stages. The key element scanner (KES) performs initial byte pattern matching with handoff to the Data Examination Engine (DXE) for elimination of false positives through more complex comparisons. As the name implies, the Stateful Rule Engine (SRE) receives confirmed basic matches from the earlier stages and monitors a stream for subsequent matches that define an event pattern.

1.6 Resource Partitioning and QorIQ Trust Architecture

Consolidation of discrete CPUs into a single multicore SoC and potential repartitioning of legacy software on those cores introduces many opportunities for unintended resource contentions to arise. A system may exhibit erratic behavior if the multiple CPUs do not effectively partition and share system resources. While it can be challenging to prevent unintended resource contention, stopping malicious software is much more difficult. Device consolidation, combined with a trend toward embedded systems becoming more open (or more likely to run third-party or open-source software on at least one of the cores), creates opportunities for malicious code to enter a system.

The P4080 offers a new level of hardware partitioning support, allowing system developers to ensure software running on any CPU only accesses the resources (memory, peripherals, etc.) that it is explicitly authorized to access. This may not seem like a challenge in a SMP environment, because the OS performs resource allocation for the applications running on it, but it is a very difficult problem in AMP environments where there may be multiple instances of the same OS or even different OSes running on the various CPU cores. Even OS protections in an SMP system may be insufficient in the presence of malicious software.

1.6.1 e500mc MMU and Embedded Hypervisor

The P4080's first line of defense against unintended interactions amongst the multiple CPUs/OSes is each e500mc core's MMU. Each e500mc core's MMU will be configured to determine which addresses in the global address map the CPU will be able to read or write. If a particular resource (portion of memory, peripheral device) is dedicated to a single CPU, that CPU's MMU is configured to allow access to those addresses (on 4-Kbyte granularity); other CPU MMUs are not configured for access to the other CPU's private memory range. When two CPUs need to share resources, their MMUs are both configured so that they have access to the shared address range.

This level of hardware support for partitioning is common today; however, it is not sufficient for many core systems running diverse software. When the functions of multiple discrete CPUs are consolidated onto a single multicore SoC, achieving strong partitioning should not require the developer to map functions onto cores that are the exclusive owners of specific platform resources. The alternative, a fully open system with no private resources, is also unacceptable. For this reason, the e500mc MMU also includes embedded Hypervisor extensions. Each e500mc MMU supports three levels of instructions: User, Supervisor (OS), and Hypervisor. An embedded Hypervisor micro-

kernel (provided by Freescale as source code) runs unobtrusively beneath the various OSes running on the CPUs, consuming CPU cycles only when an access attempt is made to an embedded Hypervisor-managed shared resource.

The embedded Hypervisor determines whether the access should be allowed, and if so, proxies the access on behalf of the original requestor. If malicious or poorly tested software on any core attempts to overwrite important device configuration registers (including CPU MMUs), the embedded Hypervisor blocks the write. Other examples of embedded Hypervisor managed resources are high- and low-speed peripheral interfaces (PCIe, UART) if those resources are not dedicated to a single CPU/partition.

1.6.2 Peripheral Access Management Unit (PAMU)

Being MMU-based, the embedded Hypervisor is only able to stop unauthorized software access attempts. Internal components with bus mastering capability (PME, SEC, FMan, etc.) also need to be prevented from reading and writing to specific memory regions.

These devices do not spontaneously generate access attempts, but if programmed to do so by buggy or malicious software, any of them could overwrite sensitive configuration registers and cause the system to fail. For this reason, the P4080 also includes a distributed function, collectively referred to as the Peripheral Access Management Unit (PAMU), which provides address translation and access control for all bus masters in the system. PAMU access control can be absolute (FMan, PME, SEC, other bus masters can never access memory range XYZ), or it can be conditional, based on the Partition ID of the CPU that programmed the bus master.

1.6.3 Secure Boot and Sensitive Data Protection

Many of the P4080's target applications involve multiple OSes or multiple instances of the same OS running asynchronously on the e500 cores. It is further expected that many of these target applications will be more "open" than traditional embedded systems, meaning that less of the system software will be created by the system developer.

The existence of multiple OSes (more precisely, multiple partitions) on the same device means that traditional OS resource management is insufficient to prevent partitions from interfering with each other, whether inadvertently or deliberately. The P4080's e500mc MMUs and PAMU create bulkheads between the partitions, while the embedded Hypervisor allows for safe sharing of resources required by multiple partitions. This combination allows P4080 users to run a mixture of OSes and applications with significant resilience against poorly tested and even malicious code.

The bulkheads offered by the e500mc MMUs, PAMU, and Hypervisor can be very effective in blocking or containing the effects of misprogrammed partitions, but given the choice, most developers would rather minimize the opportunity for misprogrammed partitions to exist in the first place. For this reason, the P4080 offers a secure boot option, in which the system developer digitally signs the code to be executed by the CPU coming out of reset, and the P4080 ensures that only an unaltered version of that code runs on the platform. The P4080 offers both boot time and run time code authenticity checking, and offers configurable consequences when the authenticity check fails. The P4080 also supports protected internal and external storage of developer-provisioned sensitive instructions and data. For example, a system developer may provision each system with a number of RSA private keys to be used in mutual authentication and key exchange. These values would initially be stored in encrypted form in external non-volatile memory, but following secure boot, these values can be decrypted into on-chip protected memory (portion of platform cache dedicated as SRAM). Session keys, which may number in the thousands to tens of thousands, are not good candidates for on-chip storage, so the P4080 offers automatic session key encryption/decryption. Encrypted session keys are stored in main memory, and are decrypted (transparently to software) as they are brought into the SEC 4.0 for decryption of session traffic.

1.7 Advanced Power Management

Power management is always a major design consideration in embedded applications, and for this reason, the P4080 is designed to dissipate <30 W Max with all 8 CPUs and platform logic running at maximum frequency (8x 1.5 GHz CPUs, 800 MHz CoreNet). <30 W max power is aligned with the power envelopes of typical network and wireless infrastructure system chassis.

The P4080 does not rely on statistical processing loads or forced voltage/frequency reductions to achieve its <30 W max operation. Dynamic frequency switching (DFS) are useful techniques for reducing typical/average power and maximizing battery life in laptop environments, but embedded applications have to be designed for rapid response to bursts of traffic and max power under worst case environmental conditions. While not implementing DFS in the PC sense, the P4080 does actively manage internal clocks to avoid wasting energy. These blocks can return to full operating frequency on the clock cycle after work is dispatched to them. Based on transistor-switching rates and clock gating opportunities associated with high-throughput packet forwarding applications, a more typical power dissipation will be ~23 W.

In addition to internal clock gating, the P4080's advanced power management capabilities are based around fine-grained static clock control, and software-controlled dynamic frequency management.

Fine-grained static control allows developers to turn off the clocks to individual logic blocks within the SoC that the system has no need for. Based on a finite number of SerDes, it is expected that any given application will have some Ethernet MACs, PCIe, or serial RapidIO controllers inactive. These blocks can be disabled by means of the DEVDIS Register. Re-enabling clocks to a logic block requires an SoC reset, which makes this type of power management operation infrequent (effectively static).

Another aspect of the P4080's Advanced Power Management is the amount of control over voltage and frequency available to the system developer. The eight e500 cores draw their power from two power rails, each of which can be set to a different supply voltage. When the usage model calls for AMP or mixed AMP/SMP, two, four, or six of the P4080's e500mc cores can be turned off. Four cores can be supplied 1.1 V and run at higher frequencies. System developers can use their knowledge of their specific application to determine the P4080's static power management configurations (CPU supply voltage, max operating frequency, unneeded components) to enable operating at the minimum power consumption for that application.

In addition to fine-grained static power management, the P4080 platform supports (under software control) dynamic changes to CPU operating frequencies and voltages. Each CPU sources its input clock from one of three independent PLLs inside the P4080. Each CPU can also source its input clock from an integer frequency divider from two of the three independent PLLs. CPUs can switch their source PLL, and their frequency divider glitchlessly and nearly instantaneously. This allows each core to operate at the minimum frequency required to perform its assigned function, saving power.

Changing PLL frequency dividers (/2) can be used to achieve large and rapid reductions in dynamic power consumptions, and with the help of external temperature detection circuitry, can serve as a thermal overload protection scheme. If the junction temperature of the P4080 (or system ambient temperature) achieves some critical level, external temperature detection circuitry can drive a high-priority interrupt into the P4080, causing it to reduce selected CPU frequencies by half. This allows the system to continue to function in a degraded mode, rather than failing entirely. This technique is much simpler than turning off selected CPUs, which can involve complex task migration in an AMP system. When system temperatures have been restored to safe ranges, all CPUs can be returned to normal frequency within a few clock cycles.

When less drastic frequency changes are desired, software can switch the CPU to a slower speed PLL, that is, 1 GHz vs. 1.5 GHz. Many cores could be switched to a slower PLL during periods of light traffic, with the ability to immediately return those cores to the full rate PLL should traffic suddenly increase. The more traditional Power Architecture single core power management modes (core doze, core nap, core sleep) are also available in the e500mc; however, core sleep is not supported on multicore devices such as the P4080.

1.8 Debug support

The reduced number of external buses enabled by the move to multicore chips greatly simplifies board level lay-out and eliminates many concerns over signal integrity. While the board designer may embrace multicore CPUs, software engineers have real concerns over the potential to lose debug visibility.

Processing on a multicore chip with shared caches and peripherals also leads to greater concurrency and an increased potential for unintended interactions between device components. To ensure that software developers have the same or better visibility into the device as they would with multiple discrete communications processors, Freescale developed an Advanced Multicore Debug Architecture.

The debugging and performance monitoring capability enabled by the device hardware coexists within a debug ecosystem that offers a rich variety of tools at different levels of the hardware/software stack. Software development and debug tools from Freescale (CodeWarrior), as well as third-party vendors, provide a rich set of options for configuring, controlling, and analyzing debug and performance related events.

Chapter 2

Memory Map

2.1 Memory Map Overview

This chapter describes the mechanisms that define the device memory map: the Local Access Windows (LAWs), the AddressTranslation and Mapping units (ATMUs), and the Configuration, Control, and Status Registers (CCSRs).

There are several address domains within the device, including the following:

- Logical, virtual, and physical (real) address spaces within the Power Architecture core(s)
- Internal local address space
 - Internal configuration, control, and status register (CCSR) address space, which is a special-purpose subset of the internal local address space
 - Internal debug control and status register (DCSR) address space, which is another special-purpose set of registers mapped in the internal local address space
- External memory, I/O, and configuration address spaces of the serial RapidIO link
- External memory, I/O, and configuration address spaces of the PCI Express links

The MMU in the core handles translation of logical (effective) addresses, into virtual addresses, and ultimately to the physical addresses for the local address space. The MMU is described in the *e500mc Core Reference Manual*.

The local address map refers to the physical address space seen by the core as it accesses memory and I/O space. The DMA engines also see this same local address map. All memory controlled by the DDR and other modules exists in this address map, as do all memory-mapped configuration, control, and status registers (CCSRs). The local address map is defined by a set of 32 local access windows (LAWs). Each of these windows maps a region of the local address space to a specified target interface, such as the DDR controller, PCI Express controller, or other targets. The internal configuration, control,

and status registers (CCSRs) for all the functional blocks are located in the local memory space at a specific CCSR window. There is also a fixed (8 MB of local memory space) default boot window from 0_FF80_0000h to 0xFFFF_FFFFh.

The internal debug control and status registers (DCSRs) control the hardware debug and performance monitoring capabilities of many functional blocks. The DCSRs are mapped into the local memory space by specifying the DCSR space as a LAW target.

If the target mapping performed by the local access windows directs the transaction to one of the external peripheral interfaces (as an outbound read or write), the transaction is then mapped into that interface's external address space by the address translation and mapping unit (ATMU) windows associated with the external interface. Outbound ATMUs perform the mapping from the local address space to the address space of the external peripheral interface; inbound ATMU windows perform the address translation from the external address space to the local address space.

2.2 Global Source and Target IDs

In many instances throughout the system, certain register or packet fields need to indicate or specify a transaction source or target. [Table 2-1](#) provides the encodings used for these fields.

Table 2-1. P4080 Global Source/Target ID Assignments

Source/Target ID Value	Transaction Source	Transaction Target	Notes
0x00	PCI-Express 1	PCI-Express 1	-
0x01	PCI-Express 2	PCI-Express 2	-
0x02	PCI-Express 3	PCI-Express 3	-
...	Reserved	Reserved	-
0x08	RapidIO Port 1	RapidIO Port 1	-
0x09	RapidIO Port 2	RapidIO Port 2	-
...	Reserved	Reserved	-
0x0F	Reserved	Local Space	Inbound ATMUs must use this target ID value for all local memory regions (that is, all regions that are not mapped to an outbound ATMU)
0x10	Reserved	Memory Complex 1	DDR controller 1 or CPC1 SRAM
0x11	Reserved	Memory Complex 2	DDR controller 2 or CPC2 SRAM
...	Reserved	Reserved	-
0x14	Reserved	Interleaved Memory Complex 1/2	Interleaved DDR controllers or CPC SRAM
...	Reserved	Reserved	-

Table continues on the next page...

Table 2-1. P4080 Global Source/Target ID Assignments (continued)

Source/Target ID Value	Transaction Source	Transaction Target	Notes
0x18	Buffer Manager (control)	Buffer Manager Software Portal	-
...	Reserved	Reserved	-
0x1C	PAMU	Reserved	All PAMUs use the same source ID when performing memory accesses.
0x1D	Reserved	DCSR	The debug facilities are accessed through a DCSR mapped LAW. Whenever a Local Access Window targets DCSR space, the size of this space must always be set to 4 Mbytes.
0x1E	Reserved	Reserved	-
0x1F	Reserved	eLBC	-
0x20	PME	Reserved	-
0x21	Security 4.0	Reserved	-
...	Reserved	Reserved	-
0x3C	Queue Manager (control)	Queue Manager Software Portal	-
...	Reserved	Reserved	-
0x40	USB 1	Reserved	-
0x41	USB 2	Reserved	-
...	-	-	-
0x44	eSDHC	Reserved	-
...	Reserved	Reserved	-
0x48	Pre-boot loader (PBL)	Reserved	-
...	Reserved	Reserved	-
0x4B	Nexus Port Controller (NPC)	Reserved	-
...	Reserved	Reserved	-
0x5D	RapidIO Message Unit (RMU)	Reserved	-
...	Reserved	Reserved	-
0x70	DMA 1	Reserved	-
0x71	DMA 2	Reserved	-
...	Reserved	Reserved	-
0x80	Core 0 (instruction)	Reserved	-
0x81	Core 0 (data)	Reserved	-
0x82	Core 1 (instruction)	Reserved	-
0x83	Core 1 (data)	Reserved	-
0x84	Core 2 (instruction)	Reserved	-
0x85	Core 2 (data)	Reserved	-
0x86	Core 3 (instruction)	Reserved	-
0x87	Core 3 (data)	Reserved	-

Table continues on the next page...

Table 2-1. P4080 Global Source/Target ID Assignments (continued)

Source/Target ID Value	Transaction Source	Transaction Target	Notes
0x88	Core 4 (instruction)	Reserved	-
0x89	Core 4 (data)	Reserved	-
0x8A	Core 5 (instruction)	Reserved	-
0x8B	Core 5 (data)	Reserved	-
0x8C	Core 6 (instruction)	Reserved	-
0x8D	Core 6 (data)	Reserved	-
0x8E	Core 7 (instruction)	Reserved	-
0x8F	Core 7 (data)	Reserved	-
...	Reserved	Reserved	-
0xC0	Frame Manager 1 ID 1	Reserved	-
0xC1	Frame Manager 1 ID 2	Reserved	-
0xC2	Frame Manager 1 ID 3	Reserved	-
0xC3	Frame Manager 1 ID 4	Reserved	-
0xC4	Frame Manager 1 ID 5	Reserved	-
0xC5	Frame Manager 1 ID 6	Reserved	-
0xC6	Frame Manager 1 ID 7	Reserved	-
0xC7	Frame Manager 1 ID 8	Reserved	-
0xC8	Frame Manager 1 ID 9	Reserved	-
0xC9	Frame Manager 1 ID 10	Reserved	-
0xCA	Frame Manager 1 ID 11	Reserved	-
0xCB	Frame Manager 1 ID 12	Reserved	-
0xCC	Frame Manager 1 ID 13	Reserved	-
0xCD	Frame Manager 1 ID 14	Reserved	-
0xCE	Frame Manager 1 ID 15	Reserved	-
0xCF	Frame Manager 1 ID 16	Reserved	-
0xD0	Frame Manager 2 ID 1	Reserved	-
0xD1	Frame Manager 2 ID 2	Reserved	-
0xD2	Frame Manager 2 ID 3	Reserved	-
0xD3	Frame Manager 2 ID 4	Reserved	-
0xD4	Frame Manager 2 ID 5	Reserved	-
0xD5	Frame Manager 2 ID 6	Reserved	-
0xD6	Frame Manager 2 ID 7	Reserved	-
0xD7	Frame Manager 2 ID 8	Reserved	-
0xD8	Frame Manager 2 ID 9	Reserved	-
0xD9	Frame Manager 2 ID 10	Reserved	-
0xDA	Frame Manager 2 ID 11	Reserved	-
0xDB	Frame Manager 2 ID 12	Reserved	-
0xDC	Frame Manager 2 ID 13	Reserved	-
0xDD	Frame Manager 2 ID 14	Reserved	-

Table continues on the next page...

Table 2-1. P4080 Global Source/Target ID Assignments (continued)

Source/Target ID Value	Transaction Source	Transaction Target	Notes
0xDE	Frame Manager 2 ID 15	Reserved	-
0xDF	Frame Manager 2 ID 16	Reserved	-

2.3 Local Access Windows (LAWs)

The local address map is defined by a set of 32 local access windows (LAWs).

Each of these windows maps a programmable region of the local address space to a specified target interface, such as the DDR controller, or other targets. This allows the internal interconnections of the device to route a transaction from its source to the proper target.

Each LAW is defined by a pair of base address registers that specify the starting address for the window, and an attribute register that specifies whether the mapping is enabled, the size of the window, a coherency subdomain, and the target interface for that window. Note that the LAWs do not perform any address translation, and therefore there are no corresponding translation address registers. The local access window registers exist as part of the local access control block in the CCSR space.

With the exception of configuration space (mapped by CCSRBAR) and the boot window, all addresses used by the system must be mapped by a LAW. This includes addresses that are mapped by inbound ATMU windows. Thus, target mappings of the LAWs and the inbound ATMU windows must be consistent.

2.3.1 Precedence of Local Access Windows

If two or more LAWs overlap, the lower-numbered window takes precedence.

For example, consider two LAWs, set up as shown in [Table 2-2](#).

Table 2-2. Overlapping Local Access Windows

LAW	Base Address	Size	Target Interface
1	0_7FF0_0000h	1 MB	Enhanced local bus controller (eLBC)
2	0_0000_0000h	2 GB	Memory complex (DDR controller)

In this case, LAW 1 governs the mapping of the 1 MB region from 0_7FF0_0000h to 0_7FFF_FFFFh, even though the window described in LAW 2 also encompasses that memory region.

NOTE

The CCSR mapping, defined by the CCSRBARs, supersedes all local access window mappings.

NOTE

The boot window is another special area. In unsecured boot, the boot window has lower priority than the LAWs. However, secure boot, will preempt accesses to the boot window and send them to the internal secure boot code (ISBC). In this case, the secure boot effectively makes the boot window higher priority than the LAWs.

2.3.2 Configuring Local Access Windows

Once a local access window is enabled, it should not be modified while any device in the system may be using the window.

Neither should a new window be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last LAW configuration register before enabling any other devices to use the window. For example, if LAWs 0-3 are being configured in order during the initialization process, the last write (to LAWAR3) should be followed by a read of LAWAR3 before any devices try to use any of these windows. If the configuration is being performed by the core, the read of LAWAR3 should be followed by an **isync** instruction.

2.3.3 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the LAWs and the additional mapping functions that occur at the target interfaces.

The LAWs define how a transaction is routed through the device's internal interconnects from the transaction's source to its target. After the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR controller has chip select registers that map a memory request to a particular external device. Similarly, the enhanced local bus controller has base registers that

perform a similar function. The external peripheral interface controllers (for example, serial RapidIO and PCI Express) have outbound address translation and mapping units (ATMUs) that map the local address into an external address space.

These other mapping functions are configured by programming the CCSR_s of the individual interfaces. Note that there is no need to have a one-to-one correspondence between LAWs and chip select regions or outbound ATMU windows. A single LAW can be further decoded to any number of chip selects or to any number of outbound ATMU windows at the target interface.

2.3.4 SRAM Windows

The CoreNet platform cache (CPC) can be configured as a memory-mapped SRAM. The CPC SRAM control registers (CPCSRCRs) in the CPC cache controller set the base addresses and sizes for these windows. See [SRAM Mode Registers](#) for information about configuring SRAM windows.

In addition to defining the SRAM regions in the CPCSRCRs, the CPC SRAM regions need to be covered by LAWs using their associated memory complex as the target ID (TRGT_ID).

2.3.5 Local Address Map Example

[Figure 2-1](#) shows what a typical local address map might look like.

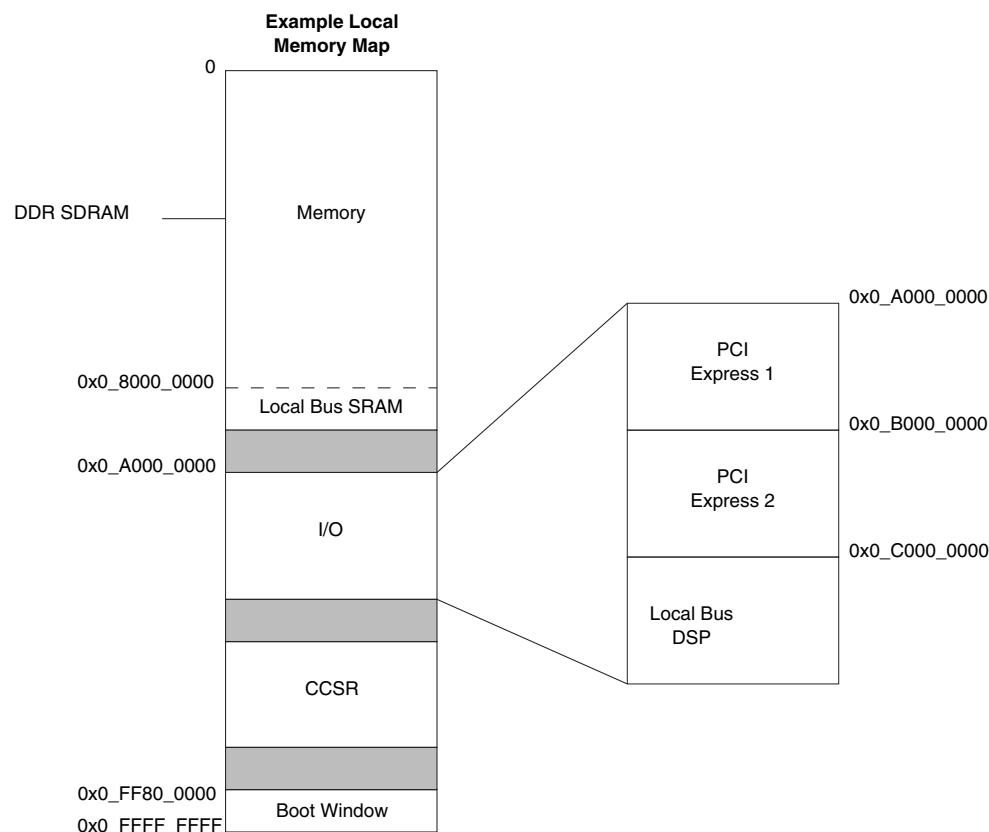


Figure 2-1. Local Address Map Example

This table shows the corresponding set of LAW settings for the example shown in [Figure 2-1](#).

Table 2-3. Local Access Window Settings Example

Window	Base Address	SIZE	Target Interface
0	0_0000_0000h	2 GB	DDR
1	0_8000_0000h	1 MB	Local bus controller (eLBC)-SRAM
2	0_A000_0000h	256 MB	PCI Express 1
3	0_B000_0000h	256 MB	PCI Express 2
4-9	Unused		

In this example, note that it is not required to define a LAW to describe the range of memory used for memory-mapped configuration, control, and status registers because these occupy a fixed 16 MB space pointed to by the CCSRBARs. See [Accessing Configuration, Control, and Status Registers](#). Also note that the boot window is always enabled and covers local addresses from 0_FF80_0000h to 0_FFFF_FFFFh.

2.4 Local Access Window (LAW) Memory Map

The table below shows the memory map for the LAW registers. The local access window registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR), plus the block base address, plus the offset of the specific register to be accessed. For the LAWs, the block base address is local access control block at 0x00_0000.

Note that all LAW registers should only be accessed a word (4 bytes) at a time.

LAW memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C00	LAWn base address register high (LAW_LAWBARH0)	32	R/W	0000_0000h	2.4.1/112
C04	LAWn base address register low (LAW_LAWBARL0)	32	R/W	0000_0000h	2.4.2/112
C08	LAWn attribute register (LAW_LAWAR0)	32	R/W	0000_0000h	2.4.3/113
C10	LAWn base address register high (LAW_LAWBARH1)	32	R/W	0000_0000h	2.4.1/112
C14	LAWn base address register low (LAW_LAWBARL1)	32	R/W	0000_0000h	2.4.2/112
C18	LAWn attribute register (LAW_LAWAR1)	32	R/W	0000_0000h	2.4.3/113
C20	LAWn base address register high (LAW_LAWBARH2)	32	R/W	0000_0000h	2.4.1/112
C24	LAWn base address register low (LAW_LAWBARL2)	32	R/W	0000_0000h	2.4.2/112
C28	LAWn attribute register (LAW_LAWAR2)	32	R/W	0000_0000h	2.4.3/113
C30	LAWn base address register high (LAW_LAWBARH3)	32	R/W	0000_0000h	2.4.1/112
C34	LAWn base address register low (LAW_LAWBARL3)	32	R/W	0000_0000h	2.4.2/112
C38	LAWn attribute register (LAW_LAWAR3)	32	R/W	0000_0000h	2.4.3/113
C40	LAWn base address register high (LAW_LAWBARH4)	32	R/W	0000_0000h	2.4.1/112
C44	LAWn base address register low (LAW_LAWBARL4)	32	R/W	0000_0000h	2.4.2/112
C48	LAWn attribute register (LAW_LAWAR4)	32	R/W	0000_0000h	2.4.3/113
C50	LAWn base address register high (LAW_LAWBARH5)	32	R/W	0000_0000h	2.4.1/112
C54	LAWn base address register low (LAW_LAWBARL5)	32	R/W	0000_0000h	2.4.2/112
C58	LAWn attribute register (LAW_LAWAR5)	32	R/W	0000_0000h	2.4.3/113
C60	LAWn base address register high (LAW_LAWBARH6)	32	R/W	0000_0000h	2.4.1/112
C64	LAWn base address register low (LAW_LAWBARL6)	32	R/W	0000_0000h	2.4.2/112
C68	LAWn attribute register (LAW_LAWAR6)	32	R/W	0000_0000h	2.4.3/113
C70	LAWn base address register high (LAW_LAWBARH7)	32	R/W	0000_0000h	2.4.1/112
C74	LAWn base address register low (LAW_LAWBARL7)	32	R/W	0000_0000h	2.4.2/112
C78	LAWn attribute register (LAW_LAWAR7)	32	R/W	0000_0000h	2.4.3/113
C80	LAWn base address register high (LAW_LAWBARH8)	32	R/W	0000_0000h	2.4.1/112

Table continues on the next page...

LAW memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C84	LAWn base address register low (LAW_LAWBARL8)	32	R/W	0000_0000h	2.4.2/112
C88	LAWn attribute register (LAW_LAWAR8)	32	R/W	0000_0000h	2.4.3/113
C90	LAWn base address register high (LAW_LAWBARH9)	32	R/W	0000_0000h	2.4.1/112
C94	LAWn base address register low (LAW_LAWBARL9)	32	R/W	0000_0000h	2.4.2/112
C98	LAWn attribute register (LAW_LAWAR9)	32	R/W	0000_0000h	2.4.3/113
CA0	LAWn base address register high (LAW_LAWBARH10)	32	R/W	0000_0000h	2.4.1/112
CA4	LAWn base address register low (LAW_LAWBARL10)	32	R/W	0000_0000h	2.4.2/112
CA8	LAWn attribute register (LAW_LAWAR10)	32	R/W	0000_0000h	2.4.3/113
CB0	LAWn base address register high (LAW_LAWBARH11)	32	R/W	0000_0000h	2.4.1/112
CB4	LAWn base address register low (LAW_LAWBARL11)	32	R/W	0000_0000h	2.4.2/112
CB8	LAWn attribute register (LAW_LAWAR11)	32	R/W	0000_0000h	2.4.3/113
CC0	LAWn base address register high (LAW_LAWBARH12)	32	R/W	0000_0000h	2.4.1/112
CC4	LAWn base address register low (LAW_LAWBARL12)	32	R/W	0000_0000h	2.4.2/112
CC8	LAWn attribute register (LAW_LAWAR12)	32	R/W	0000_0000h	2.4.3/113
CD0	LAWn base address register high (LAW_LAWBARH13)	32	R/W	0000_0000h	2.4.1/112
CD4	LAWn base address register low (LAW_LAWBARL13)	32	R/W	0000_0000h	2.4.2/112
CD8	LAWn attribute register (LAW_LAWAR13)	32	R/W	0000_0000h	2.4.3/113
CE0	LAWn base address register high (LAW_LAWBARH14)	32	R/W	0000_0000h	2.4.1/112
CE4	LAWn base address register low (LAW_LAWBARL14)	32	R/W	0000_0000h	2.4.2/112
CE8	LAWn attribute register (LAW_LAWAR14)	32	R/W	0000_0000h	2.4.3/113
CF0	LAWn base address register high (LAW_LAWBARH15)	32	R/W	0000_0000h	2.4.1/112
CF4	LAWn base address register low (LAW_LAWBARL15)	32	R/W	0000_0000h	2.4.2/112
CF8	LAWn attribute register (LAW_LAWAR15)	32	R/W	0000_0000h	2.4.3/113
D00	LAWn base address register high (LAW_LAWBARH16)	32	R/W	0000_0000h	2.4.1/112
D04	LAWn base address register low (LAW_LAWBARL16)	32	R/W	0000_0000h	2.4.2/112
D08	LAWn attribute register (LAW_LAWAR16)	32	R/W	0000_0000h	2.4.3/113
D10	LAWn base address register high (LAW_LAWBARH17)	32	R/W	0000_0000h	2.4.1/112
D14	LAWn base address register low (LAW_LAWBARL17)	32	R/W	0000_0000h	2.4.2/112
D18	LAWn attribute register (LAW_LAWAR17)	32	R/W	0000_0000h	2.4.3/113
D20	LAWn base address register high (LAW_LAWBARH18)	32	R/W	0000_0000h	2.4.1/112
D24	LAWn base address register low (LAW_LAWBARL18)	32	R/W	0000_0000h	2.4.2/112
D28	LAWn attribute register (LAW_LAWAR18)	32	R/W	0000_0000h	2.4.3/113
D30	LAWn base address register high (LAW_LAWBARH19)	32	R/W	0000_0000h	2.4.1/112
D34	LAWn base address register low (LAW_LAWBARL19)	32	R/W	0000_0000h	2.4.2/112
D38	LAWn attribute register (LAW_LAWAR19)	32	R/W	0000_0000h	2.4.3/113
D40	LAWn base address register high (LAW_LAWBARH20)	32	R/W	0000_0000h	2.4.1/112
D44	LAWn base address register low (LAW_LAWBARL20)	32	R/W	0000_0000h	2.4.2/112
D48	LAWn attribute register (LAW_LAWAR20)	32	R/W	0000_0000h	2.4.3/113

Table continues on the next page...

LAW memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D50	LAWn base address register high (LAW_LAWBARH21)	32	R/W	0000_0000h	2.4.1/112
D54	LAWn base address register low (LAW_LAWBARL21)	32	R/W	0000_0000h	2.4.2/112
D58	LAWn attribute register (LAW_LAWAR21)	32	R/W	0000_0000h	2.4.3/113
D60	LAWn base address register high (LAW_LAWBARH22)	32	R/W	0000_0000h	2.4.1/112
D64	LAWn base address register low (LAW_LAWBARL22)	32	R/W	0000_0000h	2.4.2/112
D68	LAWn attribute register (LAW_LAWAR22)	32	R/W	0000_0000h	2.4.3/113
D70	LAWn base address register high (LAW_LAWBARH23)	32	R/W	0000_0000h	2.4.1/112
D74	LAWn base address register low (LAW_LAWBARL23)	32	R/W	0000_0000h	2.4.2/112
D78	LAWn attribute register (LAW_LAWAR23)	32	R/W	0000_0000h	2.4.3/113
D80	LAWn base address register high (LAW_LAWBARH24)	32	R/W	0000_0000h	2.4.1/112
D84	LAWn base address register low (LAW_LAWBARL24)	32	R/W	0000_0000h	2.4.2/112
D88	LAWn attribute register (LAW_LAWAR24)	32	R/W	0000_0000h	2.4.3/113
D90	LAWn base address register high (LAW_LAWBARH25)	32	R/W	0000_0000h	2.4.1/112
D94	LAWn base address register low (LAW_LAWBARL25)	32	R/W	0000_0000h	2.4.2/112
D98	LAWn attribute register (LAW_LAWAR25)	32	R/W	0000_0000h	2.4.3/113
DA0	LAWn base address register high (LAW_LAWBARH26)	32	R/W	0000_0000h	2.4.1/112
DA4	LAWn base address register low (LAW_LAWBARL26)	32	R/W	0000_0000h	2.4.2/112
DA8	LAWn attribute register (LAW_LAWAR26)	32	R/W	0000_0000h	2.4.3/113
DB0	LAWn base address register high (LAW_LAWBARH27)	32	R/W	0000_0000h	2.4.1/112
DB4	LAWn base address register low (LAW_LAWBARL27)	32	R/W	0000_0000h	2.4.2/112
DB8	LAWn attribute register (LAW_LAWAR27)	32	R/W	0000_0000h	2.4.3/113
DC0	LAWn base address register high (LAW_LAWBARH28)	32	R/W	0000_0000h	2.4.1/112
DC4	LAWn base address register low (LAW_LAWBARL28)	32	R/W	0000_0000h	2.4.2/112
DC8	LAWn attribute register (LAW_LAWAR28)	32	R/W	0000_0000h	2.4.3/113
DD0	LAWn base address register high (LAW_LAWBARH29)	32	R/W	0000_0000h	2.4.1/112
DD4	LAWn base address register low (LAW_LAWBARL29)	32	R/W	0000_0000h	2.4.2/112
DD8	LAWn attribute register (LAW_LAWAR29)	32	R/W	0000_0000h	2.4.3/113
DE0	LAWn base address register high (LAW_LAWBARH30)	32	R/W	0000_0000h	2.4.1/112
DE4	LAWn base address register low (LAW_LAWBARL30)	32	R/W	0000_0000h	2.4.2/112
DE8	LAWn attribute register (LAW_LAWAR30)	32	R/W	0000_0000h	2.4.3/113
DF0	LAWn base address register high (LAW_LAWBARH31)	32	R/W	0000_0000h	2.4.1/112
DF4	LAWn base address register low (LAW_LAWBARL31)	32	R/W	0000_0000h	2.4.2/112
DF8	LAWn attribute register (LAW_LAWAR31)	32	R/W	0000_0000h	2.4.3/113

2.4.1 LAWn base address register high (LAW LAWBARHn)

Address: 0h base + C00h offset + (16d × i), where i=0d to 31d

LAW LAWBARH n field descriptions

Field	Description
0-27 -	This field is reserved. Reserved
28-31 BASE_ADDR_ HIGH	Identifies bits 0-3 of the 36-bit base address of local access window n . The specified base address should be aligned to the window size, as defined by LAWAR n [SIZE].

2.4.2 LAWn base address register low (LAW LAWBARLn)

Address: 0h base + C04h offset + (16d × i), where i=0d to 31d

LAW_LAWBARLn field descriptions

Field	Description
0–19 BASE_ADDR_LOW	Identifies bits 4–23 of the 36-bit base address of local access window n . The specified base address should be aligned to the window size, as defined by LAWAR n [SIZE].
20–31 -	This field is reserved. Write reserved, read = 0. Because the minimum window size is 4 KB, the 12 least significant bits are treated as zeros.

2.4.3 LAWn attribute register (LAW_LAWARn)

Address: 0h base + C08h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved			TRGT_ID						CSD_ID					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSD_ID			Reserved						SIZE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LAW_LAWARn field descriptions

Field	Description
0 EN	Enable 0 The local access window n (and all other LAWAR n and LAWBAR n fields) are disabled. 1 The local access window n is enabled and other LAWAR n and LAWBAR n fields combine to identify an address range for this window.
1–3 -	This field is reserved. Write reserved, read = 0
4–11 TRGT_ID	Target identifier. Identifies the target interface when a transaction hits in the address range defined by this window. See Global Source and Target IDs for the defined encodings. NOTE: CPC SRAM regions are generally mapped by LAWs using their associated memory complex target ID. However, these regions also need to be defined by the CPC SRAM control registers (CPCSRCRs) in the CPC block. NOTE: Configuration registers are mapped by the windows defined by the CCSRBARs. These mappings supersede local access window mappings, so configuration registers do not appear as a target for local access windows. NOTE: Whenever a Local Access Window targets DCSR space, the size of this space must always be set to 4 MB. NOTE: TRGT_ID = 0x0F (local space) is an illegal encoding in the LAWARn registers and causes a local access error (CEDR[LAE]) if detected.
12–19 CSD_ID	Coherency subdomain identifier. Identifies a group of one or more partitions that have access to the address space defined by this access window. Each such group may contain one or more processors and/or other caching devices representing a coherency subdomain within the system. For a coherent access via this window, coherency is enforced over this subdomain.
20–25 -	This field is reserved. Write reserved, read = 0
26–31 SIZE	Identifies the size of the window from the starting address. Window size is $2^{(\text{SIZE}+1)}$ bytes. Example values are as follows: 000000-001010 Reserved 001011 4 KB 001100 8 KB 100010 32 GB 100011 64 GB

Table continues on the next page...

LAW_LAWARn field descriptions (continued)

Field	Description
100100-111111	... Reserved

2.5 Address Translation and Mapping Units

To facilitate flexibility in defining the address maps for the external interfaces (serial RapidIO and PCI Express), the device provides address translation and mapping units (ATMUs).

The following types of translation and mapping operations are performed by the ATMUs:

- Translating the local address to an external address space
- Translating external addresses to the local address space
- Assigning attributes to transactions
- Mapping a local address to a target interface

Outbound address translation and mapping refers to the translation of addresses from the local address space to the external address space and attributes of a particular I/O interface.

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface to the local address space understood by the internal interfaces of this device. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a function similar to that of the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the LAWs. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a LAW must also map that same local address to the same target.

2.5.1 Address Translation

All of the configuration registers that define an ATMU window's translation and mapping functions follow the same general register format.

The register format is summarized in [Table 2-103](#).

Table 2-103. Format of ATMU Window Definitions

Register	Function
Translation address (TAR/TEAR)	High-order address bits defining location of the window in the target address space
Base address (BAR/BEAR)	High-order address bits defining location of the window in the initiator address space
Window attributes (WAR)	Window enable, window size, target interface, and transaction attributes

The size of the windows must be a power-of-two. To perform a translation or mapping function, the address of the transaction is compared with the base address register of each window. The number of bits used in the comparison is dictated by each window's size attribute. When an address hits a window, if address translation is being performed, the new translated address is created by concatenating the window offset to the translation address. Again, the window's size attribute dictates how many bits are translated.

2.5.2 Outbound ATMUs

If the target mapping performed by the local access windows directs the transaction to one of the external peripheral interfaces (as an outbound read or write), the transaction is then mapped into that interface's external address space by outbound ATMUs associated with the external interface.

The outbound ATMUs perform the mapping from the local address space to the address space of the external peripheral interface, which may be much larger than the local space.

The outbound ATMUs also map attributes such as the transaction type and priority level.

The serial RapidIO controller has eight outbound ATMU windows plus a default window. Each PCI Express controller has four outbound ATMU windows plus a default window. If a transaction's address does not hit any of the outbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. The outbound ATMU registers include extended translation address registers so that up to 64 bits of external address space can be supported.

2.5.3 Inbound ATMUs

The inbound ATMUs perform the address translation from the external address spaces to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface.

The serial RapidIO controller has four inbound ATMU windows plus a default window. If the inbound transaction's address does not hit any of the inbound ATMU windows, the translation actions defined by the default window are used.

Each PCI Express controller has three inbound ATMU windows, a fixed translation window to the CCSR space, and a fixed MSI ATMU window for mapping inbound message-signalled interrupt transactions to the interrupt controller (MPIC).

2.5.3.1 Illegal Interaction Between Inbound ATMUs and LAWs

Since both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another.

For example, it is considered a programming error to have an inbound ATMU map a transaction target to the local memory space if the resulting translated local address is mapped to an external peripheral interface by a local access window. Such programming errors may result in unpredictable system deadlocks.

2.6 Configuration, Control, and Status Register (CCSR) Space

To allow for flexibility, the configuration, control, and status register space is relocatable in the local address space.

The local address map location of this register block is controlled by the configuration, control, and status registers base address register (CCSRBAR), see [Accessing Configuration, Control, and Status Registers](#).

The default value for CCSRBAR is 0_FE00_0000h.

2.6.1 Accessing CCSR Memory from the Local Processor

When the local processor cores are used to configure CCSR space, the CCSR memory space should typically be marked as cache-inhibited and guarded.

In addition, many configuration registers affect accesses to other memory regions; therefore writes to these registers must be guaranteed to have taken effect before accesses are made to the associated memory regions.

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be immediately followed by a read of the same register, and that should be followed by a SYNC instruction. Then accesses can safely be made to memory regions affected by the configuration register write.

2.6.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the processor cores, the configuration, control, and status registers are accessible from external interfaces.

This allows external masters on the I/O ports to configure the device.

External masters do not need to know the location of the CCSR memory in the local address map. Rather, they access this region of the local memory map through a window defined by a register in the interface programming model that is accessible to the external master from its external memory map.

The PCI Express controller's base address for accessing the local CCSR memory is selectable through the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10, described in [PCI Express Base Address Register 0 \(PEXCSRBAR\)](#). An external PCI Express root complex host sets this register by running a PCI Express configuration transaction. Subsequent memory accesses by a PCI Express master to the PCI Express address range indicated by PEXCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

The serial RapidIO base address for accessing the local CCSR memory is selectable through the serial RapidIO LCSBA1CSR, defined in the RapidIO programming model. See [Local configuration space base address 1 command and status register](#). An external serial RapidIO master can set the value of LCSBA1CSR with a maintenance packet. Then subsequent read and write packets whose RapidIO addresses match the window defined by LCSBA1CSR are translated to the local address range indicated by CCSRBAR.

2.6.3 Accessing Reserved Registers and Bits

Reserved registers and bits in the CCSR memory space are not guaranteed to have predictable values.

In general, reads of reserved bits return zeros, but software should not rely on the value of any reserved bit being a zero or remaining consistent.

Unless otherwise specified, when writing registers in CCSR memory space, reserved bits should be written with zeros.

Writing ones to a reserved bit may result in undefined operation.

2.6.4 Organization of CCSR Memory

The configuration, control, and status registers are grouped according to functional units.

Most functional units are allocated a 4 KB address space for registers.

Some functional units have larger address spaces as defined by their programming models. The registers for these blocks are given their own larger regions of CCSR memory.

2.6.5 CCSR Address Map

The full register address of any CCSR is comprised of the CCSR window base address, specified in CCSRBAR (default address 0x0_FE00_0000), plus the functional block base address, plus the specific register's offset within that block.

Table 2-104 shows the location of the functional block base addresses for the entire CCSR space. Cross-references are provided to the CCSR maps for each individual block.

Table 2-104. CCSR Block Base Address Map

Block Base Address (Hex)	Block	Section	Comments
0x00_0000	Local access control-Local configuration control	Local Configuration Control Memory Map	-Local configuration control starts at offset 0h
	Local access control-Local access windows	Local Access Window (LAW) Memory Map	-Local Access Window (LAWs) start at offset C00h
0x00_1000-0x00_7FFF	Reserved	-	-
0x00_8000	DDR memory controller 1	DDR Memory Controller Memory Map	-
0x00_9000	DDR memory controller 2	DDR Memory Controller Memory Map	-
0x00_A000-0x00_FFFF	Reserved	-	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x01_0000	CoreNet platform cache 1 (CPC1)	CoreNet Platform Cache (CPC) Memory Map	-
0x01_1000	CoreNet platform cache 2 (CPC2)	CoreNet Platform Cache (CPC) Memory Map	-
0x01_2000-0x01_7FFF	Reserved	-	-
0x01_8000	CoreNet coherency fabric (CCF)	CoreNet Coherency Fabric (CCF) Memory Map	-
0x01_9000-0x01_FFFF	Reserved	-	-
0x02_0000	PAMU partition 1	PAMU Memory Map	The PAMU is partitioned into 16 identical instances. Not all are necessarily backed with physical hardware. However, all of them must be programmed identically or undefined behavior may result.
0x02_1000	PAMU partition 2	PAMU Memory Map	
0x02_2000	PAMU partition 3	PAMU Memory Map	
0x02_3000	PAMU partition 4	PAMU Memory Map	
0x02_4000	PAMU partition 5	PAMU Memory Map	
0x02_5000	PAMU partition 6	PAMU Memory Map	
0x02_6000	PAMU partition 7	PAMU Memory Map	
0x02_7000	PAMU partition 8	PAMU Memory Map	
0x02_8000	PAMU partition 9	PAMU Memory Map	
0x02_9000	PAMU partition 10	PAMU Memory Map	
0x02_A000	PAMU partition 11	PAMU Memory Map	
0x02_B000	PAMU partition 12	PAMU Memory Map	
0x02_C000	PAMU partition 13	PAMU Memory Map	
0x02_D000	PAMU partition 14	PAMU Memory Map	
0x02_E000	PAMU partition 15	PAMU Memory Map	
0x02_F000	PAMU partition 16	PAMU Memory Map	

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x03_0000-0x03_FFFF	Reserved	-	-
0x04_0000	PIC-Global registers	MPIC Memory Map	Gbl config: 0x04_1000 Gbl timers: 0x04_1100
0x05_0000	PIC-Interrupt source registers	MPIC Memory Map	External IRQs: 0x05_0000 Internal IRQs: 0x05_1200
0x06_0000	PIC-Processor (core) registers	MPIC Memory Map	-
0x07_0000-0x0B_FFFF	Reserved	-	-
0x0C_0000	RapidIO-Architectural registers	Serial RapidIO Memory Map/ Register Definition	-
0x0D_0000	RapidIO-Implementation registers	Serial RapidIO Memory Map/ Register Definition	-
0x0E_0000	Configuration/pin control	Device Configuration and Pin Control Memory Map/ Register Definition	-
0x0E_1000	Clocking	Clocking Memory Map/ Register Definition	-
0x0E_2000	Run control/power management (RCPM)	RCPM Memory Map/Register Definition	-
0x0E_3000-0x0E_7FFF	Reserved	-	-
0x0E_8000	Security fuse processor (SFP)	Security fuse processor (SFP) memory map	-
0x0E_9000-0x0E_9FFF	Reserved	-	-
0x0E_A000	SerDes control	SerDes Control Memory Map	-
0x0E_B000-0x0F_FFFF	Reserved	-	-
0x10_0000	DMA controller 1	DMA controller memory map	-
0x10_1000	DMA controller 2	DMA controller memory map	-
0x10_2000-0x10_FFFF	Reserved	-	-
0x11_0000	Enhanced serial peripheral interface (eSPI)	Enhanced serial peripheral interface (eSPI) memory map	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x11_1000-0x11_3FFF	Reserved	-	-
0x11_4000	Enhanced secure digital high capacity (eSDHC)	Enhanced Secure Digital Host Controller (eSDHC) Memory Map	-
0x11_5000-0x11_7FFF	Reserved	-	-
0x11_8000	Dual I2C controller 1	I2C Controller Memory Map	I ² C 1: 0x11_8000 I ² C 2: 0x11_8100
0x11_9000	Dual I2C controller 2	I2C Controller Memory Map	I ² C 3: 0x11_9000 I ² C 4: 0x11_9100
0x11_A000-0x11_BFFF	Reserved	-	-
0x11_C000	DUART controller 1	DUART Memory Map/Register Definition	UART1: 0x11_C500 (DUART1) UART2: 0x11_C600 (DUART1)
0x11_D000	DUART controller 2	DUART Memory Map/Register Definition	UART3: 0x11_D500 (DUART2) UART4: 0x11_D600 (DUART2)
0x11_E000-0x12_3FFF	Reserved	-	-
0x12_4000	Enhanced local bus controller (eLBC)	Enhanced Local Bus Controller (eLBC) Memory Map	-
0x12_5000-0x12_FFFF	Reserved	-	-
0x13_0000	GPIO controller	GPIO Memory Map/Register Definition	-
0x13_1000-0x13_7FFF	Reserved	-	-
0x13_8000	Pre-boot loader (PBL)	Reserved Address Space Used as Internal PBL Commands	Software cannot write to the PBL CCSR space directly. However, special PBL commands may be leveraged during pre-boot initialization by referencing specific CCSR offsets (unique commands have unique CCSR offsets). See Reserved Address Space Used as Internal PBL Commands , for more information.
0x13_9000-0x1F_FFFF	Reserved	-	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x20_0000	PCI Express controller 1	PCI Express memory-mapped registers	-
0x20_1000	PCI Express controller 2	PCI Express memory-mapped registers	-
0x20_2000	PCI Express controller 3	PCI Express memory-mapped registers	-
0x20_3000-0x20_FFFF	Reserved	-	-
0x21_0000	USB 1 (host only)	USB Memory Map	-
0x21_1000	USB 2 (dual role)	USB Memory Map	-
0x21_2000-0x2F_FFFF	Reserved	-	-
0x30_0000	SEC 4.0	-	See the Security (SEC 4.0) reference manual.
0x31_0000-0x31_3FFF	Reserved	-	-
0x31_4000	Security monitor	Security Monitor memory map/ register definition	
0x31_5000-0x31_5FFF	Reserved	-	-
0x31_6000	Pattern match engine (PME)	-	See the DPAA reference manual.
0x31_7000-0x31_7FFF	Reserved	-	-
0x31_8000	Queue manager (QMAN)	-	See the DPAA reference manual.
0x31_9000-0x31_9FFF	Reserved	-	-
0x31_A000	Buffer manager (BMAN)	-	See the DPAA reference manual.
0x31_B000-0x3F_FFFF	Reserved	-	-
0x40_0000	Frame manager 1	-	See the DPAA reference manual.
0x50_0000	Frame manager 2	-	See the DPAA reference manual.
0x60_0000-0xFF_FFFF	Reserved	-	-

Chapter 3

Signal Descriptions

3.1 Signals Introduction

This chapter describes the P4080 external signals and is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions
- List of reset configuration signals
- Signal multiplexing details
- List of output signal states at reset

NOTE

A trailing "_B" or a bar over a signal name indicates that the signal is active low, such as IRQ_OUT_B or IRQ_OUT (interrupt output). Active-low signals are referred to as asserted (active) when they are low and negated when they are high.

Signals that are not active low, such as IRQ (interrupt input), are referred to as asserted when they are high and negated when they are low.

NOTE

Internal signals are shown throughout this document as lower case and in italics. For example, *sys_logic_clk* is an internal signal. These are discussed only as necessary for understanding the external functionality of the device.

3.2 Signals Overview

The chip's signals are grouped as follows:

- DDR memory controller interface signals
- I²C interface signals

- Enhanced local bus interface signals
- Enhanced SPI interface signals
- Enhanced SDHC interface signals
- USB controller interface signals
- DUART interface signals
- Hi-speed serial interface signals (XAUI, SGMII, serial RapidIO, PCI Express)
- DMA controller interface signals
- dTSEC interface signals
- IEEE 1588 timestamp signals
- Ethernet management interface signals
- Ethernet clocking signals
- PIC interface signals
- System control, general-purpose input/output, power management, and debug signals
- Test, JTAG, configuration, and clock signals

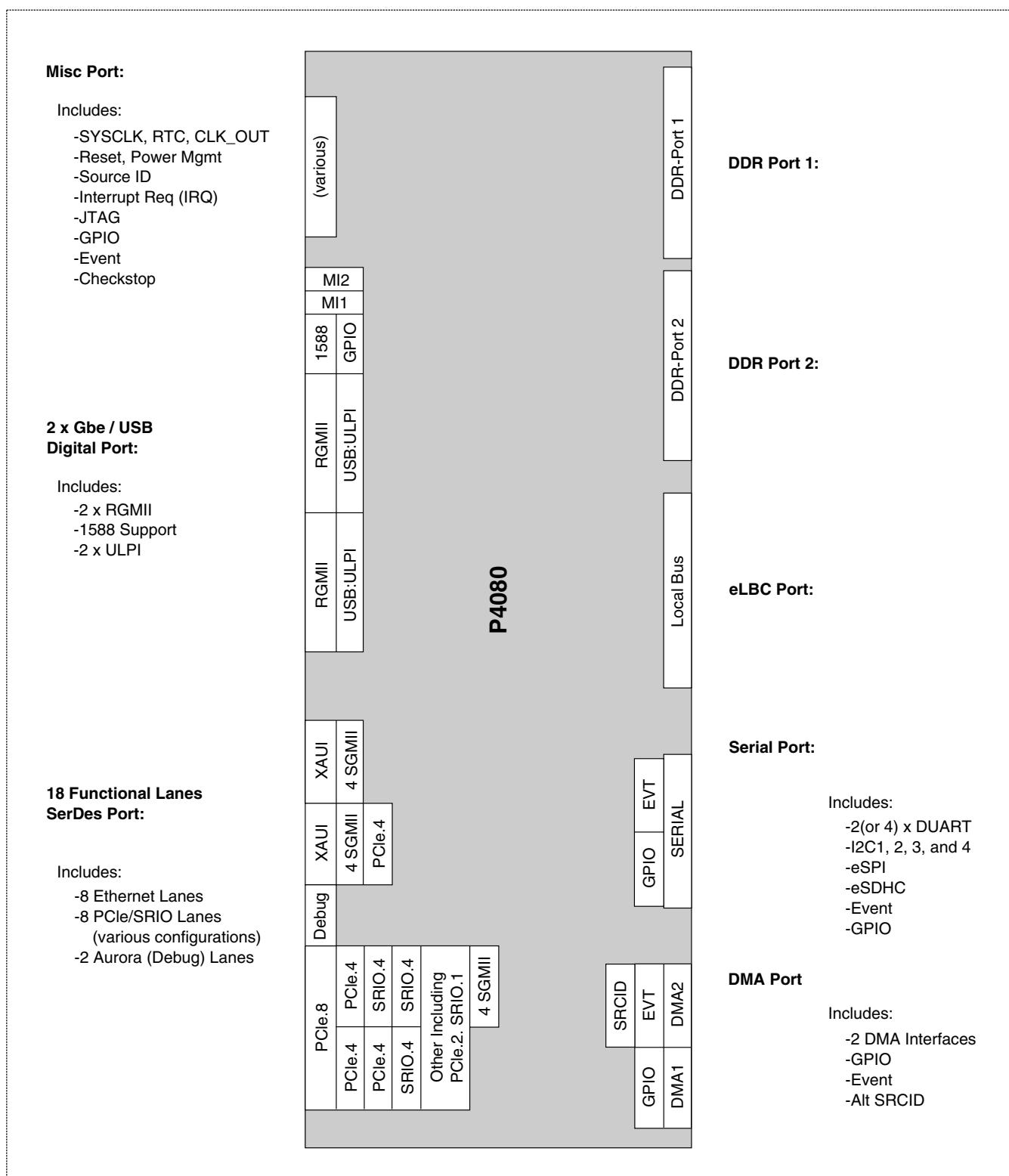


Figure 3-1. External Signal Groupings for the P4080

Note that individual chapters of this document provide details for each signal, describing each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

The following tables provides a summary of the signals grouped by function. This table details the signal name, interface, alternate functions, number of signals, and whether the signal is an input, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the tables provide a pointer to the table where the signal function is described.

Table 3-1. P4080 Signal Reference by Functional Block

Name	Description	Alternate Function(s)	No. of Signals	I/O
DDR Controller 1				
See DDR Signals Overview for more details				
D1_MDQ[0:63]	DDR controller data	-	64	I/O
D1_MECC[0:7]	DDR error correcting code	-	8	I/O
D1_MAPAR_ERR_B	Address parity error	-	1	I
D1_MAPAR_OUT	Address parity out	-	1	O
D1_MDM[0:8]	DDR data mask	-	9	O
D1_MDQS[0:8]	DDR data strobe	-	9	I/O
D1_MDQS[0:8]_B	DDR data strobe (complement)	-	9	I/O
D1_MBA[0:2]	DDR bank select	-	3	O
D1_MA[0:15]	DDR address	-	16	O
D1_MWE_B	DDR write enable	-	1	O
D1_MRAS_B	DDR row address strobe	-	1	O
D1_MCAS_B	DDR column address strobe	-	1	O
D1_MCS[0:3]_B	DDR chip select (2/DIMM)	-	4	O
D1_MCKE[0:3]	DDR clock enable	-	4	O
D1_MCK[0:5], D1_MCK[0:5]_B	DDR differential clocks (3 pairs/DIMM)	-	12	O
D1_MODT[0:3]	DRAM On-Die Termination	-	4	O
D1_MDIC[0:1]	Driver impedance calibration	-	2	I/O
DDR Controller 2				
See DDR Signals Overview for more details				
D2_MDQ[0:63]	DDR controller data	-	64	I/O
D2_MECC[0:7]	DDR error correcting code	-	8	I/O
D2_MAPAR_ERR_B	Address parity error	-	1	I
D2_MAPAR_OUT	Address parity out	-	1	O
D2_MDM[0:8]	DDR data mask	-	9	O
D2_MDQS[0:8]	DDR data strobe	-	9	I/O
D2_MDQS[0:8]_B	DDR data strobe (complement)	-	9	I/O
D2_MBA[0:2]	DDR bank select	-	3	O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
D2_MA[0:15]	DDR address	-	16	O
D2_MWE_B	DDR write enable	-	1	O
D2_MRAS_B	DDR row address strobe	-	1	O
D2_MCAS_B	DDR column address strobe	-	1	O
D2_MCS[0:3]_B	DDR chip select (2/DIMM)	-	4	O
D2_MCKE[0:3]	DDR clock enable	-	4	O
D2_MCK[0:5], D2_MCK[0:5]_B	DDR differential clocks (3 pairs/DIMM)	-	12	O
D2_MODT[0:3]	DRAM On-Die Termination	-	4	O
D2_MDIC[0:1]	Driver impedance calibration	-	2	I/O
I ² C 1				
See I²C Detailed Signal Descriptions for more details				
IIC1_SCL	I ² C serial clock	-	1	I/O
IIC1_SDA	I ² C serial data	-	1	I/O
I ² C 2				
See I²C Detailed Signal Descriptions for more details				
IIC2_SCL	I ² C serial clock	-	1	I/O
IIC2_SDA	I ² C serial data	-	1	I/O
I ² C 3				
See I²C Detailed Signal Descriptions for more details				
IIC3_SCL	I ² C serial clock	SDHC_CD_B/ GPIO[16]	1	I/O
IIC3_SDA	I ² C serial data	SDHC_WP/ GPIO[17]	1	I/O
I ² C 4				
See I²C Detailed Signal Descriptions for more details				
IIC4_SCL	I ² C serial clock	EVT[5]_B	1	I/O
IIC4_SDA	I ² C serial data	EVT[6]_B	1	I/O
eLBC				
See eLBC external signal descriptions for more details				
LAD[0:15]	Local bus address/data	<i>cfg_gpinput[0:15]</i>	16	I/O
LDP[0:1]	Local bus data parity	-	2	I/O
LA[16]	Local bus port address	-	1	O
LA[17]	Local bus port address	-	1	O
LA[18]	Local bus port address	-	1	O
LA[19]	Local bus port address	-	1	O
LA[20]	Local bus port address	-	1	O
LA[21]	Local bus port address	-	1	O
LA[22]	Local bus port address	-	1	O
LA[23]	Local bus port address	<i>cfg_elbc_ecc</i>	1	O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
LA[24]	Local bus port address	<i>cfg_dram_type</i>	1	O
LA[25]	Local bus port address	-	1	O
LA[26]	Local bus port address	-	1	O
LA[27]	Local bus port address	-	1	O
LA[28]	Local bus port address	-	1	O
LA[29]	Local bus port address	-	1	O
LA[30]	Local bus port address	-	1	O
LA[31]	Local bus port address	-	1	O
LCS[0:7]_B	Local bus chip select	-	8	O
LWE0_B/LBS0_B/ LFWE_B	Local bus write enable/byte select/flash write enable	-	1	O
LWE1_B/LBS1_B	Local bus write enable/byte select	-	1	O
LBCTL	Local bus data buffer control	-	1	O
LALE	Local bus address latch enable	-	1	O
LGPL[0]/LFCLE	Local bus UPM general purpose line/flash command latch enable	<i>cfg_rcw_src[0]</i>	1	O
LGPL[1]/LFALE	Local bus GP line/flash address latch enable	<i>cfg_rcw_src[1]</i>	1	O
LGPL[2]/LOE_B/ LFRE_B	Local bus GP line/output enable/flash read enable	<i>cfg_rcw_src[2]</i>	1	O
LGPL[3]/LFWP_B	Local bus GP line/flash write protect	<i>cfg_rcw_src[3]</i>	1	O
LGPL[4]/LGTA_B/ LUPWAIT/LPBSE/ LFRB	Local bus GP line/GPCM terminate access/UPM wait/parity byte select/flash ready-busy	-	1	O
LGPL[5]	Local bus GP line	<i>cfg_rcw_src[4]</i>	1	O
LCLK[0:1]	Local bus clock	-	2	O
eSPI				
See eSPI detailed signal descriptions for more details				
SPI_MOSI	SPI master out slave in	-	1	I/O
SPI_MISO	SPI master in slave out	-	1	I/O
SPI_CLK	SPI clock	-	1	O
SPI_CS[0:3]_B	SPI chip select	SDHC_DAT[4:7]	4	O
eSDHC				
See eSDHC external signal description for more details				
SDHC_CMD	SDHC command response	-	1	I/O
SDHC_DAT[0:3]	SDHC data	-	4	I/O
SDHC_DAT[4:7]	SDHC data	SPI_CS[0:3]_B	4	I/O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
SDHC_CLK	SDHC clock	-	1	I/O
SDHC_CD_B	SDHC card detect	IIC3_SCL/ GPIO[16]	1	I
SDHC_WP	SDHC write protect detect	IIC3_SDA/ GPIO[17]	1	I
USB				
See USB External Signals for more details				
USB1_D[7:4]	USB data	EC1_TXD[3:0]	4	I/O
USB1_STP	USB stop	EC1_TX_EN	1	O
USB1_CLK	USB clock	EC1_GTX_CLK	1	I
USB1_D[3:0]	USB data	EC1_RXD[3:0]	1	I/O
USB1_NXT	USB next	EC1_RX_DV	1	I
USB1_DIR	USB direction	EC1_RX_CLK	1	I
USB2_D[7:4]	USB data	EC2_TXD[3:0]	4	I/O
USB2_STP	USB stop	EC2_TX_EN	1	O
USB2_CLK	USB clock	EC2_GTX_CLK	1	I
USB2_D[3:0]	USB data	EC2_RXD[3:0]	1	I/O
USB2_NXT	USB next	EC2_RX_DV	1	I
USB2_DIR	USB direction	EC2_RX_CLK	1	I
DUART				
See DUART External Signal Descriptions for more details				
UART1_SOUT	DUART serial data out	GPIO[8]	1	O
UART2_SOUT	DUART serial data out	GPIO[9]	1	O
UART1_SIN	DUART serial data in	GPIO[10]	1	I
UART2_SIN	DUART serial data in	GPIO[11]	1	I
UART1_RTS_B/ UART3_SOUT	DUART ready to send/ serial data out	GPIO[12]	1	O
UART2_RTS_B/ UART4_SOUT	DUART ready to send/ serial data out	GPIO[13]	1	O
UART1_CTS_B/ UART3_SIN	DUART clear to send/serial data in	GPIO[14]	1	I
UART2_CTS_B/ UART4_SIN	DUART clear to send/serial data in	GPIO[15]	1	I
SerDes				
See PCI Express signal descriptions for more details				
SD_TX[0:17]	Transmit data	-	18	O
SD_TX[0:17]_B	Transmit data (complement)	-	18	O
SD_RX[0:17]	Receive data	-	18	I
SD_RX[0:17]_B	Receive data (complement)	-	18	I
SD_REF_CLK1	SerDes Bank 1 PLL reference clock	-	1	I

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
SD_REF_CLK1_B	SerDes Bank 1 PLL reference clock (complement)	-	1	I
SD_REF_CLK2	SerDes Bank 2 reference clock	-	1	I
SD_REF_CLK2_B	SerDes Bank 2 reference clock (complement)	-	1	I
SD_REF_CLK3	SerDes Bank 2 and 3 PLL reference clock	-	1	I
SD_REF_CLK3_B	SerDes Bank 2 and 3 PLL reference clock (complement)	-	1	I
DMA 1				
See DMA signal descriptions for more details				
DMA1_DREQ0_B	DMA1 channel 0 request	GPIO[18]	1	I
DMA1_DACK0_B	DMA1 channel 0 acknowledge	GPIO[19]	1	O
DMA1_DDONE0_B	DMA1 channel 0 done	-	1	O
DMA 2				
See DMA signal descriptions for more details				
DMA2_DREQ0_B	DMA2 channel 0 request	GPIO[20]/ALT_MDVAL	1	I
DMA2_DACK0_B	DMA2 channel 0 acknowledge	EVT7_B/ALT_MSRCID[0]	1	O
DMA2_DDONE0_B	DMA2 channel 0 done	EVT8_B/ALT_MSRCID[1]	1	O
FM1 dTSEC1 RGMII				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EC1_TXD[3:0]	FM1 dTSEC1 transmit data	USB1_D[7:4]	4	O
EC1_TX_EN	FM1 dTSEC1 transmit enable	USB1_STP	1	O
EC1_GTX_CLK	FM1 dTSEC1 transmit clock out	USB1_CLK	1	O
EC1_RXD[3:0]	FM1 dTSEC1 receive data	USB1_D[3:0]	4	I
EC1_RX_DV	FM1 dTSEC1 receive data valid	USB1_NXT	1	I
EC1_RX_CLK	FM1 dTSEC1 receive clock	USB1_DIR	1	I
FM2 dTSEC1 RGMII				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EC2_TXD[3:0]	FM2 dTSEC1 transmit data	FM1 dTSEC2 TXD[3:0]/ USB2_D[7:4]/	4	O
EC2_TX_EN	FM2 dTSEC1 transmit enable	FM1 dTSEC2 TX_EN/ USB2_STP	1	O
EC2_GTX_CLK	FM2 dTSEC1 transmit clock out	FM1 dTSEC2 GTX_CLK USB2_CLK	1	O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
EC2_RXD[3:0]	FM2 dTSEC1 receive data	FM1 dTSEC2 RXD[3:0]/ USB2_D[3:0]	4	I
EC2_RX_DV	FM2 dTSEC1 receive data valid	FM1 dTSEC2 RX_DV/ USB2_NXT	1	I
EC2_RX_CLK	FM2 dTSEC1 receive clock	FM1 dTSEC2 RX_CLK/ USB2_DIR	1	I
IEEE 1588				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
TSEC_1588_CLK_IN	1588 clock in	-	1	I
TSEC_1588_TRIG_I_N1	1588 trigger in 1	-	1	I
TSEC_1588_TRIG_I_N2	1588 trigger in 2	-	1	I
TSEC_1588_ALARM_OUT1	1588 alarm out 1	-	1	O
TSEC_1588_ALARM_OUT2	1588 alarm out 2	GPIO[30]	1	O
TSEC_1588_CLK_OUT	1588 clock out	-	1	O
TSEC_1588_PULSE_OUT1	1588 pulse out 1	-	1	O
TSEC_1588_PULSE_OUT2	1588 pulse out 2	GPIO[31]	1	O
Ethernet External Timestamping (dTSECs)				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EC_XTRNL_TX_STMP1	Ethernet External Timestamp Transmit 1	-	1	I
EC_XTRNL_RX_STMP1	Ethernet External Timestamp Receive 1	-	1	I
EC_XTRNL_TX_STMP2	Ethernet External Timestamp Transmit 2	-	1	I
EC_XTRNL_RX_STMP2	Ethernet External Timestamp Receive 2	-	1	I
Ethernet Management Interface 1 (FMan1; dTSEC1)				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EMI1_MDC	Ethernet management data clock	-	1	O
EMI1_MDIO	Ethernet management data in/out	-	1	I/O
Ethernet management Interface 2 (FMan1; 10Ge)				
See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EMI2_MDC	Ethernet management data clock	-	1	O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
EMI2_MDIO	Ethernet management data in/out	-	1	I/O
Gigabit Clock				
EC_GTX_CLK125	Gigabit reference clock	-	1	I
PIC				
See MPIC Signal Descriptions for more details				
IRQ[0:2]	External interrupt	-	3	I
IRQ[3:11]	External interrupt	GPIO[21:29]	9	I
IRQ_OUT_B	Interrupt output	EVT9_B	1	O
Security Monitor				
See Signals for more details				
TMP_DETECT_B	Tamper detect	-	1	I
System Control				
See External Signal Descriptions for more details				
PORESET_B	Power on reset	-	1	I
HRESET_B	Hard reset	-	1	I/O
RESET_REQ_B	Reset request (POR or hard)	-	1	O
CKSTP_OUT_B	Checkstop out	-	1	O
GPIO				
See GPIO signal descriptions for more details				
GPIO[0]	General-purpose I/O	-	1	I/O
GPIO[1]	General-purpose I/O	-	1	I/O
GPIO[2]	General-purpose I/O	-	1	I/O
GPIO[3]	General-purpose I/O	-	1	I/O
GPIO[4]	General-purpose I/O	-	1	I/O
GPIO[5]	General-purpose I/O	-	1	I/O
GPIO[6]	General-purpose I/O	-	1	I/O
GPIO[7]	General-purpose I/O	-	1	I/O
GPIO[8]	General-purpose I/O	UART1_SOUT	1	I/O
GPIO[9]	General-purpose I/O	UART2_SOUT	1	I/O
GPIO[10]	General-purpose I/O	UART1_SIN	1	I/O
GPIO[11]	General-purpose I/O	UART2_SIN	1	I/O
GPIO[12]	General-purpose I/O	UART1_RTS/_BUART3_SOUT	1	I/O
GPIO[13]	General-purpose I/O	UART2_RTS/_BUART4_SOUT	1	I/O
GPIO[14]	General-purpose I/O	UART1_CTS/_BUART3_SIN	1	I/O
GPIO[15]	General-purpose I/O	UART2_CTS/_BUART4_SIN	1	I/O
GPIO[16]	General-purpose I/O	IIC3_SCL/ SDHC_CD_B	1	I/O
GPIO[17]	General-purpose I/O	IIC3_SDA/ SDHC_WP	1	I/O
GPIO[18]	General-purpose I/O	DMA1_DREQ0_B	1	I/O

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
GPIO[19]	General-purpose I/O	DMA1_DACK0_B	1	I/O
GPIO[20]	General-purpose I/O	DMA2_DREQ0_B/ ALT_MDVAL	1	I/O
GPIO[21:29]	General-purpose I/O	IRQ[3:11]	9	I/O
GPIO[30]	General-purpose I/O	TSEC_1588_ALARM_OUT2	1	I/O
GPIO[31]	General-purpose I/O	TSEC_1588_PULSE_OUT2	1	I/O
Power Management				
See RCPM Signal Descriptions for more details				
ASLEEP	Asleep	-	1	O
Debug				
EVT[0:4]_B	Events 0-4	-	5	I/O
EVT5_B	Event 5	IIC4_SCL	1	I/O
EVT6_B	Event 6	IIC4_SDA	1	I/O
EVT7_B	Event 7	DMA2_DACK0_B/ ALT_MSRCID[0]	1	I/O
EVT8_B	Event 8	DMA2_DDONE0_B/ ALT_MSRCID[1]	1	I/O
EVT9_B	Event 9	IRQ_OUT_B	1	I/O
MDVAL	Memory debug data valid	-	1	O
MSRCID[0]	Memory debug source port ID 0	-	1	O
MSRCID[1]	Memory debug source port ID 1	-	1	O
MSRCID[2]	Memory debug source port ID 2	-	1	O
ALT_MDVAL	Alternate debug data valid	DMA2_DREQ0_B /GPIO[20]	1	O
ALT_MSRCID[0]	Alternate debug source port ID 0	DMA2_DACK0_B /EVT7_B	1	O
ALT_MSRCID[1]	Alternate debug source port ID 1	DMA2_DDOME0_B /EVT8_B	1	O
CLK_OUT	Clock out	-	1	O
Clocking				
See External Clock Signals for more details				
SYSCLK	System clock	-	1	I
RTC	Real time clock	-	1	I
JTAG				
TCK	Test clock	-	1	I
TDI	Test data in	-	1	I
TDO	Test data out	-	1	O
TMS	Test mode select	-	1	I
TRST_B	Test reset	-	1	I

Table continues on the next page...

Table 3-1. P4080 Signal Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	No. of Signals	I/O
Analog				
MVREF	SSTL18 Reference Voltage	-	-	
SD_IMP_CAL_TX	Tx Impedance Calibration	-	-	
SD_IMP_CAL_RX	Rx Impedance Calibration	-	-	
TEMP_ANODE	Thermal diode anode pin	-	-	-
TEMP_CATHODE	Thermal diode cathode pin	-	-	-
IO Voltage Select				
IO_VSEL[0:4]	I/O voltage select	-	5	I

3.3 Dedicated Configuration Signals

3.3.1 I/O Voltage Select

The I/O voltage select inputs, shown in [Table 3-2](#), properly configure the receivers and drivers of the I/Os associated with the BV_{DD}, CV_{DD}, and LV_{DD} power planes, respectively.

Table 3-2. I/O Voltage Selection

Signals	Value (Binary)	VDD Voltage Selection		
		BVDD	CVDD	LVDD
IO_VSEL[0:4]	0_0000	3.3 V	3.3 V	3.3 V
	0_0001	3.3 V	3.3 V	2.5 V
	0_0010	3.3 V	3.3 V	1.8 V
	0_0011	3.3 V	2.5 V	3.3 V
	0_0100	3.3 V	2.5 V	2.5 V
	0_0101	3.3 V	2.5 V	1.8 V
	0_0110	3.3 V	1.8 V	3.3 V
	0_0111	3.3 V	1.8 V	2.5 V
	0_1000	3.3 V	1.8 V	1.8 V
	0_1001	2.5 V	3.3 V	3.3 V
	0_1010	2.5 V	3.3 V	2.5 V
	0_1011	2.5 V	3.3 V	1.8 V
	0_1100	2.5 V	2.5 V	3.3 V
	0_1101	2.5 V	2.5 V	2.5 V
	0_1110	2.5 V	2.5 V	1.8 V
	0_1111	2.5 V	1.8 V	3.3 V

Table continues on the next page...

Table 3-2. I/O Voltage Selection (continued)

Signals	Value (Binary)	VDD Voltage Selection		
		BVDD	CVDD	LVDD
	1_0000	2.5 V	1.8 V	2.5 V
	1_0001	2.5 V	1.8 V	1.8 V
	1_0010	1.8 V	3.3 V	3.3 V
	1_0011	1.8 V	3.3 V	2.5 V
	1_0100	1.8 V	3.3 V	1.8 V
	1_0101	1.8 V	2.5 V	3.3 V
	1_0110	1.8 V	2.5 V	2.5 V
	1_0111	1.8 V	2.5 V	1.8 V
	1_1000	1.8 V	1.8 V	3.3 V
	1_1001	1.8 V	1.8 V	2.5 V
	1_1010	1.8 V	1.8 V	1.8 V

NOTE

The IO_VSEL[0:4] inputs do not control the DDR DRAM I/O supply voltage (GV_{DD}) or the pad power supply for the SerDes transceivers (XV_{DD}). The GV_{DD} voltage must match XV_{DD} .

The DRAM type select configuration input signal (*cfg_dram_type*) determines the voltage to be used, either 1.5 V (DDR3) or 1.8 V (DDR2), for GV_{DD} and by extension XV_{DD} . See [DRAM type select](#), for more information.

3.4 Configuration Signals Sampled at Reset

The signals that serve alternate functions as configuration input signals during system reset are summarized in [Table 3-3](#). The detailed interpretation of their voltage levels during reset is described in [Power-On Reset Configuration](#).

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of PORESET_B. However, there is a setup and hold time for these signals relative to the rising edge of PORESET_B, as described in the chip hardware specifications. Note that the PLL configuration signals have different setup and hold time requirements than the other reset configuration signals.

The reset configuration signals are multiplexed with other functional signals. The values on these signals during reset are interpreted to be logic one or zero, regardless of whether the functional signal name is defined as active-low. The reset configuration signals have internal pull-up resistors so that if the signals are not driven, the default value is high (a

one), as shown in the table. Some signals do not have pull-up resistors and must be driven high or low during the reset period. For details about all the signals that require external pull-up resistors, see the chip hardware specifications.

Table 3-3. P4080 Reset Configuration Signals

Reset Configuration Name	Functional Interface	Functional Signal Name	Default
cfg_gpininput[0:15]	eLBC	LAD[0:15]	1111 1111 1111 1111
cfg_elbc_ecc	eLBC	LA[23]	1
cfg_dram_type	eLBC	LA[24]	1
cfg_rcw_src[0]	eLBC	LGPL0/LFCLE	1
cfg_rcw_src[1]	eLBC	LGPL1/LFALE	1
cfg_rcw_src[2]	eLBC	LGPL2/LOE_B/LFRE_B	1
cfg_rcw_src[3]	eLBC	LGPL3/LFWP_B	1
cfg_rcw_src[4]	eLBC	LGPL5	1

3.5 Signal Multiplexing Details

Due to the extensive functionality present on the P4080 and the limited number of external signals available, several functional blocks share signal resources through multiplexing. Shared groups of signals are shown in [Figure 3-1](#). Signals which serve multiple functions during run-time (not including POR configuration functionality) generally fall into the following categories:

- Alternate functionality within a single functional block The functional block controls the multiple functions of a single signal. (These signals are designated using the slash notation in [Table 3-1](#).) For example, the eLBC signal LGPL[4]/LGTA/LUPWAIT/LPBSE/LFRB functions in one of five distinct ways; Local bus GP line/GPCM terminate access/UPM wait/parity byte select/flash ready-busy, respectively. Each of these functions serves the functionality of the eLBC; the specific eLBC mode and configuration determines how this signal is used. For these cases, refer to the functional block chapter for the functional configuration details.
- Alternate functionality between multiple functional blocks The multiple functions of a single signal serve different functional blocks. (These signals are designated in the Alternate Function(s) column of [Table 3-1](#).) In this case, the signal's function is determined at the device level (rather than at the block level) typically by a reset control word (RCW) option. For example, the signal SPI_CS[0:3]/SDHC_DAT[4:7] can be utilized by either the SPI block or the eSDHC block. Since these signals alternatively service two different functional blocks, their function is determined at the device level by the RCW[SPI] field. See [RCW Field Definitions](#), for details regarding external signal functional selection using the RCW.

NOTE

Some external signals fall into both of the above categories. For example, the signal DMA2_DREQ[0] can be used by one of three functional blocks: DMA controller 2 (DMA2_DREQ[0]), the general purpose I/O module (GPIO[20]), or debug (ALT_MDVAL). The selection of which block utilizes this signal is determined at the device level by the RCW[DMA2] field.

3.5.1 IEEE 1588 and GPIO Signal Multiplexing

The IEEE 1588 interface shares signals with GPIO. The functionality of these signals is determined by the 1588 field in the reset configuration word (RCW[1588]).

Table 3-4. IEEE 1588 Signal Configuration

Signal Name	Signal Function	RCW[1588]
TSEC_1588_ALARM_OUT2	TSEC_1588_ALARM_OUT2	0
	GPIO[30]	1
TSEC_1588_PULSE_OUT2	TSEC_1588_PULSE_OUT2	0
	GPIO[31]	1

3.5.2 Frame Manager 1 dTSEC1 and USB1 Signal Multiplexing

The Frame Manager 1 (FM1) dTSEC1 RGMII interface shares signals with the USB1 controller. The functionality of these signals is determined by the EC1 field in the reset configuration word (RCW[EC1]) .

Table 3-5. FM1 dTSEC RGMII Signal Configuration

Signal Name	Signal Function	RCW[EC1]
EC1_TXD[3:0]	EC1_TXD[3:0] (FM1 dTSEC1 RGMII)	00
	Reserved	01
	USB1_D[7:4]	10
	None	11
EC1_RXD[3:0]	EC1_RXD[3:0] (FM1 dTSEC1 RGMII)	00
	Reserved	01
	USB1_D[3:0]	10
	None	11
EC1_TX_EN	EC1_TX_EN (FM1 dTSEC1 RGMII)	00

Table continues on the next page...

**Table 3-5. FM1 dTSEC RGMII Signal Configuration
(continued)**

Signal Name	Signal Function	RCW[EC1]
	Reserved	01
	USB1_STP	10
	None	11
EC1_GTX_CLK	EC1_GTX_CLK (FM1 dTSEC RGMII)	00
	Reserved	01
	USB1_CLK	10
	None	11
EC1_RX_DV	EC1_RX_DV (FM1 dTSEC1 RGMII)	00
	Reserved	01
	USB1_NXT	10
	None	11
EC1_RX_CLK	EC1_RX_CLK (FM1 dTSEC1 RGMII)	00
	Reserved	01
	USB1_DIR	10
	None	11

3.5.3 Frame Manager 2 dTSEC1, Frame Manager 1 dTSEC2, and USB2 Signal Multiplexing

The FM2 dTSEC1 RGMII interface shares signals with the FM1 dTSEC2 RGMII and USB2 controller. The functionality of these signals is determined by the EC2 field in the reset configuration word (RCW[EC2]).

Table 3-6. FM2 dTSEC1 RGMII Signal Configuration

Signal Name	Signal Function	RCW[EC2]
EC2_TXD[3:0]	EC2_TXD[3:0] (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_TXD[3:0] (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_D[7:4]	100
	None	111
EC2_RXD[3:0]	EC2_RXD[3:0] (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_RXD[3:0] (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_D[3:0]	100

Table continues on the next page...

Table 3-6. FM2 dTSEC1 RGMII Signal Configuration (continued)

Signal Name	Signal Function	RCW[EC2]
	None	111
EC2_TX_EN	EC2_TX_EN (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_TX_EN (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_STP	100
	None	111
EC2_GTX_CLK	EC2_GTX_CLK (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_GTX_CLK (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_CLK	100
	None	111
EC2_RX_DV	EC2_RX_DV (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_RX_DV (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_NXT	100
	None	111
EC2_RX_CLK	EC2_RX_CLK (FM2 dTSEC1 RGMII)	000
	Reserved	001
	EC2_RX_CLK (FM1 dTSEC2 RGMII)	010
	Reserved	011
	USB2_DIR	100
	None	111

3.5.4 UART and GPIO Signal Multiplexing

For the UART signals, the following relationships apply:

- UART1 shares signals with GPIO and UART3
- UART2 shares signals with GPIO and UART4

The functionality of these signals is determined by the UART field in the reset configuration word (RCW[UART]).

Table 3-7. UART Signal Configuration

Signal Name	Signal Function	RCW[UART]
UART1_SOUT	GPIO[8]	0x0
	UART1_SOUT	0x3, 0x4, 0x5, 0x6, 0x7
UART1_SIN	GPIO[10]	0x0
	UART1_SIN	0x3, 0x4, 0x5, 0x6, 0x7
UART1_RTS	GPIO[12]	0x0, 0x3, 0x5
	UART1_RTS	0x4, 0x6
	UART3_SOUT	0x7
UART1_CTS	GPIO[14]	0x0, 0x3, 0x5
	UART1_CTS	0x4, 0x6
	UART3_SIN	0x7
UART2_SOUT	GPIO[9]	0x0, 0x3, 0x4
	UART2_SOUT	0x6, 0x5, 0x7
UART2_SIN	GPIO[11]	0x0, 0x3, 0x4
	UART2_SIN	0x6, 0x5, 0x7
UART2_RTS	GPIO[13]	0x0, 0x3, 0x4, 0x5
	UART2_RTS	0x6
	UART4_SOUT	0x7
UART2_CTS	GPIO[15]	0x0, 0x3, 0x4, 0x5
	UART2_CTS	0x6
	UART4_SIN	0x7

3.5.5 I2C3, eSDHC, and GPIO Signal Multiplexing

The I2C3 interface shares signals with the eSDHC, and GPIO. The functionality of these signals is determined by the I2C3 field in the reset configuration word (RCW[I2C3]).

Table 3-8. I2C3 Signal Configuration

Signal Name	Signal Function	RCW[I2C3]
IIC3_SCL	IIC3_SCL	00
	Reserved	01
	GPIO[16]	10
	SDHC_CD	11
IIC3_SDA	IIC3_SDA	00
	Reserved	01
	GPIO[17]	10
	SDHC_WP	11

3.5.6 I2C4 and Debug Signal Multiplexing

The I2C4 interface shares signals with debug signals $\overline{EVT}[5:6]$. The functionality of these signals is determined by the I2C4 field in the reset configuration word (RCW[I2C4]).

Table 3-9. I2C4 Signal Configuration

Signal Name	Signal Function	RCW[I2C4]
IIC4_SCL	IIC4_SCL	0
	$\overline{EVT}[5]$	1
IIC4_SDA	IIC4_SDA	0
	$\overline{EVT}[6]$	1

3.5.7 MPIC and GPIO Signal Multiplexing

The interrupt controller (MPIC) shares signals with GPIO. The functionality of these signals is determined by the IRQ field in the reset configuration word (RCW[IRQ]).

NOTE

When GPIO functionality is selected, the corresponding MPIC external interrupt vector priority registers (MPIC_EIVPRn) must have their polarity set to active high.

Table 3-10. IRQ Signal Configuration

Signal Name	Signal Function	RCW[IRQ]
IRQ[3:11]	IRQ[3:11]	0
	GPIO[21:29]	1

3.5.8 MPIC and Debug Signal Multiplexing

The interrupt controller (MPIC) shares a signal with debug signal $\overline{EVT}[9]$. The functionality of this signal is determined by the IRQ_OUT field in the reset configuration word (RCW[IRQ_OUT]).

Table 3-11. IRQ_OUT Signal Configuration

Signal Name	Signal Function	RCW[IRQ_OUT]
IRQ_OUT	IRQ_OUT	0
	EVT[9]	1

3.5.9 eSPI and eSDHC Signal Multiplexing

The eSPI shares signals with the eSDHC. The functionality of these signals is determined by the SPI field in the reset configuration word (RCW[SPI]).

Table 3-12. eSPI Signal Configuration

Signal Name	Signal Function	RCW[SPI]
SPI_CS[0:3]	SPI_CS[0:3]	0
	SDHC_DAT[4:7]	1

3.5.10 DMA1, Debug, and GPIO Signal Multiplexing

DMA controller 1, channel 0 (DMA1) shares signals with debug and GPIO. The functionality of these signals is determined by the DMA1 field in the reset configuration word (RCW[DMA1]).

Table 3-13. DMA1 Signal Configuration

Signal Name	Signal Function	RCW[DMA1]
DMA1_DREQ[0]	DMA1_DREQ[0]	0
	GPIO[18]	1
DMA1_DACK[0]	DMA1_DACK[0]	0
	GPIO[19]	1
DMA1_DDONE[0]	DMA1_DDONE[0]	0
	Reserved	1

3.5.11 DMA2, Debug, and GPIO Signal Multiplexing

DMA controller 2, channel 0 (DMA2) shares signals with debug and GPIO. The functionality of these signals is determined by the DMA2 field in the reset configuration word (RCW[DMA2]).

Table 3-14. DMA2 Signal Configuration

Signal Name	Signal Function	RCW[DMA2]
DMA2_DREQ[0]	DMA2_DREQ[0]	00
	ALT_MDVAL	01
	GPIO[20]	10
	Reserved	11
DMA2_DACK[0]	DMA2_DACK[0]	00
	ALT_MSRCID[0]	01
	EVT[7]	10
	Reserved	11
DMA2_DDONE[0]	DMA2_DDONE[0]	00
	ALT_MSRCID[1]	01
	EVT[8]	10

3.5.12 SerDes Lane Assignments and Multiplexing

The SerDes lanes correspond to the SD_TX n and SD_RX n signals as shown in [Table 3-15](#). [Table 3-15](#) also shows the lane assignments for the controllers that can use the SerDes.

Table 3-15. SerDes Bank/Lane/Signal Assignments

Bank	Lane	Signal	Controller Lane Assignments							
			PCI Express			SRI0		Debug	Frame Manager	
			PCIe1	PCIe2	PCIe3	sRIO1	sRIO2		FM1	FM2
1	A	SD_TX[0]/SD_RX[0]	0				3			
	B	SD_TX[1]/SD_RX[1]	1				2			
	C	SD_TX[2]/SD_RX[2]	2		0		1			
	D	SD_TX[3]/SD_RX[3]	3		1		0			
	E	SD_TX[4]/SD_RX[4]	4	0		3				dTSEC1
	F	SD_TX[5]/SD_RX[5]	5	1		2	0			dTSEC2
	G	SD_TX[6]/SD_RX[6]	6	2		1				dTSEC3
	H	SD_TX[7]/SD_RX[7]	7	3		0				dTSEC4
	I	SD_TX[8]/SD_RX[8]						1		

Table continues on the next page...

Table 3-15. SerDes Bank/Lane/Signal Assignments (continued)

Bank	Lane	Signal	Controller Lane Assignments							
			PCI Express			SRIO		Debug	Frame Manager	
			PCIe1	PCIe2	PCIe3	sRIO1	sRIO2		FM1	FM2
	J	SD_TX[9]/SD_RX[9] SD_RX[9]/SD_RX[9]						0		
2	A	SD_TX[10]/SD_RX[10] SD_RX[10]/SD_RX[10]			0				dTSEC1 / 10GEC0	
	B	SD_TX[11]/SD_RX[11] SD_RX[11]/SD_RX[11]			1				dTSEC2 / 10GEC1	
	C	SD_TX[12]/SD_RX[12] SD_RX[12]/SD_RX[12]			2				dTSEC3 / 10GEC2	
	D	SD_TX[13]/SD_RX[13] SD_RX[13]/SD_RX[13]			3				dTSEC4 / 10GEC3	
3	A	SD_TX[14]/SD_RX[14] SD_RX[14]/SD_RX[14]							dTSEC1 / 10GEC0	
	B	SD_TX[15]/SD_RX[15] SD_RX[15]/SD_RX[15]							dTSEC2 / 10GEC1	
	C	SD_TX[16]/SD_RX[16] SD_RX[16]/SD_RX[16]							dTSEC3 / 10GEC2	
	D	SD_TX[17]/SD_RX[17] SD_RX[17]/SD_RX[17]							dTSEC4 / 10GEC3	

Table 3-16 describes the protocol multiplexing options for the SerDes lanes. As shown in Table 3-16, each option is selected by a specific encoding of RCW[SRDS_PRTCL]. See [RCW Field Definitions](#), for more information.

Table 3-16. SerDes Lane Multiplexing/Configuration

SRDS PRTC L	Bank 1										Bank 2				Bank 3			
	A	B	C	D	E	F	G	H	I	J	A	B	C	D	A	B	C	D
0x02	PCIe1 (2.5G)										Debug (5/2.5G)				XAUI FM2 10GEC			
0x05	PCIe1 (5/2.5G)				PCIe2 (5/2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				XAUI FM1 10GEC			
0x08	PCIe1 (5/2.5G)		PCIe3 (5/2.5G)		PCIe2 (5/2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				XAUI FM1 10GEC			

Table continues on the next page...

Table 3-16. SerDes Lane Multiplexing/Configuration (continued)

SRDS PRTC L	Bank 1										Bank 2				Bank 3					
	A	B	C	D	E	F	G	H	I	J	A	B	C	D	A	B	C	D		
0x0D	PCIe1 (5/2.5G)		PCIe2 (5/2.5G)		2x SGMII FM2 dTSEC[3:4] ¹		Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC									
0x0E	PCIe1 (5/2.5G)		PCIe3 (5/2.5G)		PCIe2 (5/2.5G)		2x SGMII FM2 dTSEC[3:4] ¹		Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC							
0x0F	PCIe1 (5/2.5G)		4x SGMII FM2 dTSEC[1:4] ¹				Debug (5/2.5G)		XAUI FM2 10GEC		Reserved ²									
0x10	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	4x SGMII FM2 dTSEC[1:4] ¹				Debug (5/2.5G)		XAUI FM2 10GEC		Reserved ²									
0x13	sRIO2 (2.5G)		sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC									
0x16	sRIO2 (3.125G)		sRIO1 (3.125G)				Debug (3.125G)		4x SGMII FM2 dTSEC[1:4]		4x SGMII FM1 dTSEC[1:4]									
0x19	sRIO2 (3.125G)		sRIO1 (3.125G)				Debug (3.125G)		PCIe3 (5/2.5G)		4x SGMII FM1 dTSEC[1:4]									
0x1D	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	-	sRIO2 (2.5G)	-	sRIO1 (2.5G)	Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC									
0x22	PCIe1 (5/2.5G)		sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC									
0x25	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC		XAUI FM1 10GEC									

- When SerDes bank 1 is configured for 5G operation (by RCW[SRDS_RATIO_B1] and the reference clock), lanes configured as SGMII operate at 1.25Gbps regardless of the corresponding SRDS_DIV_B1 settings.
- Bank 3 cannot be used in this configuration.

3.6 SerDes Control Memory Map

The SerDes control register map occupies 1024 bytes of memory-mapped space. Reserved bits should always be written with the value they return when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. All the registers are 32 bits wide and, although (mostly) located on 128-bit address boundaries, should be accessed only as 32-bit quantities.

The SerDes control address offset, shown in the following table, is divided into the following areas:

- 000h-01Fh-Bank 1 PLL control registers
- 020h-03Fh-Bank 2 PLL control registers

SerDes Control Memory Map

- 040h-05Fh-Bank 3 PLL control registers
- 060h-08Fh-Reserved
- 090h-0FFh-calibration/miscellaneous control/status registers
- 100h-3FFh-Reserved
- 400h-6FFh-SerDes control registers for lanes on Bank 1 lanes A-J
- 700h-7FFh-Reserved
- 800h-8FFh-SerDes control registers for lanes on Bank 2 lanes A-D
- 900h-9FFh-SerDes control registers for lanes on Bank 3 lanes A-D

Signals memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_A000	SerDes Bank n Reset Control Register (Signals_SRDSB1RSTCTL)	32	R/W	0047_4500h	3.6.1/151
E_A004	SerDes Bank n PLL Control Register 0 (Signals_SRDSB1PLLCR0)	32	R/W	0000_0000h	3.6.2/154
E_A008	SerDes Bank1 PLL Control Register 1 (Signals_SRDSB1PLLCR1)	32	R/W	0000_0100h	3.6.3/155
E_A020	SerDes Bank n Reset Control Register (Signals_SRDSB2RSTCTL)	32	R/W	0047_4500h	3.6.1/151
E_A024	SerDes Bank n PLL Control Register 0 (Signals_SRDSB2PLLCR0)	32	R/W	0000_0000h	3.6.2/154
E_A028	SerDes Bank1 PLL Control Register 1 (Signals_SRDSB2PLLCR1)	32	R/W	0000_0100h	3.6.3/155
E_A040	SerDes Bank n Reset Control Register (Signals_SRDSB3RSTCTL)	32	R/W	0047_4500h	3.6.1/151
E_A044	SerDes Bank n PLL Control Register 0 (Signals_SRDSB3PLLCR0)	32	R/W	0000_0000h	3.6.2/154
E_A048	SerDes Bank1 PLL Control Register 1 (Signals_SRDSB3PLLCR1)	32	R/W	0000_0100h	3.6.3/155
E_A090	SerDes Transmit Calibration Control Register (Signals_SRDSTCALCR)	32	R/W	0000_0000h	3.6.4/157
E_A0A0	SerDes Receive Calibration Control Register (Signals_SRDSRCALCR)	32	R/W	0000_0000h	3.6.5/158
E_A0B0	SerDes General Register 0 (Signals_SRDSGR0)	32	R/W	0000_0000h	3.6.6/159
E_A0E0	SerDes Protocol Converter Configuration Register 0 (Signals_SRDSPCCR0)	32	R/W	0000_0000h	3.6.7/160
E_A0E4	SerDes Protocol Converter Configuration Register 1 (Signals_SRDSPCCR1)	32	R/W	0000_0000h	3.6.8/162
E_A0E8	SerDes Protocol Converter Configuration Register 2 (Signals_SRDSPCCR2)	32	R/W	ECC0_FF00h	3.6.9/164
E_A400	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRA0)	32	R/W	004A_0000h	3.6.10/167
E_A404	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRA1)	32	R/W	0000_0000h	3.6.11/170

Table continues on the next page...

Signals memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_A410	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRA0)	32	R/W	See section	3.6.12/173
E_A410	Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRA0)	32	R/W	See section	3.6.13/176
E_A410	Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRA0)	32	R/W	See section	3.6.14/178
E_A418	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRA0)	32	R/W	1000_0000h	3.6.15/180
E_A420	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRB0)	32	R/W	See section	3.6.16/182
E_A440	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRB0)	32	R/W	004A_0000h	3.6.10/167
E_A444	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRB1)	32	R/W	0000_0000h	3.6.11/170
E_A450	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRB0)	32	R/W	See section	3.6.12/173
E_A450	Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRB0)	32	R/W	See section	3.6.13/176
E_A450	Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRB0)	32	R/W	See section	3.6.14/178
E_A458	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRB0)	32	R/W	1000_0000h	3.6.15/180
E_A460	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRB0)	32	R/W	See section	3.6.16/182
E_A480	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRC0)	32	R/W	004A_0000h	3.6.10/167
E_A484	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRC1)	32	R/W	0000_0000h	3.6.11/170
E_A490	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRC0)	32	R/W	See section	3.6.12/173
E_A490	Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRC0)	32	R/W	See section	3.6.13/176
E_A490	Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRC0)	32	R/W	See section	3.6.14/178
E_A498	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRC0)	32	R/W	1000_0000h	3.6.15/180
E_A4A0	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRC0)	32	R/W	See section	3.6.16/182
E_A4C0	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRD0)	32	R/W	004A_0000h	3.6.10/167
E_A4C4	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRD1)	32	R/W	0000_0000h	3.6.11/170
E_A4D0	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRD0)	32	R/W	See section	3.6.12/173

Table continues on the next page...

Signals memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_A4D0	Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRD0)	32	R/W	See section	3.6.13/176
E_A4D0	Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRD0)	32	R/W	See section	3.6.14/178
E_A4D8	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRD0)	32	R/W	1000_0000h	3.6.15/180
E_A4E0	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCDR0)	32	R/W	See section	3.6.16/182
E_A500	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRE0)	32	R/W	004A_0000h	3.6.10/167
E_A504	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRE1)	32	R/W	0000_0000h	3.6.11/170
E_A510	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRE0)	32	R/W	See section	3.6.12/173
E_A518	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRE0)	32	R/W	1000_0000h	3.6.15/180
E_A520	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRE0)	32	R/W	See section	3.6.16/182
E_A540	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRF0)	32	R/W	004A_0000h	3.6.10/167
E_A544	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRF1)	32	R/W	0000_0000h	3.6.11/170
E_A550	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRF0)	32	R/W	See section	3.6.12/173
E_A558	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRF0)	32	R/W	1000_0000h	3.6.15/180
E_A560	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRF0)	32	R/W	See section	3.6.16/182
E_A580	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRG0)	32	R/W	004A_0000h	3.6.10/167
E_A584	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRG1)	32	R/W	0000_0000h	3.6.11/170
E_A590	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRG0)	32	R/W	See section	3.6.12/173
E_A598	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRG0)	32	R/W	1000_0000h	3.6.15/180
E_A5A0	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRG0)	32	R/W	See section	3.6.16/182
E_A5C0	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRH0)	32	R/W	004A_0000h	3.6.10/167
E_A5C4	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRH1)	32	R/W	0000_0000h	3.6.11/170
E_A5D0	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRH0)	32	R/W	See section	3.6.12/173

Table continues on the next page...

Signals memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_A5D8	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRH0)	32	R/W	1000_0000h	3.6.15/180
E_A5E0	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRH0)	32	R/W	See section	3.6.16/182
E_A600	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRJ0)	32	R/W	004A_0000h	3.6.10/167
E_A604	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRJ1)	32	R/W	0000_0000h	3.6.11/170
E_A610	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRJ0)	32	R/W	See section	3.6.12/173
E_A618	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRJ0)	32	R/W	1000_0000h	3.6.15/180
E_A620	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRJ0)	32	R/W	See section	3.6.16/182
E_A640	Bank 1 General Control Register 0 - Lane x (Signals_B1GCRJ0)	32	R/W	004A_0000h	3.6.10/167
E_A644	Bank 1 General Control Register 1 - Lane x (Signals_B1GCRJ1)	32	R/W	0000_0000h	3.6.11/170
E_A650	Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRJ0)	32	R/W	See section	3.6.12/173
E_A658	Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRJ0)	32	R/W	1000_0000h	3.6.15/180
E_A660	Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRJ0)	32	R/W	See section	3.6.16/182
E_A800	Bank 2 General Control Register 0 - Lane x (Signals_B2GCRA0)	32	R/W	004A_0000h	3.6.17/184
E_A804	Bank 2 General Control Register 1 - Lane x (Signals_B2GCRA1)	32	R/W	0000_0000h	3.6.18/187
E_A818	Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRA0)	32	R/W	1000_0000h	3.6.19/190
E_A820	Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRRA0)	32	R/W	See section	3.6.20/192
E_A840	Bank 2 General Control Register 0 - Lane x (Signals_B2GCRB0)	32	R/W	004A_0000h	3.6.17/184
E_A844	Bank 2 General Control Register 1 - Lane x (Signals_B2GCRB1)	32	R/W	0000_0000h	3.6.18/187
E_A858	Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRB0)	32	R/W	1000_0000h	3.6.19/190
E_A860	Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRB0)	32	R/W	See section	3.6.20/192
E_A880	Bank 2 General Control Register 0 - Lane x (Signals_B2GCRC0)	32	R/W	004A_0000h	3.6.17/184
E_A884	Bank 2 General Control Register 1 - Lane x (Signals_B2GCRC1)	32	R/W	0000_0000h	3.6.18/187

Table continues on the next page...

Signals memory map (continued)

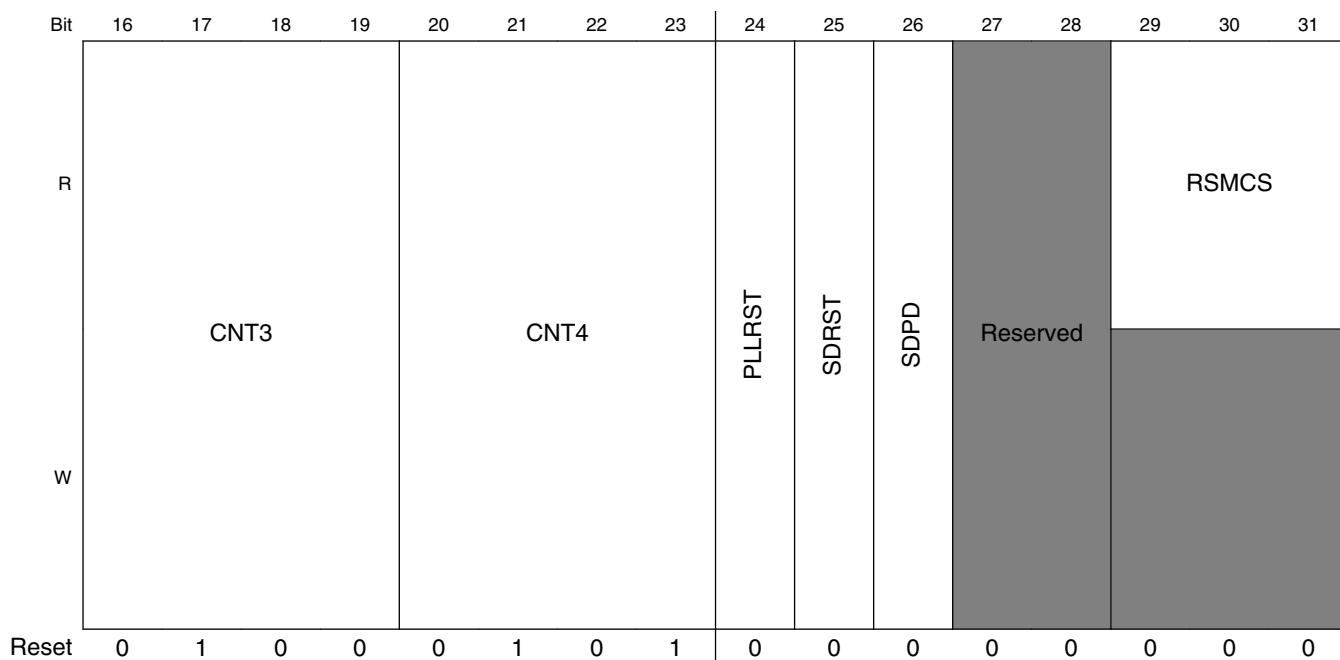
Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_A898	Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRC0)	32	R/W	1000_0000h	3.6.19/190
E_A8A0	Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRC0)	32	R/W	See section	3.6.20/192
E_A8C0	Bank 2 General Control Register 0 - Lane x (Signals_B2GCRD0)	32	R/W	004A_0000h	3.6.17/184
E_A8C4	Bank 2 General Control Register 1 - Lane x (Signals_B2GCRD1)	32	R/W	0000_0000h	3.6.18/187
E_A8D8	Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRD0)	32	R/W	1000_0000h	3.6.19/190
E_A8E0	Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLRD0)	32	R/W	See section	3.6.20/192
E_A900	Bank 3 General Control Register 0 - Lane x (Signals_B3GCRA0)	32	R/W	004A_0000h	3.6.21/194
E_A904	Bank 3 General Control Register 1 - Lane x (Signals_B3GCRA1)	32	R/W	0000_0000h	3.6.22/197
E_A918	Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRA0)	32	R/W	1000_0000h	3.6.23/200
E_A920	Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRRA0)	32	R/W	See section	3.6.24/202
E_A940	Bank 3 General Control Register 0 - Lane x (Signals_B3GCRB0)	32	R/W	004A_0000h	3.6.21/194
E_A944	Bank 3 General Control Register 1 - Lane x (Signals_B3GCRB1)	32	R/W	0000_0000h	3.6.22/197
E_A958	Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRB0)	32	R/W	1000_0000h	3.6.23/200
E_A960	Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRB0)	32	R/W	See section	3.6.24/202
E_A980	Bank 3 General Control Register 0 - Lane x (Signals_B3GCRD0)	32	R/W	004A_0000h	3.6.21/194
E_A984	Bank 3 General Control Register 1 - Lane x (Signals_B3GCRD1)	32	R/W	0000_0000h	3.6.22/197
E_A998	Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRC0)	32	R/W	1000_0000h	3.6.23/200
E_A9A0	Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRC0)	32	R/W	See section	3.6.24/202
E_A9C0	Bank 3 General Control Register 0 - Lane x (Signals_B3GCRD0)	32	R/W	004A_0000h	3.6.21/194
E_A9C4	Bank 3 General Control Register 1 - Lane x (Signals_B3GCRD1)	32	R/W	0000_0000h	3.6.22/197
E_A9D8	Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRD0)	32	R/W	1000_0000h	3.6.23/200
E_A9E0	Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLRD0)	32	R/W	See section	3.6.24/202

3.6.1 SerDes Bank n Reset Control Register (Signals_SRDSBnRSTCTL)

SRDSB n RSTCTL contains the control/status bits for SerDes reset state machine on banks 1, 2, and 3.

Address: E_A000h base + 0h offset + (32d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RSTREQ	RST_DONE	RST_ERR	Reserved	PSCL		Reserved		CNT1				CNT2			
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1



Signals_SRDSBnRSTCTL field descriptions

Field	Description
0 RSTREQ	To initiate SerDes soft reset, software writes a 1. CB reset state machine clears bit when reset is done. Software can assert this bit, but not clear it. If cleared in the middle, the reset state machine ignores the change.
1 RST_DONE	SerDes reset done from SerDes state machine
2 RST_ERR	No PLL lock before counter time_out
3 -	This field is reserved. Reserved
4–6 PSCL	Determines how many platform cycles equal one state machine tick. This value should be changed only when RSTREQ = 1 All other settings not shown are reserved 000 Up to 200-MHz platform (1 platform cycle per tick) 001 Up to 400-MHz platform (2 platform cycle per tick) 010 Up to 600-MHz platform (3 platform cycle per tick) 011 Up to 800-MHz platform (4 platform cycle per tick) 100 Up to 1000-MHz platform (5 platform cycle per tick)
7 -	This field is reserved. Reserved
8–11 CNT1	/2 Number of ticks before exiting states using cnt1 Default value is 0x4 (meaning 8 nominal ticks). 0x0 = 16 (32 200-MHz ticks)
12–15 CNT2	/8 Number of ticks before exiting states using cnt2 Default value is 0x4 (meaning 32 nominal ticks).

Table continues on the next page...

Signals_SRDSBnRSTCTL field descriptions (continued)

Field	Description
	0x0 = 16 (128 200-MHz ticks)
16–19 CNT3	/64 Number of ticks before exiting states using cnt3 Default value is 0x4 (meaning 256 nominal ticks) 0x0 = 16 (1024 200-MHz ticks)
20–23 CNT4	/4096 Number of ticks before exiting states using cnt4 Default value is 0x5 (meaning 20480 nominal ticks). 0x0 = 16 (65536 200-MHz ticks)
24 PLLRST	PLL master reset (active_high) 0 Application Mode 1 Reset
25 SDRST	SerDes master reset (active_high) 0 Application Mode 1 Reset
26 SDPD	SerDes power down. This power down signal shuts down the PLL, all of the receiver amplifiers, all of the samplers and places the transmitters in 3-state. Recommended setting: 0 0 Application Mode 1 Block power down
27–28 -	This field is reserved. Reserved
29–31 RSMCS	SerDes reset state machine current state for PLL

3.6.2 SerDes Bank n PLL Control Register 0 (Signals_SRDSBnPLLCR0)

SRDSB_nPLLCR0 contains the SerDes/test control bits for the SerDes on banks 1, 2 and 3.

Address: E_A000h base + 4h offset + (32d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		RFCK_SEL		RFCK_EN		Reserved		PLL_LCK		Reserved		Reserved		FRATE_SEL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_SRDSBnPLLCR0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 RFCK_SEL	Defines reference clock frequency: Recommended settings for all banks: 00 00 100-MHz ref clk 01 125-MHz ref clk 10 156-.25-MHz ref clk 11 Reserved
4 RFCK_EN	Enable buffered version of SD_REF_CLK _n Recommended settings for all banks: 0

Table continues on the next page...

Signals_SRDSBnPLLCR0 field descriptions (continued)

Field	Description
5–7 -	This field is reserved. Reserved
8 PLL_LCK	Indicates Bank <i>n</i> SerDes PLL has locked 0 No PLL lock 1 PLL lock
9–13 -	This field is reserved. Reserved
14–15 FRATE_SEL	Select frequency of PLL VCO of the Bank. All lanes within a bank must operate at a multiple of this frequency and within specified limits of the protocol. 00 5.00 GHz 01 6.25 GHz 10 Reserved 11 Reserved
16–31 -	This field is reserved. Reserved

3.6.3 SerDes Bank1 PLL Control Register 1 (Signals_SRDSBnPLLCR1)

SRDSBnPLLCR1 contains the SerDes/test control bits for the SerDes on banks 1, 2, and 3.

Address: E_A000h base + 8h offset + (32d × i), where i=0d to 2d

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

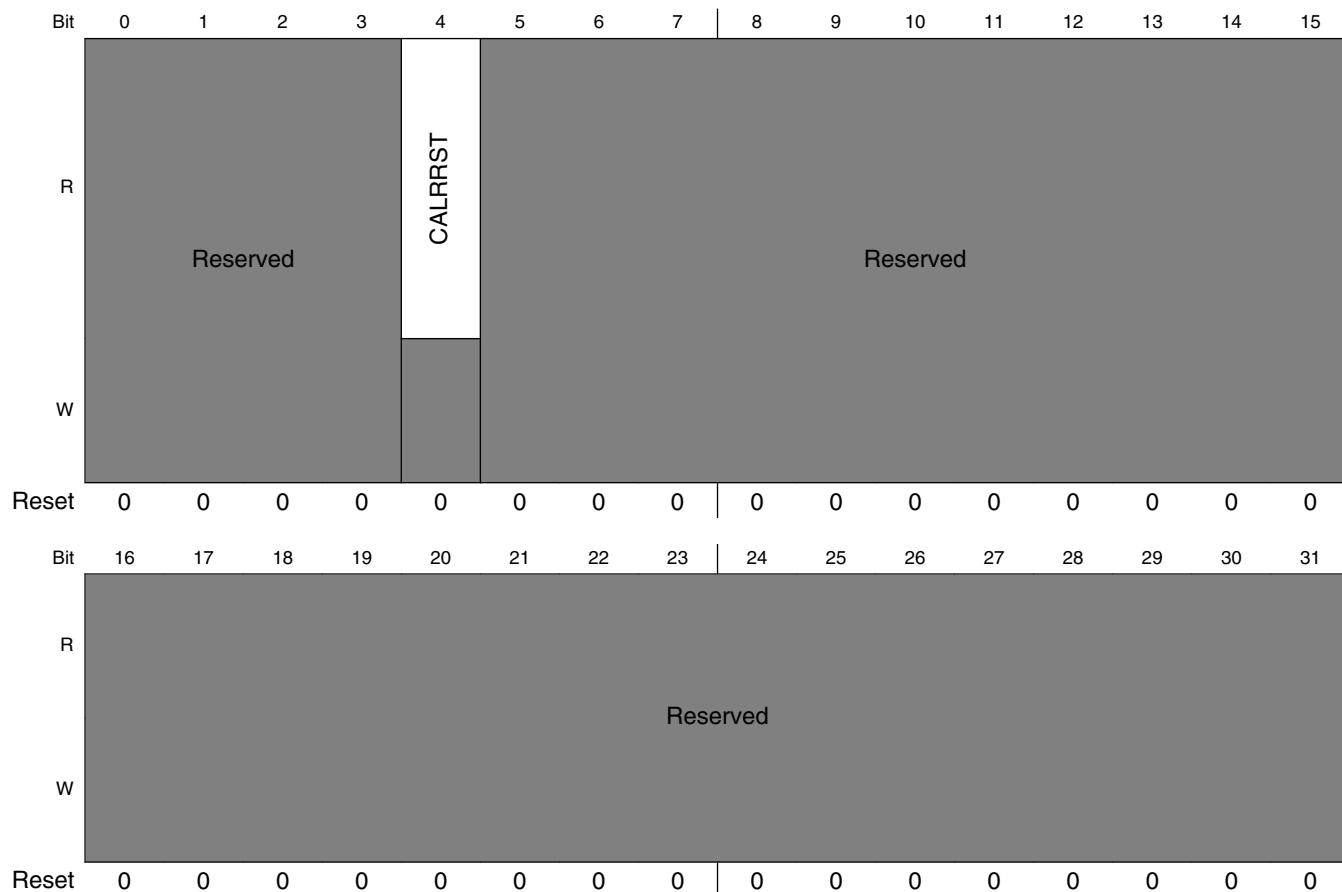
Signals_SRDSBnPLLCR1 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 PLLBW_SEL	Selects the SerDes PLL bandwidth. Recommended setting per Bank: 1 Bank1 1 Bank2 0 Bank3 0 Nominal PLL Bandwidth 1 PLL Bandwidth Setting for PCI-EXP
5–21 -	This field is reserved. Reserved
22–23 RFCK_PTRM_VCM	Recommended settings per bank: 01 Select RX termination common mode: 00 If srds_pd or lane_pd asserted, common mode in HiZ, else common mode is calibrated termination to xcorevss 01 Common mode is always calibrated termination to xcorevss 10 Common mode is HiZ, RX termination is uncalibrated 120 Ω differential 11 Common mode is 0.7*xcorevdd through 3 kΩ, RX termination is uncalibrated 120 Ω differential
24–31 -	This field is reserved. Reserved

3.6.4 SerDes Transmit Calibration Control Register (Signals_SRDSTCALCR)

SRDSTCALCR contains the functional control bits used for the transmit calibration logic.

Address: E_A000h base + 90h offset = E_A090h



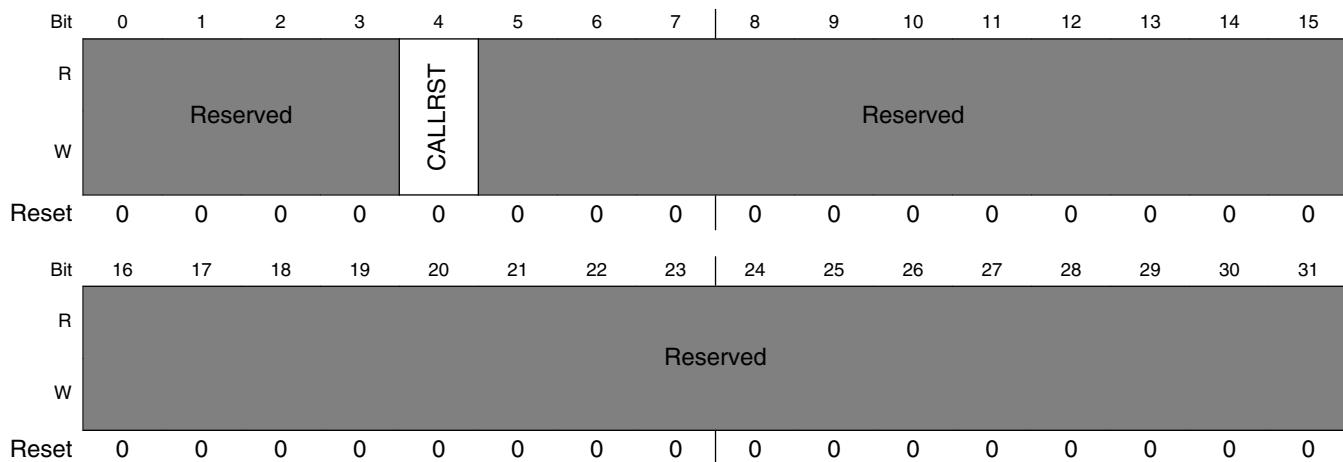
Signals_SRDSTCALCR field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 CALRRST	Reset to the calibration logic in the right endcap cell 0 Reset 1 Application Mode
5–31 -	This field is reserved. Reserved

3.6.5 SerDes Receive Calibration Control Register (Signals_SRDSRCALCR)

SRDSRCALCR contains the functional control bits used for the receive calibration logic.

Address: E_A000h base + A0h offset = E_A0A0h



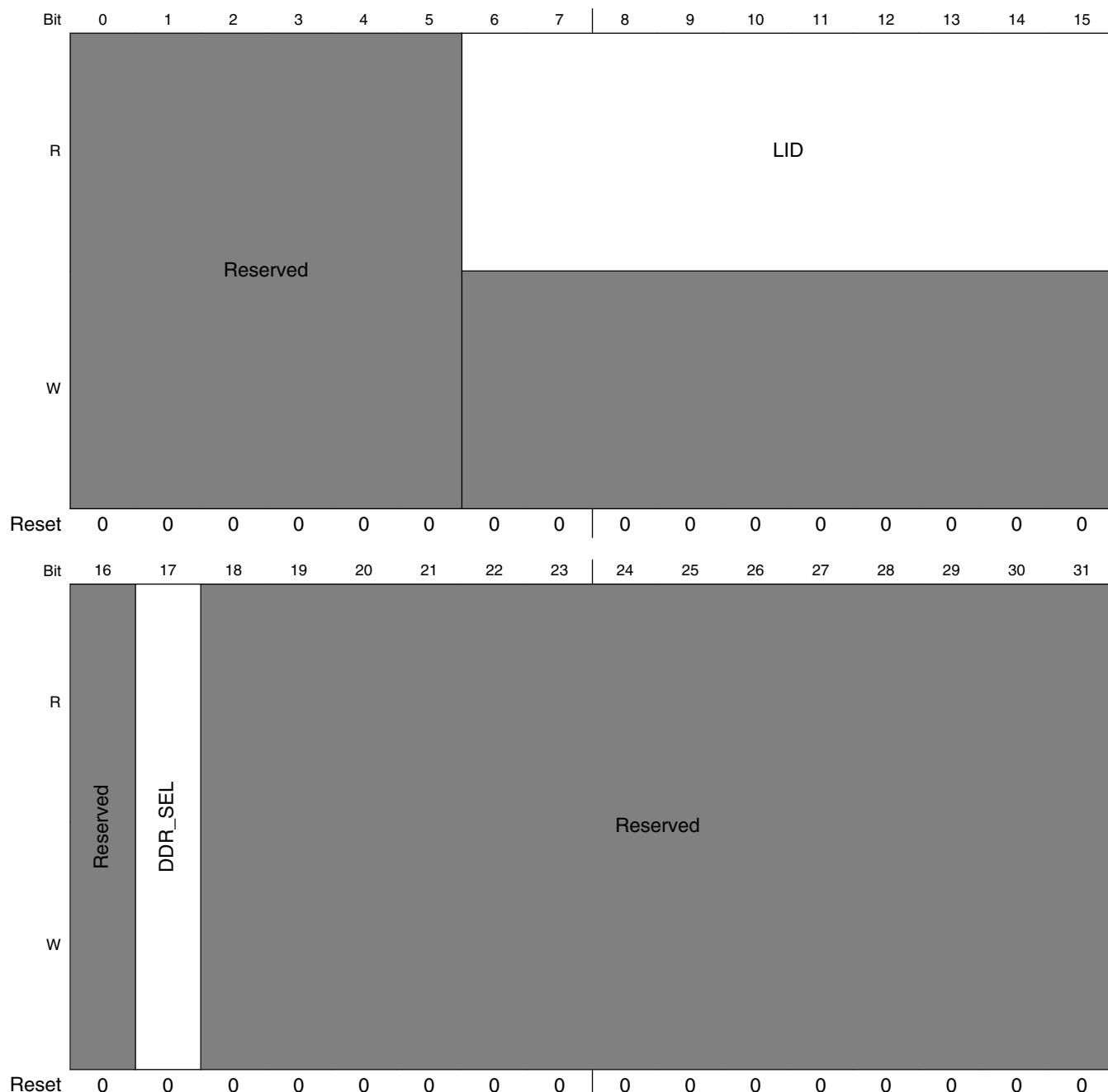
Signals_SRDSRCALCR field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 CALLRST	Reset to the calibration logic in the left endcap cell During POR (warm reset sequence) it is active and is released when first PLL (of the 3 banks) come out of reset 0 Reset 1 Application Mode
5–31 -	This field is reserved. Reserved

3.6.6 SerDes General Register 0 (Signals_SRDSGR0)

SRDSGR0 contains the control/status bits used for the SerDes logic.

Address: E_A000h base + B0h offset = E_A0B0h



Signals_SRDSGR0 field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6–15 LID	SerDes Block identification Register.
16 -	This field is reserved. Reserved
17 DDR_SEL	Describes to SerDes block the value of the DDR supply being used by the SoC: 0 1.8 V Vdd 1 1.5 V Vdd
18–31 -	This field is reserved. Reserved

3.6.7 SerDes Protocol Converter Configuration Register 0 (Signals_SRDSPCCR0)

SRDSPCCR0 contains the protocol converter configuration on all lanes across the three banks in SerDes logic.

Address: E_A000h base + E0h offset = E_A0E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved	PEX1_CFG	Reserved	PEX2_CFG	Reserved	PEX3_CFG	Reserved									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved	SRIO1_CFG	Reserved	SRIO2_CFG	Reserved											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_SRDSPCCR0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 PEX1_CFG	PEX1 Configuration: 00 Disabled 01 x2 on lanes AB (Bank1) 10 x4 on lanes ABCD (Bank1) 11 x8 on lanes ABCDEFGH (Bank1)
4–5 -	This field is reserved. Reserved

Table continues on the next page...

Signals_SRDSPCCR0 field descriptions (continued)

Field	Description
6–7 PEX2_CFG	PEX2 Configuration: 00 Disabled 01 x2 on lanes EF (Bank1) 10 x4 on lanes EFGH (Bank1) 11 Reserved
8–9 -	This field is reserved. Reserved
10–11 PEX3_CFG	PEX3 Configuration: 00 Disabled 01 x2 on lanes CD (Bank1) 10 x4 on lanes ABCD (Bank2) 11 Reserved
12–17 -	This field is reserved. Reserved
18–19 SRIO1_CFG	SRIO1 Configuration: 00 Disabled 01 x1 on lane H (Bank1) 10 Reserved 11 x4 on lanes EFGH (Bank1)
20–21 -	This field is reserved. Reserved
22–23 SRIO2_CFG	SRIO2 Configuration: 00 Disabled 01 x1 on lane F (Bank1) 10 Reserved 11 x4 on lanes ABCD (Bank1)
24–31 -	This field is reserved. Reserved

3.6.8 SerDes Protocol Converter Configuration Register 1 (Signals_SRDSPCCR1)

SRDSPCCR1 contains the protocol converter configuration on all lanes across the three banks in SerDes logic.

Address: E_A000h base + E4h offset = E_A0E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	SGMII1_CFG	Reserved	SGMII2_CFG	Reserved	SGMII3_CFG	Reserved	SGMII4_CFG	SGMII5_CFG	SGMII6_CFG	SGMII7_CFG	SGMII8_CFG				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	XAU1_CFG	Reserved	XAU12_CFG	Reserved						AURORA1_CFG	Reserved				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_SRDSPCCR1 field descriptions

Field	Description
0 -	This field is reserved. Reserved
1 SGMII1_CFG	SGMII1 Configuration: 0 Disabled 1 x1 on lanes A (Bank3)
2 -	This field is reserved. Reserved
3 SGMII2_CFG	SGMII2 Configuration: 0 Disabled 1 x1 on lanes B (Bank3)

Table continues on the next page...

Signals_SRDSPCCR1 field descriptions (continued)

Field	Description
4 -	This field is reserved. Reserved
5 SGMII3_CFG	SGMII3 Configuration: 0 Disabled 1 x1 on lanes C (Bank3)
6 -	This field is reserved. Reserved
7 SGMII4_CFG	SGMII4 Configuration: 0 Disabled 1 x1 on lanes D (Bank3)
8–9 SGMII5_CFG	SGMII5 Configuration: 00 Disabled 01 x1 on lane E (Bank 1) 10 x1 on lane A (Bank2) 11 Reserved
10–11 SGMII6_CFG	SGMII6 Configuration: 00 Disabled 01 x1 on lane F (Bank 1) 10 x1 on lane B (Bank2) 11 Reserved
12–13 SGMII7_CFG	SGMII7 Configuration: 00 Disabled 01 x1 on lane G (Bank 1) 10 x1 on lane C (Bank2) 11 Reserved
14–15 SGMII8_CFG	SGMII8 Configuration: 00 Disabled 01 x1 on lane H (Bank 1) 10 x1 on lane D (Bank2) 11 Reserved
16 -	This field is reserved. Reserved
17 XAUI1_CFG	XAUI1 Configuration: 0 Disabled 1 x4 on lanes ABCD (Bank3)
18 -	This field is reserved. Reserved
19 XAUI2_CFG	XAUI2 Configuration: 0 Disabled 1 x4 on lanes ABCD (Bank2)

Table continues on the next page...

Signals_SRDSPCCR1 field descriptions (continued)

Field	Description
20–25 -	This field is reserved. Reserved
26–27 AURORA1_CFG	AURORA1 Configuration: 00 Disabled 01 x2 on lanes IJ (Bank1) 10 Reserved 11 Reserved
28–31 -	This field is reserved. Reserved

3.6.9 SerDes Protocol Converter Configuration Register 2 (Signals_SRDSPCCR2)

SRDSPCCR2 contains the protocol converter configuration on all lanes across the three banks in SerDes logic.

Address: E_A000h base + E8h offset = E_A0E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	RST_PEX1	RST_PEX2	RST_PEX3	Reserved	RST_SRIO1	RST_SRIO2		Reserved	RST_XGM1	RST_XGM2						Reserved
Reset	1	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	RST_SGM1	RST_SGM2	RST_SGM3	RST_SGM4	RST_SGM5	RST_SGM6	RST_SGM7	RST_SGM8								Reserved
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Signals_SRDSPCCR2 field descriptions

Field	Description
0 RST_PEX1	Asynchronous Reset to PEX1 0 Reset 1 Application Mode
1 RST_PEX2	Asynchronous Reset to PEX2 0 Reset 1 Application Mode
2 RST_PEX3	Asynchronous Reset to PEX3 0 Reset 1 Application Mode
3 -	This field is reserved. Reserved
4 RST_SRIO1	Asynchronous Reset to SRIO1 0 Reset 1 Application Mode
5 RST_SRIO2	Asynchronous Reset to SRIO2 0 Reset 1 Application Mode
6–7 -	This field is reserved. Reserved
8 RST_XGM1	Asynchronous Reset to XGM1 0 Reset 1 Application Mode
9 RST_XGM2	Asynchronous Reset to XGM2 0 Reset 1 Application Mode
10–15 -	This field is reserved. Reserved
16 RST_SGM1	Asynchronous Reset to SGM1 0 Reset 1 Application Mode
17 RST_SGM2	Asynchronous Reset to SGM2 0 Reset 1 Application Mode
18 RST_SGM3	Asynchronous Reset to SGM3 0 Reset 1 Application Mode
19 RST_SGM4	Asynchronous Reset to SGM4

Table continues on the next page...

Signals_SRDSPCCR2 field descriptions (continued)

Field	Description
	0 Reset 1 Application Mode
20 RST_SGM5	Asynchronous Reset to SGM5 0 Reset 1 Application Mode
21 RST_SGM6	Asynchronous Reset to SGM6 0 Reset 1 Application Mode
22 RST_SGM7	Asynchronous Reset to SGM7 0 Reset 1 Application Mode
23 RST_SGM8	Asynchronous Reset to SGM8 0 Reset 1 Application Mode
24–31 -	This field is reserved. Reserved

3.6.10 Bank 1 General Control Register 0 - Lane x (Signals_B1GCRn0)

The bank 1 general control register 0 contains the functional control bits for the SerDes logic on lanes A to J. Individual lanes can be powered down using B1GCRA0[PD]-B1GCRJ0[PD]. A complete SerDes reset is necessary to activate a lane from a powered down state.

Address: E_A000h base + 400h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		RRAT_SEL	Reserved	TRAT_SEL				HALT_RCLK	RRST	TRST	Reserved	PD	X3S	IACC_EN	1STLANE
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		TTRM_VM_SEL		PROTS											Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_B1GCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 RRAT_SEL	Receiver speed selection: 00 Full speed 01 Half speed 10 Quarter speed 11 Reserved
4–5 -	This field is reserved. Reserved

Table continues on the next page...

Signals_B1GCRn0 field descriptions (continued)

Field	Description
6–7 TRAT_SEL	Transmitter speed selection: 00 Full speed 01 Half speed 10 Quarter speed 11 Reserved
8 HALT_RCLK	During auto negotiation halts receiver clock outputs while RX speed changed via B1GCR(lane)0[RRAT_SEL] 0 = rx_clk clock outputs always running Recommended setting per protocol: PCI Express: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0
9 RRST	Receiver reset. Writing a 0 to this bit resets the receiver. Note that software must explicitly write a 1 to remove the receiver from reset.
10 TRST	Transmitter Reset. Writing a 0 to this bit resets the transmitter/receiver. Note that with one exception, software must explicitly write a 1 to remove the transmitter from reset. The one exception is during power-on reset, the transmitter is initially in reset, but the bit is set to 1 automatically when the PLL locks.
11 -	This field is reserved. Reserved
12 PD	Lane powerdown Coming out of POR, PD is set and is cleared at the same time srds_pd deasserts (during warm reset sequence) Lane Disable is also combined to power down the lane during POR 0 Normal 1 Powerdown lane
13 X3S	Recommended setting per protocol: PCI Express: 1 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0 Lane transmitter three-state 0 Normal 1 The transmitter output is disabled and place in a three-state condition
14 IACC_EN	Used to set on-chip AC coupling in the receiver in the lane Recommended setting per protocol:

Table continues on the next page...

Signals_B1GCRn0 field descriptions (continued)

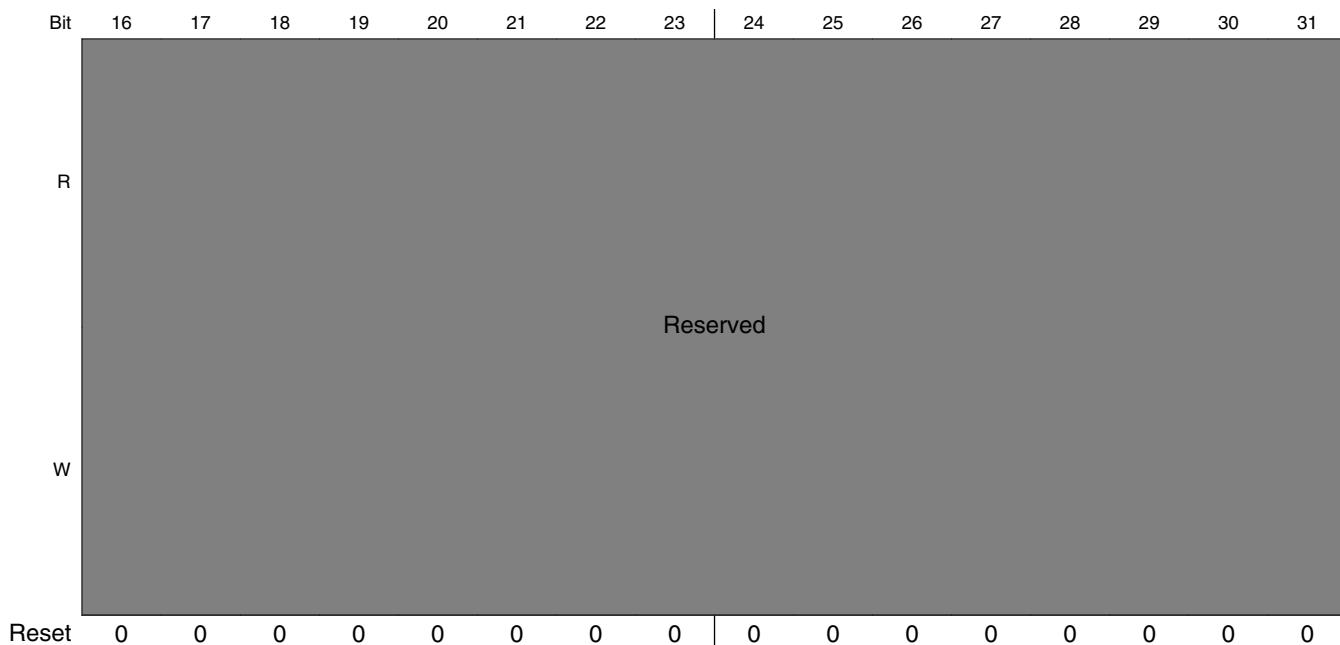
Field	Description
	PCI Express: 1 SGMII: 1 SRIO: 1 0 Disable on-chip AC coupling 1 Enable on-chip AC coupling
15 1STLANE	Indicates this lane is the first of a group of lanes. This is configured at POR based on the protocol selected - hardcoded 0 tx_clk not used by the protocol converter 1 tx_clk used by the protocol converter
16–17 -	This field is reserved. Reserved
18–19 TTRM_VM_SEL	Select RX termination common mode: Recommended settings per protocol: 00 PCI Express 01 SRIO 01 SGMII 01 XAUI 01 Aurora 00 If srds_pd or lane_pd asserted, common mode in HiZ, else common mode is calibrated termination to xcorevss 01 Common mode is always calibrated termination to xcorevss 10 Common mode is HiZ, RX termination is uncalibrated 120 Ω differential 11 Common mode is 0.7*xcorevdd through 3kΩ, RX termination is uncalibrated 120 Ω differential
20–23 PROTS	Lane Protocol Select 0000 PCIe 0001 SGMII 0010 Reserved 0011 SRIO 0100 XAUI 0101 Aurora 0110 Reserved 0111 Reserved 1xxx Reserved
24–31 -	This field is reserved. Reserved

3.6.11 Bank 1 General Control Register 1 - Lane x (Signals_B1GCRn1)

The bank 1 general control register 1 contains the functional control bits for the SerDes logic on lanes A to J.

Address: E_A000h base + 404h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	0	0	TDET	REIDL	Reserved	OPAD_CTL	0	TDET_EN	REIDL_EN	Reserved	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Signals_B1GCRn1 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2 TDET	Transmit Detection Flag for Lane
3 REIDL	Lane receiver detected Vdff p-p less than value selected via B1GCR(lane)1[11:13]
4 -	This field is reserved. Reserved
5 OPAD_CTL	TX Output pad control signal for common mode Recommended setting per protocol: 0 0 Transmitter Enabled 1 Force Transmitter Output to Common
6 TDET_EN	Enable transmitter detection of rcvr (for PCI-Express)
7 REIDL_EN	When asserted, places the specified lane into Receiver Elec Idle state Recommended setting per protocol: PCI Express: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0

Table continues on the next page...

Signals_B1GCRn1 field descriptions (continued)

Field	Description
	<p>0 The specified lane is not "forced" into receive electrical idle state.</p> <p>1 For PCI Express, place the specified lane into electrical idle state. For Aurora, shut down RX side of SerDes for TX only lanes.</p>
8–10 -	This field is reserved. Reserved
11–15 REIDL_CTL	<p>REIDL_CTL[4:2]. Represents the threshold for the Loss of Signal detector within the receive amplifier</p> <p>000 Loss of signal detect function is disabled</p> <p>001 Default SGMII levels (Low=30mV, High=100mV)</p> <p>010 Intermediate level (Low=38mV, High=120mV)</p> <p>011 Intermediate level (Low=50mV, High=150mV)</p> <p>100 Default PEX levels (Low=65mV, High=175mV)</p> <p>101 Reserved</p> <p>110 Intermediate level (Low=88mV, High=225mV)</p> <p>111 Intermediate level (Low=100mV, High=250mV)</p> <p>Recommended setting per protocol:</p> <p>PCI Express: 100</p> <p>SGMII: 001</p> <p>SRIO: 000</p> <p>XAUI: 000</p> <p>Aurora: 000</p> <p>REIDL_CTL[1:0]. Used to set filter depths appropriate to protocol and bit rate.</p> <p>For PCI-Express 5 Gbps:</p> <p>00 Exit from Idle ~88UI (Application Mode)</p> <p>01 Exit from Idle ~88UI and Unexpected Idle Detect ~10us</p> <p>10 Exit from Idle ~48UI and Unexpected Idle Detect ~1us</p> <p>11 Bypass</p> <p>For PCI-Express 2.5 Gbps and SGMII:</p> <p>00 Exit from Idle ~88UI and Unexpected Idle Detect ~1us (Application Mode)</p> <p>01 Exit from Idle ~88UI and Unexpected Idle Detect ~10us</p> <p>10 Exit from Idle ~88UI</p> <p>11 Bypass</p> <p>Recommended setting per protocol: 00</p>
16–31 -	This field is reserved. Reserved

3.6.12 Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRn0)

The SerDes receive equalization control register 0 contains the functional control bits for the SerDes on bank 1, lanes A to J.

Address: E_A000h base + 410h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			GK2OVD				Reserved			GK3OVD					
W																
Reset	0	0	0	n	n	n	n	n	0	0	0	0	n	n	n	n
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GK2OVD_EN	GK3OVD_EN	OSETOVD_EN	Reserved			ZERSET	Reserved		OSETOVD						
W	n	n	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Reset	n	n	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Signals_B1RECRn0 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 GK2OVD	Overrides adaptive equalization k2 gain control. Recommended settings per protocol: PCI Express: 00000 SGMII: 01111 XAUI: 00000 SRIO 3.125 Gbps: 00000 SRIO 2.5 Gbps: 00000 Aurora 5.0 Gbps: 00000 Aurora 2.5 Gbps: 00000 00000 Maximum Gaink2 equalization 01111 Minimum Gaink2 equalization
8–11 -	This field is reserved. Reserved
12–15 GK3OVD	Overrides adaptive equalization k3 gain control.

Table continues on the next page...

Signals_B1RECRn0 field descriptions (continued)

Field	Description
	<p>Recommended settings per protocol</p> <p>PCI Express: 0000</p> <p>SGMII: 1111</p> <p>XAUI: 0000</p> <p>SRIO 3.125 Gbps: 0000</p> <p>SRIO 2.5 Gbps: 0000</p> <p>Aurora 5.0 Gbps: 0000</p> <p>Aurora 2.5 Gbps: 0000</p> <p>0000 Maximum Gaink3 equalization 1111 Minimum Gaink3 equalization</p>
16 GK2OVD_EN	<p>Enables GK2OVD to override adaptive equalization k2 gain control.</p> <p>Recommended settings per protocol:</p> <p>PCI Express: 0</p> <p>SGMII: 1</p> <p>XAUI: 0</p> <p>SRIO 3.125 Gbps: 0</p> <p>SRIO 2.5 Gbps: 0</p> <p>Aurora 5.0 Gbps: 0</p> <p>Aurora 2.5 Gbps: 0</p> <p>0 Use rxeq adaption derived gaink2 1 Fix gaink2 according to B1RECR(lane)0[GK2OVD]</p>
17 GK3OVD_EN	<p>Enables GK3OVD to override adaptive equalization k3 gain control.</p> <p>Recommended settings per protocol:</p> <p>PCI Express: 0</p> <p>SGMII: 1</p> <p>XAUI: 0</p> <p>SRIO 3.125 Gbps: 0</p> <p>SRIO 2.5 Gbps: 0</p> <p>Aurora 5.0 Gbps: 0</p> <p>Aurora 2.5 Gbps: 0</p> <p>0 Use rxeq adaption derived gaink3 1 Fix gaink3 according to B1RECR(lane)0[GK3OVD]</p>
18 OSETOVD_EN	<p>Enables OSETOVD to override adaptive equalization offset control.</p> <p>Recommended settings per protocol:</p> <p>PCI Express: 0</p> <p>SGMII: 0</p> <p>XAUI: 0</p>

Table continues on the next page...

Signals_B1RECRn0 field descriptions (continued)

Field	Description
	SRIO: 0 Aurora: 0 0 Use rxreq adaption derived offset control 1 Fix equalization offset control according to B1RECR(lane)0[OSETOVD]
19–21 -	This field is reserved. Reserved
22–23 ZERSET	Adjusts adaptive equalization zero settings. Recommended settings per protocol: PCI Express: 00 SGMII: 00 XAUI: 00 SRIO: 00 Aurora: 00
24–25 -	This field is reserved. Reserved
26–31 OSETOVD	Overrides adaptive equalization offset control. Recommended settings per protocol: PCI Express: 01_1111 SGMII: 01_1111 XAUI: 01_1111 SRIO: 01_1111 Aurora: 01_1111 000000 + Maximum imposed offset 011111 0 Imposed offset 111111 - Maximum imposed offset

3.6.13 Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRn0)

The SerDes receive equalization control register 0 contains the functional control bits for the SerDes on Bank 2, lanes A to D.

Address: E_A000h base + 410h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			GK2OVD				Reserved			GK3OVD					
W																
Reset	0	0	0	n	n	n	n	n	0	0	0	0	n	n	n	n
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GK2OVD_EN	GK3OVD_EN	OSETOVD_EN	Reserved			ZERSET	Reserved			OSETOVD					
W	n	n	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Reset	n	n	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Signals_B2RECRn0 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 GK2OVD	Overrides adaptive equalization k2 gain control. Recommended settings per protocol: PCI Express: 00000 SGMII: 01111 XAUI: 00000 00000 Maximum Gaink2 equalization 01111 Minimum Gaink2 equalization
8–11 -	This field is reserved. Reserved
12–15 GK3OVD	Overrides adaptive equalization k3 gain control. Recommended settings per protocol PCI Express: 0000 SGMII: 1111 XAUI: 0000

Table continues on the next page...

Signals_B2RECRn0 field descriptions (continued)

Field	Description
	0000 Maximum Gaink3 equalization 1111 Minimum Gaink3 equalization
16 GK2OVD_EN	Enables GK2OVD to override adaptive equalization k2 gain control. Recommended settings per protocol: PCI Express: 0 SGMII: 1 XAUI: 0 0 Use rxreq adaption derived gaink2 1 Fix gaink2 according to B2RECR(lane)0[GK2OVD]
17 GK3OVD_EN	Enables GK3OVD to override adaptive equalization k3 gain control. Recommended settings per protocol: PCI Express: 0 SGMII: 1 XAUI: 0 0 Use rxreq adaption derived gaink3 1 Fix gaink3 according to B2RECR(lane)0[GK3OVD]
18 OSETOVD_EN	Enables OSETOVD to override adaptive equalization offset control. Recommended settings per protocol: PCI Express: 0 SGMII: 0 XAUI: 0 0 Use rxreq adaption derived offset control 1 Fix equalization offset control according to B2RECR(lane)0[OSETOVD]
19–21 -	This field is reserved. Reserved
22–23 ZERSET	Adjusts adaptive equalization zero settings. Recommended settings per protocol: PCI Express: 00 SGMII: 00 XAUI: 00
24–25 -	This field is reserved. Reserved
26–31 OSETOVD	Overrides adaptive equalization offset control. Recommended settings per protocol: PCI Express: 01_1111 SGMII: 01_1111 XAUI: 01_1111 000000 + Maximum imposed offset

Table continues on the next page...

Signals_B2RECRn0 field descriptions (continued)

Field	Description
	011111 0 Imposed offset 111111 - Maximum imposed offset

3.6.14 Bank 3 Receive Equalization Control Register 0 Lane n (Signals_B3RECRn0)

The SerDes receive equalization control register 0 contains the functional control bits for the SerDes on Bank 3, lanes A to D.

Address: E_A000h base + 410h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved			GK2OVD				Reserved			GK3OVD					
W																
Reset	0	0	0	n	n	n	n	n	0	0	0	0	n	n	n	n
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	GK2OVD_EN	GK3OVD_EN	OSETOVD_EN	Reserved			ZERSET	Reserved		OSETOVD						
W																
Reset	n	n	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Signals_B3RECRn0 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 GK2OVD	Overrides adaptive equalization k2 gain control. Recommended settings per protocol: SGMII: 01111 XAUI: 00000 00000 Maximum Gaink2 equalization 01111 Minimum Gaink2 equalization
8–11 -	This field is reserved. Reserved
12–15 GK3OVD	Overrides adaptive equalization k3 gain control.

Table continues on the next page...

Signals_B3RECRn0 field descriptions (continued)

Field	Description
	<p>Recommended settings per protocol</p> <p>SGMII: 1111</p> <p>XAUI: 0000</p> <p>0000 Maximum Gaink3 equalization 1111 Minimum Gaink3 equalization</p>
16 GK2OVD_EN	<p>Enables GK2OVD to override adaptive equalization k2 gain control.</p> <p>Recommended settings per protocol:</p> <p>SGMII: 1</p> <p>XAUI: 0</p> <p>0 Use rxeq adaption derived gaink2 1 Fix gaink2 according to B3RECR(lane)0[GK2OVD]</p>
17 GK3OVD_EN	<p>Enables GK3OVD to override adaptive equalization k3 gain control.</p> <p>Recommended settings per protocol:</p> <p>SGMII: 1</p> <p>XAUI: 0</p> <p>0 Use rxeq adaption derived gaink3 1 Fix gaink3 according to B3RECR(lane)0[GK3OVD]</p>
18 OSETOVD_EN	<p>Enables OSETOVD to override adaptive equalization offset control.</p> <p>Recommended settings per protocol:</p> <p>SGMII: 0</p> <p>XAUI: 0</p> <p>0 Use rxeq adaption derived offset control 1 Fix equalization offset control according to B3RECR(lane)0[OSETOVD]</p>
19–21 -	This field is reserved. Reserved
22–23 ZERSET	<p>Adjusts adaptive equalization zero settings.</p> <p>Recommended settings per protocol:</p> <p>SGMII: 00</p> <p>XAUI: 00</p>
24–25 -	This field is reserved. Reserved
26–31 OSETOVD	<p>Overrides adaptive equalization offset control.</p> <p>Recommended settings per protocol:</p> <p>SGMII: 01_1111</p> <p>XAUI: 01_1111</p> <p>000000 + Maximum imposed offset 011111 0 Imposed offset 111111 - Maximum imposed offset</p>

3.6.15 Bank 1 Transmit Equalization Control Register 0 - Lane x (Signals_B1TECRn0)

The transmit equalization control register 0 contains the functional control bits for the SerDes logic on lanes A to J.

Address: E_A000h base + 418h offset + (64d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved	TEQ_TYPE	Reserved	RATIO_PREQ				Reserved	RATIO_PST1Q							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				RATIO_PST2Q				Reserved	AMP_RED						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Signals_B1TECRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 TEQ_TYPE	Recommended settings per protocol converter: SGMII: 00 All other protocols: 01 Selects amount/type of Transmit Equalization 00 No TX Equalization 01 2 Levels of TX Equalization (+1 postcursor) 10 3 Levels of TX Equalization (+1 precursor and +1 postcursor) 11 4 Levels of TX Equalization (+1 precursor and +2 postursors)
4–5 -	This field is reserved. Reserved
6–9 RATIO_PREQ	Ratio of full swing transition bit to precursor
10–11 -	This field is reserved. Reserved
12–15 RATIO_PST1Q	Ratio of full swing transition bit to postcursor for 2-tap equalization B1TECR(lane)0[12]: 0 Negative Sign 1 Positive Sign B1TECR(lane)0[13:15]: 000 No Equalization 001 1.09x 010 1.20x

Table continues on the next page...

Signals_B1TECRn0 field descriptions (continued)

Field	Description
	011 1.33x 100 1.50x 101 1.71x 110 2.00x 111 Reserved Recommend settings per protocol converter: 2.5 Gbps PCI Express 1100 5.0 Gbps PCI Express 1110 SRIO: 1011 SGMII: 1000 XAUI: 1100 Aurora: 1100
16–19 -	This field is reserved. Reserved
20–23 RATIO_PST2Q	ratio of full swing transition bit to postcursor for 4-tap equalization
24–25 -	This field is reserved. Reserved
26–31 AMP_RED	Amount of amplitude reduction for all bits B1TECR(lane)0[26] = Reserved B1TECR(lane)0[27:31]: 00000 = Full Swing Vdiffpk- pk 01000 = 0.75x Full Swing Vdiffpk- pk 01011 = 0.68x Full Swing Vdiffpk- pk 10011 = 0.50x Full Swing Vdiffpk- pk Recommend settings per protocol converter: PCI Express: 000000 SRIO: 000000 SGMII: 001011 XAUI: 000000 Aurora: 000000

3.6.16 Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRn0)

The transition tracking loop control register contains the functional control bits used for the Transition Tracking Loop (TTL) logic for lanes A to J.

Address: E_A000h base + 420h offset + (64d × i), where i=0d to 9d

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

Signals_B1TTLCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 FLT_SEL	TTL channel filter bandwidth selection NOTE: The reset value for FLT_SEL depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 01_1011. For lanes configured for all other protocols (non-PCI Express), the reset value is either 01_1000 (P4080 Rev 2) or 00_0011 (P4080 Rev 3). FLT_SEL[2:3] - Selects the gain 'Kfr' in the CDR for lane 00 2 ⁻⁵ 01 2 ⁻⁶ 10, 11 Reserved Recommended setting per protocol:

Table continues on the next page...

Signals_B1TTLCRn0 field descriptions (continued)



Field	Description
	<p>PCI Express: 01 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 01 (Note that this setting differs from the reset value.) FLT_SEL[4:5] - Selects the gain 'Kph' in the CDR for lane</p> <p>00 Reserved 01 2^-7 10 2^-8 11 Reserved</p> <p>Recommended setting per protocol: PCI Express: 10 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 10 (Note that this setting differs from the reset value.) FLT_SEL[6:7] - Select Markov filter lengths in Transition Tracking Loop for Lane.</p> <p>For B1TTLCR(lane)0[PM_DIS] = 0 00, 01, 10, 11 TBD</p> <p>For B1TTLCR(lane)0[PM_DIS] = 1 00 +/-200 ppm 01 +/-400 ppm 10 +/-500 ppm 11 +/-750 ppm</p> <p>Recommended setting per protocol: PCI Express 11 SGMII, SRIO, XAUI, Aurora - P4080 Rev2: 11 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev3: 00 (Note that this setting differs from the reset value.)</p>
8–16 -	This field is reserved. Reserved
17 PM_DIS	<p>Disable slow-phase-modulation portion of transition tracking loop (TTL)</p> <p>NOTE: The reset value for PM_DIS depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 0. For lanes configured for all other protocols (non-PCI Express), the reset value is either 0 (P4080 Rev 2) or 1 (P4080 Rev 3).</p> <p>Recommended setting per protocol: PCI Express: 0 SGMII, SRIO, XAUI, Aurora: 0 (Note that this setting differs from the reset value on P4080 Rev3.)</p>
18–30 -	This field is reserved. Reserved
31 -	<p>This field is reserved. Reserved</p> <p>Recommended settings per protocol: PCI Express: 0</p>

Table continues on the next page...

Signals_B1TTLCRn0 field descriptions (continued)

Field	Description
	SGMII, SRIO, XAUI, Aurora: 1 (Note that this setting differs from the reset value)

3.6.17 Bank 2 General Control Register 0 - Lane x (Signals_B2GCRn0)

The general control register 0 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 800h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved		RRAT_SEL		Reserved		TRAT_SEL		HALT_RCLK		RRST	TRST		Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	Reserved		TTRM_VM_SEL			PROTS								Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_B2GCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 RRAT_SEL	Receiver speed selection: 00 Full-speed 01 Half-speed 10 Quarter-speed 11 Reserved

Table continues on the next page...

Signals_B2GCRn0 field descriptions (continued)

Field	Description
4–5 -	This field is reserved. Reserved
6–7 TRAT_SEL	Transmitter speed selection: 00 Full-speed 01 Half-speed 10 Quarter-speed 11 Reserved
8 HALT_RCLK	During auto negotiation halts receiver clock outputs while RX speed changed via B2GCR(lane)0[RRAT_SEL] 0 rx_clk clock outputs always running Recommended setting per protocol: PCI Express: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0
9 RRST	Receiver reset. Writing a 0 to this bit resets the receiver. Note that software must explicitly write a 1 to remove the receiver from reset.
10 TRST	Transmitter Reset. Writing a 0 to this bit resets the transmitter/receiver. Note that with one exception, software must explicitly write a 1 to remove the transmitter from reset. The one exception is during power-on reset, the transmitter is initially in reset, but the bit is set to 1 automatically when the PLL locks.
11 -	This field is reserved. Reserved
12 PD	Lane powerdownComing out of POR, it is asserted and deasserts same time srds_pd deasserts (during warm reset sequence) Lane Disable is also combined to power down the lane during POR 0 Normal 1 Powerdown lane
13 X3S	Recommended setting per protocol: PCI Express: 1 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0 Lane transmitter three-state 0 Normal 1 The transmitter output is disabled and place in a three-state condition
14 IACC_EN	Recommended setting per protocol: PCI Express: 1

Table continues on the next page...

Signals_B2GCRn0 field descriptions (continued)

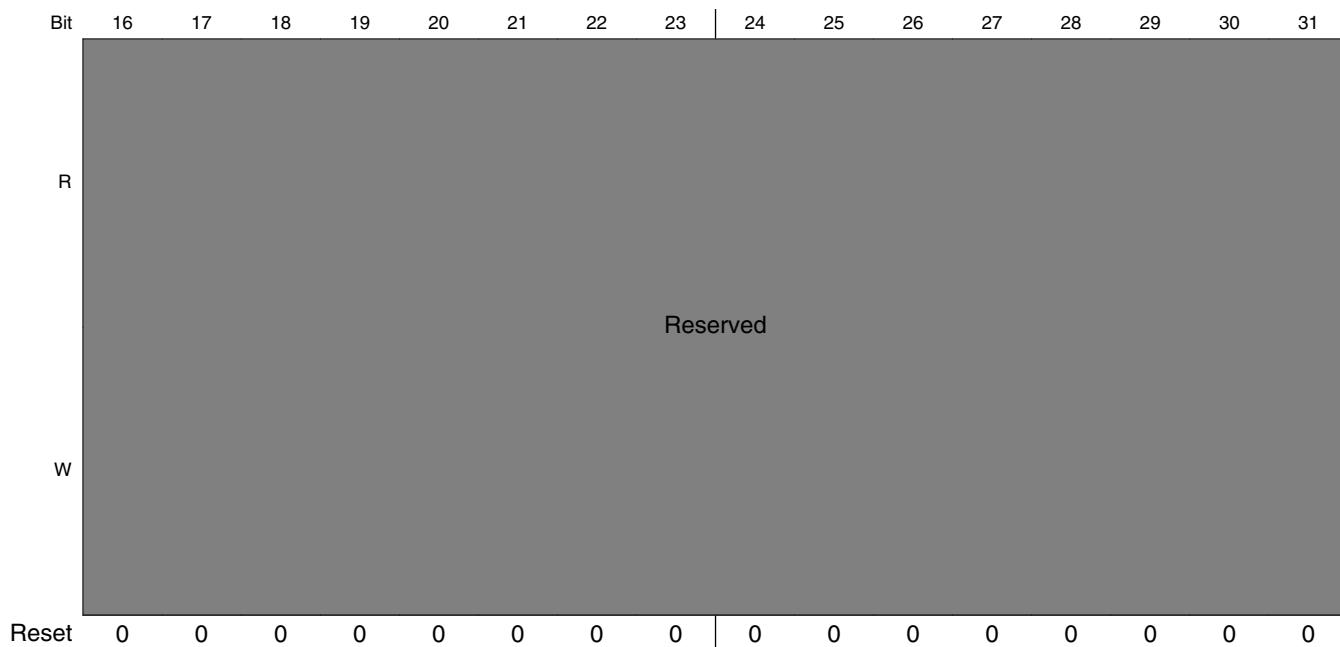
Field	Description
	<p>SGMII: 1 SRIO: 1</p> <p>Used to set on-chip AC coupling in the receiver in the lane</p> <p>0 Disable on-chip AC coupling 1 Enable on-chip AC coupling</p>
15 1STLANE	<p>Indicates this lane is the first of a group of lanes. This is configured at POR based on the protocol selected.</p> <p>0 tx_clk not used by the protocol converter 1 tx_clk used by the protocol converter</p>
16–17 -	This field is reserved. Reserved
18–19 TTRM_VM_SEL	<p>Select RX termination common mode:</p> <p>Recommended settings per Protocol:</p> <ul style="list-style-type: none"> 00 PCI Express 01 SRIO 01 SGMII 01 XAUI 01 Aurora <p>00 If srds_pd or lane_pd asserted, common mode in HiZ, else common mode is calibrated termination to xccorevss</p> <p>01 Common mode is always calibrated termination to xccorevss</p> <p>10 Common mode is HiZ, RX termination is uncalibrated $120\ \Omega$ differential</p> <p>11 Common mode is $0.7 \times \text{xccorevdd}$ through $3k\Omega$, RX termination is uncalibrated $120\ \Omega$ differential</p>
20–23 PROTS	<p>Lane Protocol Select</p> <ul style="list-style-type: none"> 0000 PCIe 0001 SGMII 0010 Reserved 0011 SRIO 0100 XAUI 0101 Aurora 0110 Reserved 0111 Reserved 1xxx Reserved
24–31 -	This field is reserved. Reserved

3.6.18 Bank 2 General Control Register 1 - Lane x (Signals_B2GCRn1)

The general control register 1 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 804h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		TDET	REIDL	Reserved	OPAD_CTL		TDET_EN	REIDL_EN	Reserved						REIDL_CTL
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Signals_B2GCRn1 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2 TDET	Transmit Detection Flag for Lane
3 REIDL	Lane receiver detected Vdff p-p less than value selected via B2GCR(lane)1[11:13]
4 -	This field is reserved. Reserved
5 OPAD_CTL	TX Output pad control signal for common mode Recommended setting per protocol: 0 0 Transmitter Enabled 1 Force Transmitter Output to Common
6 TDET_EN	Enable transmitter detection of rcvr (for PCI-Express)
7 REIDL_EN	When asserted, places the specified lane into Receiver Elec Idle state Recommended setting per protocol: PCIe: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0

Table continues on the next page...

Signals_B2GCRn1 field descriptions (continued)

Field	Description
	<p>0 The specified lane is not "forced" into receive electrical idle state.</p> <p>1 For PCI Express, place the specified lane into electrical idle state. For Aurora, shut down RX side of SerDes for TX only lanes.</p>
8–10 -	This field is reserved. Reserved
11–15 REIDL_CTL	<p>REIDL_CTL[4:2]. Represents the threshold for the Loss of Signal detector within the receive amplifier</p> <p>000 Loss of signal detect function is disabled</p> <p>001 Default SGMII levels (Low=30mV, High=100mV)</p> <p>010 Intermediate level (Low=38mV, High=120mV)</p> <p>011 Intermediate level (Low=50mV, High=150mV)</p> <p>100 Default PEX levels (Low=65mV, High=175mV)</p> <p>101 Reserved</p> <p>110 Intermediate level (Low=88mV, High=225mV)</p> <p>111 Intermediate level (Low=100mV, High=250mV)</p> <p>Recommended setting per protocol:</p> <p>PCI Express: 100</p> <p>SGMII: 001</p> <p>SRIO: 000</p> <p>XAUI: 000</p> <p>Aurora: 000</p> <p>REIDL_CTL[1:0]. Used to set filter depths appropriate to protocol and bit rate.</p> <p>For PCI-Express 5 Gbps:</p> <p>00 Exit from Idle ~88UI (Application Mode)</p> <p>01 Exit from Idle ~88UI and Unexpected Idle Detect ~10us</p> <p>10 Exit from Idle ~48UI and Unexpected Idle Detect ~1us</p> <p>11 Bypass</p> <p>For PCI-Express 2.5 Gbps and SGMII:</p> <p>00 Exit from Idle ~88UI and Unexpected Idle Detect ~1us (Application Mode)</p> <p>01 Exit from Idle ~88UI and Unexpected Idle Detect ~10us</p> <p>10 Exit from Idle ~88UI</p> <p>11 Bypass</p> <p>Recommended setting per protocol: 00</p> <p>SGMII: 001</p>
16–31 -	This field is reserved. Reserved

3.6.19 Bank 2 Transmit Equalization Control Register 0 - Lane x (Signals_B2TECRn0)

The transmit equalization control register 0 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 818h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	TEQ_TYPE	Reserved	RATIO_PREQ				Reserved	RATIO_PST1Q							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				RATIO_PST2Q				Reserved	AMP_RED						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Signals_B2TECRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 TEQ_TYPE	Recommended settings per protocol converter: SGMII: 00 All other protocols: 01 Selects amount/type of Transmit Equalization 00 No TX Equalization 01 2 Levels of TX Equalization (+1 postcursor) 10 3 Levels of TX Equalization (+1 precursor and +1 postcursor) 11 4 Levels of TX Equalization (+1 precursor and +2 postursors)
4–5 -	This field is reserved. Reserved
6–9 RATIO_PREQ	Ratio of full swing transition bit to precursor
10–11 -	This field is reserved. Reserved
12–15 RATIO_PST1Q	Ratio of full swing transition bit to postcursor for 2-tap equalization B2TECR(lane)0[12]: 0 Negative Sign 1 Positive Sign B2TECR(lane)0[13:15]: 000 No Equalization 001 1.09x

Table continues on the next page...

Signals_B2TECRn0 field descriptions (continued)

Field	Description
	010 1.20x 011 1.33x 100 1.50x 101 1.71x 110 2.00x 111 Reserved Recommended settings per protocol converter: 2.5 Gbps PCI Express: 1100 5.0 Gbps PCI Express: 1110 SRIO: 1011 SGMII: 1000 XAUI: 1100 Aurora: 1100
16–19 -	This field is reserved. Reserved
20–23 RATIO_PST2Q	ratio of full swing transition bit to postcursor for 4-tap equalization
24–25 -	This field is reserved. Reserved
26–31 AMP_RED	Amount of amplitude reduction for all bits B2TECR(lane)0[26] = Reserved B2TECR(lane)0[27:31]: 00000 = Full Swing Vdfffpk- pk 01000 = 0. 75x Full Swing Vdfffpk- pk 01011 = 0. 68x Full Swing Vdfffpk- pk 10011 = 0. 50x Full Swing Vdfffpk- pk Recommend settings per protocol converter: PCI Express: 000000 SRIO: 000000 SGMII: 001011 XAUI: 000000 Aurora: 000000

3.6.20 Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRn0)

The transition tracking loop control register contains the functional control bits used for the Transition Tracking Loop (TTL) logic for lanes A to D.

Address: E_A000h base + 820h offset + (64d × i), where i=0d to 3d

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

Signals_B2TTLCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 FLT_SEL	TTL channel filter bandwidth selection NOTE: The reset value for FLT_SEL depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 01_1011. For lanes configured for all other protocols (non-PCI Express), the reset value is either 00_0011 (P4080 Rev 3) or 01_1000 (P4080 Rev 2). FLT_SEL[2:3] - Selects the gain 'Kfr' in the CDR for lane 00 2^−5 01 - 2^−6 10, 11 - Reserved Recommended setting per protocol:

Table continues on the next page...

Signals_B2TTLCRn0 field descriptions (continued)

Field	Description
	<p>PCI Express: 01 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 01 (Note that this setting differs from the reset value.) FLT_SEL[4:5] - Selects the gain 'Kph' in the CDR for lane 00 Reserved 01 2^-7 10 2^-8 11 Reserved Recommended setting per protocol: PCI Express: 10 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 10 (Note that this setting differs from the reset value.) FLT_SEL[6:7] - Select Markov filter lengths in Transition Tracking Loop for Lane. For B2TTLCR(lane)0[PM_DIS] = 0 00, 01, 10, 11 - TBD For B2TTLCR(lane)0[PM_DIS] = 1 00 +/-200ppm 01 +/-400ppm 10 +/-500ppm 11 +/-750ppm Recommended setting per protocol: PCI Express: 11 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 11 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 00 (Note that this setting differs from the reset value.)</p>
8–16 -	This field is reserved. Reserved
17 PM_DIS	<p>Disable slow-phase-modulation portion of transition tracking loop (TTL) NOTE: The reset value for PM_DIS depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 0. For lanes configured for all other protocols (non-PCI Express), the reset value is either 0 (P4080 Rev 2) or 1 (P4080 Rev 3). Recommended setting per protocol" PCI Express: 0 SGMII, SRIO, XAUI, Aurora: 0 (Note that this setting differs from the reset value on P4080 Rev3.)</p>
18–30 -	This field is reserved. Reserved
31 -	<p>This field is reserved. Reserved Recommended settings per protocol: PCI Express: 0</p>

Table continues on the next page...

Signals_B2TTLCRn0 field descriptions (continued)

Field	Description
	SGMII, SRIO, XAUI, Aurora: 1 (Note that this setting differs from the reset value)

3.6.21 Bank 3 General Control Register 0 - Lane x (Signals_B3GCRn0)

The general control register 0 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 900h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved		RRAT_SEL		Reserved		TRAT_SEL		HALT_RCLK		RRST	TRST		Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	Reserved		TTRM_VM_SEL			PROTS								Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_B3GCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 RRAT_SEL	Receiver speed selection: 00 full-speed 01 half-speed 10 quarter-speed 11 reserved

Table continues on the next page...

Signals_B3GCRn0 field descriptions (continued)

Field	Description
4–5 -	This field is reserved. Reserved
6–7 TRAT_SEL	Transmitter speed selection: 00 full-speed 01 half-speed 10 quarter-speed 11 reserved
8 HALT_RCLK	During auto negotiation halts receiver clock outputs while RX speed changed via B3GCR(lane)0[RRAT_SEL] Recommended setting per protocol: PCI Express: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0 0 rx_clk clock outputs always running
9 RRST	Receiver reset. Writing a 0 to this bit resets the receiver. Note that software must explicitly write a 1 to remove the receiver from reset.
10 TRST	Transmitter Reset. Writing a 0 to this bit resets the transmitter/receiver. Note that with one exception, software must explicitly write a 1 to remove the transmitter from reset. The one exception is during power-on reset, the transmitter is initially in reset, but the bit is set to 1 automatically when the PLL locks.
11 -	This field is reserved. Reserved
12 PD	Lane powerdown Coming out of POR, it is asserted and deasserts same time srds_pd deasserts (during warm reset sequence) Lane Disable is also combined to power down the lane during POR 1 Powerdown lane
13 X3S	Lane transmitter three-state Recommended setting per protocol: PCI Express: 1 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0 0 Normal 1 The transmitter output is disabled and placed in a three-state condition
14 IACC_EN	Used to set on-chip AC coupling in the receiver in the lane

Table continues on the next page...

Signals_B3GCRn0 field descriptions (continued)

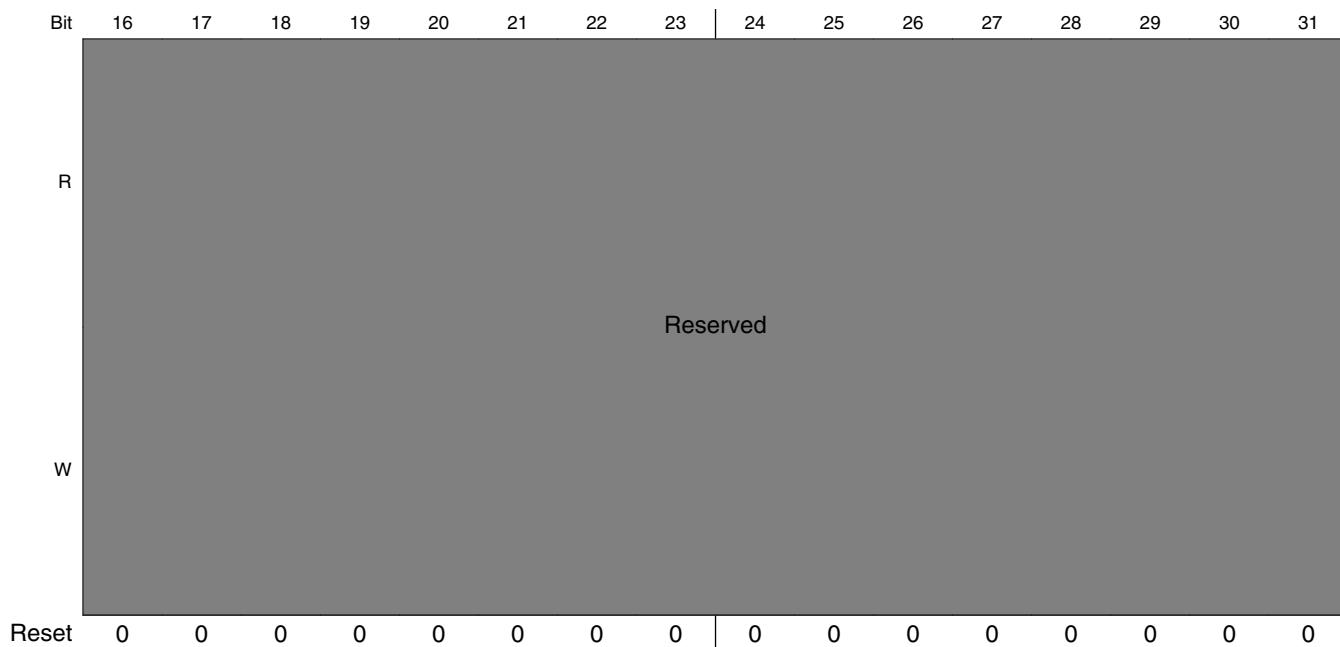
Field	Description																		
	<p>Recommended setting per protocol:</p> <p>PCI Express: 1</p> <p>SGMII: 1</p> <p>SRIO: 1</p> <p>0 Disable on-chip AC coupling 1 Enable on-chip AC coupling</p>																		
15 1STLANE	<p>Indicates this lane is the first of a group of lanes.. This is configured at POR based on the protocol selected.</p> <p>0 tx_clk not used by the protocol converter 1 tx_clk used by the protocol converter</p>																		
16–17 -	<p>This field is reserved. Reserved</p>																		
18–19 TTRM_VM_SEL	<p>Select RX termination common mode:</p> <p>Recommended settings per Protocol:</p> <p>00 PCIe 01 SRIO 01 SGMII 01 XAUI 01 Aurora</p> <p>00 If srds_pd or lane_pd asserted, common mode in HiZ, else common mode is calibrated termination to xccorevss 01 Common mode is always calibrated termination to xccorevss 10 Common mode is HiZ, RX termination is uncalibrated 120 Ω differential 11 Common mode is 0.7*xcorevdd through 3kΩs, RX termination is uncalibrated 120 Ω differential</p>																		
20–23 PROTS	<p>Lane Protocol Select</p> <table> <tbody> <tr><td>0000</td><td>PCIe</td></tr> <tr><td>0001</td><td>SGMII</td></tr> <tr><td>0010</td><td>Reserved</td></tr> <tr><td>0011</td><td>SRIO</td></tr> <tr><td>0100</td><td>XAUI</td></tr> <tr><td>0101</td><td>Aurora</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>1xxx</td><td>Reserved</td></tr> </tbody> </table>	0000	PCIe	0001	SGMII	0010	Reserved	0011	SRIO	0100	XAUI	0101	Aurora	0110	Reserved	0111	Reserved	1xxx	Reserved
0000	PCIe																		
0001	SGMII																		
0010	Reserved																		
0011	SRIO																		
0100	XAUI																		
0101	Aurora																		
0110	Reserved																		
0111	Reserved																		
1xxx	Reserved																		
24–31 -	<p>This field is reserved. Reserved</p>																		

3.6.22 Bank 3 General Control Register 1 - Lane x (Signals_B3GCRn1)

The general control register 1 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 904h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		TDET	REIDL	Reserved	OPAD_CTL		TDET_EN	REIDL_EN	Reserved						REIDL_CTL
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Signals_B3GCRn1 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2 TDET	Receive Detection Flag for Lane
3 REIDL	Lane receiver detected Vdff p-p less than value selected via B3GCR(lane)1[11:13]
4 -	This field is reserved. Reserved
5 OPAD_CTL	TX Output pad control signal for common mode Recommended setting per protocol: 0 0 Transmitter Enabled 1 Force Transmitter Output to Common
6 TDET_EN	Enable transmitter detection of rcvr (for PCI-Express)
7 REIDL_EN	When asserted, places the specified lane into Receiver Elec Idle state Recommended setting per protocol: PCI Express: 0 SGMII: 0 SRIO: 0 XAUI: 0 Aurora: 0

Table continues on the next page...

Signals_B3GCRn1 field descriptions (continued)

Field	Description
	<p>0 The specified lane is not "forced" into receive electrical idle state.</p> <p>1 For PCI Express, place the specified lane into electrical idle state. For Aurora, shut down RX side of SerDes for TX only lanes.</p>
8–10 -	This field is reserved. Reserved
11–15 REIDL_CTL	<p>REIDL_CTL[4:2]. Represents the threshold for the Loss of Signal detector within the receive amplifier</p> <p>000 Loss of signal detect function is disabled</p> <p>001 Default SGMII levels (Low=30mV, High=100mV)</p> <p>010 Intermediate level (Low=38mV, High=120mV)</p> <p>011 Intermediate level (Low=50mV, High=150mV)</p> <p>100 Default PEX levels (Low=65mV, High=175mV)</p> <p>101 Reserved</p> <p>110 Intermediate level (Low=88mV, High=225mV)</p> <p>111 Intermediate level (Low=100mV, High=250mV)</p> <p>Recommended setting per protocol:</p> <p>PCI Express: 100</p> <p>SGMII: 001</p> <p>SRIO: 000</p> <p>XAUI: 000</p> <p>Aurora: 000</p> <p>REIDL_CTL[1:0]. Used to set filter depths appropriate to protocol and bit rate.</p> <p>For SGMII:</p> <p>00 Exit from Idle ~88UI and Unexpected Idle Detect ~1us (Application Mode)</p> <p>01 Exit from Idle ~88UI and Unexpected Idle Detect ~10us</p> <p>10 Exit from Idle ~88UI</p> <p>11 Bypass</p> <p>Recommended setting per protocol: 00</p>
16–31 -	This field is reserved. Reserved

3.6.23 Bank 3 Transmit Equalization Control Register 0 - Lane x (Signals_B3TECRn0)

The transmit equalization control register 0 contains the functional control bits for the SerDes logic on lanes A to D.

Address: E_A000h base + 918h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved	TEQ_TYPE	Reserved	RATIO_PREQ				Reserved	RATIO_PST1Q							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				RATIO_PST2Q				Reserved	AMP_RED						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Signals_B3TECRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–3 TEQ_TYPE	Selects amount/type of transmit equalization Recommended settings per protocol converter: SGMII: 00 All other protocols: 01 00 No TX Equalization 01 2 Levels of TX Equalization (+1 postcursor) 10 3 Levels of TX Equalization (+1 precursor and +1 postcursor) 11 4 Levels of TX Equalization (+1 precursor and +2 postursors)
4–5 -	This field is reserved. Reserved
6–9 RATIO_PREQ	Ratio of full swing transition bit to precursor
10–11 -	This field is reserved. Reserved
12–15 RATIO_PST1Q	Ratio of full swing transition bit to postcursor for 2-tap equalization B3TECR(lane)0[12]: 0 Negative Sign 1 Positive Sign B3TECR(lane)0[13:15]: 000 No Equalization 001 1.09x

Table continues on the next page...

Signals_B3TECRn0 field descriptions (continued)

Field	Description
	010 1.20x 011 1.33x 100 1.50x 101 1.71x 110 2.00x 111 Reserved Recommended settings per protocol converter: 2.5 Gbps PCI Express: 1100 5.0 Gbps PCI Express: 1110 SRIO: 1011 SGMII: 1000 XAUI: 1100 Aurora: 1100
16–19 -	This field is reserved. Reserved
20–23 RATIO_PST2Q	Ratio of full swing transition bit to postcursor for 4-tap equalization
24–25 -	This field is reserved. Reserved
26–31 AMP_RED	Amount of amplitude reduction for all bits B3TECR(lane)0[26] = Reserved B3TECR(lane)0[27:31]: 00000 = Full Swing Vdfffpk- pk 01000 = 0. 75x Full Swing Vdfffpk- pk 01011 = 0. 68x Full Swing Vdfffpk- pk 10011 = 0. 50x Full Swing Vdfffpk- pk Recommend settings per protocol converter: PCI Express - 000000 SRIO: 000000 SGMII: 001011 XAUI: 000000 Aurora: 000000

3.6.24 Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRn0)

The transition tracking loop control register contains the functional control bits used for the Transition Tracking Loop (TTL) logic for lanes A to D.

Address: E_A000h base + 920h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved				FLT_SEL									Reserved		
W																
Reset	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	Reserv ed	PM_DIS							Reserved							Reserved
W																
Reset	0	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Signals_B3TTLCRn0 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 FLT_SEL	TTL channel filter bandwidth selection NOTE: The reset value for FLT_SEL depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 01_1011. For lanes configured for all other protocols (non-PCI Express), the reset value is either 00_0011 (P4080 Rev 3) or 01_1000 (P4080 Rev 2). FLT_SEL[2:3] - Selects the gain 'Kfr' in the CDR for lane 00 2 ⁻⁵ 01 2 ⁻⁶ 10, 11 Reserved Recommended setting per protocol:

Table continues on the next page...

Signals_B3TTLCRn0 field descriptions (continued)

Field	Description
	<p>PCI Express: 01 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 01 (Note that this setting differs from the reset value.) FLT_SEL[4:5] - Selects the gain 'Kph' in the CDR for lane 00 Reserved 01 2^-7 10 2^-8 11 Reserved Recommended setting per protocol: PCI Express: 10 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 00 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 10 (Note that this setting differs from the reset value.) FLT_SEL[6:7] - Select Markov filter lengths in Transition Tracking Loop for Lane. For B3TTLCR(lane)0[PM_DIS] = 0 00, 01, 10, 11 - TBD For B3TTLCR(lane)0[PM_DIS] = 1 00 +/-200 ppm 01 +/-400 ppm 10 +/-500 ppm 11 +/-750 ppm Recommended setting per protocol: PCI Express: 11 SGMII, SRIO, XAUI, Aurora - P4080 Rev 2: 11 (Note that this setting differs from the reset value.) SGMII, SRIO, XAUI, Aurora - P4080 Rev 3: 00 (Note that this setting differs from the reset value.)</p>
8–16 -	This field is reserved. Reserved
17 PM_DIS	<p>Disable slow-phase-modulation portion of transition tracking loop (TTL) NOTE: The reset value for PM_DIS depends on the protocol configuration for the associated lane. For lanes configured for PCI Express, the reset value is 0. For lanes configured for all other protocols (non-PCI Express), the reset value is either 0 (P4080 Rev 2) or 1 (P4080 Rev 3). Recommended setting per protocol" PCI Express: 0 SGMII, SRIO, XAUI, Aurora: 0 (Note that this setting differs from the reset value on P4080 Rev3.)</p>
18–30 -	This field is reserved. Reserved
31 -	<p>This field is reserved. Reserved Recommended settings per protocol: PCI Express: 0</p>

Table continues on the next page...

Signals_B3TTLCRn0 field descriptions (continued)

Field	Description
	SGMII, SRIO, XAUI, Aurora: 1 (Note that this setting differs from the reset value)

3.7 SerDes PLL reset and reconfiguration

To reconfigure a SerDes PLL, perform the following sequence:

1. Disable all lanes using the PLL to be reconfigured by setting SRDSxPLLnRSTCTL[SDRST]=1
2. Wait at least 240 ns
3. Disable the PLL by setting SRDSxPLLnRSTCTL[SDPD]=1 and SRDSxPLLnRSTCTL[PLLRST]=1
4. Wait at least 240 ns
5. Change the desired PLL settings
6. Reset the PLL by setting SRDSxPLLnRSTCTL[RSTREQ]=1
7. Wait a few cycles
8. Set SRDSxPLLnRSTCTL[SDPD]=0, SRDSxPLLnRSTCTL[PLLRST]=0, and SRDSxPLLnRSTCTL[SDRST]=0

3.8 Output Signal States During Reset

When a system reset is initiated ($\overline{\text{HRESET}}$ or $\overline{\text{PORRESET}}$ sampled asserted by the P4080), the P4080 aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See [Reset, Clocking, and Initialization](#) for a complete description of the reset functionality.

While the chip is in reset, it drives $\overline{\text{HRESET}}$ asserted and ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state.

Note that signals associated with all potential RCW source interfaces are enabled and active while the P4080 is in reset (that is, while $\overline{\text{HRESET}}$ is being driven asserted by the P4080, but after $\overline{\text{PORRESET}}$ is deasserted). This is necessary to allow the interfaces to be used for fetching configuration information from non-volatile memory devices. See [Reset Configuration Word Source](#) for more information.

Chapter 4

Reset, Clocking, and Initialization

4.1 Reset, clocking, and initialization overview

This content describes the reset, clocking, and initialization, including a definition of the reset configuration signals and the options they select. Additionally, the configuration, control, and status registers are described. Note that other chapters in this book may describe specific aspects of initialization for individual blocks.

The reset, clocking, and control signals provide many options for operation. Additionally, many modes are selected with reset configuration signals during a power on reset (assertion of PORESET_B) and by using the reset configuration word (RCW) functionality.

4.2 External Signal Descriptions

The table below summarizes the external signals described in this chapter. [Table 4-3](#) and [Table 4-4](#) have detailed signal descriptions, but this table contains references to additional sections that contain more information.

Table 4-1. Reset and Control Signals Summary

Signal	I/O	Description	References (Section/Page)
PORESET_B	I	Power on reset input.	System Control Signals
HRESET_B	I/O	Hard reset input. Functions as an output during initial steps in the power on reset sequence. See Power-On Reset Sequence for more information.	System Control Signals
RESET_REQ_B	O	Reset request output. An internal block requests that either HRESET_B or PORESET_B be asserted.	System Control Signals
CKSTP_OUT_B	O	Checkstop out.	System Control Signals

Table 4-2. Clock Signals Summary

Signal	I/O	Description	References
SYSCLK	I	Primary clock input.	External Clock Signals
RTC	I	Real time clock input.	External Clock Signals
SD_REF_CLK n _P/ SD_REF_CLK n _N	I	SerDes high-speed interface reference clock n .	External Clock Signals
CLK_OUT	O	Diagnostic clock output.	External Clock Signals

The following sections describe the reset and clock signals in detail.

4.2.1 System Control Signals

The table below describes some of the system control signals. [Power-On Reset Configuration](#) describes the signals that also function as reset configuration signals.

Table 4-3. System Control Signals: Detailed Signal Descriptions

Signal	I/O	Description	
PORESET_B	I	Power on reset. Causes the chip to abort all current internal and external transactions and set all registers to their default values. PORESET_B may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated-See Signals Introduction and Power-On Reset Configuration for more information on the interpretation of the other signals during reset.
		Timing	Assertion/Negation-The chip gives specific timing information for this signal and the reset configuration signals.
HRESET_B	I/O	Hard reset. Causes the chip to abort all current internal and external transactions and set all registers to their default values. HRESET_B may be asserted completely asynchronously with respect to all other signals. HRESET_B is driven as an output during the first part of the power on reset sequence, after which, it becomes an input, allowing external devices to stall/hold the reset sequence. See Hard Reset Sequence for more information.	
		State Meaning	Asserted/Negated-See Signals Introduction for more information on the interpretation of the other chip signals during reset.
		Timing	Assertion/Negation-The chip gives specific timing information for this signal.
RESET_REQ_B	O	Reset request. Indicates to the board (system in which the chip is embedded) that a condition requiring the assertion of HRESET_B or PORESET_B has been detected.	
		State Meaning	Asserted-An event has triggered a request for either a hard reset or a power on reset. See Reset request status register (DCFG_RSTQSR1) for more information. Negated-Indicates no reset request.
		Timing	Assertion/Negation-May occur any time. Once asserted, RESET_REQ_B does not negate until either HRESET_B or PORESET_B is asserted.

Table continues on the next page...

**Table 4-3. System Control Signals: Detailed Signal Descriptions
(continued)**

Signal	I/O	Description	
CKSTP_OUT_B	O	Checkstop out.	
		State Meaning	Asserted-Indicates that the chip is in a checkstop state. Negated-Indicates normal operation. After CKSTP_OUT_B has been asserted, it is negated after the next negation (low-to-high transition) of PORESET_B.
		Timing	Assertion-May occur at any time; may be asserted asynchronously to the input clocks. Negation-Must remain asserted until the chip has been reset with a PORESET_B.
ASLEEP	O	Power Management Signal. See RCPM Signal Descriptions .	
TMP_DETECT_B	I	Tamper Detect.	

4.2.2 External Clock Signals

The table below describes some of the external clock signals of the chip. Note that some clock signals are specific to modules within the chip, and although some of their functionality is described here, they are defined in detail in their respective chapters.

Table 4-4. Clock External Signals-Detailed Signal Descriptions

Signal	I/O	Description	
SYSCLK	I	System clock (SYSCLK). SYSCLK is the primary clock input to the chip.	
		Timing	Assertion/Negation-See the chip for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the cores. The RTC timing specifications are given in the chip . This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).	
		Timing	Assertion/Negation-See the chip for specific timing information for this signal.
SD_REF_CLKn_P, SD_REF_CLKn_N	I	SerDes high-speed interface differential reference clocks. These differential clock inputs are used to independently clock the banks/ports of high-speed differential signal lanes available on the chip. The SerDes reference clock timing specifications are given in the chip . See Reference Clocks for SerDes Protocols .	
		Timing	Assertion/Negation-See the chip for specific timing information for these signals.
CLK_OUT	O	Diagnostic clock output. This output may be configured to offer one of a variety of internal system clocks to external hardware for diagnostic or debug purposes. See CLK_OUT Configuration .	

4.3 Local Configuration Control Registers

4.3.1 Accessing Configuration, Control, and Status Registers

The memory-mapped configuration, control, and status registers (CCSRs) occupy a 16 MB region of memory. Their location is programmable using CCSR base address register high (CCSRBARTH) and CCSR base address register low (CCSRBTRL); together, these registers are referred to as CCSRBAR. The default base address for the CCSR is 0x0_FE00_0000. CCSRBARH and CCSRBARL are part of the local access block of CCSR memory, which begins at offset 0x0 from CCSRBAR. Because CCSRBAR is at offset 0x0 from the beginning of the local access registers, CCSRBAR always points to itself.

4.3.1.1 Updating CCSRBARs

Updating the CCSRBARs that relocate the entire 16 MB region of CCSR registers requires special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen.

To ensure this happens, use the following guidelines:

- CCSRBARH and CCSRBARL should be updated during initial configuration of the device when only one host or controller has access to the device.
 - If an external host on PCI Express or RapidIO is configuring the device, it should set CCSRBARH and CCSRBARL to the desired final values before the core is released to boot.
 - If the core is initializing the device, it should set CCSRBARH and CCSRBARL to the desired final values before enabling other I/O devices to access the device.

When updating CCSRBARH and CCSRBARL, use the following sequence:

1. Place a temporary LAW over the new planned CCSRBAR space (because the CCSR space has not been moved yet, the address of the LAW register is still relative to the old CCSR space). Program the LAWAR n [TRGT_ID] = 0x1E (a reserved setting) for this temporary LAW. When the LAW registers for this temporary LAW are read, it forces the initial write to the CCSR space to complete. Note that this LAW is only temporary.
2. The application that is performing the update must ensure that all other cores and applications are not accessing the configuration space during the procedure.
3. Read the current CCSRBARH and CCSRBARL using load word instructions.

4. Follow this with an **isync** instruction. This forces any outstanding accesses to configuration space to completion.
5. Write the new values for CCSRBARH and CCSRBARL to their old locations.

The CCSRBARH has a shadow register. When the CCSRBARH has a new value written it loads a CCSRBARH shadow register. When the CCSRBARL is written, the CCSRBARH shadow register contents along with the CCSRBARL value are loaded into the CCSRBARH and CCSRBARL registers, respectively.

6. Follow this with a **sync** instruction.
7. Write a 1 to the commit bit (C) of CCSRAR at the old location.
8. Follow this with a **sync** instruction.
9. The temporary LAW configured in Step 1 above should now be disabled and reused elsewhere because it is no longer needed.

4.3.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBARH, ALTCBARL, and ALTCAR registers. These are intended to be used with the pre-boot loader (PBL) to allow the PBL to access a 16 MB region of the memory map. By loading the proper PBL command in the serial ROM, the base address in ALTCBARH and ALTCBARL can be combined with the 24 bits of address offset supplied from the serial ROM to generate a 36-bit address that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the PBL has access to the entire memory map, one 16 MB block at a time.

NOTE

Updates to the alternate configuration access window must be performed with care. The ALTCBARL and ALTCBARH must be written before setting ALTCAR[EN] or undefined results will occur.

While enabled, the alternate configuration space always takes precedence over the LAWs. The CoreNet Coherency Fabric (CCF) automatically disables the alternate configuration window when the PBL has completed its pre-boot initialization (PBI) process. Windows (including LAWs and the boot space translation window) can be enabled by the PBL during PBI to establish an initial address map that the cores see during their boot process.

The alternative configuration mechanism cannot be used to access external devices on the peripheral interfaces (SRIO or PCI Express). ALTCAR does not support SRIO or PCIe target IDs.

4.3.3 Boot Space Translation

When each core comes out of reset, its MMU has one 4 KB page defined at `0x0_FFFF_Fnnn`. Each core begins execution with the instruction at effective address `0x0_FFFF_FFFC`. To get this instruction, the core's first instruction fetch is a burst read of boot code from effective address `0x0_FFFF_FFC0`. For systems in which the boot code resides at a different address, the chip provides boot space translation capability. Note that boot space translation affects transactions initiated by all cores in the same manner.

The pre-boot loader can enable boot space translation, or the boot space translation can be set up by an external host when the device is configured to be in boot holdoff mode. If translation is to be performed to a page outside the default boot window (8 MB at `0x0_FF80_0000` to `0x0_FFFF_FFFF`), the external host or pre-boot loader must then also set up the target ID defining the target destination.

When an address falls within the range for the boot window space, boot space translation is applied (if enabled) to translate the address per [Boot space translation register high \(LCC_BSTRH\)](#), [Boot space translation register low \(LCC_BSTRL\)](#), and [Boot space translation attribute register \(LCC_BSTAR\)](#). If LCC_BSTAR[EN] bit is set and the access falls within the boot translation window size, a translation of the incoming address occurs. The lower address bits that represent the byte offset within the programmed size remain unchanged. The LCC_BSTRH and LCC_BSTRL are combined to form a translation address to replace the remaining upper address bits of the incoming address. The boot window location starts at address `0x0_FF80_0000` and extends to `0x0_FFFF_FFFF`. The address range within the boot window that is translated by the boot space translation registers covers the region from 4 GB-BSTAR[SIZE] to 4 GB-1.

4.4 Local Configuration Control Memory Map

This section describes the CCSR_s that control access to the configuration space, the alternate configuration space, and boot space translation as described in the previous sections.

LCC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
0	Configuration, control, and status registers base address register high (LCC_CCSRBARH)	32	R/W	0000_0000h	4.4.1/211
4	Configuration, control, and status registers base address register low (LCC_CCSRBARL)	32	R/W	FE00_0000h	4.4.2/211
8	Configuration, control, and status registers attribute register (LCC_CCSRAR)	32	R/W	0000_0000h	4.4.3/212
10	Alternate configuration base address register high (LCC_ALTCBARH)	32	R/W	0000_0000h	4.4.4/212
14	Alternate configuration base address register low (LCC_ALTCBARL)	32	R/W	0000_0000h	4.4.5/213
18	Alternate configuration attribute register (LCC_ALTCAR)	32	R/W	0000_0000h	4.4.6/213
20	Boot space translation register high (LCC_BSTRH)	32	R/W	0000_0000h	4.4.7/214
24	Boot space translation register low (LCC_BSTRL)	32	R/W	0000_0000h	4.4.8/214
28	Boot space translation attribute register (LCC_BSTAR)	32	R/W	01F0_000Bh	4.4.9/214

4.4.1 Configuration, control, and status registers base address register high (LCC_CCSRBARH)

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

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

LCC_CCSRBARH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 BASE_ADDR_HIGH	Identifies the upper portion (the 4 most significant bits) of the CCSR window base address.

4.4.2 Configuration, control, and status registers base address register low (LCC_CCSRBARL)

Address: 0h base + 4h offset = 4h

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

LCC_CCSRBARL field descriptions

Field	Description
0–31 BASE_ADDR_LOW	Identifies the lower portion of the CCSR window base address. Because the CCSR base address must be aligned on a 16 MB boundary, bits 8–31 are ignored by hardware and always return 0x00_0000.

4.4.3 Configuration, control, and status registers attribute register (LCC_CCSRAR)

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

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

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

LCC_CCSRAR field descriptions

Field	Description
0 C	Commit. Commits the CCSR base address configuration. Writing a 1 to this bit causes the CCSR base address programmed in CCSRBARH and CCSRBARL to take effect. When read, this bit always returns zero.
1–31 -	This field is reserved. Reserved

4.4.4 Alternate configuration base address register high (LCC_ALTCBARTH)

Address: 0h base + 10h offset = 10h

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

LCC_ALTCBARTH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 BASE_ADDR_HIGH	Identifies the upper portion (the 4 most significant bits) of the alternate configuration window base address.

4.4.5 Alternate configuration base address register low (LCC_ALTCBRL)

Address: 0h base + 14h offset = 14h

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

Reset 0

LCC_ALTCBRL field descriptions

Field	Description																													
0–31 BASE_ADDR_LOW	Identifies the lower portion of the alternate configuration window base address. Because the base address must be aligned on a 16 MB boundary, bits 8–31 are ignored by hardware and always return 0x00_0000.																													

4.4.6 Alternate configuration attribute register (LCC_ALTCAR)

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

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

LCC_ALTCAR field descriptions

Field	Description														
0 EN	Enable for alternate configuration window. The alternate configuration window has a fixed size of 16 MB. 0 Alternate configuration window is disabled. 1 Alternate configuration window is enabled.														
1–3 -	Write reserved, read = 0														
4–11 TRGT_ID	Identifies the device ID to target when a transaction hits in the 16 MB address range defined by the alternate configuration window. See Global Source and Target IDs for encodings. NOTE: SRIO and PEX targets are not supported.														
12–31 -	Write reserved, read = 0														

4.4.7 Boot space translation register high (LCC_BSTRH)

Address: 0h base + 20h offset = 20h

LCC BSTRH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 <u>TRANS_ADDR_HIGH</u>	Identifies the upper portion (the 4 most significant bits) of the boot space translation address.

4.4.8 Boot space translation register low (LCC_BSTRL)

Address: 0h base + 24h offset = 24h

LCC_BSTRL field descriptions

Field	Description
0-31 TRANS_ADDR_LOW	<p>Identifies the lower portion of the boot space translation address.</p> <p>NOTE: The least significant bits of the address representing offsets within the boot space (as defined by BSTAR[SIZE]) are treated as zero by the hardware. Reading them returns zeros.</p>

4.4.9 Boot space translation attribute register (LCC_BSTAR)

Address: 0h base + 28h offset = 28h

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

LCC_BSTAR field descriptions

Field	Description
0 EN	Enable for boot space translation. 0 Boot space translation is disabled. 1 Boot space translation is enabled.
1–3 -	Write reserved, read = 0
4–11 TRGT_ID	Identifies the destination for accesses covered by boot space translation. See Global Source and Target IDs for encodings.
12–25 -	Write reserved, read = 0
26–31 SIZE	Boot space translation window size. The window size is $2^{(\text{SIZE}+1)}$ bytes. Minimum window size is 4 KB; maximum window size is 8 MB. Example settings: 000000-001010 Reserved 001011 4 KB 001100 8 KB 001101 16 KB 010101 4 MB 010110 8 MB 010111-111111 Reserved

4.5 Clocking Memory Map/Register Definition

The following table summarizes the memory mapped registers which are used to configure clocking features.

Clocking memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_1000	Core n clock control/status register (Clocking_CLKC0CSR)	32	R/W	0000_0000h	4.5.1/216
E_1020	Core n clock control/status register (Clocking_CLKC1CSR)	32	R/W	0000_0000h	4.5.1/216
E_1040	Core n clock control/status register (Clocking_CLKC2CSR)	32	R/W	0000_0000h	4.5.1/216
E_1060	Core n clock control/status register (Clocking_CLKC3CSR)	32	R/W	0000_0000h	4.5.1/216
E_1080	Core n clock control/status register (Clocking_CLKC4CSR)	32	R/W	0000_0000h	4.5.1/216
E_10A0	Core n clock control/status register (Clocking_CLKC5CSR)	32	R/W	0000_0000h	4.5.1/216
E_10C0	Core n clock control/status register (Clocking_CLKC6CSR)	32	R/W	0000_0000h	4.5.1/216
E_10E0	Core n clock control/status register (Clocking_CLKC7CSR)	32	R/W	0000_0000h	4.5.1/216
E_1800	PLL cluster n general status register (Clocking_PLL1CGSR)	32	R/W	0000_0000h	4.5.2/218
E_1820	PLL cluster n general status register (Clocking_PLL2CGSR)	32	R/W	0000_0000h	4.5.2/218

Table continues on the next page...

Clocking memory map (continued)

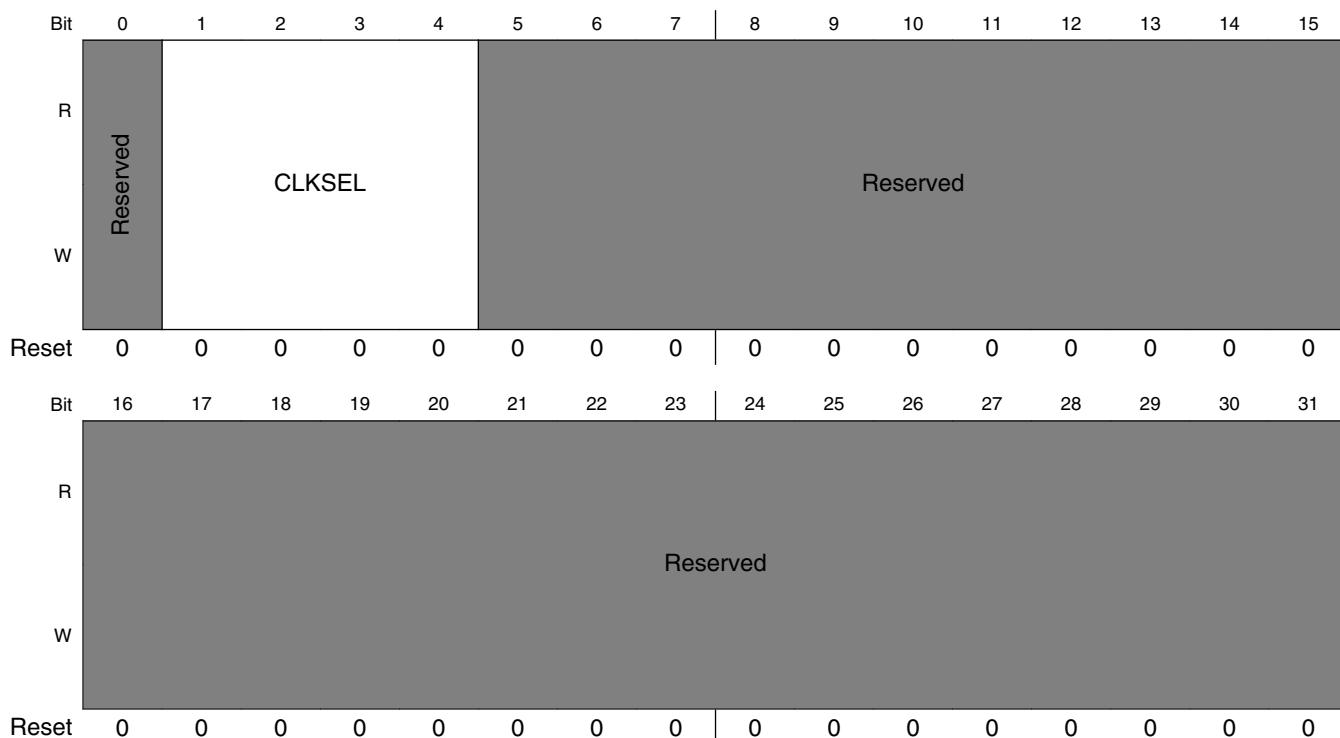
Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
E_1840	PLL cluster n general status register (Clocking_PLL3CGSR)	32	R/W	0000_0000h	4.5.2/218
E_1860	PLL cluster n general status register (Clocking_PLL4CGSR)	32	R/W	0000_0000h	4.5.2/218
E_1A00	Platform clock domain control/status register. (Clocking_CLKPCSR)	32	R/W	0000_0000h	4.5.3/219
E_1C00	Platform PLL general status register. (Clocking_PLLPGSR)	32	R/W	0000_0000h	4.5.4/221
E_1C20	DDR PLL general status register (Clocking_PLLDGSR)	32	R/W	0000_0000h	4.5.5/223

4.5.1 Core n clock control/status register (Clocking_CLKCnCSR)

The CLKCnCSR registers control the clock frequency selection for each core. CLKC0CSR corresponds to core 0, and CLKC7CSR corresponds to core 7.

The CLKCnCSR registers' fields are described below. Note that cores 0-3 and cores 4-7 have a different set of options for their clock sources.

Address: E_1000h base + 0h offset + (32d × i), where i=0d to 7d



Clocking_CLKCnCSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–4 CLKSEL	Clock Select. Selects the clock source for this clock domain. Cores 0-3 0000 Cluster PLL1 output 0001 Cluster PLL1 divide-by-2 0010 Reserved 0011 Reserved 0100 Cluster PLL2 output 0101 Cluster PLL2 divide-by-2 0110 Reserved 0111 Reserved 1000 Cluster PLL3 output 1001-1111 Reserved Cores 4-7 0000 Cluster PLL1 output 0001-0111 Reserved 1000 Cluster PLL3 output 1001 Cluster PLL3 divide-by-2 1010 Reserved 1011 Reserved 1100 Cluster PLL4 output 1101 Cluster PLL4 divide-by-2 1110 Reserved 1111 Reserved
5–31 -	This field is reserved. Reserved

4.5.2 PLL cluster n general status register (Clocking_PLLnCGSR)

The PLLCnGSR registers provide information regarding the cluster PLL configuration.

Address: E_1000h base + 800h offset + (32d × i), where i=0d to 3d

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

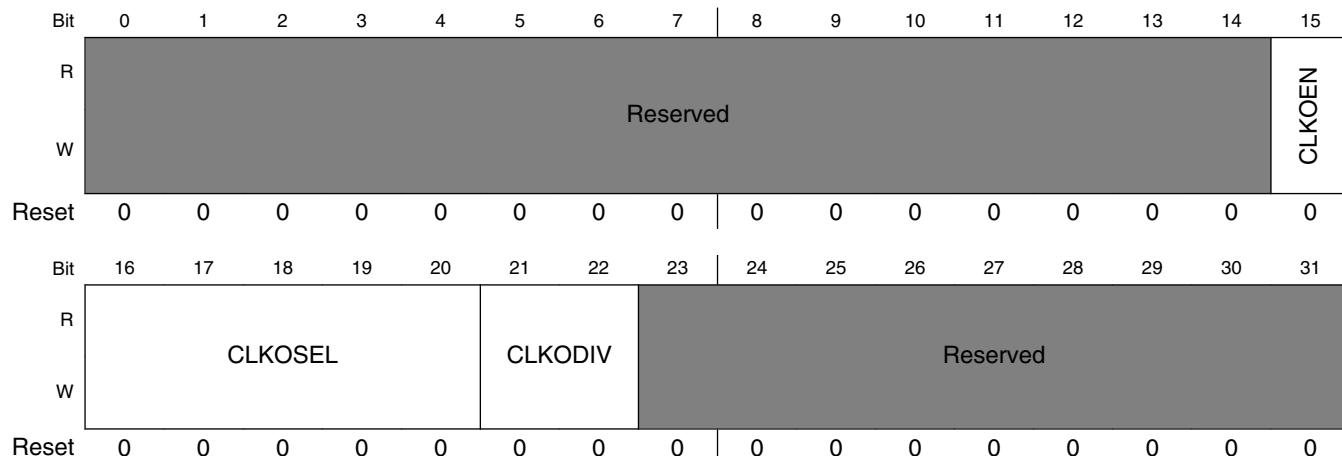
Clocking_PLLnCGSR field descriptions

Field	Description
0 KILL	Writing a 1 to this bit disables the PLL. If PLLnGSR[CFG] indicates 0b00000, the PLL is bypassed and this bit (KILL) is set. 0 PLL is active. 1 PLL is disabled.
1–24 -	This field is reserved. Reserved
25–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. If CFG = 0b00000, the PLL is bypassed and KILL is set.
31 -	This field is reserved. Reserved

4.5.3 Platform clock domain control/status register. (Clocking_CLKPCSR)

The CLKPCSR register selects which signal to observe on the CLK_OUT1 pad.

Address: E_1000h base + A00h offset = E_1A00h



Clocking_CLKPCSR field descriptions

Field	Description
0–14 -	This field is reserved. Reserved
15 CLKOEN	CLK_OUT pad enable 0 Release CLK_OUT pad to high impedance 1 Enable CLK_OUT pad

Table continues on the next page...

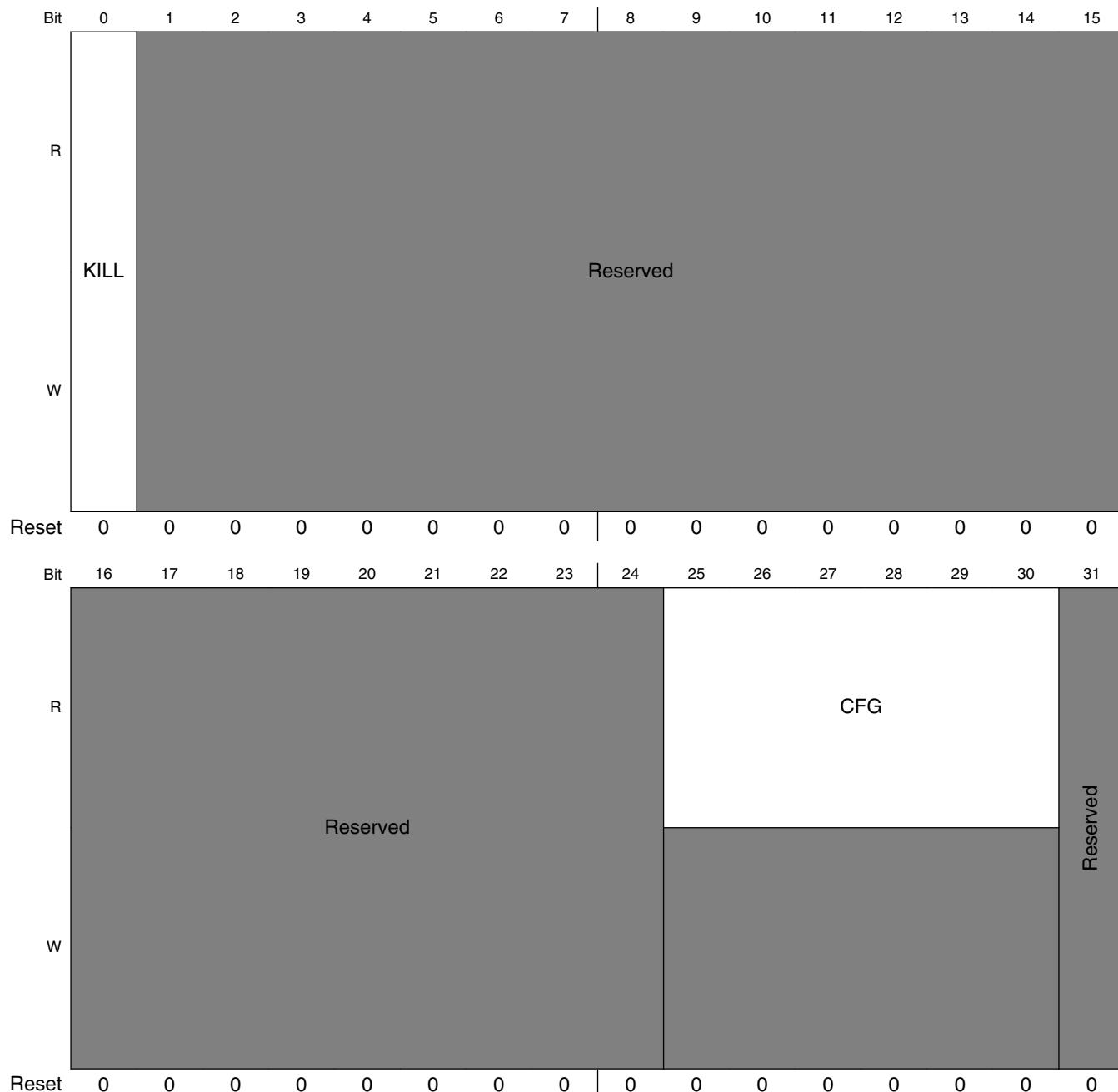
Clocking_CLKPCSR field descriptions (continued)

Field	Description
16–20 CLKOSEL	Selects core related clock signal for observation on CLK_OUT pad 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved 00100 Reserved 00101 Reserved 00110 Reserved 00111 Reserved 01000 Reserved 01001 Reserved 01010 Reserved 01011 Reserved 01100 Reserved 01101 Reserved 01110 Reserved 01111 Reserved 10000 Reserved 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Reserved 11001 Platform Clock /8 11010 Platform Clock /4 11011 Platform Clock /2 11100 Reserved 11101 Reserved 11110 SYSCLK (demonstrates Platform PLL lock) 11111 SYSCLK (echoes SYSCLK pin)
21–22 CLKODIV	Selects division setting for clock-out mux output to reduce the frequency of the signal to a more observable/measurable range. 00 no division (divide-by-1) 01 divide-by-2 10 Reserved 11 Reserved
23–31 -	This field is reserved. Reserved

4.5.4 Platform PLL general status register. (Clocking_PLLPGSR)

The PLLPGSR register provides information regarding the platform PLL configuration.

Address: E_1000h base + C00h offset = E_1C00h



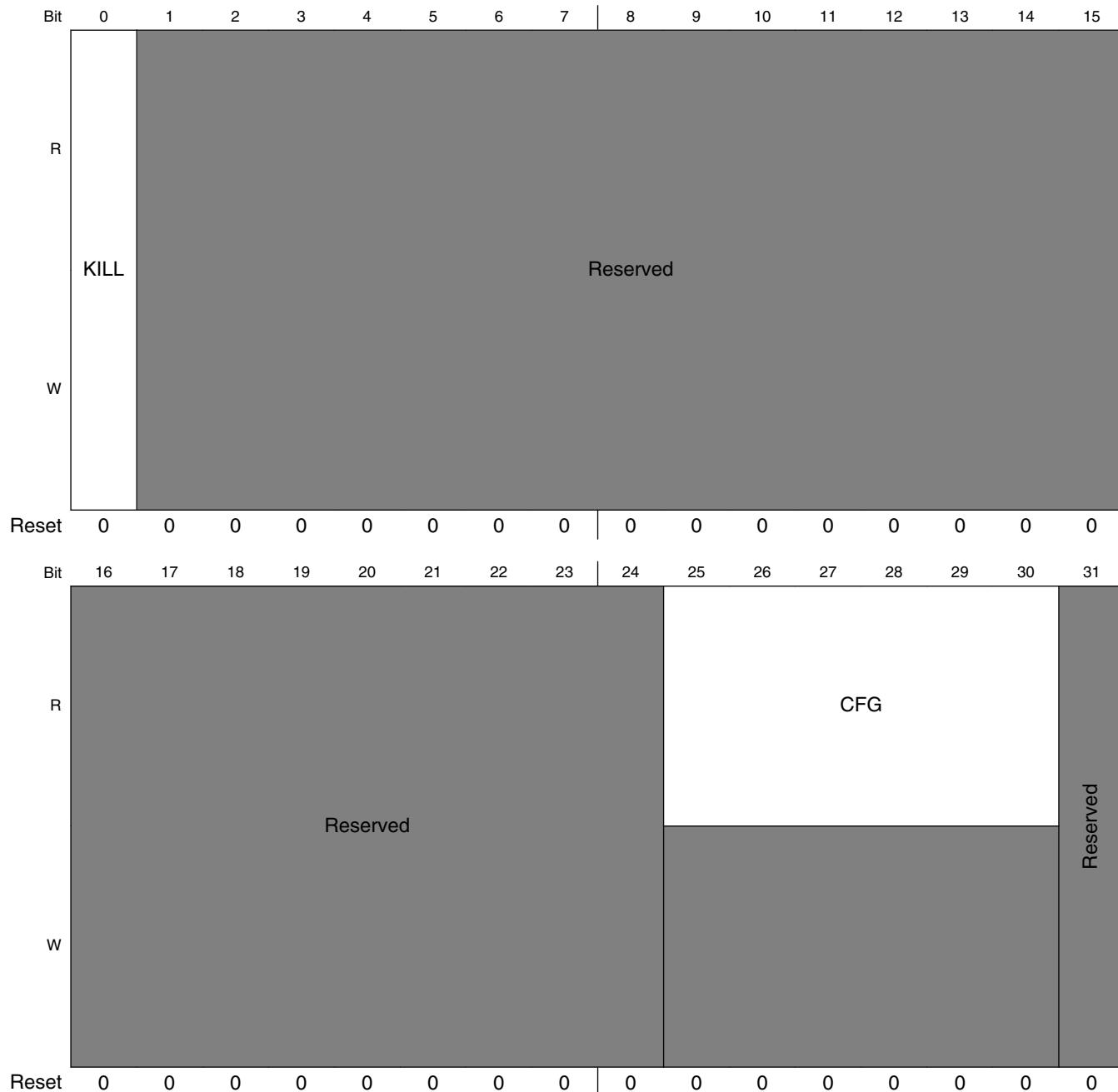
Clocking_PLLPGSR field descriptions

Field	Description
0 KILL	Writing a 1 to this bit disables the PLL. If PLLnGSR[CFG] indicates 0b00000, the PLL is bypassed and this bit (KILL) is set. 0 PLL is active. 1 PLL is disabled.
1–24 -	This field is reserved. Reserved
25–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. If CFG = 0b00000, the PLL is bypassed and KILL is set.
31 -	This field is reserved. Reserved

4.5.5 DDR PLL general status register (Clocking_PLLDGSR)

The PLLDGSR register provides information regarding the DDR PLL configuration.

Address: E_1000h base + C20h offset = E_1C20h



Clocking_PLLDGSR field descriptions

Field	Description
0 KILL	Writing a 1 to this bit disables the PLL. If PLLnGSR[CFG] indicates 0b00000, the PLL is bypassed and this bit (KILL) is set. 0 PLL is active. 1 PLL is disabled.
1–24 -	This field is reserved. Reserved
25–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. If CFG = 0b00000, the PLL is bypassed and KILL is set.
31 -	This field is reserved. Reserved

4.6 Functional Description

This section describes the various ways to reset the chip, the POR sequence, the hard reset sequence, the POR configurations, the reset configuration word (RCW), and the clocking on the device.

4.6.1 Power-On Reset Sequence

Assertion of the external PORESET_B signal initiates the Power-On Reset flow. See the chip for more information regarding power sequencing and PORESET_B input requirements.

After the negation of PORESET_B, the reset control logic begins cycling the device through its full reset and RCW configuration process. The chip asserts HRESET_B throughout the power-on reset process, including RCW configuration. Reset and RCW configuration time varies depending on the configuration source and SYSCLK frequency. Initially, the RCW source POR configuration inputs are sampled to determine the configuration source. Next, the device begins loading the RCW data. The system PLL begins to lock according to the clock ratio/mode values communicated in the RCW data. Once the PLL locks and the RCW data is loaded, HRESET_B is released by the device and the clocking unit begins distributing PLL outputs throughout the device. Pre-boot initialization is then optionally performed, and the cores are permitted to boot.

The detailed POR sequence for the device is as follows:

1. The external system logic asserts PORESET_B and power is applied to comply with the chip's .

2. PORESET_B asserted causes all registers to be initialized to their default states and most I/O drivers to be released to high impedance (some clock, clock enables, and system control signals are active).

NOTE

The common on-chip processor (COP) requires the ability to independently assert PORESET_B and TRST_B to fully control the processor. If a JTAG/COP port is used, follow the JTAG/COP interface connection recommendations given in the chip's . If the JTAG interface and COP header are not being used, Freescale recommends that TRST_B be tied to PORESET_B so that TRST_B is asserted when PORESET_B is asserted, ensuring that the JTAG scan chain is initialized during the power-on reset flow. See the JTAG configuration signals section in the chip's for more information.

3. The system applies a toggling SYSCLK signal and stable POR configuration inputs. At this point, SYSCLK is propagated throughout the device; the platform PLL is running in bypass mode.
4. The device begins driving HRESET_B asserted after sampling the assertion of PORESET_B.
5. External system logic negates PORESET_B after its required hold time and after POR configuration inputs have been valid for their required setup times.
6. The device samples the RCW source POR configuration inputs (cfg_rcw_src[0:n]) on deassertion of PORESET_B to determine the RCW source. Note that the POR configuration inputs are sampled only on a PORESET_B.
7. The device initiates and completes reset of the rest of the platform logic. Note that this platform reset step is the point where the device hard reset process (HRESET_B) begins if an external device asserts HRESET_B (assuming the device is not already sequencing through the power-on reset process).
8. Some of the I/O drivers are enabled; specifically, those pins associated with any interface potentially usable as the source of RCW data. All of the DDR I/Os become enabled at this point (though MCKE, MCK, MODT are enabled from the beginning). The ASLEEP signal is also enabled at this point.
9. If the eLBC's FCM/NAND Flash interface is configured as the RCW source, the reset block instructs the eLBC to load a boot block from Flash into the internal buffer RAM of the eLBC. Once complete, the reset block proceeds to instruct the Pre-Boot Loader to begin reading in RCW data. Note that if the eLBC FCM reports an ECC error, the device reset sequence is halted indefinitely, waiting for another PORESET_B or hard reset.

10. The pre-boot loader (PBL) starts loading the RCW data from the interface specified by cfg_rcw_src[0:n] configuration inputs and stores that 64 bytes of data to the RCWSR registers within the device configuration block. Loading time varies and depends on the source of the RCW. Note that if a hard-coded RCW option is used, this PBL RCW loading process is effectively bypassed and the device is automatically configured according to the specific RCW field encodings pre-assigned for the given hard-coded RCW option (see [Hard Coded RCW Options](#) for more information).
11. The PLLs begin to lock.
12. The sequence then waits for the PBL RCW process to be completed (loading of all 512 bits). If the PBL reports an error during its process of loading the RCW data, the device reset sequence is halted indefinitely, waiting for another PORESET_B or hard reset.
13. The platform clock tree is then switched over and is driven by the output of the platform PLL.
14. The device stops driving HRESET_B at this point. All other I/O drivers are enabled at this point.
15. If the eLBC's FCM/NAND Flash interface is:
 - configured as the pre-boot initialization source
 - the eLBC's FCM/NAND Flash interface was NOT previously used as the RCW source,then the reset block informs the eLBC to load a boot block from Flash into the internal buffer RAM of the eLBC. Once complete, the eLBC signals back to the reset block, and the reset block can proceed. Note that if the eLBC FCM reports an ECC error, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B.
16. The PBL performs Pre-Boot Initialization, if enabled by RCW, by reading data from either the eSDHC, SPI, I²C1, or eLBC interface and writing to CCSR space or local memory space (SRAM, DDR). If the PBL reports an error during its pre-boot initialization process, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B. Note that non-CCSR-mapped memory space of a module on OCN must not be the target of any pre-boot initialization.
17. Any external device optionally driving HRESET_B negates it if not done earlier. If other external devices do not release HRESET_B, the device reset sequence stalls at this point.
18. System ready state. The peripheral interfaces (serial RapidIO and PCI Express) are released to accept external requests, and the boot vector fetches by the cores are allowed to proceed unless processor booting is further held off by the boot release register (BRR) in the device configuration module. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state of the system.

After reaching this system ready state, the ASLEEP signal transitions to the asserted state when the device enters sleep mode.

NOTE

After completing reset, software should check the SRDSBnRSTCTL[RST_DONE] field to make sure that each active SerDes PLL on the device has locked. Transactions or packet data cannot be transferred through the targeted lane(s) of the SerDes interface if the PLL associated with the lane(s) does not lock properly. Note that a SerDes PLL will not lock if the corresponding reference clock is not provided.

Figure below shows a timing diagram of the POR sequence.

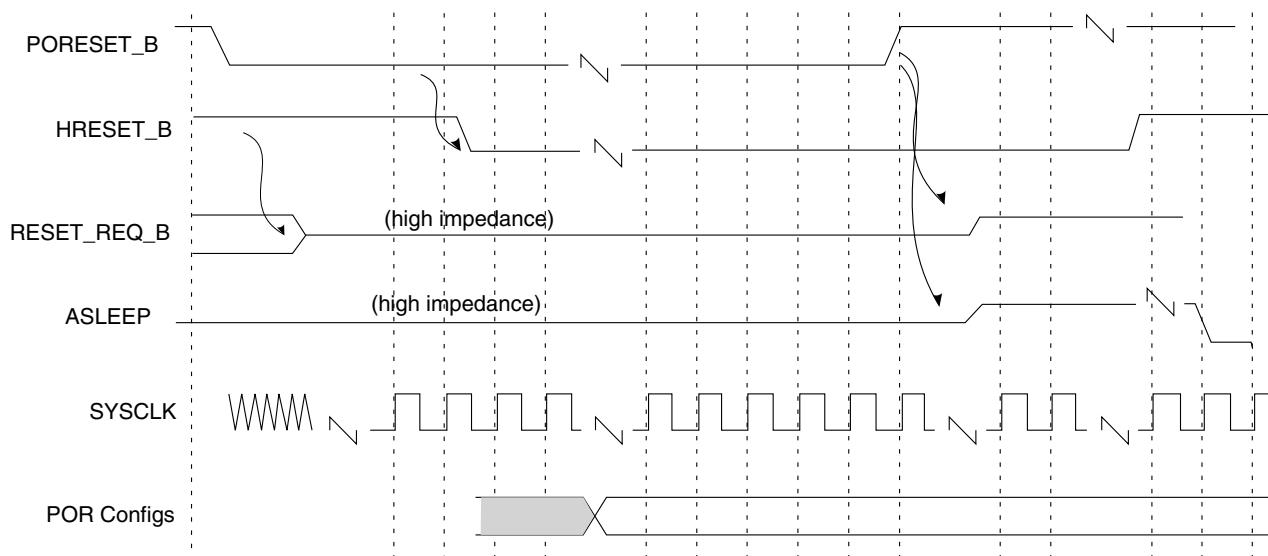


Figure 4-27. Power-On Reset Sequence

4.6.2 Hard Reset Sequence

The hard reset sequence is initiated by the external system asserting HRESET_B. Upon sampling the HRESET_B asserted, the device then begins driving HRESET itself throughout the hard reset state.

Reset and RCW configuration time varies subject to the configuration source and SYSCLK frequency. The reset configuration input signals are not sampled by a hard reset (only a power-on reset), so the device immediately begins loading RCW data and configures the device as explained in [Reset Configuration Word \(RCW\)](#). After the configuration sequence completes, the device releases the HRESET signal and exits the

hard reset state. After negation is detected, a 16-cycle period is taken before testing for the presence of an external reset. The hard reset sequence begins with reset of the device at step 7 in [Power-On Reset Sequence](#).

4.6.3 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during power on reset. The values of all these signals are sampled into registers when PORESET_B is deasserted. These inputs are to be pulled high or low by external resistors. During PORESET_B, all other signal drivers connected to these signals must be in the high-impedance state.

All POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board. See the chip for proper pull-down resistor values for POR configuration signals, when necessary.

This section describes the functions and modes configured by the POR configuration signals.

NOTE

In the following tables, the binary value 0 represents a signal pulled down to GND and a value of 1 represents a signal pulled up to that signal's corresponding V_{DD} voltage level, regardless of the sense of the functional signal name.

4.6.3.1 Reset Configuration Word Source

The reset configuration word (RCW) source inputs (values 0_0000 through 0_1111), shown in [Table 4-31](#), select the source for the RCW data. The hard-coded configuration options shown in [Table 4-31](#) (values 1_0000 through 1_1011) select specific preconfigured options for the device without requiring an external RCW source. See [Hard Coded RCW Options](#), for more information. The encoded values latched on these signals during POR are accessible in PORSR1[RCW_SRC], as described in [POR status register 1 \(DCFG_PORSR1\)](#).

Table 4-31. RCW Source Location

Functional Signals	Reset Configuration Name	Value (Binary)	RCW Source
LGPL0/LFCLE, LGPL1/LFALE, LGPL2/LOE/LFRE, LGPL3/LFWP, LGPL5 Default (1_1111)	cfg_rcw_src[0:4]	0_0000	I ² C1 normal addressing (supports ROMs up to 256 bytes)
		0_0001	I ² C1 extended addressing
		0_0010	Reserved
		0_0011	Reserved
		0_0100	SPI 16-bit addressing
		0_0101	SPI 24-bit addressing
		0_0110	eSDHC
		0_0111	Reserved
		0_1000	eLBC FCM (NAND flash, 8-bit small page)
		0_1001	eLBC FCM (NAND flash, 8-bit large page)
		0_1010	Reserved
		0_1011	Reserved
		0_1100	eLBC GPCM (NOR flash, 8-bit)
		0_1101	eLBC GPCM (NOR flash, 16-bit)
		0_1110	Reserved
		0_1111	Reserved
		1_0000 - 1_1011	Hard-coded RCW options (See Hard Coded RCW Options , for more information.)
		1_1100 - 1_1111	Reserved

4.6.3.2 General-Purpose Input

The general-purpose inputs, shown below, are available for application-specific use. The encoded values latched on these signals during POR are accessible in GPPORCR1[POR_CFG_VEC], as described in [General-purpose POR configuration register \(DCFG_GPPORCR1\)](#).

Table 4-32. General-Purpose Input

Functional Signals	Reset Configuration Name	Value (Binary)	General Purpose Input
LAD[0:15] Default (1111_1111_1111_1111)	cfg_gpininput[0:15]	all	Application-defined

4.6.3.3 eLBC FCM ECC Control

The eLBC FCM ECC control input, shown below, controls the ECC functionality of the NAND flash machine (FCM) within the eLBC. The encoded value latched on this signal during POR is accessible in [POR status register 1 \(DCFG_PORSR1\)](#).

Table 4-33. eLBC FCM ECC Control

Functional Signals	Reset Configuration Name	Value (Binary)	FCM ECC Behavior
Default (1)	cfg_elbc_ecc	0	NAND Flash FCM ECC disabled
		1	NAND Flash FCM ECC enabled

4.6.3.4 DRAM type select

The DRAM type select input, described in this table, specifies the DDR technology and, thus, voltage (GV_{DD}) to be used with the DDR memory controllers. The encoded value latched on this signal during POR is accessible in [PORSR2\[DRAM_TYPE\]](#), as described in [PORSR2](#).

Table 4-34. DRAM type select

Functional signals	Reset configuration name	Value (binary)	DRAM type
LA[24] Default (1)	cfg_dram_type	0	DDR3 technology (1.5 V)
		1	DDR2 technology (1.8 V)

NOTE

The DDR DRAM I/O supply voltage (GV_{DD}) must match the pad power supply for the SerDes transceivers (XV_{DD}).

4.6.4 Reset Configuration Word (RCW)

The chip uses an external memory interface to import a subset of the reset configuration information from a memory device during reset.

Such information is called reset configuration word (RCW) data.

The pre-boot loader (PBL) loads RCW data from a non-volatile memory device interface, as specified by the RCW source configuration inputs (cfg_rcw_src[0:8]-see [Reset Configuration Word Source](#) for more information). See [Pre-Boot Loader](#) for details on the operation of the PBL. Note that this approach does not completely remove the

necessity for at least a few power-on reset (POR) configuration signals. As noted, POR config signals are used to control RCW source information in addition to other low-level system configuration.

The logic involved is clocked directly from SYSCLK since RCW importing takes place before on-chip PLLs are configured.

The RCW is 512 bits long in order to contain all necessary configuration information for the chip. RCW data is read from external memory and written to the RCW status registers (see [Reset control word status registers 1-16 \(DCFG_RCWSR \$n\$ \)](#)) contained in the Device Configuration and Pin Control module , after which the device is configured as specified in the RCW. [Required Format of Data Structure Consumed by PBL](#) provides details of the data structure that is required to reside in non-volatile memory.

4.6.4.1 RCW Field Definitions

This table describes the function of the individual bits of the 512-bit (64-byte) RCW data structure.

NOTE

Unless noted otherwise, any bit ranges in [Table 4-35](#) listed as reserved must be populated with 0.

Table 4-35. RCW Field Descriptions

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
PLL CONFIGURATION (BITS 0-127)			
0-1	SYS_PLL_CFG	System PLL Configuration	Options: 00 Platform freq/system PLL targeting 667 MHz and above operation 01 Platform freq/system PLL targeting 533-666 MHz operation 10 Reserved 11 Reserved
2-6	SYS_PLL_RAT	System PLL multiplier/ratio	This field selects the platform clock:SYSCLK ratio. Options: 0_0000 Reserved 0_0100 Reserved 0_0101 5:1 0_0110 6:1 0_0111 7:1 0_1000 8:1

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			<p>0_1001 9:1 0_1010 10:1 0_1011 11:1 0_1100 12:1</p> <p>NOTE: All ratios may not be supported due to frequency restrictions. See the chip hardware specifications for the supported frequencies.</p>
7	Reserved		
8-9	MEM_PLL_CFG	Memory Controller Complex PLL Configuration (applies to both DDR controllers).	<p>This field configures the Memory Complex PLLs for the frequency of the reference clock applied. Note that in the case of synchronous mode operation, the reference clock to the memory complex PLLs is the platform clock (that is, the output of the system PLL). In the case of asynchronous mode operation, the reference clock is SYSCLK. See the chip hardware specifications for the definition of the boundary or cutoff point between the lower and higher reference clock frequency for each specific ratio/multiplier selected.</p> <p>Options:</p> <ul style="list-style-type: none"> 00 Lower frequency reference clock 01 Higher frequency reference clock 10 Reserved 11 Reserved
10-14	MEM_PLL_RAT	Memory Controller Complex PLL multiplier/ratio (applies to both DDR controllers). Note that certain ratios are only valid in either synchronous mode or asynchronous mode. The DDR_SYNC RCW field is used to enable synchronous mode operation for both DDR Complexes.	<p>RCW[MEM_PLL_CFG] must be configured to use the appropriate reference clock frequency option (higher or lower) for the cutoff frequency of the selected ratio. See the chip hardware specifications for the specific cutoff reference clock frequency for each ratio. Any reference clock frequency applied at or below the stated cutoff point requires configuring MEM_PLL_CFG for a lower frequency reference clock. Any reference clock frequency applied above the stated cutoff point requires configuring MEM_PLL_CFG for a higher frequency reference clock. See the MEM_PLL_CFG field definition for more details.</p> <p>NOTE: The memory controller complex frequency (which equals the DDR data rate) must not be configured to run faster than 2x the platform clock rate which is determined by SYS_PLL_RAT.</p> <p>Options:</p> <ul style="list-style-type: none"> 0_0000 Reserved 0_0001 1:1 (sync mode only)

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			0_0010 Reserved 0_0011 Reserved 0_0100 Reserved 0_0101 5:1 (async mode only) 0_0110 6:1 (async mode only) 0_0111 Reserved 0_1000 8:1 (async mode only) 0_1001 9:1 (async mode only) 0_1010 10:1 (async mode only) 0_1011 Reserved 0_1100 12:1 (async mode only) 0_1101 13:1 (async mode only) 0_1110 Reserved 0_1111 Reserved 1_0000 16:1 (async mode only) 1_0001 Reserved 1_0010 18:1 (async mode only) 1_0011 Reserved 1_0100 20:1 (async mode only) 1_0101 Reserved 1_0110 Reserved 1_0111 Reserved 1_1000 24:1 (async mode only)
15-63	Reserved		
64-65	CC1_PLL_CFG	Core cluster PLL 1 configuration.	Options: 00 Core cluster PLL 1 output frequency targeting 1 GHz and above 01 Core cluster PLL 1 output frequency targeting below 1 GHz 10 Reserved 11 Reserved
66-70	CC1_PLL_RAT	Core cluster PLL 1 multiplier/ratio.	Note that if CC1 PLL is ever used by any core 4-7, then CC1's maximum allowed frequency is 80% of the maximum rated frequency of the core at nominal voltage. Options: 0_1000 8:1 Async 0_1001 9:1 Async

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			0_1010 10:1 Async 0_1011 11:1 Async 0_1100 12:1 Async 0_1110 14:1 Async 0_1111 15:1 Async 1_0000 16:1 Async All other encodings are reserved
71	Reserved		
72-73	CC2_PLL_CFG	Core cluster PLL 2 configuration.	Same options/notes as CC1_PLL_CFG.
74-78	CC2_PLL_RAT	Core cluster PLL 2 multiplier/ratio.	Same options/notes as CC1_PLL_RAT.
79	Reserved		
80-81	CC3_PLL_CFG	Core cluster PLL 3 configuration.	Same options/notes as CC1_PLL_CFG. NOTE: P4040-based systems should set this to the correct value depending on the frequency of SYSCLK if CC3 is not used.
82-86	CC3_PLL_RAT	Core cluster PLL 3 multiplier/ratio.	Same options/notes as CC1_PLL_RAT. Note that if CC3 PLL is ever used by any core 0-3, then CC3's maximum allowed frequency is 80% of the maximum rated frequency of the core at nominal voltage. NOTE: P4040-based systems should set this to 0_1000 if CC3 is not used.
87	Reserved		
88-89	CC4_PLL_CFG	Core cluster PLL 4 configuration.	Same options/notes as CC1_PLL_CFG. NOTE: P4040-based systems should set this to 00.
90-94	CC4_PLL_RAT	Core cluster PLL 4 multiplier/ratio.	Same options/notes as CC1_PLL_RAT NOTE: P4040-based systems should set this to 0_0000.
95	Reserved		
96-99	C0_PLL_SEL	Core complex 0 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 0001 CC1 PLL /2 0010 Reserved 0100 CC2 PLL /1 0101 CC2 PLL /2 0110 Reserved 1000 CC3 PLL /1

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
100-103	C1_PLL_SEL	Core complex 1 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 0001 CC1 PLL /2 0010 Reserved 0100 CC2 PLL /1 0101 CC2 PLL /2 0110 Reserved 1000 CC3 PLL /1
104-107	C2_PLL_SEL	Core complex 2 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 0001 CC1 PLL /2 0010 Reserved 0011 Reserved 0100 CC2 PLL /1 0101 CC2 PLL /2 0110 Reserved 1000 CC3 PLL /1
108-111	C3_PLL_SEL	Core complex 3 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 0001 CC1 PLL /2 0010 Reserved 0100 CC2 PLL /1 0101 CC2 PLL /2 0110 Reserved 1000 CC3 PLL /1
112-115	C4_PLL_SEL	Core complex 4 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 1000 CC3 PLL /1 1001 CC3 PLL /2 1010 Reserved 1100 CC4 PLL /1 1101 CC4 PLL /2 1110 Reserved NOTE: P4040-based systems should set this to 1101

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
116-119	C5_PLL_SEL	Core complex 5 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 1000 CC3 PLL /1 1001 CC3 PLL /2 1010 Reserved 1100 CC4 PLL /1 1101 CC4 PLL /2 1110 Reserved NOTE: P4040-based systems should set this to 1101
120-123	C6_PLL_SEL	Core complex 6 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 1000 CC3 PLL /1 1001 CC3 PLL /2 1010 Reserved 1100 CC4 PLL /1 1101 CC4 PLL /2 1110 Reserved NOTE: P4040-based systems should set this to 1101
124-127	C7_PLL_SEL	Core complex 7 PLL select. Selects one of three available core complex PLLs and one of the two available VCO dividers of that PLL for the given core.	Options: 0000 CC1 PLL /1 1000 CC3 PLL /1 1001 CC3 PLL /2 1010 Reserved 1100 CC4 PLL /1 1101 CC4 PLL /2 1110 Reserved NOTE: P4040-based systems should set this to 1101
SerDes PLL and Protocol Configuration (Bits 128-183)			
128-133	SRDS_PRTCL	SerDes protocol select	See SerDes Lane Assignments and Multiplexing , for a complete list of the options and the definitions of this encoded field. See Reference Clocks for SerDes Protocols , for a list of the legal combinations of reference clocks and SRDS_DIV_Bn and SRDS_RATIO_Bn encodings for each given SRDS_PRTCL encoding.
134-135	Reserved		
136-138	SRDS_RATIO_B1	SerDes PLL ratio-bank 1	Options:

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			000 10:1 001 20:1 010 25:1 011 40:1 100 50:1
139-143	SRDS_DIV_B1	SerDes PLL dividers-bank 1	1 bit for each of the five pairs of lanes in bank 1. The first bit corresponds to lanes A/B, while the last bit corresponds to lanes I/J. Options for each bit: 0 Divide by 1 off of bank 1 PLL 1 Divide by 2 off of bank 1 PLL NOTE: When SerDes bank 1 is configured for a 5-Gbps operation (by SRDS_RATIO_B1 and the reference clock), lanes configured as SGMII operate at 1.25 Gbps regardless of the corresponding SRDS_DIV_B1 settings.
144-146	SRDS_RATIO_B2	SerDes PLL ratio-bank 2	Options: 000 10:1 001 20:1 010 25:1 011 40:1 100 50:1
147	SRDS_DIV_B2	SerDes PLL divider-bank 2	1 bit controls all four lanes (A-D) in bank 2. Options: 0 Divide by 1 off of bank 2 PLL 1 Divide by 2 off of bank 2 PLL
148-150	SRDS_RATIO_B3	SerDes PLL ratio-bank 3	Options: 000 10:1 001 20:1 010 25:1
151	SRDS_DIV_B3	SerDes PLL divider-bank 3	1 bit controls all four lanes (A-D) in bank 3. Options: 0 Divide by 1 off of bank 3 PLL 1 Divide by 2 off of bank 3 PLL
152-161	SRDS_LPD_B1	SerDes lane power down-bank 1.	1 bit per lane for the ten lanes in bank 1. Option for each bit: 0 Lane not powered down 1 Lane powered down
162-165	SRDS_LPD_B2	SerDes lane power down-bank 2.	1 bit per lane for the four lanes in bank 2.

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			Option for each bit: 0 Lane not powered down 1 Lane powered down
166-169	Reserved		
170-173	SRDS_LPD_B3	SerDes lane power down-bank 3.	1 bit per lane for the four lanes in bank 3. Option for each bit: 0 Lane not powered down 1 Lane powered down
174-177	Reserved		
178	SRDS_EN	SerDes enable.	Options: 0 SerDes disabled 1 SerDes enabled
179-183	Reserved		
Miscellaneous PLL-Related Configuration (Bits 184-191)			
184	DDR_SYNC	DDR synchronous mode.	Options: 0 Both memory controller complexes run in asynchronous mode. In this mode, the reference clock to the memory controller complex PLL is SYSCLK. 1 Both memory controller complexes run in synchronous mode. In this mode, the reference clock to the memory controller complex PLL is the platform clock. NOTE: Synchronous mode (DDR_SYNC=1) configures the memory controller complexes to run synchronously with the platform clock.
185-191	Reserved		
Boot Configuration (Bits 192-223)			
192-195	PBI_SRC	Pre-boot initialization source. The pre-boot loader fetches addr/data pairs from the selected interface for the purpose of pre-boot initialization of CCSR and/or local memory space	The following restrictions apply: <ul style="list-style-type: none">RCW and pre-boot initialization data must be loaded from the same non-volatile memory deviceAt most, only one given eLBC option can be used for RCW, PBI_SRC, and BOOT_LOC. This does not mean that BOOT_LOC cannot be different from RCW and PBI_SRC (provided restriction 2 is observed). For example, the device can be configured to load RCW and pre-boot initialization data from NAND Flash (eLBC FCM) and boot from DDR. However, the device cannot load RCW and pre-boot initialization data from NAND Flash (eLBC FCM) and boot from NOR Flash (eLBC

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			<p>GPCM). Also, the hard-coded RCW source options are not considered their own memory interface for this purpose.</p> <p>Options:</p> <ul style="list-style-type: none"> 0000 I²C1 normal addressing (up to 256 byte ROMs) 0001 I²C1 extended addressing 0100 SPI 16-bit addressing 0101 SPI 24-bit addressing 0110 SD/MMC 1000 eLBC FCM 8-bit small page NAND Flash 1001 eLBC FCM 8-bit large page NAND Flash 1100 eLBC GPCM 8-bit 1101 eLBC GPCM 16-bit 1111 disabled
196-200	BOOT_LOC	Boot location	<p>Note the restrictions in PBI_SRC above. Specifically, restriction 2 that at most, only one given eLBC option can be used/configured as the source of RCW, PBI_SRC, and BOOT_LOC.</p> <p>Options:</p> <ul style="list-style-type: none"> 0_0000 PCIe1 (see note below) 0_0001 PCIe2 (see note below) 0_0010 PCIe3 (see note below) 0_1000 sRIO1 0_1001 sRIO2 1_0000 Memory complex 1 1_0001 Memory complex 2 1_0100 Interleaved memory complexes 1_1000 eLBC FCM 8-bit small page NAND Flash 1_1001 eLBC FCM 8-bit large page NAND Flash 1_1100 eLBC GPCM 8-bit 1_1101 eLBC GPCM 16-bit <p>NOTE: Special consideration must be given in the case where the RCW specifies booting from one of the PCI Express interfaces and the corresponding PCI Express controller is configured for RC mode. The first instruction fetch is a memory read from a downstream PCI Express endpoint device where the boot code resides. In this case there are two caveats:</p> <ul style="list-style-type: none"> • The SoC must initialize at least one LAW and one of the outbound ATMU windows of the

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			<p>target PCI Express controller using the pre-boot initialization (PBI) function. Note that this will require additional nonvolatile memory for the PBI source.</p> <ul style="list-style-type: none"> All downstream PCI Express devices in the path from the RC to the EP that contains the boot code must be powered up ahead of the root complex and configured (either by themselves or by some external means). This is because the SoC cannot issue configuration transactions to the PCI Express downstream devices prior to booting.
201	BOOT_HO	Boot holdoff mode	<p>Options:</p> <p>0 All cores except core 0 in holdoff</p> <p>1 All cores in holdoff</p>
202	SB_EN	Secure boot enable	<p>Note that secure boot is enabled if either this RCW bit is set or the Intent to secure fuse value is set. See Secure Boot and Trust Architecture for more information.</p> <p>Options:</p> <p>0 Secure boot is not enabled</p> <p>1 Secure boot is enabled-boot from internal boot ROM</p>
203-223	Reserved		
Clocking Configuration (Bits 224-255)			
224	PME_CLK_SEL	Pattern match engine clock select.	<p>Allows for PME frequency to be maximized</p> <p>Options:</p> <p>0 Platform clock /2</p> <p>1 Core cluster PLL 3 /2 (PME runs asynchronous with respect to platform)</p>
225	FM1_CLK_SEL	Frame manager 1 clock select.	<p>Options:</p> <p>0 Platform clock /2</p> <p>1 Core cluster PLL 3 /2 (FM1 runs asynchronous with respect to platform)</p>
226	FM2_CLK_SEL	Frame manager 2 clock select.	<p>Options:</p> <p>0 Platform clock /2</p> <p>1 Core cluster PLL 3 /2 (FM2 runs asynchronous with respect to platform)</p>
227-229	Reserved		
230-231	DRAM_LAT	DDR latency. This applies to both DDR Controllers.	<p>Options:</p> <p>00 6-6-6 or 7-7-7 DRAMs</p> <p>01 8-8-8, 9-9-9, 10-10-10, or 11-11-11 DRAMs</p> <p>10 Reserved</p>

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			11 5-5-5 DRAMs NOTE: If DDR latency is not known at the time the RCW is loaded, it is acceptable to conservatively configure this field as 01. This field is used for optimizing the DDR interface. No functional issues result if this field does not match the latency of the actual DRAM's used.
232	DDR_RATE	DDR data rate.	See the chip hardware specifications for settings.
233	MCK_TO_PLAT_RAT	MCK to platform (freq) ratio. This field properly configures the DDR controllers to line up with the ratio of the MCK (DDR bus clock) frequency to the platform frequency, as determined by the settings of the SYS_PLL_RAT and MEM_PLL_RAT RCW fields.	Note that the user should round up to the nearest ratio. Options: 0 Ratio of 1 1 Ratio of 1/2
234	DDR_RSV0	Reserved	See the chip hardware specifications for settings.
235-255	Reserved		
Memory and High-Speed I/O Configuration (Bits 256-287)			
256-259	Reserved		
260-262	RIO_DEVICE_ID	RapidIO device ID	-
263	RIO_SYS_SIZE	RapidIO system size	Options: 0 Small system size (256 devices) 1 Large system size (65536 devices)
264-266	HOST_AGT_B1	Host/agent bank 1. Configures Host (Root Complex) or Agent (Endpoint) mode for selected sRIO/PCIe interfaces on SerDes Bank 1.	Options: 000 All host mode 001 All agent mode 010 First two agent mode 011 First and third agent mode 100 First and fourth agent mode 101 Second and third agent mode 110 Second and fourth agent mode 111 Third and fourth agent mode For these options, "first" refers to the PCIe or sRIO controller assigned to the lanes closest to lane A, as shown in the SerDes Lane Multiplexing/Configuration table in SerDes Lane Assignments and Multiplexing ; for the given SerDes Bank 1 configuration; "second" refers to the next PCIe or sRIO controller assigned to the lanes to the right of the "first," and so on. NOTE: Any dual sRIO configurations must have both interfaces defined as host or both

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			interfaces defined as agent. Results are undefined if an illegal host/agent configuration is used.
267	HOST_AGT_B2	Host/agent bank 2. Configures Root Complex/Endpoint mode for selected PCIe interface on SerDes Bank 2.	Options: 0 Root Complex (RC) 1 Endpoint (EP)
268-287	Reserved	General Purpose Information (Bits 288-319)	
288-319	GP_INFO	General purpose information. This field has no affect on functional logic; it may be used by software.	
320-351	Reserved	Pin Multiplexing Configuration (Bits 352-415)	
Group A Pin Configuration (Bits 352-383)			
352-353	1588	These bits determine whether GPIO or 1588 functionality is selected for the given pin.	Options for each bit: 0 1588 functionality-TSEC_1588_ALARM_OUT2, TSEC_1588_PULSE_OUT2 1 GPIO functionality-GPIO[30:31]
354-359	Reserved		
360-361	EC1	This configures the functionality assigned to the EC1 parallel mode pins.	Options: 00 Frame Manager 1 dTSEC1 RGMII 01 Reserved 10 USB 1 Host 11 No parallel mode Ethernet, no USB NOTE: If SGMII mode is to be used (with the appropriate SerDes lane powered on in SRDS_LPDn) for FM1-dTSEC1, then RGMII mode must not be enabled in EC1. However, if RGMII is to be used by FM1-dTSEC1 and EC1 is set accordingly, then SGMII mode must not be enabled for FM1-dTSEC1 (and the corresponding SerDes lane must be powered down in SRDS_LPD_Bn if SGMII is selected for that bank n in the SRDS_PRTCL setting).
362	Reserved		
363-365	EC2	This configures the functionality assigned to the EC2 parallel mode pins.	Options: 000 Frame Manager 2 dTSEC1 RGMII. 001 Reserved 010 Frame Manager 1 dTSEC2 RGMII 011 Reserved 100 USB 2 Host/Device 101-110 Reserved

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			111 No parallel mode Ethernet, no USB NOTE: If SGMII mode is to be used (with the appropriate SerDes lane powered on in SRDS_LPdn) for FM2-dTSEC1 or FM1-dTSEC2, then RGMII mode for that dTSEC must not be enabled in EC2. However, if RGMII is to be used by FM2-dTSEC1 or FM1-dTSEC2 and EC2 is set accordingly, then SGMII mode must not be enabled for FM2-dTSEC1 or FM1-dTSEC2 (and the corresponding SerDes lane must be powered down in SRDS_LPD_Bn if SGMII is selected for that bank n in the SRDS_PRTCL setting).
366-368	UART	Configures functionality of the UART pins: UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, UART2_SOUT, UART2_SIN, UART2_RTS_B, UART2_CTS_B	Options: 000 GPIO[8], GPIO[10], GPIO[12], GPIO[14], GPIO[9], GPIO[11], GPIO[13], GPIO[15] 001-010 Reserved 011 UART1_SOUT, UART1_SIN, GPIO[12], GPIO[14], GPIO[9], GPIO[11], GPIO[13], GPIO[15] 100 UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, GPIO[9], GPIO[11], GPIO[13], GPIO[15] 101 UART1_SOUT, UART1_SIN, GPIO[12], GPIO[14], UART2_SOUT, UART2_SIN, GPIO[13], GPIO[15] 110 UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, UART2_SOUT, UART2_SIN, UART2_RTS_B, UART2_CTS_B 111 UART1_SOUT, UART1_SIN, UART3_SOUT, UART3_SIN, UART2_SOUT, UART2_SIN, UART4_SOUT, UART4_SIN
369-370	I2C3	Configures functionality of the I2C3 pins.	Options: 00 IIC3_SCL, IIC3_SDA 01 Reserved 10 GPIO[16:17] 11 SDHC_CD_B, SDHC_WP
371	I2C4	Configures functionality of the I2C4 pins.	Options: 0 IIC4_SCL, IIC4_SDA 1 EVT[5:6]_B Note: EVT[5:6]_B functionality, if selected by this field, defaults to input functionality until reconfigured by software to output functionality.
372	Reserved		
373-381	IRQ	Configures functionality of IRQ[3:11] pins—the corresponding GPIOs for these pins are GPIO[21:29].	Options for each bit: 0 IRQ

Table continues on the next page...

Table 4-35. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			1 GPIO NOTE: When GPIO functionality is selected, the corresponding MPIC external interrupt vector priority registers (MPIC_EIVPRn) must have their polarity set to active high.
382	IRQ_OUT	Configures functionality of the IRQ_OUT_B pin.	Options: 0 IRQ_OUT_B 1 EVT[9]_B Note: EVT[9]_B functionality, if selected by this field, defaults to input functionality until software reconfigures to output functionality.
383	SPI	Configures functionality of the SPI_CS[0:3] pins.	Options: 0 SPI_CS[0:3]_B 1 SDHC_DAT[4:7] for 8-bit MMC card support
Group B Pin Configuration (Bits 384-415)			
384	DMA1	Configures functionality of the DMA1 pins.	Options: 0 DMA1_DREQ0_B, DMA1_DACK0_B, DMA1_DDONE0_B 1 GPIO[18:19], -(Reserved)
385	Reserved		
386-387	DMA2	Configures functionality of the DMA2 pins.	Options: 00 DMA2_DREQ0_B, DMA2_DACK0_B, DMA2_DDONE0_B 01 ALT_MDVAL, ALT_MSRCID[0:1] 10 GPIO[20], EVT[7]_B, EVT[8]_B Note: EVT[7:8]_B functionality, if selected by this field, defaults to input functionality until reconfigured by software to output functionality.
388-511	Reserved		

4.6.4.1.1 Hard Coded RCW Options

The device can be initialized with a hard-coded RCW option, selected by the cfg_rcw_src[0:4] configuration inputs. When the RCW source is set to a hard-coded RCW option, the device is automatically configured according to the specific RCW field encodings pre-assigned for the given option.

All hard-coded RCW options allow the reset sequence to bypass using the PBL to load in RCW data from a non-volatile memory device. One envisioned use of the hard-coded RCW options is to facilitate booting with enough basic functionality to allow reprogramming of a corrupted (or perhaps uninitialized) non-volatile memory device that originally contained RCW information.

Table 4-36 describes hard-coded RCW configuration options. All configuration options listed also define the following characteristics:

- Interfaces or subsets of signals that offer multiplexed functionality are not driven (that is, high-impedance). Unused SerDes lanes are powered down. Exceptions are:
 - USB2, RGMII, and UART signals as noted in [Table 4-36](#)
 - IRQ[0:2]
 - GPIO[0:7]
- The DDR PLL ratio is set to 10:1 (asynchronous mode).
- The I²C, SPI (with 4 CSs), eSDHC (4-bit SD/MMC), eLBC (all 8 CS), and IEEE 1588 (except TSEC_1588_ALARM_OUT2 and TSEC_1588_PULSE_OUT2) interfaces are enabled. Once the cores boot from eLBC, they can communicate with the non-volatile memory devices that may reside on I²C1, SPI, or eSDHC interfaces.

Table 4-36. RCW Hard-Coded Configuration Options

cfg_rcw_src[0:4] Encoding	RCW Hard-Coded Configuration Options
1_0000	Hard-coded RCW: Large page NAND flash as boot location; USB2 and dual 4-pin UARTs enabled; platform ratio of 8:1; core PLL ratio of 14:1
1_0001	Hard-coded RCW: Large page NAND flash as boot location; RGMII FM1-dTSEC1 and dual 4-pin UARTs enabled; platform ratio of 8:1; core PLL ratio of 14:1
1_0010	Hard-coded RCW: 16-bit NOR flash as boot location; USB2 and dual 4-pin UARTs enabled; platform ratio of 8:1; core PLL ratio of 14:1
1_0011	Hard-coded RCW: 16-bit NOR flash as boot location; RGMII FM1-dTSEC1 and dual 4-pin UARTs enabled; Platform ratio of 8:1; Core PLL ratio of 14:1
1_0100	Hard-coded RCW: Three (x2, x2, x4) PCI Express at 2.5 G, 100-MHz ref clk, all agent mode; all cores in boot hold-off; dual 4-pin UARTs enabled; Platform ratio of 8:1; Core PLL ratio of 14:1
1_0101	Hard-coded RCW: Two (x4, x4) serial RapidIO at 2.5 G, 100-MHz ref clk, all agent mode; all cores in boot hold-off; dual 4-pin UARTs enabled; Platform ratio of 8:1; Core PLL ratio of 14:1
1_0110	Hard-coded RCW: Large page NAND flash as boot location; USB2 and dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_0111	Hard-coded RCW: Large page NAND flash as boot location; RGMII FM1-dTSEC1 and dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_1000	Hard-coded RCW: 16-bit NOR flash as boot location; USB2 and dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_1001	Hard-coded RCW: 16-bit NOR flash as boot location; RGMII FM1-dTSEC1 and dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_1010	Hard-coded RCW: Three (x2, x2, x4) PCI Express at 2.5 G, 100-MHz ref clk, all agent mode; all cores in boot hold-off; dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_1011	Hard-coded RCW: Two (x4, x4) serial RapidIO at 2.5G, 100-MHz ref clk, all agent mode; all cores in boot hold-off; dual 4-pin UARTs enabled; platform ratio of 6:1; core PLL ratio of 10:1
1_1100-1_1111	Reserved

4.6.5 Clocking

The following sections describe the clocking within the chip.

4.6.5.1 IP Logic Clock Distribution and Configuration

The chip takes primary clocking input from the external SYSCLK signal. As shown in [Figure 4-28](#), the SYSCLK input (frequency) is multiplied using multiple phase locked loops (PLL) to create a variety of frequencies which can then be passed to a variety of internal logic, including cores and peripheral IP modules.

The DDR PLL is used to provide clocking to the DDR memory controller complexes. The DDR PLL may use the platform clock (or a synchronous fraction thereof) as an input reference, allowing both DDR interfaces to be synchronous with the platform. Alternately, the DDR PLL may use the SYSCLK input clock as a reference to create a unique DDR memory controller complex clock. In this case, the DDR complex operates asynchronously with respect to the platform clock.

Note that many of the IP modules contain logic allowing software to further modify their external interface clocks within the IP module. See the applicable IP module chapter for details.

[Figure 4-28](#) describes internal logic clock distribution along with the means to configure the various ratios and clock sources. (See [Table 4-38](#) for correlation between clocking configuration and status registers that may be read by software to assess how clocking has been configured.) Note that [Figure 4-28](#) is not intended to reflect external interface clocking. Although sometimes dependent on or derived from logic clocking, a full description of external interface clocking is described within the applicable IP module chapter of this reference manual. Each of the three SerDes bank external interfaces are clocked by dedicated SerDes reference clock inputs. (See [Reference Clocks for SerDes Protocols](#) for details regarding valid combinations of external reference clocks and RCW configurations.)

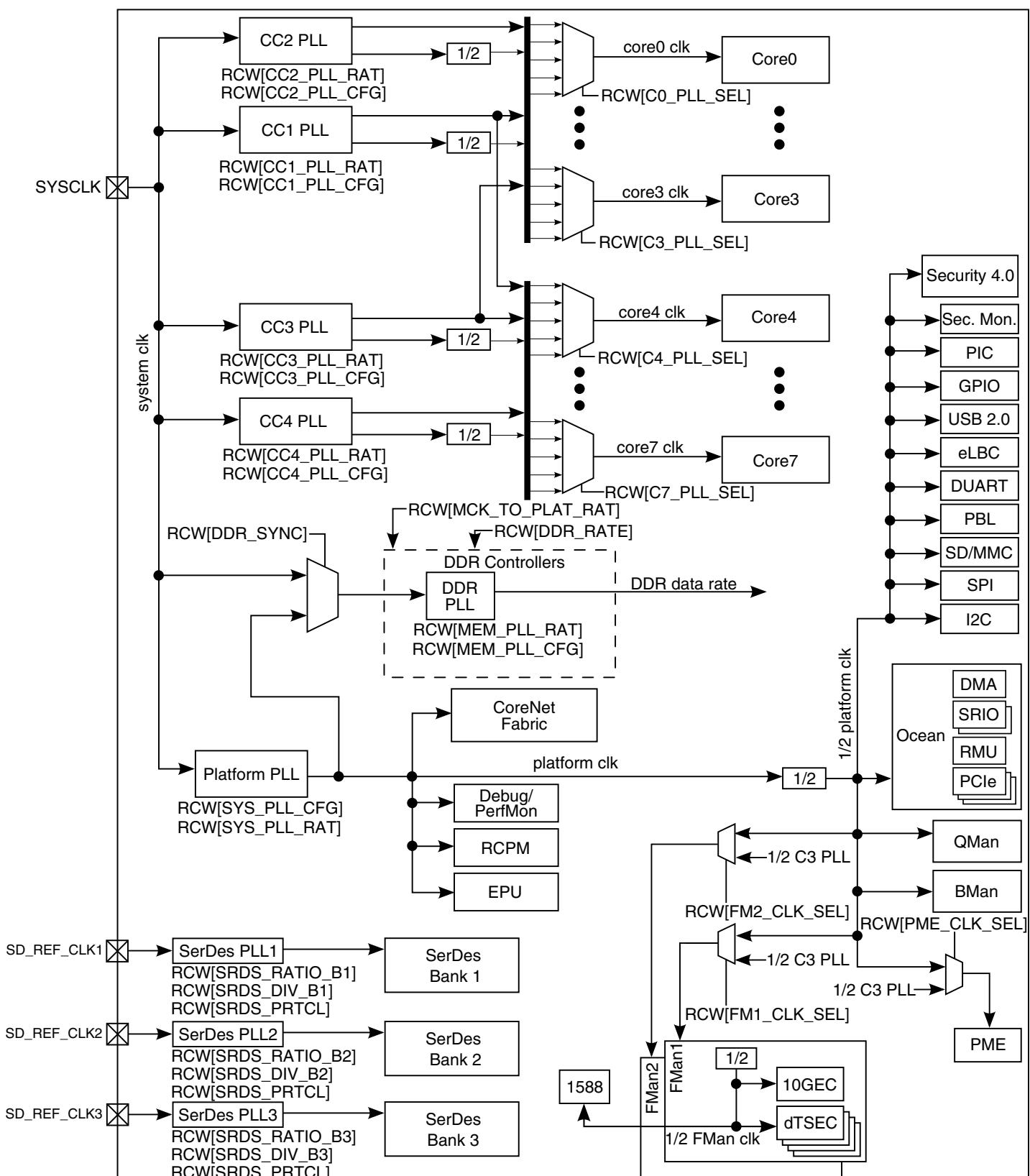


Figure 4-28. P4080 Logic Clock Subsystem Block Diagram

This table summarizes the available core clocking options subject to clock source selection and PLL configurations depicted in [Figure 4-28](#).

Table 4-37. Core Clocking Options

Core	Allowed Clock Sources (PLLs and Dividers)							
	CC1 and CC2 PLLs				CC3 and CC4 PLLs			
	CC1	CC1/2	CC2	CC2/2	CC3	CC3/2	CC4	CC4/2
0	Y	Y	Y	Y	Y ¹	-	-	-
1	Y	Y	Y	Y	Y ¹	-	-	-
2	Y	Y	Y	Y	Y ¹	-	-	-
3	Y	Y	Y	Y	Y ¹	-	-	-
4	Y ¹	-	-	-	Y	Y	Y	Y
5	Y ¹	-	-	-	Y	Y	Y	Y
6	Y ¹	-	-	-	Y	Y	Y	Y
7	Y ¹	-	-	-	Y	Y	Y	Y

- See the chip hardware specifications for limitations related to using these PLL/Core combinations.

This table describes the register fields which may be read by software to assess how various clocking options have been configured via RCW.

Table 4-38. Clocking Configuration Status

RCW Bit(s)	Clock Control Input (RCW Field)	CCSR Field
0-1	SYS_PLL_CFG	RCWSRn[SYS_PLL_CFG]
2-6	SYS_PLL_RAT	PLLPGSR[CFG]
8-9	MEM_PLL_CFG	RCWSRn[MEM_PLL_CFG]
10-14	MEM_PLL_RAT	PLLDGSR[CFG]
64-65	CC1_PLL_CFG	RCWSRn[CC1_PLL_CFG]
66-70	CC1_PLL_RAT	PLLC1GSR[CFG]
72-73	CC2_PLL_CFG	RCWSRn[CC2_PLL_CFG]
74-78	CC2_PLL_RAT	PLLC2GSR[CFG]
80-81	CC3_PLL_CFG	RCWSRn[CC3_PLL_CFG]
82-86	CC3_PLL_RAT	PLLC3GSR[CFG]
88-89	CC4_PLL_CFG	RCWSRn[CC4_PLL_CFG]
90-94	CC4_PLL_RAT	PLLC4GSR[CFG]
96-99	C0_PLL_SEL	CLKC0CSR[CLKSEL]
100-103	C1_PLL_SEL	CLKC1CSR[CLKSEL]
104-107	C2_PLL_SEL	CLKC2CSR[CLKSEL]
108-111	C3_PLL_SEL	CLKC3CSR[CLKSEL]
112-115	C4_PLL_SEL	CLKC4CSR[CLKSEL]
116-119	C5_PLL_SEL	CLKC5CSR[CLKSEL]
120-123	C6_PLL_SEL	CLKC6CSR[CLKSEL]

Table continues on the next page...

**Table 4-38. Clocking Configuration Status
(continued)**

RCW Bit(s)	Clock Control Input (RCW Field)	CCSR Field
124-127	C7_PLL_SEL	CLKC7CSR[CLKSEL]
136-138	SRDS_RATIO_B1	RCWSRn[SRDS_RATIO_B1]
139-143	SRDS_DIV_B1	RCWSRn[SRDS_DIV_B1]
144-146	SRDS_RATIO_B2	RCWSRn[SRDS_RATIO_B2]
147	SRDS_DIV_B2	RCWSRn[SRDS_DIV_B2]
148-150	SRDS_RATIO_B3	RCWSRn[SRDS_RATIO_B3]
151	SRDS_DIV_B3	RCWSRn[SRDS_DIV_B3]
184	DDR_SYNC	RCWSRn[DDR_SYNC]
224	PME_CLK_SEL	RCWSRn[PME_CLK_SEL]
225	FM1_CLK_SEL	RCWSRn[FM1_CLK_SEL]
226	FM2_CLK_SEL	RCWSRn[FM2_CLK_SEL]
232	DDR_RATE	-
233	MCK_TO_PLAT_RAT	-

4.6.5.2 CLK_OUT Configuration

The CLK_OUT signal can be configured to offer external hardware one of a variety of internal clock signals for debug or diagnostic purposes. (See [Platform clock domain control/status register. \(Clocking_CLKPCSR\)](#) for details.)

4.6.5.3 Reference Clocks for SerDes Protocols

Each SerDes protocol allows for a finite set of valid SerDes-related RCW fields subject to external reference clock frequencies, as shown below. Note that the SRDS_RATIO_B_n (for n=1, 2, 3) RCW fields allow for PLL control per SerDes bank, and that the SRDS_DIV_B_n (for n=1, 2, 3) RCW fields allow for bit rate control per pair of lanes (for bank 1) or per quad set of lanes (for banks 2 and 3).

Table 4-39. Valid SerDes RCW Encodings and Reference Clocks

SerDes Protocol (given lane)	Valid Reference Clock Frequency	Valid Setting as Determined by SRDS_RATIO_B _n	Valid Setting as Determined by SRDS_DIV_B _n
PCI Express (2.5 Gbps) ¹	100 MHz	25:1	/1
		50:1	/2
	125 MHz	20:1	/1
		40:1	/2

Table continues on the next page...

Table 4-39. Valid SerDes RCW Encodings and Reference Clocks (continued)

SerDes Protocol (given lane)	Valid Reference Clock Frequency	Valid Setting as Determined by SRDS_RATIO_Bn	Valid Setting as Determined by SRDS_DIV_Bn
PCI Express (5 Gbps) ¹	100 MHz	50:1	/1
	125 MHz	40:1	/1
Serial RapidIO (2.5 Gbps)	100 MHz	25:1	/1
		50:1	/2
	125 MHz	20:1	/1
		40:1	/2
Serial RapidIO (3.125 Gbps)	125 MHz	25:1	/1
SGMII (1.25 Gbps) ²	100 MHz	25:1	/2
	125 MHz	10:1	/1
		20:1	/2
XAUI (3.125 Gbps)	125 MHz	25:1	/1
	156.25 MHz	20:1	/1
Debug (2.5 Gbps)	100 MHz	25:1	/1
		50:1	/2
	125 MHz	20:1	/1
		40:1	/2
Debug (3.125 Gbps)	125 MHz	25:1	/1
Debug (5 Gbps)	100 MHz	50:1	/1
	125 MHz	40:1	/1

1. A spread-spectrum reference clock is permitted for PCI Express. However, if serial RapidIO or debug is used concurrently on the same SerDes bank, spread-spectrum clocking is not permitted.
2. When SerDes bank 1 is configured for 5 Gbps operation (by SRDS_RATIO_B1 and the reference clock), lanes configured as SGMII operate at 1.25 Gbps regardless of the corresponding SRDS_DIV_B1 settings.

Chapter 5

Pre-Boot Loader (PBL)

5.1 PBL Overview

The pre-boot loader (PBL) performs configuration register reads and writes to initialize the I²C, eLBC FCM (NAND Flash), eSDHC, or SPI interface, loads the RCW and pre-boot initialization commands from external memory device through I²C, eLBC FCM, eLBC GPCM, eSDHC, or SPI and writes data to configuration registers or memory before the local cores are permitted to boot.

[Figure 5-1](#) shows a block diagram of PBL, showing the functional organization of the module, which includes interface command modules (ICMs) for eSDHC, I2C1, eSPI, eLBC, and a common control module.

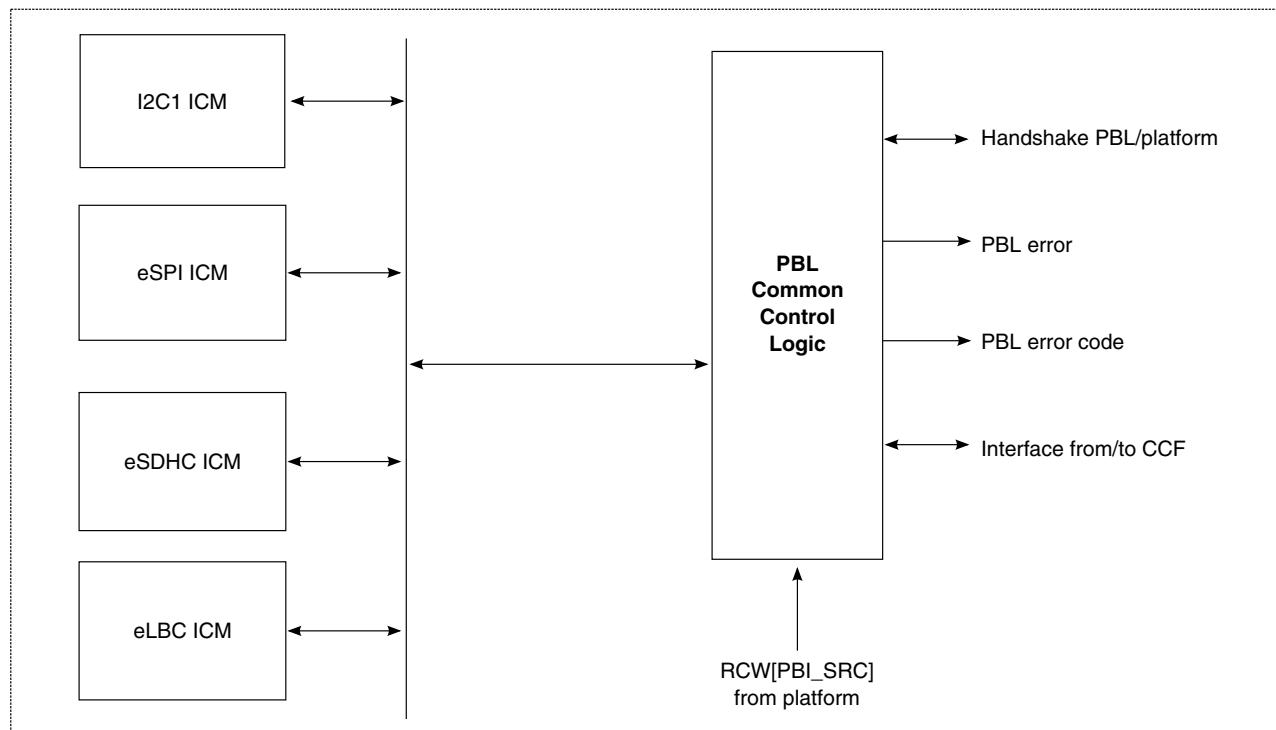


Figure 5-1. Pre-Boot Loader (PBL) Block Diagram

5.2 PBL Features Summary

- Initializes and loads RCW/pre-boot initialization commands from eSDHC SD/MMC interface
 - Supports multi-media card (MMC)
 - 1-bit data transfer on SD interface
 - Compliant with version 4.2 of the specification
 - Standard capacity (< 2 Gbytes)
 - High-capacity (> 2 Gbytes)
 - 3.3 V only
 - Supports Secure Digital (SD) memory card
 - 1-bit data transfer on MMC interface
 - Compliant with version 2.00 of the specification
 - Standard capacity (< 2 Gbytes)
 - High-capacity (> 2 Gbytes)
 - 3.3 V only
- Initializes and load RCW/pre-boot initialization commands from I2C1 interface
- Initializes and load RCW/pre-boot initialization commands from eLBC FCM/GPCM interface
- Initializes and load RCW/pre-boot initialization commands from eSPI interface
- Reports error upon receiving error response from each interface
- Reports error if RCW byte count is not 64 bytes
- Reports error if RCW address offset is not the specific 64-byte aligned offset to RCW status registers (RCWSR n) to which RCW contents are written
- Reports error in secure boot mode if pre-boot initialization address offset is inside of protected CCSR spaces
- Reports byte count and address misalignment error during pre-boot initialization
- Reports error if PBL command is in invalid format
- Reports error if there is no response from platform when timeout counter expires

5.3 PBL Modes of Operation

- Load both RCW and pre-boot initialization commands from individual interface.
- Load RCW only.
- Load pre-boot initialization commands only if default RCW is to be used.

5.4 PBL Functional Description

This section contains the following subsections:

- Configuration of Device via Reset Configuration Word (RCW)
- Device Initialization by PBL
- Required Format of Data Structure Consumed by PBL
- RCW Loading by PBL
- Pre-Boot Initialization Command Loading by PBL
- Reserved Address Space Used as Internal PBL Commands
- PBL Error Codes

5.4.1 Configuration of Device via Reset Configuration Word (RCW)

The reset configuration word (RCW) data contains reset configuration information that is either loaded by the PBL from a memory device during reset or set by the platform according to a hard-coded RCW. The size of the RCW is 64 bytes (512 bits). All of the data read from the RCW source is written to the RCW status registers (RCWSR n) in the device configuration and pin control block. See [Reset control word status registers 1-16 \(DCFG_RCWSR \$n\$ \)](#) for more information. The RCWSRs have specific 64-byte aligned address offsets that the PBL should ensure is written. Any address offset that is not that specific 64-byte aligned address offset is reported by the PBL as an error to the platform, which asserts the RESET_REQ output pin upon receiving the error from the PBL. Refer the Reset configuration word (RCW) source section in Reset, Clocking, and Initialization chapter to see how each encoding for cfg_rcw_src[0:8] selects a given source of RCW data.

5.4.2 Device Initialization by PBL

The cfg_rcw_src configuration input determines which interface to use for retrieving the RCW data. Similarly, RCW[PBI_SRC] determines the interface to use for pre-boot initialization. In these cases, the PBL initializes the I2C1, eSPI, eSDHC, or eLBC FCM (NAND flash) interface before accessing the RCW or PBI commands. Note that eLBC GPCM (NOR flash) does not require initialization by the PBL even though it may be selected as the source for the RCW or PBI commands. Each interface command module (ICM) inside the PBL issues CCSR space accesses to setup each interface properly. If there is error response during initialization, the PBL reports the error to the platform.

NOTE

Only one interface can be used as the source of RCW and pre-boot initialization data-the PBL data structure must not be split across two interfaces.

5.4.2.1 CCSR Registers Blocked from PBL During Secure Boot

Table 5-1 lists the CCSR address ranges that are blocked from PBL during secure boot.

Table 5-1. CCSR Registers Blocked from PBL During Secure Boot

Address Range	Size	Description
0x00_0000	4 bytes	CCSRBARD
0x00_0004	4 bytes	CCSRBARL
0x00_0008	4 bytes	CCSRAR
0x02_0000-0x02_FFFF	64 Kbytes	PAMU partitions 1-16
0x0E_8000-0x0E_8FFF	4 Kbytes	Security fuse processor (SFP)
0x30_0000-0x30_FFFF	64 Kbytes	Security Engine (SEC)
0x31_4000- 0x31_4FFF	4 Kbytes	Security monitor

5.4.3 Required Format of Data Structure Consumed by PBL

The RCW and pre-boot initialization commands share the same data structure format, regardless of the type of external memory device used as the source.

Table 5-2. Required Format of Data Structure Consumed by PBL

	0	1	2	3	4	5	6	7						
Preamble (required)	1	0	1	0	1	0	1	0						
	0	1	0	1	0	1	0	1						
	1	0	1	0	1	0	1	0						
	0	1	0	1	0	1	0	1						
RCW Data	ACS=0	BYTE_CNT = 000000 (64 bytes)						CONT=1						
	SYS_ADDR[23-16] ¹													
	SYS_ADDR[15-8] ¹													
	SYS_ADDR[7-0] ¹													
	BYTE0													
	BYTE1													
	BYTE2													
													

Table continues on the next page...

Table 5-2. Required Format of Data Structure Consumed by PBL (continued)

	0	1	2	3	4	5	6	7											
	BYTE63																		
First Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT				CONT=1													
	SYS_ADDR[23:16]																		
	SYS_ADDR[15:8]																		
	SYS_ADDR[7:0]																		
	BYTE0																		
	BYTE1																		
	BYTE2																		
																		
	BYTE N-1 (up to 63)																		
Second Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT				CONT=1													
	SYS_ADDR[23:16]																		
	SYS_ADDR[15:8]																		
	SYS_ADDR[7:0]																		
	BYTE0																		
	BYTE1																		
	BYTE2																		
																		
	BYTE N-1 (up to 63)																		
.....																			
.....																			
Last Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT				CONT=1													
	SYS_ADDR[23:16]																		
	SYS_ADDR[15:8]																		
	SYS_ADDR[7:0]																		
	BYTE0																		
	BYTE1																		
	BYTE2																		
																		
	BYTE N-1 (up to 63)																		
End Command (required, special CRC Check command with CONT=0)	ACS=0	BYTE_CNT = 00100 (4 bytes)				CONT=0													
	SYS_ADDR[23:16] = 0x13 ²																		
	SYS_ADDR[15:8] = 0x80 ²																		
	SYS_ADDR[7:0] = 0x40 ²																		
	CRC0																		
	CRC1																		
	CRC2																		
	CRC3																		

- SYS_ADDR for the RCW should point to the RCW status register (RCWSR[1-16]) in the device configuration and pin control block.

2. SYS_ADDR = 0x13_8040 is a PBI CRC check command (PBL block base address 0x13_8000 with offset 0x040). See [Pre-Boot Initialization Command Loading by PBL](#) for more information.

Table 5-3 provides field descriptions for the PBL data structure.

Table 5-3. PBL Data Structure Field Descriptions

Field Name	Description
ACS	Alternate Configuration Space. Logic 1 for Alternate Configuration Space. Logic 0 for CCSR Space. Note that this field must be logic 0 for the RCW datum.
BYTE_CNT	Byte Count. Represents the number of bytes associated with this addr/data pair. Note that an encoding of all zeros represents 64 bytes. Only power of 2 multiples are legal (1, 2, 4, 8, 16, 32, and 64 bytes). The associated SYS_ADDR must be aligned to the byte count of the command. 000000 64 bytes 000001 1 byte 000010 2 bytes 000100 4 bytes 001000 8 bytes 010000 16 bytes 100000 32 bytes all other encoding are reserved
CONT	Continue. This bit should be logic 1 for all commands except for the "End Command."
SYS_ADDR	System Address. The lowest order system address bits. Note that this permits addressability down to a byte for single byte transactions. SYS_ADDR must be aligned to the byte count of the command. The upper bits are either selected from CCSRBAR or Alt Config BAR (depending on value of ACS) and then concatenated with SYS_ADDR to form the full address associated with the command.
BYTEn	Byte number n where n may range from 0 up to 63. Byte 0 corresponds to addr 0 relative to the SYS_ADDR (starting address), byte 1 for addr 1..., and byte 63 for addr 63. Note the first command must be the RCW addr/data pair and must be 64 bytes of data . Note that BYTEn is only present/valid in the data structure if n is less than the BYTE_CNT.
CRCn	Cyclic Redundancy Check data. CRC value is calculated using CRC-32 algorithm with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC covers all bytes stored in the ROM prior to the CRC. The polynomial used is $1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$

5.4.4 RCW Loading by PBL

If a hard-coded RCW is not used and initialization of the source memory device completes without error, the PBL fetches RCW data from the source memory device and writes it to the RCW status registers (RCWSR n) in the device configuration and pin control block.

The PBL reports an error and sends a corresponding error code to the platform when one of the following conditions occurs:

- No preamble is detected.

- Alternate configuration space (ACS) is selected (ACS = 1).
- The byte count (BYTE_CNT) is not 64 bytes.
- The system address (SYS_ADDR) is not the specific address offset to the RCWSR to which the RCW contents are written.
- The system does not respond before internal time-out counter (32-bit) expires.

5.4.5 Pre-Boot Initialization Command Loading by PBL

If PBI is selected by the RCW[PBI_SRC] field, the PBL PBI commands are processed and routed to CCSR space, DDR, or other memory space.

The PBL reports an error and sends a corresponding error code to the platform when one of the following conditions occurs:

- CRC check error
- The byte count is not one of the supported values (1, 2, 4, 8, 16, 32, or 64 bytes). See [Table 5-3](#) for the supported encodings. Note that if byte count is 1 or 2, the PBL expects 3 or 2 bytes of padding (zeros) after the real data.
- The system address is misaligned to the byte count
- The system address is in an offset range of protected CCSR space. See [CCSR Registers Blocked from PBL During Secure Boot](#), for more information.
- The system does not respond before internal time-out counter expires.
- Invalid internal PBL commands.
- Invalid End command, for example, not valid CRC command with CONT=0.

5.4.6 Reserved Address Space Used as Internal PBL Commands

When a PBL command accesses the 4-Kbyte CCSR address space that is assigned to the PBL, the PBL treats this command entry as a PBL internal command. There are 4 bytes allocated for each command for command parameters. The byte count must be 4 bytes and unused data must be filled with 0s.

[Table 5-4](#) defines each PBL command.

Table 5-4. Description of PBL Commands

Command Name	Offset from CCSRBAR	Parameter(s)	Command Description
Flush	0x13_8000	N/A	Chases the previous write with a read from the same address. The purpose of this command is to ensure previous write has taken effect.

Table continues on the next page...

Table 5-4. Description of PBL Commands (continued)

Command Name	Offset from CCSRBAR	Parameter(s)	Command Description
CRC check	0x13_8040	The 4 bytes of data indicate the cyclic redundancy check (CRC) value. CRC value is calculated using CRC-32 algorithm with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000.	Checks CRC for data blocks from the point where the last CRC check was performed until the data block before this CRC check command.
Jump	0x13_8080	<p>The 4-bytes of data indicate the target address for the jump destination.</p> <p>For I²C/eSPI/eSDHC/NOR flash, this address is 4-byte address relative to current address. The address after the jump is a 4-byte aligned address.</p> <p>For 2K large page NAND flash, the higher 17 bits are used as a non-bad block index relative to current block, the lower 15 bits are ignored. The address after the jump is the start of target non-bad block.</p> <p>For small page NAND flash, the higher 20 bits are used as a non-bad block index relative to the current block, The lower 12 bits are ignored. The address after the jump is the start of target non-bad block.</p>	Upon receiving this command, the PBL reads next data from address starting from jump pointer indicated by the command parameter.
Wait	0x13_80C0	The 4 bytes of data indicate the number of cycles to wait. The clock source for the wait cycle is a PBL clock cycle (platform/2)	Upon receiving this command, the PBL waits a number of cycles indicated by the command parameter before reading the next data

5.4.7 PBL Error Codes

The PBL reports errors to the platform under various conditions. PBL error codes are also sent along with the error indication. The platform stores the error code in RSTRQPBLSR[ERR_CODE] in the device configuration and pin control block. See [Reset request preboot loader status register \(DCFG_RSTRQPBLSR\)](#) for more information.

The following table shows the encoding for each pre-boot error condition.

Table 5-5. Pre-Boot Error Encoding

Error Code[0:6]	Error Condition
0x00	Reserved
0x01	I ² C timeout while waiting for I2CSR[MIF] to assert
0x02	I ² C lost arbitration
0x03	I ² C did not receive ACK during address transmit

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
0x04	I ² C SCL signal was stuck low at POR
0x05-0x0F	Reserved
0x10	eSPI timeout while waiting for SPIE[DON] to assert
0x11-0x20	Reserved
0x21	eLBC detects error in LTESR register, user can scan out LTESR upon receiving this error. NOTE: In cases where LTESR fields BM, PAR, WP, or CS are set, the PBL will indicate a data transfer error from memory devices (encoding 0x78).
0x22	eLBC timeout error
0x23-0x3F	Reserved
0x40	eSDHC: Err_Reset_1 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x40	eSDHC: Err_Reset_2 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x41	eSDHC: Err_Card_Ins Indicates that eSDHC has not detected an inserted card. PRSSTAT:CINS (offset: 0x024, mask: 0x0001_0000) did not set within 1 second. It should set when eSDHC detects that a card has been inserted.
0x42	eSDHC: Err_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD0_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD0. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_CMD8_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD CMD8. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD55_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD55. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_ACMD41_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD ACMD41. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_MMC_CMD1_Cmd_Data_Rdy

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has not detected an idle command and data interface before attempting to send MMC CMD1. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD2_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD2. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_CMD3_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD CMD3. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_MMC_CMD3_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send MMC CMD3. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD9_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD9. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD7_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD7. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD18_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD18. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD12_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD12. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x43	eSDHC: Err_Init_Clk Indicates that eSDHC has not completed its initialization clock sequence to the card. SYSCTL:INITA (offset: 0x02C, mask: 0x0800_0000) did not clear within 1 second. It should clear when eSDHC completes transmission of initialization clocks to the card.
0x44	eSDHC: Err_CMD0_Cmd_Stat

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has not detected an error or command complete after sending CMD0. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_SD_CMD8_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD CMD8. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD55_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD55. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_SD_CMD41_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD ACMD41. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_MMC_CMD1_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending MMC CMD1. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD2_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD2. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_SD_CMD3_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_MMC_CMD3_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending MMC CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD9_Cmd_Stat

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has not detected an error or command complete after sending CMD9. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD7_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD7. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD18_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD18. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD12_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD12. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x45	eSDHC: Err_CMD0_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD0. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD8_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD CMD8. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD55_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD55. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_SD_ACMD41_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD ACMD41. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_MMC_CMD1_Cmd_Line_Conflict

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has detected a command line conflict after sending MMC CMD1. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD2_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD2. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_SD_CMD3_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD CMD3. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_MMC_CMD3_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending MMC CMD3. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD9_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD9. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD7_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD7. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD18_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD18. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD12_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD12. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x46	eSDHC: Err_SD_CMD8_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for SD CMD8. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons.

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	<ul style="list-style-type: none"> - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	<p>eSDHC: Err_CMD55_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for CMD55. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	<p>eSDHC: Err_SD_ACMD41_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for SD ACMD41. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) set. It will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command End Bit Error (CEBE)
0x46	<p>eSDHC: Err_MMC_CMD1_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for MMC CMD1. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) set. It will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command End Bit Error (CEBE)
0x46	<p>eSDHC: Err_CMD2_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for CMD2. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	<p>eSDHC: Err_SD_CMD3_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for SD CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	<p>eSDHC: Err_MMC_CMD3_Cmd_Rsp</p> <p>Indicates that eSDHC has detected an error with the command response for MMC CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons.</p> <ul style="list-style-type: none"> - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD9_Cmd_Rsp

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has detected an error with the command response for CMD9. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD7_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD7. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD18_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD18. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD12_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD12. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x47	eSDHC: Err_SD_CMD8_Rsp_Data_Match Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of SD CMD8. CMDRSP0:VHS (offset: 0x010, mask: 0x0000_0F00) was not equal to the configured SD 'coarse' voltage or CMDRSP0:CHKPTRN (offset: 0x010, mask: 0x0000_00FF) was not equal to the configured SD check pattern. Both fields have to match in order to guarantee proper communication with the card.
0x47	eSDHC: Err_SD_ACMD41_Rsp_Data_Match Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of SD ACMD41. CMDRSP0:VDDVW (offset: 0x010, mask: 0x00FF_FFFF) was not covered (matches at least one bit) by the configured SD 'fine' voltage. At least one bit must match in order to guarantee proper communication with the card.
0x47	eSDHC: Err_MMC_CMD1_Rsp_Data_Match Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of MMC CMD1. CMDRSP0:VDDVW (offset: 0x010, mask: 0x00FF_FFFF) was not covered (matches at least one bit) by the configured MMC voltage. At least one bit must match in order to guarantee proper communication with the card.

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
0x48	eSDHC: Err_SD_ACMD41_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for SD ACMD41. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_MMC_CMD1_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for MMC CMD1. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD2_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD2. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_SD_CMD3_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for SD CMD3. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_MMC_CMD3_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for MMC CMD3. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD9_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD9. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD7_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD7. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD18_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD18. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD12_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD12. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x49	eSDHC: Err_SD_ACMD41_Rdy_Timeout Indicates that the card always indicates busy by examining the response of SD ACMD41. CMDRSP0:RDY (offset: 0x010, mask: 0x8000_0000) did not set within 1 second. It should set when the card completes its initialization sequence.
0x49	eSDHC: Err_MMC_CMD1_Rdy_Timeout

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that the card always indicates busy by examining the response of MMC CMD1. CMDRSP0:RDY (offset: 0x010, mask: 0x8000_0000) did not set within 1 second. It should set when the card completes its initialization sequence.
0x4A	eSDHC: Err_Rd_Buf_Rdy Indicates that eSDHC has not received the watermark's amount (configured to 4 bytes) of data in its read buffer from CMD18. PRSSTAT:BREN (offset: 0x024, mask: 0x0000_0800) did not set within 1 second. It should set when eSDHC receives the watermark's amount of data in its read buffer.
0x4B	eSDHC: Err_Data Indicates that eSDHC has detected an error with the read data from CMD18. IRQSTAT:DEBE (offset: 0x030, mask: 0x0040_0000) or IRQSTAT:DCE (offset: 0x030, mask: 0x0020_0000) set. Each will set for one of the corresponding reasons. - Data End Bit Error (DEBE) - Data CRC Error (DCE)
0x4C	eSDHC: Err_Data_Timeout Indicates that eSDHC has detected a timeout with the read data from CMD18. IRQSTAT:DTOE (offset: 0x030, mask: 0x0010_0000) set. It will set for one of the corresponding reasons. - Data Timeout Error (DTOE)
0x4D	eSDHC: Err_Data_Reset Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTD (offset: 0x02C, mask: 0x0400_0000) did not clear within 1 second. It should clear when eSDHC completes its data 'reset' sequence.
0x4E	eSDHC: Err_PBI_Timeout Indicates that pre-boot initialization (PBI) has not started within 1 second. An indication should be received when PBI starts.
0x4F	eSDHC: Reserved
0x50-0x6F	Reserved
0x70	No preamble is detected
0x71	ACS is logic 1 during RCW
0x72	Byte count is not 64 for RCW or 4 for PBL commands or 1/2/4/8/16/32/64 for pre-boot initialization commands
0x73	Misaligned address
0x74	Address is in protected space
0x75	CRC Error
0x76	Time out counter expires
0x77	Unrecognized PBL commands, including unrecognized offset and invalid parameter.
0x78	Data Transfer error from memory devices
0x79	Invalid End command error
0x7A	Invalid RCW or pre-boot initialization word source encoding
0x7B-0x7F	Reserved

5.5 Registers Written by PBL During RCW and PBI Phases

This section lists registers written by PBL during RCW and PBI phases.

If the interface is the PBI/RCW source, the registers listed below are reset back to their POR values, when PBL is completed. Alternatively, if the registers are modified by some external source, then its value does not reset back to POR value when PBL is completed.

5.5.1 I²C Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- I2CFDR
- I2CCR
- I2CSR
- I2CDR
- I2CDFSRR

5.5.2 eSPI Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- SPMODE
- SPIE
- SPCOM
- SPITF
- SPIRF (read only)
- SPMODE0

5.5.3 eLBC Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- MDR
- LSOR
- LTESR

- LTEATR
- FMR
- FIR
- FCR
- FBAR
- FPAR
- FBCR

5.5.4 eSDHC Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed. The following eSDHC registers are reset by performing a software reset, asserting SYSCTL:RSTA:

- BLKATTR (Offset: 0x004)
- CMDARG (Offset: 0x008)
- XFERTYP (Offset: 0x00C)
- CMDRSP0 (Offset: 0x010)
- CMDRSP2 (Offset: 0x018)
- DATPORT (Offset: 0x020)
- PRSSTAT (Offset: 0x024)
- PROCTL (Offset: 0x028)
- SYSCTL (Offset: 0x02C)
- IRQSTAT (Offset: 0x030)
- IRQSTATEN (Offset: 0x034)
- WML (Offset: 0x044)

5.6 Addressing Multiple I²C EEPROMs

When standard I²C EEPROMs are used, it is assumed that each EEPROM holds 256 bytes of data. After 256 bytes have been received, the PBL will increment the slave address and access the next EEPROM. A maximum of eight standard I²C EEPROMs may be used with the PBL.

Extended I²C EEPROMs have variable sizes. However, the I²C controller increments the slave address after 64 Kbytes have been received. In some cases, the least significant bit of the slave address refers to a different page address on an extended EEPROM (for the larger density EEPROMs). However, the PBL should still be able to access the page successfully. Alternatively, the least significant bit may reference a second extended

EEPROM on the I²C bus. Multiple extended EEPROMs of sizes less than 64 Kbytes are not supported (unless a jump command is used to force the PBL to the next EEPROM device).

5.7 eSPI Mode Assumptions

eSPI mode assumptions are listed as follows:

- Chip select-when using eSPI as the source of PBL, the first chip select (CS0) is used. PBL uses the SPMODE0 register in the eSPI block to use CS0 during initialization.
- Supported SPI mode-when using eSPI as the source for PBL, only SPI Mode 0 is supported (referring to the SPI clock phase and polarity). The SPI slave device must also support SPI Mode 0.
- SPI Read command-when using eSPI as the source for PBL, a value of 0x03 is used for the read command. The SPI slave device must decode 0x03 as a read command.

5.8 PBL Initialization/Application Information

The PBL starts searching for the first good block. It begins to fetch the PBL image at the starting address of that block. It automatically searches for the next good block should the PBL image extend past one block.

If booting from NAND, the boot image must be placed on the next 4-Kbyte boundary. The PBL searches for the next good block should that 4-Kbyte boundary fall on a block boundary. The PBL then loads 4 Kbytes into the buffer.

This PBL block search routine continues to search as necessary until reaching the end of the memory device.

5.8.1 Starting Addresses

[Table 5-6](#) lists the starting addresses of data fetched from each interface. Note that in case of eLBC NAND flash, the address starts from the first good block.

Table 5-6. Starting Addresses

Interface	Starting Address
eLBC	0x00000000
I ² C ¹	0x00000000
eSPI	0x00000000

Table continues on the next page...

Table 5-6. Starting Addresses (continued)

Interface	Starting Address
eSDHC	0x00001000

1. The 7-bit calling address for the I²C slave is 0x50.

5.8.2 Software Restrictions

The following are restrictions for programming the PBI commands:

- CCSRBAR cannot be updated through PBI commands. Software must wait for the cores to boot to change CCSRBAR.
- Software must issue a Flush command after updating the alternate configuration space registers (ALTCBARH, ALTCBARL, and ALTCAR) using PBI commands. This ensures that the logic has seen the update prior to any dependent writes being issued by the PBL.
- Use of the Flush command is restricted to CCSR space. Software should use the Wait command after commands to non-CCSR space to allow them time to complete before issuing subsequent commands to non-CCSR space.
- The SRIO memory-mapped configuration registers cannot be updated through PBI commands. Software must wait for the core(s) to boot before initializing the SRIO configuration registers.

5.8.3 Software Recommendations

The following are recommended for programming the PBL data structure:

- A CRC check command should always be placed immediately following the RCW command to ensure that a corrupted RCW is not consumed.



Chapter 6

Secure Boot and Trust Architecture 1.0

Chapter 7

e500mc Core Integration

7.1 Introduction

This chapter provides details about how the eight e500mc microprocessor cores are integrated into the P4080 . The e500mc core provides features that the integrated device may not implement or may implement in a more specific way.

NOTE

The following documents are necessary for understanding how to program the core: The *EREF: A Programmer's Reference Manual for Freescale Embedded Processors* describes the instruction, register, and interrupt models, as well as other functionality defined at the architecture level and implemented on e500 family processors. The *e500mc Reference Manual* describes functionality that is specific to the e500mc and that is not defined by the architecture, such as core-specific instructions, registers, and register fields. It also provides e500mc-specific details about how the e500mc implements functionality that is defined by the architecture.

7.2 e500mc Core Overview

The e500mc core is a low-power implementation of the resources for embedded processors defined by the Power ISA. The core is a 32-bit implementation and implements 32-bit general-purpose register; however it supports accesses to 36-bit physical addresses.

The e500mc is designed to be implemented in multicore integrated devices, and many of the new features are defined to support multicore implementations, in particular to partition the cores in such a way that multiple operating systems can be run with the integrated device, as shown in the following figure.

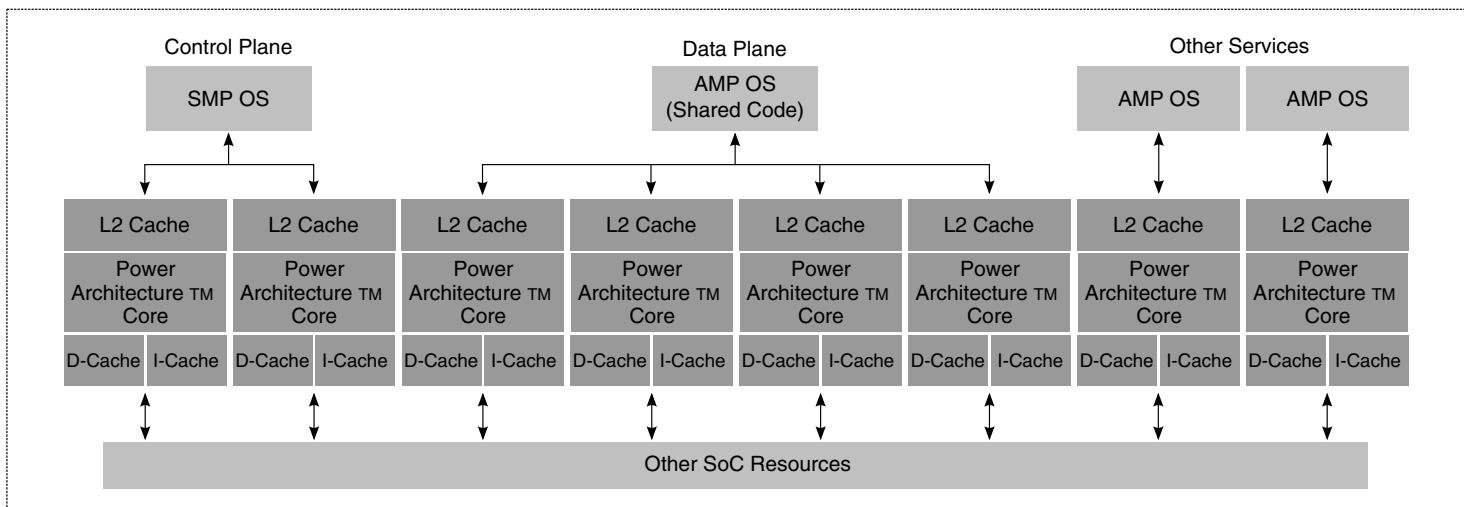


Figure 7-1. Example Partitioning Scenario of a Multicore Integrated Device

The architecture defines the resources required to allow orderly and secure interactions between the cores, memory, peripheral devices, and virtual machines. These include a hypervisor and guest supervisor privilege levels, that determine whether certain activities, such as memory accesses and management, cache management, and interrupt handling, are to be carried on at a system-wide level (hypervisor level) or by the operating system within a partition (guest supervisor level).

Each e500mc is a superscalar processor that can issue two instructions (out of order) and complete two instructions (in order) per clock cycle. Instructions complete in order, but can execute out of order. Execution results are available to subsequent instructions through the rename buffers, but those results are recorded into architected registers in program order, maintaining a precise exception model.

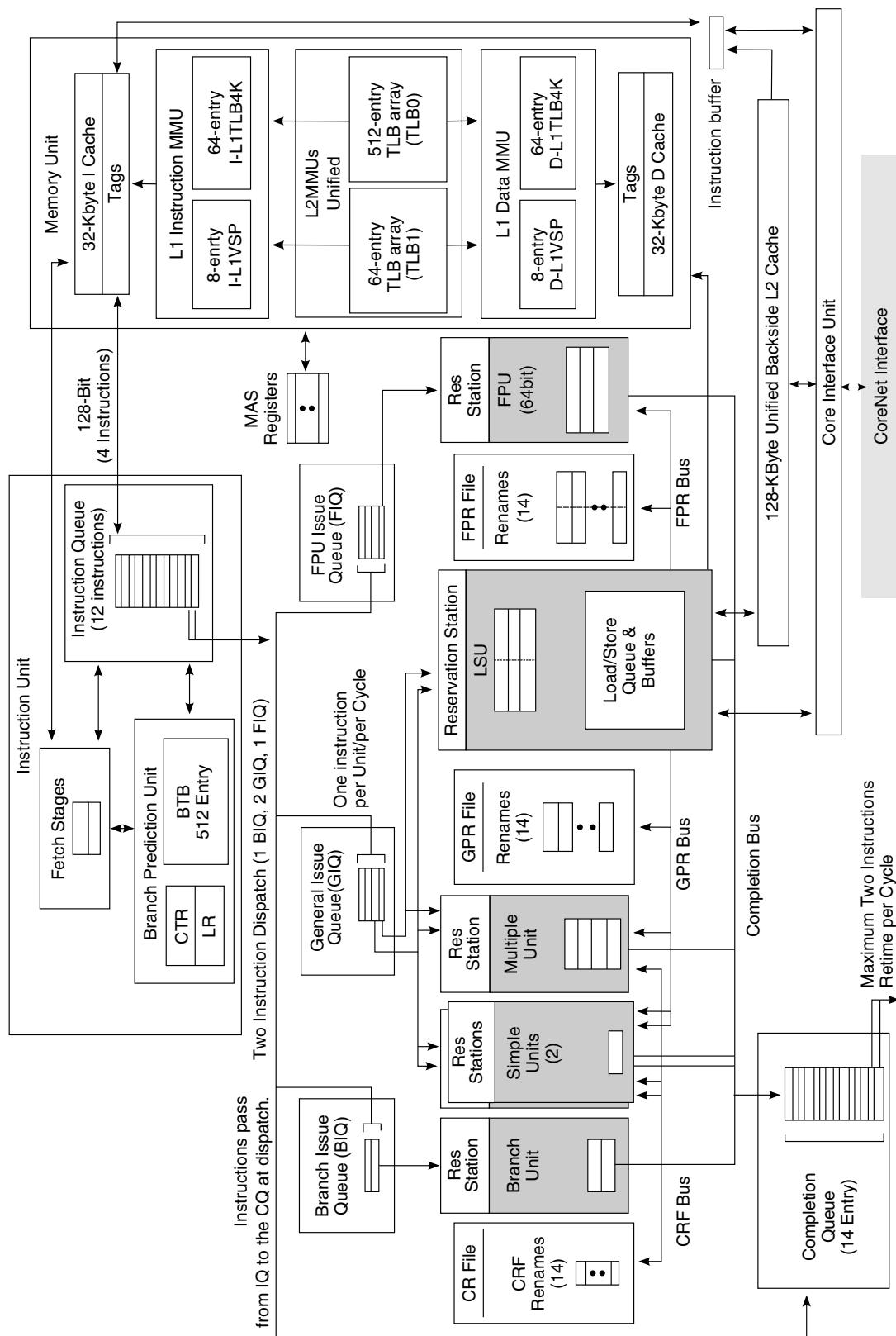


Figure 7-2. e500mc Block Diagram

7.3 P4080 -Specific Core Implementation Details

This section describes any aspects of the implementation of the e500mc cores that are specific to the P4080 . Information in this section supersedes any more general descriptions provided in either *EREF: A Programmer's Reference Manual for Freescale Embedded Processors* or the *e500mc Reference Manual*.

7.3.1 Eight e500mc Cores

The eight e500mc cores implemented on the device can be combined as a fully symmetric multiprocessing system on a chip, or they can be operated with variable degrees of independence to perform asymmetric multiprocessing. Full processor independence, including the ability to independently boot and reset each e500mc core, is a defining characteristic of the P4080 . The ability of the cores to run different operating systems, or run without an OS, provides significant flexibility in partitioning among control, data path, and applications processing. It also simplifies consolidation of functions previously spread across multiple discrete processors onto a single device.

7.3.2 CoreNet Coherency Fabric

The CoreNet interface fabric provides interconnections among the cores, peripheral devices, and system memory in a multicore implementation. [CoreNet Coherency Fabric \(CCF\)](#), describes the CoreNet coherency fabric in detail, and in particular it explains how it facilitates communication between the e500mc cores, the L3 cache, and the other blocks that comprise the coherent memory domain.

7.3.3 Reset and Clocking

[Reset, Clocking, and Initialization](#) provides information about coordination between the core and the device as part of the boot process.

The following signals are important to consider regarding the integration of the cores:

- The system clock (SYSCLK), is the primary clock input and is the clock source for the core complex PLLs. Core complex PLL outputs are assigned to specific cores

using their associated *C_n_PLL_SEL* fields in the RCW. For information, see [Reset Configuration Word \(RCW\)](#).

- The real time clock, RTC, may be used (optionally) to clock the time base of the e500mc cores. The RTC timing specifications are given in the chip hardware specifications. This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC) and IEEE 1588.

e500mc timer facilities are described in the *e500mc Reference Manual*.

7.3.4 Register Model Implementation Details

The device implements the e500mc register model as defined for the e500mc and described in the *EREF* and the *e500mc Reference Manual*. Details that are specific to how the e500mc registers are implemented in the device are described in the following subsections.

7.3.4.1 Processor Version Register (PVR)

[Table 7-1](#) lists the revision codes in the processor version register (PVR). The PVR value is source provided by the e500mc processor itself. Its value is reflected in the PVR register, which is described in [Processor version register \(DCFG_PVR\)](#). The value is also visible to software running on the e500mc in a read-only SPR within the core, SPR287.

Table 7-1. Processor Version Register

ID	Bits	Assigned Value(Binary)	Description
PVR	[0:3]	1000	Manufacturer ID
	[4:5]	00	Reserved
	[6:9]	0000	Processor type
	[10:15]	10_0011	Processor ID
	[16:19]	0000	Process revision
	[20:23]	0000	Manufacturing revision
	[24:27]	<i>nnnn</i>	Processor major revision number
	[28:31]	<i>mmmm</i>	Processor minor revision number

7.3.4.2 System Version Register (SVR)

The SVR register, described in [System version register \(DCFG_SVR\)](#), is routed to input terminals on the e500mc as well where it is accessible as a read-only register within the core, SPR1023. [Table 7-2](#) contains the assigned values for the device.

Table 7-2. System Version Register

ID	Bits	Assigned Value (Binary)	Description	Comments
SVR	[0:3]	1000	Manufacturer ID	Freescale
	[4:11]	0010_0000	SoC device ID	P4
	[12]	<i>n</i>	Security	0 No security 1 Security enabled
	[13:15]	<i>nnn</i>	SoC variant ID	000 P4080 001 P4040
	[16:23]	0000_0000	Personality	0x00 P4080/P4040
	[24:27]	<i>nnnn</i>	Device major revision number	For first silicon, this is set to 4'h1 for major revision 1.
	[28:31]	<i>mmmm</i>	Device minor revision number	For first silicon, this is set to 4'h0 for minor revision 0.

7.3.4.3 (Guest) Processor ID register (PIR/GPIR)

The value in PIR/GPIR is used to distinguish processors in a system from one another. The processor control category augments this definition to make the PIR writable so software can store information specific to its needs.

In a multiprocessor system, each PIR register should be initialized at power-on-reset to a unique value. The contents of the PIR register are used as a tag for matching requested doorbell interrupts sent to the processor. Note that SPR 286 was assigned to the system version register (SVR) on some earlier devices.

Software can safely assume that only core 0's PIR is 0 out of reset.

7.3.4.4 Timer Control Register (TCR)

TCR[WRC] is defined more specifically for the implementation of the core in the integrated device, as shown in [Table 7-3](#).

Table 7-3. TCR[WRC] Descriptions

Bits	Name	Description
34-35	WRC	See Watchdog Timer Expiration Out to Platform .

7.3.5 Cache Model Implementation Details

The device's implemented caches can be configured and locked using instructions defined by the Power Architecture and described in the *EREF* and in the *e500mc Core Reference Manual*. These instructions specify the backsideL2cache by setting the CT operand to 2 and the CoreNet platform cache (CPC) by setting the CT operand to 1.

Caches in a system may be specific targets of *cache stashing*, an operation initiated by the processor or other device in the system, specifying a hint that specified addresses should be prefetched into a target cache. Each cache in the system that responds to such requests has a cache identifier set by system software or predefined by hardware. For the L1 data cache, the identifier is defined in L1CSR2[DCSTASHID]. For the backside cache, the identifier is defined in L2CSR1[L2STASHID]. A cache identifier value of 0 indicates that cache stashing operations are not accepted or performed by the cache.

NOTE

The L2 cache must have ECC enabled as follows:

- L2CSR0[L2PE] = 1
- L2ERRDIS[MBECCDIS] = 0 and L2ERRDIS[SBECCDIS] = 0

7.3.6 Interrupt Model Implementation Details

The e500mc implements the external proxy functionality, which can be used to eliminate the need for an interrupt handler to perform a software read of the MPIC IACK register. See [Multicore Programmable Interrupt Controller \(MPIC\)](#) for information about how this functionality is implemented on the chip.

The architecture defines resources to eliminate the need to read the IACK register from an interrupt handler responding to the assertion of the external interrupt signal, *int*, from a programmable interrupt controller (PIC) in the integrated device. The Open PIC standards define a software handshake in response to the assertion of this signal that requires the handler to read the memory-mapped IACK register, which holds interrupt vector offset bits provided by the source of the interrupt, typically one of the peripheral devices incorporated in the integrated device. In addition to providing the interrupt vector, this read also informs internal logic of the PIC to treat this interrupt source as "in-service" rather than pending and it signals the PIC to deassert its internal *int* output, making it possible for the PIC to deliver an external interrupt from another peripheral device that had been configured as higher priority

The architecture defines the external proxy register (EPR) used when the interrupt is handled at the hypervisor level, and a guest EPR, GEPR, when the interrupt is handled at the guest supervisor level. The EPR and GEPR are SPRs, and the value that appears in the PIC's IACK[VECTOR] fields is automatically loaded into the EPR (or GEPR), eliminating the cache-inhibited guarded load to the memory mapped IACK register. The EPR is considered valid only from the time that the external input interrupt occurs until MSR[EE] is set as the result of a **mtmsr** or a return from interrupt instruction.

The interrupt proxy functionality can be enabled and disabled through the global configuration register in the PIC. See [Multicore Programmable Interrupt Controller \(MPIC\)](#) for more information.

7.3.6.1 Watchdog Timer Expiration Out to Platform

Each core has an internal watchdog timer that can be configured to signal the platform upon "first" or "second" expiration; see the *e500mc Core Reference Manual* for details. Such an indication to the platform typically identifies runaway or faulty software or that something has gone wrong interacting with the platform.

There are four different actions taken in response to such a watchdog timer expiration depending on the value programmed in TCR[WRC] and EDBCR0[EDM] of the core:

- The platform indicates to the MPIC that a core watchdog timer event has occurred for WRC/WRS==01 while EDM core output == 0
- The platform indicates to the MPIC that a core watchdog timer event has occurred and automatic reset of that core for WRC/WRS==11 while EDM core output == 0
- The platform indicates to the MPIC that a core watchdog timer event has occurred and device output RESET_REQ_B is asserted to external world: WRC/WRS==10 while EDM core output == 0
- SoC-specific debug action taken when WRC/WRS==01, 10, 11 while EDM core output == 1.

The platform indicates to the MPIC that a core watchdog timer event has occurred by updating the core's associated field (WRSn) in the MPIC's Watchdog Status Register Summary Register (WSRSR). If any of the WRSn bits in WSRSR are set (essentially ORing all watchdog timer expiration events), an internal interrupt is generated. The internal interrupt can be configured to be delivered as either a regular, critical, or machine check type interrupt to a designated core. See [Multicore Programmable Interrupt Controller \(MPIC\)](#).

7.3.7 Performance Monitor Details

The performance monitor facility provides the ability to monitor and count predefined events such as processor clocks, misses in the instruction cache or data cache, types of instructions decoded, or mispredicted branches. The count of such events can be used to trigger the performance monitor interrupt or a Nexus event. Additional performance monitor registers (PMRs) similar to SPRs are used to configure and track performance monitor operations. These registers are accessed with the Move to PMR and Move from PMR instructions (**mtpmr** and **mfpmr**).

Note that the core's performance monitor is separate from the SoC-level performance monitor which is similar in its design, but uses memory-mapped performance monitor registers.

The performance monitor can be configured to trigger either a performance monitor interrupt or an event to the Nexus facility when configured conditions are met.

NOTE

On P4080, Rev 2, when using the PMLCx[EVENT] field and the corresponding counter, PMCx, is not frozen, the following event counts are inaccurate. These event counts are accurate on P4080, Rev 3.

- Event 16 - the number of branches in the BTB mispredicted due to direction prediction are not counted correctly.
- Event 73 - The snoop hit event only counts snoop hits in the cast out buffers, the line fill buffers, and the L1 Data cache. It does not count snoop hits in the L1 cast away and Backside-L2.
- Event 126 - The L2 castout event only counts castouts from the DLFB. It does not count cast outs from the Backside L2 as expected.

NOTE

On P4080, Rev 2, when a performance monitor counter is set up to count during user mode (PMLCx[FCS]=1,MSR[PR]=1), the following events (PMLCal[EVENT]) are not counted. These events are counted on P4080, Rev 3 silicon.

- Event 86 - Interrupts taken
- Event 87 - External input interrupts taken
- Event 88 - Critical input interrupts taken
- Event 89 - System call and trap interrupts

7.3.8 Decorated Load and Store Operations

The architecture defines a set of decorated load and store instructions that allow efficient, SoC-specific operations, such as atomically sampling and updating packet-counting statistics. The SoC defines specific semantics understood by an SoC-customized resource that requires them. See [Decorated Storage](#) for more information.

The architecture defines a set of load/store byte, half-word, and word decorated integer operations and load/store floating-point double-word instructions, which provide the EA in **rB** and the decoration in **rA**.

The Decorated Storage Notify (**dsn**) instruction is an address-only operation that sends a decoration without sending data.

7.3.9 Debug Features

The e500mc provides a comprehensive, extensively integrated set of internal and external debug features to support multicore implementations:

- Debug interrupt is a separate interrupt type
- Nexus facility
- Performance monitor now provides the ability to trigger either a performance monitor interrupt or a Nexus event.
- Debug resources are extensively integrated at the SoC level. Implementation-specific details about e500mc debug resources are described in the *EREF* and the *e500mc Core Reference Manual*.

7.3.10 Power Management

The e500mc defines registers and instructions that provide power management that is programmable from the core. See the register and instruction model chapters of the *e500mc Reference Manual*. Integration details are provided in [Run Control/Power Management \(RCPM\)](#), which describes the run control and power management (RCPM) block, which allows the device to operate at various frequencies and execution modes to optimize power consumption.

Chapter 8

CoreNet Platform Cache (CPC)

8.1 Introduction

8.1.1 CPC Overview

The CoreNet platform cache (CPC) is a CoreNet-compliant target device that functions as a general purpose write-back cache, I/O stash and memory mapped SRAM device, or a combination of these functions. As a general purpose cache, it manages allocations and victimizations to improve read latency and bandwidth over accesses to backing store (for example, DRAM). As an I/O stash, it can accept and allocate writes from an I/O device in order to reduce latency and improve bandwidth for future read operations to the same address. As an SRAM device, it acts as a low-latency, high-bandwidth memory that occupies a programmable address range.

The CPC connects between the CoreNet coherency fabric (CCF) and the memory controller in an inline configuration and cannot function as a lookaside cache. Therefore, the CoreNet platform cache can only cache address ranges that are present in the memory controller behind it. This limitation does not apply to SRAM, which does not have backing store.

8.1.2 Features

The CPC includes the following features:

- In-line configuration
- 1-Mbyte, 32-way set associative, 64-byte coherency granule
- Configurable pseudo-least recently used (PLRU), streaming PLRU with aging, streaming PLRU without aging, and first-in/first-out (FIFO) replacement policies with programmable allocation policy and update options
- Support for individual line locking

- Write-back or write-through operation
- Low latency interface to memory controller including a dedicated 128-bit data bus in each direction
- Partial SRAM and partial I/O stash mode with programmable sizes
- Stashing support
 - Stashing of all transactions and sizes supported (not limited to 64-byte or 32-byte)
 - Explicit (CoreNet signalled) and implicit (address range based) stash allocation
- Support for decorated storage

8.2 CoreNet Platform Cache (CPC) Memory Map

The following table shows the memory map for the CoreNet platform cache (CPC) registers. The CPC registers are accessed by reading and writing to an address comprised of the base address (specified by the CCSRBAR), plus the CPC block base address, plus the offset of the specific register to be accessed.

CPC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_0000	CPC configuration and status register 0 (CPC1_CPCCSR0)	32	R/W	0000_0000h	8.2.1/349
1_0008	CPC configuration register 0 (CPC1_CPCCFG0)	32	R	50B7_C010h	8.2.2/352
1_0010	CPC external write control register n (CPC1_CPCCEWCR0)	32	R/W	0000_0000h	8.2.3/354
1_0014	CPC external write base address register n (CPC1_CPCCEWBAR0)	32	R/W	0000_0000h	8.2.4/355
1_0020	CPC external write control register n (CPC1_CPCCEWCR1)	32	R/W	0000_0000h	8.2.3/354
1_0024	CPC external write base address register n (CPC1_CPCCEWBAR1)	32	R/W	0000_0000h	8.2.4/355
1_0100	CPC SRAM control register 1 (CPC1_CPCCSR1)	32	R/W	0000_0000h	8.2.5/356
1_0104	CPC SRAM control register 0 (CPC1_CPCCSR0)	32	R/W	0000_0000h	8.2.6/356
1_0200	CPC partition ID register 0 (CPC1_CPCPIR0)	32	R/W	FFFF_FFFFh	8.2.7/357
1_0208	CPC partition allocation register 0 (CPC1_CPCPAR0)	32	R/W	FFFF_FBFH	8.2.8/358
1_020C	CPC partition way register 0 (CPC1_CPCPWR0)	32	R/W	FFFF_FFFFh	8.2.9/359
1_0210	CPC partition ID register n (CPC1_CPCPIR1)	32	R/W	0000_0000h	8.2.10/360
1_0218	CPC partition allocation register n (CPC1_CPCPAR1)	32	R/W	0000_0000h	8.2.11/360
1_021C	CPC partition way register 0 (CPC1_CPCPWR1)	32	R/W	0000_0000h	8.2.12/362
1_0220	CPC partition ID register n (CPC1_CPCPIR2)	32	R/W	0000_0000h	8.2.10/360
1_0228	CPC partition allocation register n (CPC1_CPCPAR2)	32	R/W	0000_0000h	8.2.11/360

Table continues on the next page...

CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_022C	CPC partition way register 0 (CPC1_CPCPWR2)	32	R/W	0000_0000h	8.2.12/362
1_0230	CPC partition ID register n (CPC1_CPCPIR3)	32	R/W	0000_0000h	8.2.10/360
1_0238	CPC partition allocation register n (CPC1_CPCPAR3)	32	R/W	0000_0000h	8.2.11/360
1_023C	CPC partition way register 0 (CPC1_CPCPWR3)	32	R/W	0000_0000h	8.2.12/362
1_0240	CPC partition ID register n (CPC1_CPCPIR4)	32	R/W	0000_0000h	8.2.10/360
1_0248	CPC partition allocation register n (CPC1_CPCPAR4)	32	R/W	0000_0000h	8.2.11/360
1_024C	CPC partition way register 0 (CPC1_CPCPWR4)	32	R/W	0000_0000h	8.2.12/362
1_0250	CPC partition ID register n (CPC1_CPCPIR5)	32	R/W	0000_0000h	8.2.10/360
1_0258	CPC partition allocation register n (CPC1_CPCPAR5)	32	R/W	0000_0000h	8.2.11/360
1_025C	CPC partition way register 0 (CPC1_CPCPWR5)	32	R/W	0000_0000h	8.2.12/362
1_0260	CPC partition ID register n (CPC1_CPCPIR6)	32	R/W	0000_0000h	8.2.10/360
1_0268	CPC partition allocation register n (CPC1_CPCPAR6)	32	R/W	0000_0000h	8.2.11/360
1_026C	CPC partition way register 0 (CPC1_CPCPWR6)	32	R/W	0000_0000h	8.2.12/362
1_0270	CPC partition ID register n (CPC1_CPCPIR7)	32	R/W	0000_0000h	8.2.10/360
1_0278	CPC partition allocation register n (CPC1_CPCPAR7)	32	R/W	0000_0000h	8.2.11/360
1_027C	CPC partition way register 0 (CPC1_CPCPWR7)	32	R/W	0000_0000h	8.2.12/362
1_0280	CPC partition ID register n (CPC1_CPCPIR8)	32	R/W	0000_0000h	8.2.10/360
1_0288	CPC partition allocation register n (CPC1_CPCPAR8)	32	R/W	0000_0000h	8.2.11/360
1_028C	CPC partition way register 0 (CPC1_CPCPWR8)	32	R/W	0000_0000h	8.2.12/362
1_0290	CPC partition ID register n (CPC1_CPCPIR9)	32	R/W	0000_0000h	8.2.10/360
1_0298	CPC partition allocation register n (CPC1_CPCPAR9)	32	R/W	0000_0000h	8.2.11/360
1_029C	CPC partition way register 0 (CPC1_CPCPWR9)	32	R/W	0000_0000h	8.2.12/362
1_02A0	CPC partition ID register n (CPC1_CPCPIR10)	32	R/W	0000_0000h	8.2.10/360
1_02A8	CPC partition allocation register n (CPC1_CPCPAR10)	32	R/W	0000_0000h	8.2.11/360
1_02AC	CPC partition way register 0 (CPC1_CPCPWR10)	32	R/W	0000_0000h	8.2.12/362
1_02B0	CPC partition ID register n (CPC1_CPCPIR11)	32	R/W	0000_0000h	8.2.10/360
1_02B8	CPC partition allocation register n (CPC1_CPCPAR11)	32	R/W	0000_0000h	8.2.11/360
1_02BC	CPC partition way register 0 (CPC1_CPCPWR11)	32	R/W	0000_0000h	8.2.12/362
1_02C0	CPC partition ID register n (CPC1_CPCPIR12)	32	R/W	0000_0000h	8.2.10/360
1_02C8	CPC partition allocation register n (CPC1_CPCPAR12)	32	R/W	0000_0000h	8.2.11/360
1_02CC	CPC partition way register 0 (CPC1_CPCPWR12)	32	R/W	0000_0000h	8.2.12/362
1_02D0	CPC partition ID register n (CPC1_CPCPIR13)	32	R/W	0000_0000h	8.2.10/360
1_02D8	CPC partition allocation register n (CPC1_CPCPAR13)	32	R/W	0000_0000h	8.2.11/360
1_02DC	CPC partition way register 0 (CPC1_CPCPWR13)	32	R/W	0000_0000h	8.2.12/362
1_02E0	CPC partition ID register n (CPC1_CPCPIR14)	32	R/W	0000_0000h	8.2.10/360
1_02E8	CPC partition allocation register n (CPC1_CPCPAR14)	32	R/W	0000_0000h	8.2.11/360
1_02EC	CPC partition way register 0 (CPC1_CPCPWR14)	32	R/W	0000_0000h	8.2.12/362
1_02F0	CPC partition ID register n (CPC1_CPCPIR15)	32	R/W	0000_0000h	8.2.10/360

Table continues on the next page...

CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_02F8	CPC partition allocation register n (CPC1_CPCPAR15)	32	R/W	0000_0000h	8.2.11/360
1_02FC	CPC partition way register 0 (CPC1_CPCPWR15)	32	R/W	0000_0000h	8.2.12/362
1_0E00	CPC error injection high register (CPC1_CPCERRINJHI)	32	R/W	0000_0000h	8.2.13/362
1_0E04	CPC error injection low register (CPC1_CPCERRINJLO)	32	R/W	0000_0000h	8.2.14/362
1_0E08	CPC error injection control register (CPC1_CPCERRINJCTL)	32	R/W	0000_0000h	8.2.15/363
1_0E20	CPC capture data high register (CPC1_CPCCAPTDATAHI)	32	R	0000_0000h	8.2.16/364
1_0E24	CPC capture data low register (CPC1_CPCCAPTDATALO)	32	R	0000_0000h	8.2.17/364
1_0E28	CPC capture ECC register (CPC1_CPCCAPTECC)	32	R	0000_0000h	8.2.18/364
1_0E40	CPC error detect register (CPC1_CPCERRDET)	32	w1c	0000_0000h	8.2.19/366
1_0E44	CPC error disable register (CPC1_CPCERRDIS)	32	R/W	0000_0000h	8.2.20/368
1_0E48	CPC error interrupt enable register (CPC1_CPCERRINTEN)	32	R/W	0000_0000h	8.2.21/369
1_0E50	CPC error extended address register (CPC1_CPCERREADR)	32	R/W	0000_0000h	8.2.22/370
1_0E54	CPC error address register (CPC1_CPCERRADDR)	32	R/W	0000_0000h	8.2.23/370
1_0E58	CPC error control register (CPC1_CPCERRCTL)	32	R/W	0000_0000h	8.2.24/371
1_0F00	CPC hardware debug control register 0 (CPC1_CPCHDBCR0)	32	R/W	001E_0000h	8.2.25/371
1_1000	CPC configuration and status register 0 (CPC2_CPCCSR0)	32	R/W	0000_0000h	8.2.1/349
1_1008	CPC configuration register 0 (CPC2_CPCCFG0)	32	R	50B7_C010h	8.2.2/352
1_1010	CPC external write control register n (CPC2_CPCEWCR0)	32	R/W	0000_0000h	8.2.3/354
1_1014	CPC external write base address register n (CPC2_CPCEWBAR0)	32	R/W	0000_0000h	8.2.4/355
1_1020	CPC external write control register n (CPC2_CPCEWCR1)	32	R/W	0000_0000h	8.2.3/354
1_1024	CPC external write base address register n (CPC2_CPCEWBAR1)	32	R/W	0000_0000h	8.2.4/355
1_1100	CPC SRAM control register 1 (CPC2_CPCSRCR1)	32	R/W	0000_0000h	8.2.5/356
1_1104	CPC SRAM control register 0 (CPC2_CPCSRCR0)	32	R/W	0000_0000h	8.2.6/356
1_1200	CPC partition ID register 0 (CPC2_CPCPIR0)	32	R/W	FFFF_FFFFh	8.2.7/357
1_1208	CPC partition allocation register 0 (CPC2_CPCPAR0)	32	R/W	FFFF_FBFFh	8.2.8/358
1_120C	CPC partition way register 0 (CPC2_CPCPWR0)	32	R/W	FFFF_FFFFh	8.2.9/359
1_1210	CPC partition ID register n (CPC2_CPCPIR1)	32	R/W	0000_0000h	8.2.10/360
1_1218	CPC partition allocation register n (CPC2_CPCPAR1)	32	R/W	0000_0000h	8.2.11/360
1_121C	CPC partition way register 0 (CPC2_CPCPWR1)	32	R/W	0000_0000h	8.2.12/362
1_1220	CPC partition ID register n (CPC2_CPCPIR2)	32	R/W	0000_0000h	8.2.10/360
1_1228	CPC partition allocation register n (CPC2_CPCPAR2)	32	R/W	0000_0000h	8.2.11/360
1_122C	CPC partition way register 0 (CPC2_CPCPWR2)	32	R/W	0000_0000h	8.2.12/362
1_1230	CPC partition ID register n (CPC2_CPCPIR3)	32	R/W	0000_0000h	8.2.10/360
1_1238	CPC partition allocation register n (CPC2_CPCPAR3)	32	R/W	0000_0000h	8.2.11/360

Table continues on the next page...

CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_123C	CPC partition way register 0 (CPC2_CPCPWR3)	32	R/W	0000_0000h	8.2.12/362
1_1240	CPC partition ID register n (CPC2_CPCPIR4)	32	R/W	0000_0000h	8.2.10/360
1_1248	CPC partition allocation register n (CPC2_CPCPAR4)	32	R/W	0000_0000h	8.2.11/360
1_124C	CPC partition way register 0 (CPC2_CPCPWR4)	32	R/W	0000_0000h	8.2.12/362
1_1250	CPC partition ID register n (CPC2_CPCPIR5)	32	R/W	0000_0000h	8.2.10/360
1_1258	CPC partition allocation register n (CPC2_CPCPAR5)	32	R/W	0000_0000h	8.2.11/360
1_125C	CPC partition way register 0 (CPC2_CPCPWR5)	32	R/W	0000_0000h	8.2.12/362
1_1260	CPC partition ID register n (CPC2_CPCPIR6)	32	R/W	0000_0000h	8.2.10/360
1_1268	CPC partition allocation register n (CPC2_CPCPAR6)	32	R/W	0000_0000h	8.2.11/360
1_126C	CPC partition way register 0 (CPC2_CPCPWR6)	32	R/W	0000_0000h	8.2.12/362
1_1270	CPC partition ID register n (CPC2_CPCPIR7)	32	R/W	0000_0000h	8.2.10/360
1_1278	CPC partition allocation register n (CPC2_CPCPAR7)	32	R/W	0000_0000h	8.2.11/360
1_127C	CPC partition way register 0 (CPC2_CPCPWR7)	32	R/W	0000_0000h	8.2.12/362
1_1280	CPC partition ID register n (CPC2_CPCPIR8)	32	R/W	0000_0000h	8.2.10/360
1_1288	CPC partition allocation register n (CPC2_CPCPAR8)	32	R/W	0000_0000h	8.2.11/360
1_128C	CPC partition way register 0 (CPC2_CPCPWR8)	32	R/W	0000_0000h	8.2.12/362
1_1290	CPC partition ID register n (CPC2_CPCPIR9)	32	R/W	0000_0000h	8.2.10/360
1_1298	CPC partition allocation register n (CPC2_CPCPAR9)	32	R/W	0000_0000h	8.2.11/360
1_129C	CPC partition way register 0 (CPC2_CPCPWR9)	32	R/W	0000_0000h	8.2.12/362
1_12A0	CPC partition ID register n (CPC2_CPCPIR10)	32	R/W	0000_0000h	8.2.10/360
1_12A8	CPC partition allocation register n (CPC2_CPCPAR10)	32	R/W	0000_0000h	8.2.11/360
1_12AC	CPC partition way register 0 (CPC2_CPCPWR10)	32	R/W	0000_0000h	8.2.12/362
1_12B0	CPC partition ID register n (CPC2_CPCPIR11)	32	R/W	0000_0000h	8.2.10/360
1_12B8	CPC partition allocation register n (CPC2_CPCPAR11)	32	R/W	0000_0000h	8.2.11/360
1_12BC	CPC partition way register 0 (CPC2_CPCPWR11)	32	R/W	0000_0000h	8.2.12/362
1_12C0	CPC partition ID register n (CPC2_CPCPIR12)	32	R/W	0000_0000h	8.2.10/360
1_12C8	CPC partition allocation register n (CPC2_CPCPAR12)	32	R/W	0000_0000h	8.2.11/360
1_12CC	CPC partition way register 0 (CPC2_CPCPWR12)	32	R/W	0000_0000h	8.2.12/362
1_12D0	CPC partition ID register n (CPC2_CPCPIR13)	32	R/W	0000_0000h	8.2.10/360
1_12D8	CPC partition allocation register n (CPC2_CPCPAR13)	32	R/W	0000_0000h	8.2.11/360
1_12DC	CPC partition way register 0 (CPC2_CPCPWR13)	32	R/W	0000_0000h	8.2.12/362
1_12E0	CPC partition ID register n (CPC2_CPCPIR14)	32	R/W	0000_0000h	8.2.10/360
1_12E8	CPC partition allocation register n (CPC2_CPCPAR14)	32	R/W	0000_0000h	8.2.11/360
1_12EC	CPC partition way register 0 (CPC2_CPCPWR14)	32	R/W	0000_0000h	8.2.12/362
1_12F0	CPC partition ID register n (CPC2_CPCPIR15)	32	R/W	0000_0000h	8.2.10/360
1_12F8	CPC partition allocation register n (CPC2_CPCPAR15)	32	R/W	0000_0000h	8.2.11/360
1_12FC	CPC partition way register 0 (CPC2_CPCPWR15)	32	R/W	0000_0000h	8.2.12/362
1_1E00	CPC error injection high register (CPC2_CPCERRINJHI)	32	R/W	0000_0000h	8.2.13/362

Table continues on the next page...

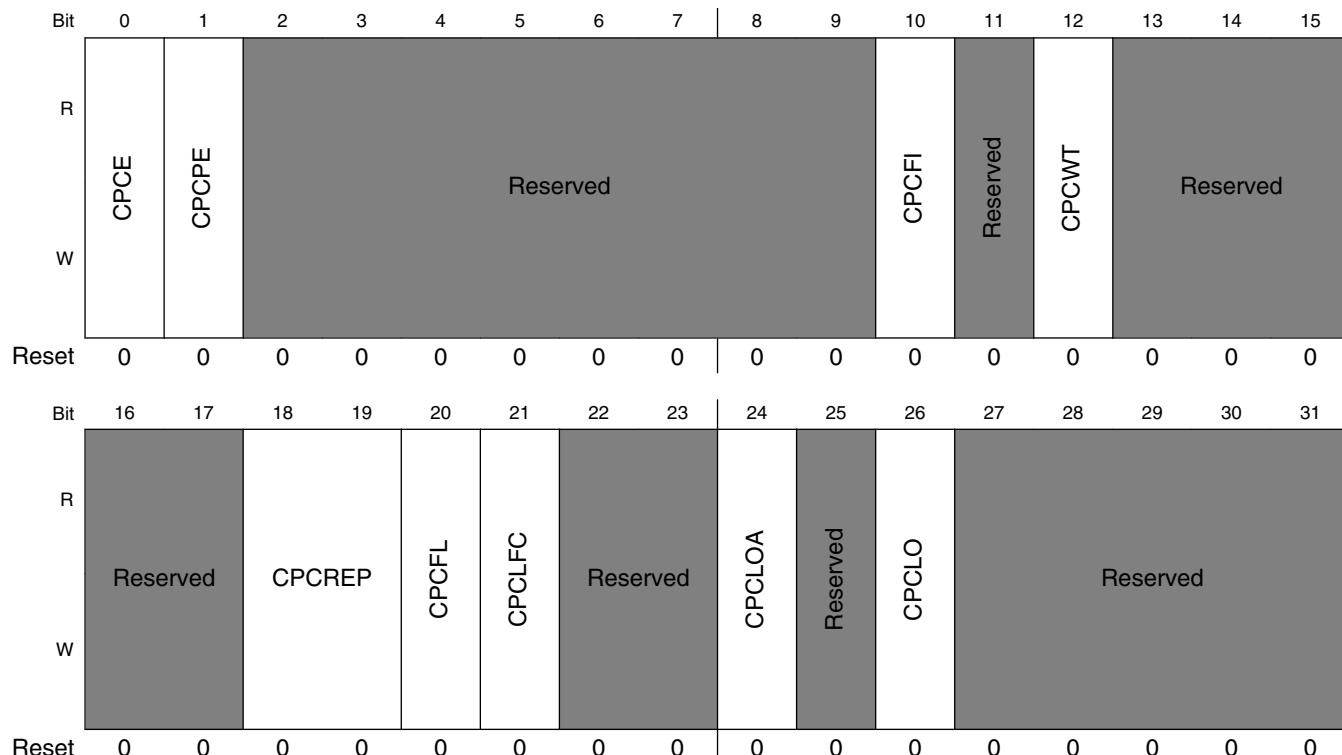
CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_1E04	CPC error injection low register (CPC2_CPCERRINJLO)	32	R/W	0000_0000h	8.2.14/362
1_1E08	CPC error injection control register (CPC2_CPCERRINJCTL)	32	R/W	0000_0000h	8.2.15/363
1_1E20	CPC capture data high register (CPC2_CPCCAPTDATAHI)	32	R	0000_0000h	8.2.16/364
1_1E24	CPC capture data low register (CPC2_CPCCAPTDATALO)	32	R	0000_0000h	8.2.17/364
1_1E28	CPC capture ECC register (CPC2_CPCCAPTECC)	32	R	0000_0000h	8.2.18/364
1_1E40	CPC error detect register (CPC2_CPCERRDET)	32	w1c	0000_0000h	8.2.19/366
1_1E44	CPC error disable register (CPC2_CPCERRDIS)	32	R/W	0000_0000h	8.2.20/368
1_1E48	CPC error interrupt enable register (CPC2_CPCERRINTEN)	32	R/W	0000_0000h	8.2.21/369
1_1E50	CPC error extended address register (CPC2_CPCERREADR)	32	R/W	0000_0000h	8.2.22/370
1_1E54	CPC error address register (CPC2_CPCERRADDR)	32	R/W	0000_0000h	8.2.23/370
1_1E58	CPC error control register (CPC2_CPCERRCTL)	32	R/W	0000_0000h	8.2.24/371
1_1F00	CPC hardware debug control register 0 (CPC2_CPHDBCR0)	32	R/W	001E_0000h	8.2.25/371

8.2.1 CPC configuration and status register 0 (CPCx_CPCCSR0)

The CPC configuration and status register (CPCCSR0) controls configuration and operation of the CPC array. The sequence for modifying CPCCSR0 can be found at [Modifying CPC Control and Status Registers](#).

Address: Base address + 0h offset



CPCx_CPCCSR0 field descriptions

Field	Description
0 CPCE	CPC enable. Used to enable the CPC array (cache or memory-mapped SRAM). 0 CPC is disabled. All transactions will behave as if they miss in the cache. 1 CPC is enabled
1 CPCPE	CPC parity/ECC error checking enable. Used to enable error checking in the CPC arrays. This bit must be set for normal operation. NOTE: The value of CPCPE should not be changed after the CPC is enabled. CPCPE bit should only be set or cleared simultaneously with setting or clearing CPCE. 0 Error checking disabled 1 Error checking enabled
2–9 -	This field is reserved. Reserved

Table continues on the next page...

CPCx_CPCCSR0 field descriptions (continued)

Field	Description
10 CPCFI	<p>CPC flash invalidate. Invalidation occurs regardless whether the cache is enabled or not. .</p> <p>NOTE: Attempting to set CPCFI during an invalidation operation causes undefined results; attempting to clear CPCFI during an invalidation operation is ignored.</p> <p>NOTE: Note: If CPCFI and CPCLFC are set simultaneously with the same register write operation, then the CPC flash invalidate and CPC lock flash clear functions are performed in parallel. In this case, hardware clears both the valid and lock bits, as well as all the tag and LRU bits, with a single pass through all indices, writing each entry in the CPC tag, status and victim arrays. In addition, all data arrays are cleared by writing 0s to all locations.</p> <p>NOTE: Before enabling the CPC through CPCE, the CPC tags should be cleared by setting both CPCFI and CPCLFC. This insures that all entries in the cache are invalidated and unlocked.</p> <ul style="list-style-type: none"> 0 Normal operation. No cache invalidation. 1 Cache flash invalidate. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. All the lines in the CPC that are designated as cache (that is, not SRAM) will be invalidated.
11 -	This field is reserved. Reserved
12 CPCWT	<p>CPC write-through mode.</p> <p>NOTE: This bit must not be changed once the CPC has been enabled through the CPCE bit. The value of CPCWT must be set at the same time as CPCE.</p> <ul style="list-style-type: none"> 0 CPC is operating in write-back mode. 1 CPC is operating in write-through mode.
13–17 -	This field is reserved. Reserved
18–19 CPCREP	<p>CPC line replacement algorithm.</p> <ul style="list-style-type: none"> 00 SPLRU (Streaming Pseudo Least Recently Used) with Aging. 01 FIFO (First In, First Out). 10 SPLRU (Streaming Pseudo Least Recently Used). 11 PLRU (Pseudo Least Recently Used).
20 CPCFL	<p>CPC flush. Setting this bit initiates a CPC flush operation causing all dirty lines to be written out to backing store (memory).</p> <p>NOTE: Writing a 1 while a flush operation is in progress is ignored. Writing a 0 during a flash clear operation is ignored. Writing a 1 to CPCFL is ignored if the CPC is not enabled (CPCE = 0).</p> <p>NOTE: To flush the CPC cache and ensure that no valid entries exist after the flush (that is, for the purposes of powering down or disabling) the following high level sequence of operations should be used: 1. Clear all bits in CPCPAR0-CPCPAR15 to prevent new transactions from allocating in the CPC 2. Set CPCCSR0[CPCFL] 3. Wait for CPCCSR0[CPCFL] to be cleared by hardware 4. Clear CPCCSR0[CPCE]</p> <ul style="list-style-type: none"> 0 A CPC flush is not being performed. 1 Hardware initiates a CPC flush operation. This bit is cleared when the operation is complete. All lines in the CPC tag array that are dirty will be written back to backing store.
21 CPCLFC	<p>CPC lock flash clear. Clearing of lock bits occurs regardless of the cache enable (CPCE) value.</p> <p>NOTE: Writing a 1 during a lock clear operation causes undefined results. Writing a 0 during a lock clear operation is ignored.</p>

Table continues on the next page...

CPCx_CPCCSR0 field descriptions (continued)

Field	Description
	<p>NOTE: If CPCFI and CPCLFC are set simultaneously with the same register write operation, then the CPC flash invalidate and CPC lock flash clear functions will be performed in parallel. In this case, hardware will clear both the valid and lock bits, as well as all the tag and LRU bits, with a single pass through all indices, writing each entry in the CPC tag, status and victim arrays. In addition, all data arrays are cleared by writing 0's to all locations.</p> <p>NOTE: Before enabling the CPC through CPCCSR0[CPCE], the CPC tags should be cleared by setting both CPCCSR0[CPCFI] and CPCCSR0[CPCLFC]. This insures that all entries in the cache are invalidated and unlocked</p> <ul style="list-style-type: none"> 0 No flash lock clear. . . 1 Cache flash lock clear operation. A cache flash lock clear operation is initiated by hardware. Once complete, this bit is cleared. All the lines in the CPC that are designated as cache (that is, not SRAM) will clear their lock bits.
22–23 -	This field is reserved. Reserved
24 CPCLOA	<p>CPC lock overflow allocate. Controls the behavior of a transaction that wishes to establish a coherency granule in the locked state.</p> <ul style="list-style-type: none"> 0 Indicates a lock overflow condition will not replace an existing locked granule with the requested granule. 1 Indicates a lock overflow condition will replace an existing locked granule with the requested granule using the selected replacement algorithm.
25 -	This field is reserved. Reserved
26 CPCLO	<p>CPC lock overflow. This sticky bit is set by hardware if a cache lock overflow situation is detected. Overflow conditions occur as the result of a transaction that wishes to establish a coherency granule in the cache in a locked state, but all available ways are already locked and as a result an existing locked coherency granule is replaced.</p> <ul style="list-style-type: none"> 0 The cache has not encountered a lock overflow condition. 1 The cache has encountered a lock overflow condition.
27–31 -	This field is reserved. Reserved

8.2.2 CPC configuration register 0 (CPCCFG0)

The CPC configuration register 0 (CPCCFG0) controls the initial configuration of the CPC.

Address: Base address + 8h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	CPCTEHA	CPCDEHA		Reserved	CPCBSIZE	CPCREPL	CPCLA					CPCLAWAY			
W													Reserved			
Reset	0	1	0	1	0	0	0	0	1	0	1	1	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CPCNWAY								CPCSIZE							
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0

CPCx_CPCCFG0 field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–2 CPCTEHA	CPC tag array error handling available. 10 Single-bit ECC correction, double-bit ECC detection available
3–4 CPCDEHA	CPC data array error handling available. 10 Single-bit ECC correction, double-bit ECC detection available
5 -	This field is reserved. Reserved
6–8 CPCBSIZE	Coherency granule size. 001 64 bytes
9–10 CPCREPL	Cache default replacement policy. 01 Pseudo LRU
11 CPCLA	Cache Line Locking APU available. The CPC responds to a CT field equal to 1. 1 Available
12 -	This field is reserved. Reserved
13–17 CPCNWAY	Cache number of ways. The actual value is the number of ways - 1. 11111 32 way set associative
18–31 CPCSIZE	Cache size in 64 Kbyte increments. A value of 1 denotes a 64-Kbyte cache 00_0000_0001_0000 1 Mbyte

8.2.3 CPC external write control register n (CPCx_CPCEWCRn)

Address: Base address + 10h offset + (16d × i), where i=0d to 1d

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

CPCx_CPCEWCRn field descriptions

Field	Description
0 E	Enable
1 LOCK	Lock
2–9 -	This field is reserved. Reserved
10–15 SIZE	Size of the address range used for implicit stashing. This field is used to generate an address mask that is applied to both the transaction address and the address base formed from EADDRBASE and ADDRBASE before comparing the two. The size of the address range is equal to 2^{SIZE+6} . 000000 64 bytes 000001 128 bytes 000010 256 bytes 000011 512 bytes 000100 1 Kbytes 000101 2 Kbytes 000110 4 Kbytes 000111 8 Kbytes 001000 16 Kbytes 001001 32 Kbytes 001010 64 Kbytes 001011 128 Kbytes 001100 256 Kbytes 001101 512 Kbytes 001110 1 Mbytes 001111 2 Mbytes 010000 4 Mbytes 010001 8 Mbytes

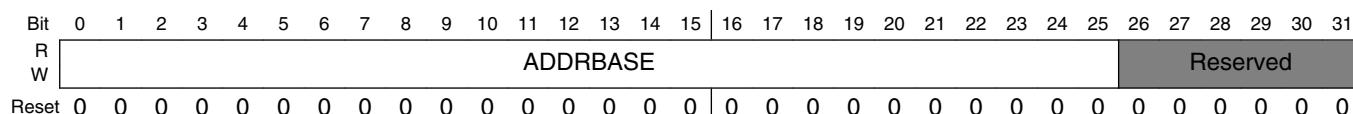
Table continues on the next page...

CPCx_CPCEWCRn field descriptions (continued)

Field	Description
	010010 16 Mbytes 010011 32 Mbytes 010100 64 Mbytes 010101 128 Mbytes 010110 256 Mbytes 010111 512 Mbytes 011000 1 Gbytes 011001 2 Gbytes 011010 4 Gbytes 011011 8 Gbytes 011100 16 Gbytes 011101 32 Gbytes 011110 64 Gbytes 011111 128 Gbytes
16–27 -	This field is reserved. Reserved
28–31 EADDRBASE	Extended base address corresponding to physical address 0:3.

**8.2.4 CPC external write base address register n
(CPCx_CPCEWBArn)**

Address: Base address + 14h offset + (16d × i), where i=0d to 1d

**CPCx_CPCEWBArn field descriptions**

Field	Description
0–25 ADDRBASE	Base address corresponding to physical address 4:29.
26–31 -	This field is reserved. Reserved

8.2.5 CPC SRAM control register 1 (CPCx_CPCSRCR1)

The CPC SRAM control register 1(CPCSRCR1) determines the upper base address when the CPC is used as SRAM.

Address: Base address + 100h offset

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

CPCx_CPCSRCR1 field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 SRBARU	SRAM base address high. Corresponds to physical address bits 0:3.

8.2.6 CPC SRAM control register 0 (CPCx_CPCSRCR0)

The CPC SRAM control register 0 (CPCSRCR0) configures the CPC when it is used as SRAM.

Address: Base address + 104h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									SRBARL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRBA RL			Reserved					INTLVE N	Reserve d	SRAMSZ					SRA MEN
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPCx_CPCSRCR0 field descriptions

Field	Description
0–16 SRBTRL	SRAM base address low. Corresponds to physical address bits 4:20. The base address should be aligned to the size specified in SRAMSZ.
17–22 -	This field is reserved. Reserved
23 INTLVEN	SRAM interleaving enable.
	NOTE: SRAM address interleaving is only supported when the memory controllers are interleaved on cache-line or page boundaries.
	0 SRAM address interleaving is disabled
	1 SRAM address interleaving is enabled. In this case, memory controller interleaving must also be enabled in the DDR controller registers. SRAM address space is interleaved at the same address boundary as the DDR controllers. These parameters are controlled by the DDR controller configuration registers.
24–25 -	This field is reserved. Reserved
26–30 SRAMSZ	SRAM size. This determines the size of the SRAM space .
	All other settings not shown are reserved.
	00001 64 Kbytes,uses ways 30-31 for SRAM
	00011 256 Kbytes,uses ways 24-31 for SRAM
	00101 1 Mbyte (1024 Kbytes) uses ways 0-31 for SRAM
31 SRAMEN	SRAM mode enable

8.2.7 CPC partition ID register 0 (CPCx_CPCPIR0)

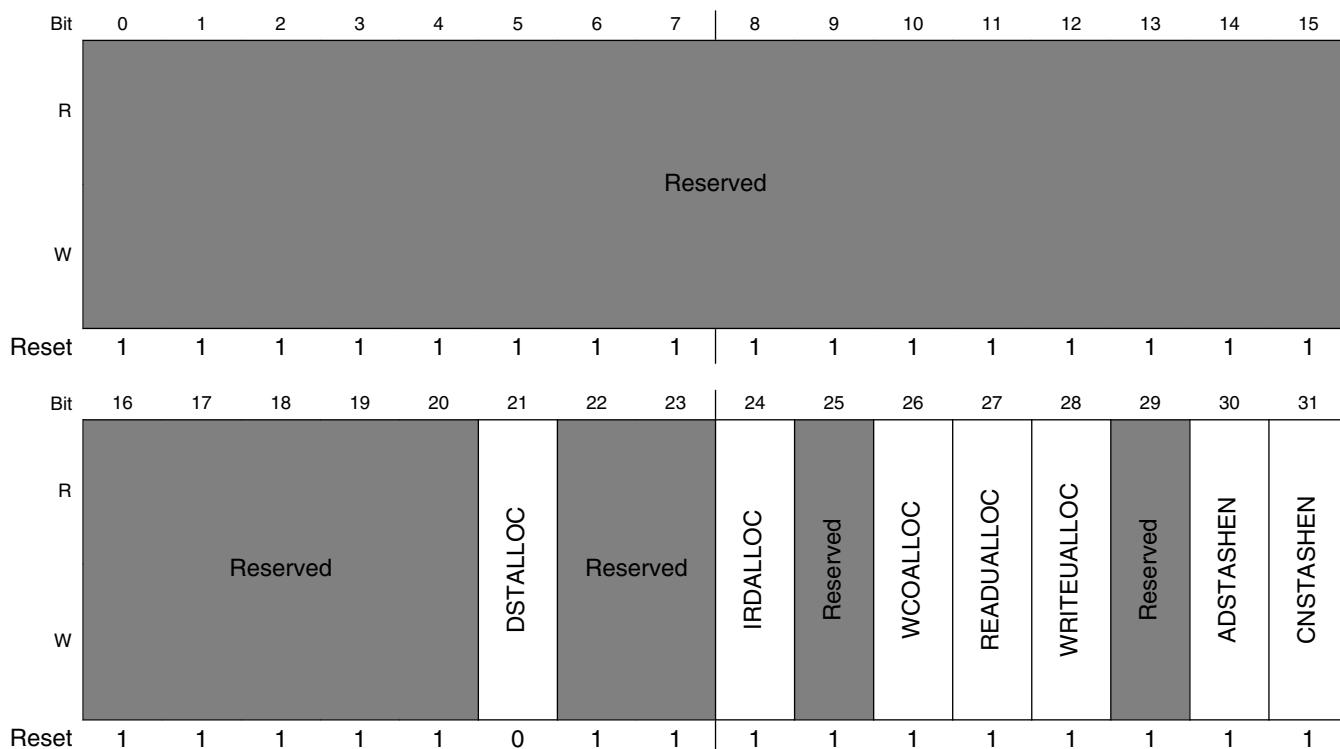
Address: Base address + 200h offset

CPCx CPCPIR0 field descriptions

Field	Description
0-31 PID	<p>Partition ID</p> <p>A 1 in a particular bit position indicates that the particular partition ID can match on this entry.</p> <p>For example, a PID of 0xF001_0000 indicates that transactions with partition IDs of 0, 1, 2, 3 and 15 can match on this table entry. Partition ID equates to the CSD_ID assigned by the LAWs. See Local Access Window (LAW) Memory Map.</p>

8.2.8 CPC partition allocation register 0 (CPCx_CPCPAR0)

Address: Base address + 208h offset



CPCx_CPCPAR0 field descriptions

Field	Description
0–20 -	This field is reserved. Reserved
21 DSTALLOC	Data store allocation control 0 Transactions resulting from a Store operation that missed in a core cache will not allocate. 1 Transactions resulting from a Store operation that missed in a core cache will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
22–23 -	This field is reserved. Reserved
24 IRDALLOC	Instruction read allocation control 0 Instruction read transactions that miss in the CPC will not allocate. 1 Instruction read transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
25 -	This field is reserved. Reserved
26 WCOALLOC	Writes carrying cast-out data from core caches transaction allocation control NOTE: Setting this bit and clearing IRDALLOC and DRDALLOC will effectively make the CPC act like a victim cache for CoreNet masters that contain write-back caches.

Table continues on the next page...

CPCx_CPCPAR0 field descriptions (continued)

Field	Description
	0 Writes carrying cast-out data from core caches transactions that miss in the CPC will not allocate. 1 Writes carrying cast-out data from core caches transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
27 READUALLOC	Decorated Load operations allocation control. See Decorated Storage 0 Decorated Load operations that miss in the CPC will not allocate. 1 Decorated Load operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
28 WRITEUALLOC	Decorated Store operations allocation control. See Decorated Storage 0 Decorated Store operations that miss in the CPC will not allocate. 1 Decorated Store operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
29 -	This field is reserved. Reserved
30 ADSTASHEN	Address based stashing enabled 0 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR n and CPCEWABR n will not allocate. 1 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR n and CPCEWABR n will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
31 CNSTASHEN	Explicitly signalled stashing enabled 0 Transactions that miss in the CPC and request stashing will not allocate. 1 Transactions that miss in the CPC and request stashing will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .

8.2.9 CPC partition way register 0 (CPCx_CPCPWR0)

Address: Base address + 20Ch offset

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

Reset 1

CPCx_CPCPWR0 field descriptions

Field	Description
0–31 WAY	Partition way(s) A 1 in a bit position indicates that transactions that match the PID defined in the corresponding CPCPIR n register can allocate the indicated way. NOTE: This field in no way prevents a transaction to any partition ID from hitting to a specified way and either reading or writing data.

8.2.10 CPC partition ID register n (CPCx_CPCPIRn)

Address: Base address + 210h offset + (16d × i), where i=0d to 14d

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

Reset 0

CPCx_CPCPIRn field descriptions

Field	Description
0–31 PID	<p>Partition ID</p> <p>A 1 in a particular bit position indicates that the particular partition ID can match on this entry.</p> <p>For example, a PID of 0xF001_0000 indicates that transactions with partition IDs of 0, 1, 2, 3 and 15 can match on this table entry. Partition ID equates to the CSD_ID assigned by the LAWs. See Local Access Window (LAW) Memory Map.</p>

8.2.11 CPC partition allocation register n (CPCx_CPCPARn)

Address: Base address + 218h offset + (16d × i), where i=0d to 14d

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

Reserved

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						
R						DSTALLOC		Reserved			IRDALLOC		WCOALLOC		READUALLOC		WRITEUALLOC		ADSTASHEN		CNSTASHEN	
W																						

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Reserved

DSTALLOC

IRDALLOC

WCOALLOC

READUALLOC

WRITEUALLOC

ADSTASHEN

CNSTASHEN

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

CPCx_CPCPARn field descriptions

Field	Description
0–20 -	This field is reserved. Reserved
21 DSTALLOC	Data store allocation control 0 Transactions resulting from a Store operation that missed in a core cache will not allocate. 1 Transactions resulting from a Store operation that missed in a core cache will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
22–23 -	This field is reserved. Reserved
24 IRDALLOC	Instruction read allocation control 0 Instruction read transactions that miss in the CPC will not allocate. 1 Instruction read transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
25 -	This field is reserved. Reserved
26 WCOALLOC	Writes carrying cast-out data from core caches transaction allocation control NOTE: Setting this bit and clearing IRDALLOC and DRDALLOC will effectively make the CPC act like a victim cache for CoreNet masters that contain write-back caches. 0 Writes carrying cast-out data from core caches transactions that miss in the CPC will not allocate. 1 Writes carrying cast-out data from core caches transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
27 READUALLOC	Decorated Load operations transaction allocation control. See Decorated Storage 0 Decorated Load operations transactions that miss in the CPC will not allocate. 1 Decorated Load operations transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
28 WRITEUALLOC	Decorated Store operations transaction allocation control. See Decorated Storage 0 Decorated Store operations that miss in the CPC will not allocate. 1 Decorated Store operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
29 -	This field is reserved. Reserved
30 ADSTASHEN	Address based stashing enabled 0 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR n and CPCEWABR n will not allocate. 1 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR n and CPCEWABR n will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .
31 CNSTASHEN	Explicitly signalled stashing enabled 0 Transactions that miss in the CPC and request stashing will not allocate. 1 Transactions that miss in the CPC and request stashing will attempt to allocate in one of the ways defined by the corresponding CPCPWR n .

8.2.12 CPC partition way register 0 (CPCx_CPCPWRn)

Address: Base address + 21Ch offset + (16d × i), where i=0d to 14d

CPCx_CPCPWRn field descriptions

Field	Description
0-31 WAY	<p>Partition way(s)</p> <p>A 1 in a bit position indicates that transactions that match the PID defined in the corresponding CPCPIR n register can allocate the indicated way.</p> <p>NOTE: This field in no way prevents a transaction to any partition ID from hitting to a specified way and either reading or writing data.</p>

8.2.13 CPC error injection high register (CPCx_CPCERRINJHI)

Address: Base address + E00h offset

CPCx CPCERRINJHI field descriptions

Field	Description
0-31 EIMASKHI	Error injection mask for high word. When a bit is set and CPCERRINJCTL[DERRIEN] = 1, the corresponding bit in the data path is inverted during data array writes.

8.2.14 CPC error injection low register (CPCx_CPCERRINJLO)

Address: Base address + E04h offset

CPCx_CPCERRINJLO field descriptions

Field	Description
0-31 EIMASKLO	Error injection mask for low word. When a bit is set and CPCERRINJCTL[DERRIEN] = 1, the corresponding bit in the data path is inverted during data array writes.

8.2.15 CPC error injection control register (CPCx_CPCERRINJCTL)

Address: Base address + E08h offset

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

CPCx_CPCERRINJCTL field descriptions

Field	Description
0–13 -	This field is reserved. Reserved
14 SERRIEN	Status error injection enable. Status error injection is determined by CPCERRINJLO[23:31]; status ECC error injection is determined by the five least-significant bits of ECCERRIM (that is, CPCERRINJCTL[27:31]). 0 No status errors are injected 1 Subsequent entries written to the CPC status arrays have status or ECC bits inverted as specified in the status and ECC error injection masks.
15 TERRIEN	Tag error injection enable. Tag error injection is determined by CPCERRINJHI[31] and CPCERRINJLO[11:31]; tag ECC error injection is determined by the seven least-significant bits of ECCERRIM (that is, CPCERRINJCTL[25:31]). 0 No tag errors are injected 1 Subsequent entries written to the CPC tag arrays have tag or ECC bits inverted as specified in the data and ECC error injection masks.
16–22 -	This field is reserved. Reserved
23 DERRIEN	CPC data array error injection enable. Data error injection is determined by CPCERRINJHI[0:31] and CPCERRINJLO[0:31]; data ECC error injection is determined by ECCERRIM. 0 No data errors are injected 1 Subsequent entries written to the CPC data arrays have data or ECC bits inverted as specified in the data and ECC error injection masks.
24–31 ECCERRIM	Error injection mask for the ECC bits. When DERRIEN=1, the eight ECCERRIM bits map to the eight ECC bits for each 64 bits of data. When TERRIEN=1, the low order seven ECCERRIM bits map to the seven ECC bits for each 33-bit tag. When SERRIEN=1, the low order five ECCERRIM bits map to the five ECC bits for each 9-bit tag.

8.2.16 CPC capture data high register (CPCx_CPCCAPTDATAHI)

Address: Base address + E20h offset

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

CPCx_CPCCAPTDATAHI field descriptions

Field	Description
0-31 DATAH	If error is data ECC, then bits 0-31 are the upper order 32 bits of data associated with the error. If the error is tag ECC, then bits 0-15 are zeros and bits 16-31 are the upper order 16 bits of the tag associated with the error. If the error is status ECC, then this field is all zeros.

8.2.17 CPC capture data low register (CPCx_CPCCAPTDATALO)

Address: Base address + E24h offset

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

CPCx_CPCCAPTDATALO field descriptions

Field	Description
0-31 DATAL	If error is data ECC, the bits 0-31 are the lower order 32 bits of data associated with the error. If the error is tag ECC, then bits 0-16 are the lower order 17 bits of the tag associated with the error and bits 17-31 are zeros. If the error is status ECC, then bits 0-22 are zeros and bits 23-31 are the status bits associated with the error.

8.2.18 CPC capture ECC register (CPCx_CPCCAPTECC)

Address: Base address + E28h offset

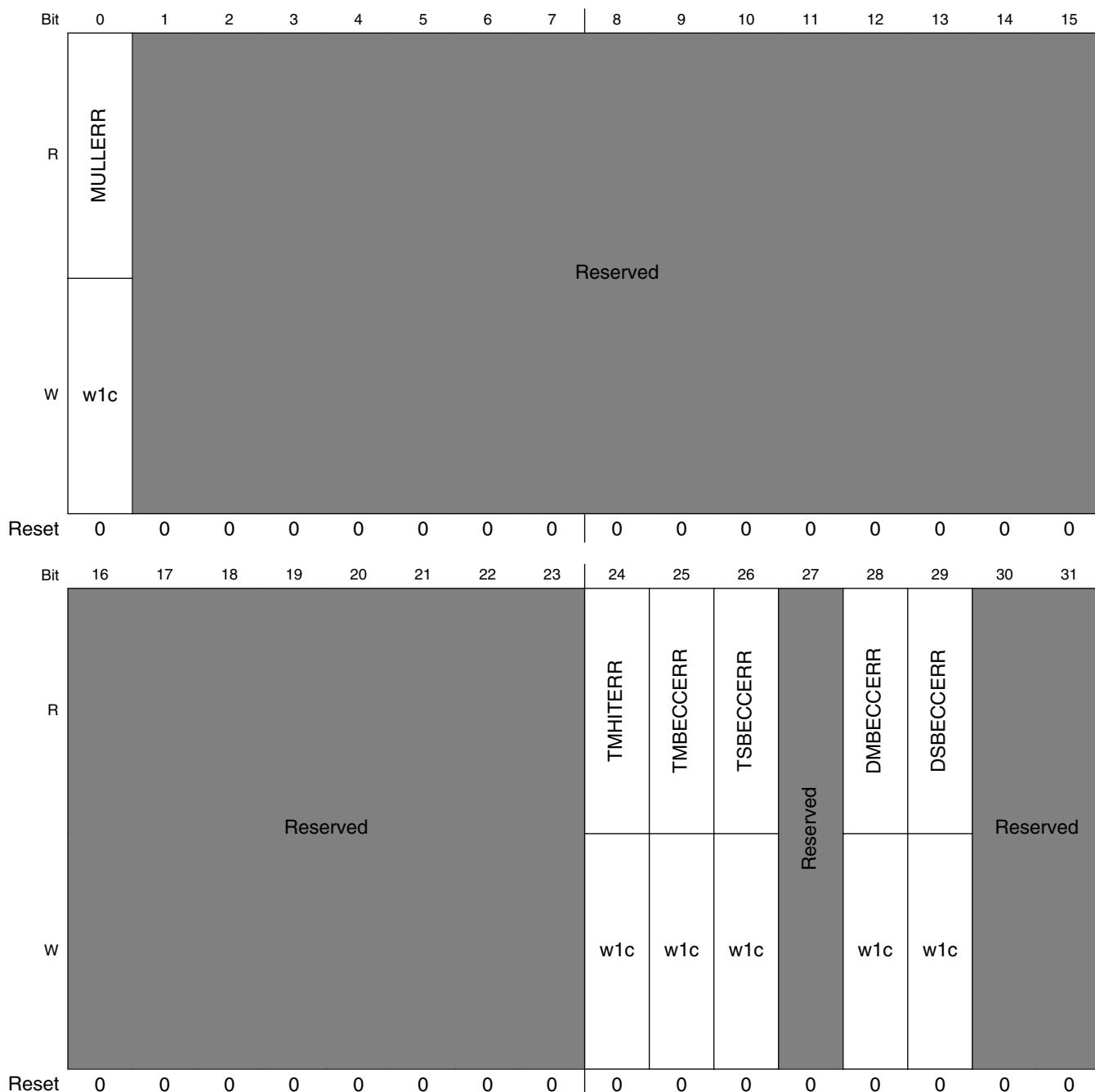
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ECCSYND																								ECCCHKSUM							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPCx_CPCCAPTECC field descriptions

Field	Description
0–7 ECCSYND	The calculated ECC syndrome of the captured error. Tag and status ECC are left padded with 1 or 3 zeros, respectively.
8–23 -	This field is reserved. Reserved
24–31 ECCCCHKSUM	The stored ECC syndrome of the captured error. Tag and status ECC are left padded with 1 or 3 zeros, respectively.

8.2.19 CPC error detect register (CPCx_CPCERRDET)

Address: Base address + E40h offset



CPCx_CPCERRDET field descriptions

Field	Description
0 MULLERR	Multiple CPC errors

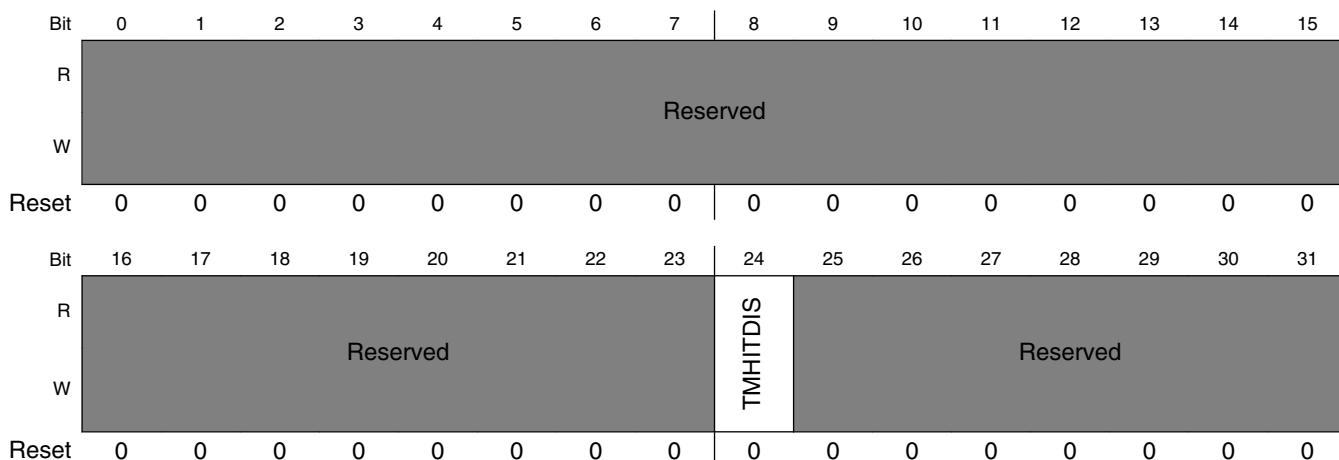
Table continues on the next page...

CPCx_CPCERRDET field descriptions (continued)

Field	Description
	0 Multiple CPC errors of the same type were not detected. 1 Multiple CPC errors to the same type were detected.
1–23 -	This field is reserved. Reserved
24 TMHITERR	Tag multi-way hit 0 Tag multi-way hit error not detected. 1 Tag multi-way hit error detected.
25 TMBECCERR	Tag or status multiple-bit ECC error. CPCERRATTR[Status_Tag] indicates whether the error occurred in the tag or status. 0 Tag/status multiple-bit ECC error not detected. 1 Tag/status multiple-bit ECC error detected.
26 TSBECCERR	Tag or status single-bit ECC error. CPCERRATTR[Status_Tag] indicates whether the error occurred in the tag or status. 0 Tag/status single-bit ECC error not detected. 1 Tag/status single-bit ECC error detected.
27 -	This field is reserved. Reserved
28 DMBECCERR	Data multiple-bit ECC error 0 Data multiple-bit ECC error not detected. 1 Data multiple-bit ECC error detected.
29 DSBECCERR	Data single-bit ECC error 0 Data single-bit ECC error not detected. 1 Data single-bit ECC error detected.
30–31 -	This field is reserved. Reserved

8.2.20 CPC error disable register (CPCx_CPCERRDIS)

Address: Base address + E44h offset

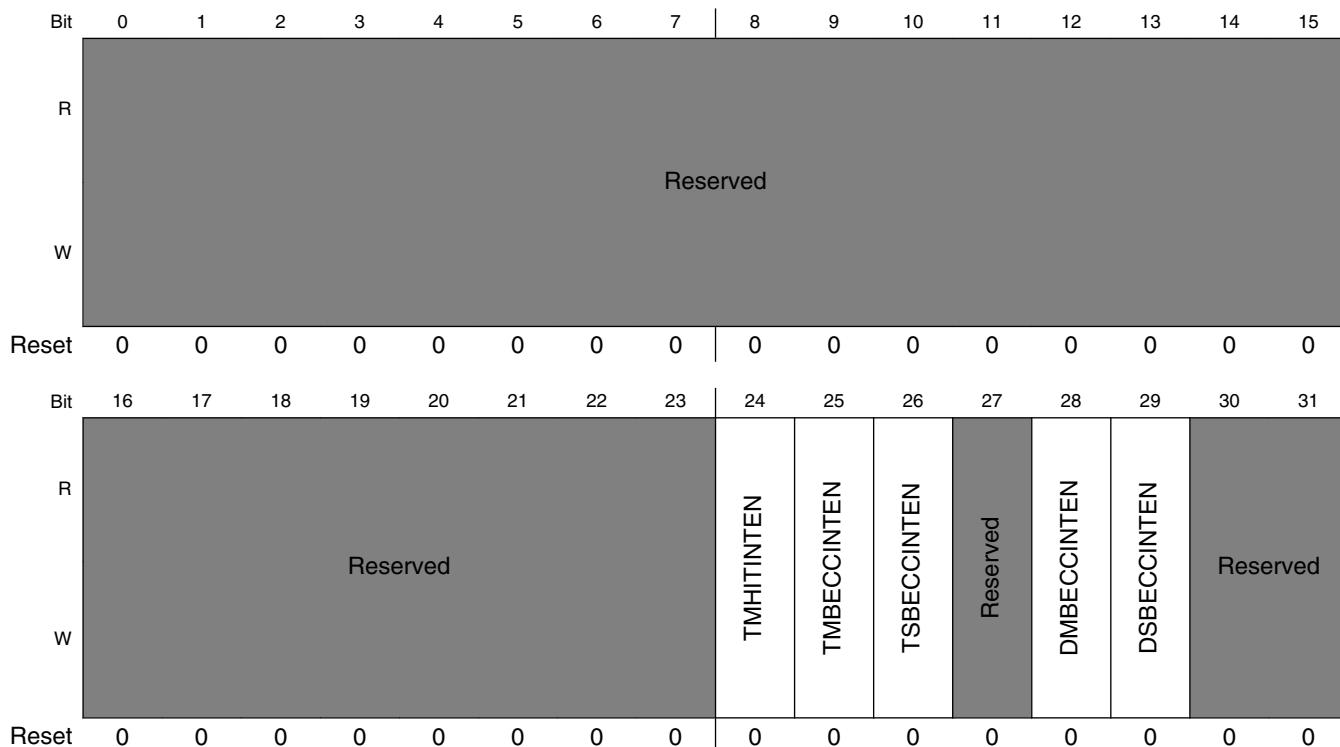


CPCx_CPCERRDIS field descriptions

Field	Description	
0–23 -	This field is reserved. Reserved	
24 TMHITDIS	Tag multi-way hit error disable. 0 Tag multi-way hit detection enabled. 1 Tag multi-way hit detection disabled.	
25–31 -	This field is reserved. Reserved	

8.2.21 CPC error interrupt enable register (CPCx_CPCERRINTEN)

Address: Base address + E48h offset



CPCx_CPCERRINTEN field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 TMHITINTEN	Tag multi-way hit error interrupt reporting enable. 0 Tag multi-way hit errors are not reported. 1 Tag multi-way hit errors are reported.
25 TMBECCINTEN	Tag multiple-bit ECC error interrupt reporting enable. 0 Tag multiple-bit ECC errors are not reported. 1 Tag multiple-bit ECC errors are reported.
26 TSBECCINTEN	Tag single-bit ECC error interrupt reporting enable. 0 Tag single-bit ECC errors are not reported. 1 Tag single-bit ECC errors are reported.
27 -	This field is reserved. Reserved
28 DMBECCINTEN	Data multiple-bit ECC error interrupt reporting enable. 0 Data multiple-bit ECC errors are not reported. 1 Data multiple-bit ECC errors are reported.

Table continues on the next page...

CPCx_CPCERRINTEN field descriptions (continued)

Field	Description
29 DSBECCINTEN	Data single-bit ECC error interrupt reporting enable. 0 Data single-bit ECC errors are not reported. 1 Data single-bit ECC errors are reported.
30–31 -	This field is reserved. Reserved

8.2.22 CPC error extended address register (CPCx_CPCERREADR)

Address: Base address + E50h offset

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

CPCx_CPCERREADR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 ADDRH	High-order 4 bits of the physical address (PA[0:3]) associated with a captured error.

8.2.23 CPC error address register (CPCx_CPCERRADDR)

Address: Base address + E54h offset

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

CPCx_CPCERRADDR field descriptions

Field	Description
0–31 ADDRL	Lower-order 32 bits of the physical address (PA[4:35]) associated with a captured error.

8.2.24 CPC error control register (CPCx_CPCERRCTL)

Address: Base address + E58h offset

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

CPCx_CPCERRCTL field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 CPCTHRESH	CPC ECC single-bit error threshold. Threshold value for the number of ECC single-bit errors that are detected before reporting an error condition. CPCTHRESH is compared to CPCTCOUNT and CPCCOUNT each time a single-bit ECC error is detected. A value of 0 in this field will cause the reporting of a single bit ECC error upon the first occurrence of such an error.
16–23 CPCTCOUNT	CPC tag ECC single-bit error count. Counts ECC single-bit errors detected in the CPC tags. If CPCTCOUNT equals the ECC single-bit error threshold (CPCTHRESH), an error is reported if single-bit error reporting for tags is enabled. Software should clear this value when such an error is reported to reset the count.
24–31 CPCDCOUNT	CPC data ECC single-bit error count. Counts ECC single-bit errors detected in the CPC data. If CPCCOUNT equals the ECC single-bit error threshold (CPCTHRESH), an error is reported if single-bit error reporting for data is enabled. Software should clear this value when such an error is reported to reset the count.

8.2.25 CPC hardware debug control register 0 (CPCx_CPCHDBCR0)

Address: Base address + F00h offset

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

CPCx_CPCHDBCR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 SPEC_DIS	Speculation disable 0 Read type requests speculatively pass to the memory controllers before determining whether there is a hit in the CPC. NOTE: This bit should be set to a 1 if the CPC SRAM is enabled and the SRAM address space does not overlap an enabled address region in the DRAM controller. 1 Read type requests must wait to determine hit in the CPC before making a memory controller request.
5–31 -	This field is reserved. Reserved

8.3 CPC Functional Description

8.3.1 Register Sections

8.3.1.1 SRAM Mode Registers

The CoreNet platform cache can be configured to use some of its ways as direct-mapped SRAM address space. The SRAM mode registers enable and configure the CPC to be used as SRAM space. SRAM mode is enabled by CPCSRCR0[SRAMEN], the size of the SRAM address space is configured by CPCSRCR0[SRAMSZ], and the base address of the SRAM address space is specified using CPCSRCR0[SRBTRL] and CPCSRCR1[SRBARU].

The CPC sends speculative requests (that is, before the CPC determines hit or miss) to the memory controller. If the CPC is configured as SRAM space, the memory controller may not respond to speculative SRAM requests that don't hit DRAM (or may generate an error). Therefore, when the CPC is configured as SRAM and the SRAM address space does not overlap an enabled memory controller address region, disable speculative requests by setting CPCHDBCR0[SPEC_DIS]. See [CPC hardware debug control register 0 \(CPC_CPCHDBCR0\)](#) for more information.

8.3.1.2 Partitioning Control Registers

The CPC supports a flexible way partition/allocation control scheme. Each transaction that misses in the cache looks in a table to determine whether or not to allocate and which ways are available for allocation. Table entries are matched by comparing Partition IDs. Allocation is then controlled dependent on transaction type and address qualifiers as well as stashing control signals.

Each partition control is composed of three registers: CPCPIR n , CPCPAR n and CPCPWR n . There are a total of 16 triples of these registers.

8.3.1.3 Error Registers

8.3.1.3.1 Error Detection and Reporting

Error detection for tags and data are enabled by default.

When an error is detected in the CPC the appropriate bit in the CPC error detect register (CPCERRDET) is set and, if enabled by CPCERRINTEN, a machine check interrupt is generated for software to notice and handle the error. The generation of the interrupt when an error is detected is controlled by the CPC error interrupt enable register (CPCERRINTEN). Bits in this register control whether an interrupt is to be generated based on the type and location of the error in the CPC. Software may read from the CPCERRDET and may clear bits by writing a 1 into a bit position, but cannot directly write the contents of the register. CPCERRDET is described in [CPC error detect register \(CPC_CPCERRDET\)](#); CPCERRINTEN is described in [CPC error interrupt enable register \(CPC_CPCERRINTEN\)](#).

Single-bit errors that are corrected by ECC are not reported unless the appropriate bit in the CPCERRDET register is set and the number of errors reach a threshold specified in CPCERRCTL, described in [CPC error control register \(CPC_CPCERRCTL\)](#). This prevents normal correctable ECC errors from generating machine check exceptions, but will still cause errors to be generated when there may be a persistent faulty condition in the CPC.

The machine check exception is generated based on the CPCERRDET status bits and CPCERRINTEN enable bits. If an CPCERRDET status bit and its corresponding CPCERRINTEN enable bit are set at the same time, a machine check exception is generated. Therefore, upon taking the machine check, software should clear CPCERRDET before setting MSR[ME], otherwise another machine check can occur when MSR[ME] is set.

8.3.1.3.2 Error Capture

When an error is detected and reported, information about the error is posted into several CPC error capture registers (CPCERRADDR, CPCERREADDR, CPCCAPTDATAHI, CPCCAPTDATALO, and CPCCAPTECC). Only the first reported error information is saved. The error capture registers are as follows:

- CPCERRADDR and CPCERREADDR contain the physical address associated with the captured error. CPCERRADDR and CPCERREADDR are described in [CPC error extended address register \(CPC_CPCERREADDR\)](#) and [CPC error address register \(CPC_CPCERRADDR\)](#).
- CPCCAPTDATAHI and CPCCAPTDATALO contain data or tag value associated with the captured error and are described in [CPC error injection high register \(CPC_CPCERRINJHI\)](#) and [CPC error injection low register \(CPC_CPCERRINJLO\)](#).
- CPCCAPTECC contains ECC information from the data or tag (the syndrome and the checksum) and is described in [CPC capture ECC register \(CPC_CPCCAPTECC\)](#).

8.3.1.3.3 Error Injection

The CPC includes support for injecting errors into the data, data ECC, tag, tag ECC, status and status ECC. This may be used to test error recovery software by deterministically creating error scenarios. The error injection registers are as follows:

- Data or tag errors are injected into the line, according to the error injection settings in CPCERRINJHI, CPCERRINJLO, and CPCERRINJCTL, at allocation. Note that error injection enable bits in CPCERRINJCTL must be cleared by software and the CPC must be invalidated (by setting CPCCSR0[CPCFI]) before resuming CPC normal operation.
- Data, tag and status use the same error injection masks as defined by CPCERRINJHI, CPCERRINJLO and CPCERRINJCTL, but have separate error injection enables as defined in CPCERRINJCTL.
- CPCERRINJCTL, CPCERRINJLO, and CPCERRINJHI are described in [CPC error injection control register \(CPC_CPCERRINJCTL\)](#), [CPC error injection low register \(CPC_CPCERRINJLO\)](#), and [CPC error injection high register \(CPC_CPCERRINJHI\)](#).

8.3.2 Line Locking

The CoreNet platform cache supports persistent cache locking as defined by Freescale Book E Implementation Standards for Storage (EIS Storage). A single lock bit is associated with each coherency granule stored in the cache.

The CPC supports explicit lock and unlock commands carried through transactions. Processor master devices can use **dcbtls**, **dcbtstls**, **icbtls**, **dcblc**, **icblc** instructions to generate LoadEC and Unlock transactions that target the CPC through the respective instructions' CT field. I/O master devices can use a wider variety of transactions that support explicit stashing transaction attributes.

The CPC also supports implicit locking through specified address ranges as defined by the Stashing Control Registers. See [CPC external write control register n \(CPC_CPCEWCRn\)](#).

8.3.3 Cache Operation Instructions and Transactions

Even though the CoreNet platform cache is not directly tied to a processor, it does respect the traditional cache operation instructions through their CoreNet transaction. **dcbf** instructions cause Flush transactions on CoreNet and the CPC performs the appropriate action in order to flush any dirty data to memory. **dcbst** instructions cause Clean transactions on CoreNet and the CPC performs the appropriate action in order to clean any dirty data to memory, while leaving a valid copy in the cache. **dcbi** instructions cause Kill transactions on CoreNet and the CPC performs the appropriate action in order to invalidate the addressed coherency granule.

icbi instructions cause Kill transactions on CoreNet. Since the CPC is a unified cache, Kill transactions have no affect on the state of the addressed coherency granule in the CPC.

Note that cache management transactions such as Flush, Clean, and Kill have no affect to addresses configured as SRAM.

8.3.4 Decorated Storage

The *Freescale Book E Implementation Standards for Storage*, referred to as the EIS, defines an extension to the base Power Architecture instruction set called the Decorated Storage APU. This APU defines new instructions that include an attribute called a decoration that provides additional semantics for the device. The CoreNet platform cache supports the Decorated Storage APU as defined in EIS through specially-coded CoreNet transactions. Note that the decoration does not have any direct meaning to the core itself, but is interpreted by the CPC on behalf of the memory complex. A decorated operation carries up to four parameters:

- Type of access (load, store, or notify)
- Address

- Data (for stores)
- Decoration (encoding that defines the operation)

Decorated storage is only supported to addresses mapped by big-endian memory pages. In addition, the decorated storage operations are performed only on addresses that have been marked as caching-inhibited and guarded.

8.3.4.1 Decorated Load Operations

Decorated load operations read data from the memory complex (either from the CPC data arrays or from DDR), returns it to the core, and then performs a selectable update to the data in the memory complex. The overall flow through the data pipeline is similar to a read-modify-write (RMW) operation. The main difference is in how the new write data is generated. A decorated load operation can perform four different operations (clear, increment, decrement or set) on four different data types (1, 2, 4 or 8 bytes). The timing of these operations is the same as for RMW.

8.3.4.2 Decorated Store Operations

Decorated store operations are similar to decorated loads. The main difference is that decorated stores do not return data to the core and the update operations can perform an add and subtract (instead of just a increment or decrement) and minimum and maximum functions using data received from the core. The timing of these operations is the same as for RMW.

Decorated store operations require that the write data be presented in the proper byte lanes as required to perform the desired operation. [Table 8-223](#) shows the address alignment restrictions for the specific decorated store instructions. Note that the maximum and minimum functions only support one size value for each decoration value.

If a decorated store does not carry the proper address alignment for its decoration, the decorated store writes back the existing data to the memory complex (that is, it will nop).

[Table 8-223](#) describes the decorated storage instructions with supported decorations and the operations that are performed as a result.

Table 8-223. Decorated Storage Operations

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63] ¹	Update Operation ²	Function
lidx	0000	-	MEM(addr[0:35])[0:7] = 0x00	Clear byte
	0001	-	MEM(addr[0:35])[0:7] = 0xFF	Set byte
	0010	-	MEM(addr[0:35])[0:7]--	Decrement byte
	0011	-	MEM(addr[0:35])[0:7]++	Increment byte
lhdx	0000	-	MEM({addr[0:34], 0})[0:15] = 0x0000	Clear half-word
	0001	-	MEM({addr[0:34], 0})[0:15] = 0xFFFF	Set half-word
	0010	-	MEM({addr[0:34], 0})[0:15]--	Decrement half-word
	0011	-	MEM({addr[0:34], 00})[0:15]++	Increment half-word
lwdx	0000	-	MEM({addr[0:33], 00})[0:31] = 0x0000_0000	Clear word
	0001	-	MEM({addr[0:33], 00})[0:31] = 0xFFFF_FFFF	Set word
	0010	-	MEM({addr[0:33], 00})[0:31]--	Decrement word
	0011	-	MEM({addr[0:33], 00})[0:31]++	Increment word
lfddx	0000	-	MEM({addr[0:32], 000})[0:31] = 0x0000_0000_0000	Clear doubleword
	0001	-	MEM({addr[0:32], 000})[0:31] = 0xFFFF_FFFF_FFFF_FFFF	Set doubleword
	0010	-	MEM({addr[0:32], 000})[0:31]--	Decrement doubleword
	0011	-	MEM({addr[0:32], 000})[0:31]++	Increment doubleword
stbdx	0000	0111	MEM({addr[0:32], 000})[0:63] += exts64(din[56:63])	Doubleword accumulate with signed byte
		1111	MEM({addr[0:32], 000})[0:63] += exts64(din[120:127])	
	0001	0011	MEM({addr[0:33], 00})[0:31] += exts32(din[24:31])	Word accumulate with signed byte
		0111	MEM({addr[0:33], 00})[0:31] += exts32(din[56:63])	
		1011	MEM({addr[0:33], 00})[0:31] += exts32(din[88:95])	
		1111	MEM({addr[0:33], 00})[0:31] += exts32(din[120:127])	
	0010	1111	MEM({addr[0:31], 0000})[0:63]++ and MEM({addr[0:31], 0000})[64:127] += exts64(din[120:127])	Doubleword increment and doubleword accumulate with signed byte
	0011	0111	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += exts32(din[56:63])	Word increment and word accumulate with signed byte
		1111	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += exts32(din[120:127])	
sthdx	0000	0110	MEM({addr[0:32], 000})[0:63] += exts64(din[48:63])	Doubleword accumulate with signed half-word
		1110	MEM({addr[0:32], 000})[0:63] += exts64(din[112:127])	
	0001	0010	MEM({addr[0:33], 00})[0:31] += exts32(din[16:31])	Word accumulate with signed half-word
		0110	MEM({addr[0:33], 00})[0:31] += exts32(din[48:63])	
		1010	MEM({addr[0:33], 00})[0:31] += exts32(din[80:95])	

Table continues on the next page...

Table 8-223. Decorated Storage Operations (continued)

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63] ¹	Update Operation ²	Function
stwdx	0010	1110	MEM({addr[0:33], 00})[0:31] += exts32(din[112:127])	
		1110	MEM({addr[0:31], 0000})[0:63]++ and MEM({addr[0:31], 0000})[64:127] += exts64(din[112:127])	Doubleword increment and doubleword accumulate with signed half-word
	0011	0110	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += exts32(din[48:63])	Word increment and word accumulate with signed half-word
		1110	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += exts32(din[112:127])	
0000	0100	0100	MEM({addr[0:32], 000})[0:63] += exts64(din[32:63])	Doubleword accumulate with signed word
		1100	MEM({addr[0:32], 000})[0:63] += exts64(din[96:127])	
	0001	0000	MEM({addr[0:33], 00})[0:31] += din[0:31]	Word accumulate with signed word
		0100	MEM({addr[0:33], 00})[0:31] += din[32:63]	
		1000	MEM({addr[0:33], 00})[0:31] += din[64:95]	
		1100	MEM({addr[0:33], 00})[0:31] += din[96:127]	
	0010	1100	MEM({addr[0:31], 0000})[0:63]++ and MEM({addr[0:31], 0000})[64:127] += exts64(din[96:1'tw27])	Doubleword increment and doubleword accumulate with signed word
		0100	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += din[32:63]	
	0011	1100	MEM({addr[0:32], 000})[0:31]++ and MEM({addr[0:32], 000})[32:63] += din[96:127]	Word increment and word accumulate with signed word
		0000	MEM({addr[0:33], 00}) = unsigned_max(MEM({addr[0:33], 2'b00}), DIn[0:31])	32-bit maximum with unsigned word
	0101	0100	MEM({addr[0:33], 00}) = unsigned_max(MEM({addr[0:33], 00}), DIn[32:63])	
		1000	MEM({addr[0:33], 00}) = unsigned_max(MEM({addr[0:33], 00}), DIn[64:95])	
		1100	MEM({addr[0:33], 00}) = unsigned_max(MEM({addr[0:33], 00}), DIn[96:127])	
	0111	0000	MEM({addr[0:33], 00}) = unsigned_min(MEM({addr[0:33], 00}), DIn[0:31])	32-bit minimum with unsigned word
		0100	MEM({addr[0:33], 00}) = unsigned_min(MEM({addr[0:33], 00}), DIn[32:63])	
		1000	MEM({addr[0:33], 00}) = unsigned_min(MEM({addr[0:33], 00}), DIn[64:95])	
		1100	MEM({addr[0:33], 00}) = unsigned_min(MEM({addr[0:33], 00}), DIn[96:127])	

Table continues on the next page...

Table 8-223. Decorated Storage Operations (continued)

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63] ¹	Update Operation ²	Function
stfddx	0000	0000	MEM({addr[0:32], 000})[0:63] += din[0:63]	Doubleword accumulate with signed Doubleword
		1000	MEM({addr[0:32], 000})[0:63] += din[64:127]	
	0010	1000	MEM({addr[0:31], 0000})[0:63]++ and MEM({addr[0:31], 0000})[64:127] += din[64:127]	Doubleword increment and doubleword accumulate with signed doubleword
		0000	MEM({addr[0:32], 000}) = unsigned_max(MEM({addr[0:32], 000}), DIn[0:63])	
	0100	1000	MEM({addr[0:32], 000}) = unsigned_max(MEM({addr[0:32], 000}), DIn[64:127])	64-bit maximum with unsigned doubleword
		0000	MEM({addr[0:32], 000}) = unsigned_min(MEM({addr[0:32], 000}), DIn[0:63])	
	0110	1000	MEM({addr[0:32], 000}) = unsigned_min(MEM({addr[0:32], 000}), DIn[64:127])	64-bit minimum with unsigned doubleword
		-	MEM({addr[0:32], 000})[0:63]++	Doubleword increment
dsn	0000	-	MEM({addr[0:33], 00})[0:31]++	Word increment
	0001	-	MEM({addr[0:33], 00})[0:31] = 0000_0000_0000_0000	Doubleword clear
	0010	-	MEM({addr[0:33], 00})[0:31] = 0000_0000	Word clear
	0011	-	MEM({addr[0:33], 00})[0:31] = 0000_0000	

1. Addr[60:63] in this case is the core effective address, not the physical address.
2. The update operations are given as C-code equivalents. The following notes apply: exts64() represents a sign extension of the operand to 64 bits exts32() represents a sign extension of the operand to 32 bits. unsigned_max(a,b)=(a>b) ? a : b, where a and b are unsigned integers of the appropriate size unsigned_min(a,b)=(a>b) ? b : a, where a and b are unsigned integers of the appropriate size

8.3.4.3 Notify Operations

Notify is a simplification of a decorated store operation that does not require data from the core. Because of this, only increment, decrement, set and clear functions are possible. The timing of these operations is the same as for RMW.

8.3.5 Cache/SRAM Data Clear

Upon Hard Fail condition, the security monitor drives an interrupt to the CPC that causes the CPC to immediately initiate a write of all zeros to all locations in the data arrays. These write operations have the highest priority to the data pipeline. This operation is non-coherent and the system may be non-recoverable after this has occurred.

8.4 Initialization/Application Information

8.4.1 Supported SRAM Mode Configurations

Table 8-224. Supported SRAM Mode Configurations

CPC1			CPC2			DDR Controller 1	DDR Controller 2	
SRAM Mode Enabled (SRAMEN)	Interleaving (INTLVEN)	SRAM Size (SRAMSZ) ¹	SRAM Enabled (SRAMEN)	Interleaving (INTLVEN)	SRAM Size (SRAMSZ) ¹	See Supported DDR Interleaving Configurations , for more information.		
No	-	-	No	-	-	Any supported configuration	Any supported configuration	
No	-	-	Yes ²	No	1 Mbyte	Used	Not used ²	
Yes ³	No	1 Mbyte	No	-	-	Not used ³	Used	
Yes	No	64 Kbytes	Yes	No	64 Kbytes	Used, but memory controller interleaving disabled	Used, but memory controller interleaving disabled	
		256 Kbytes			256 Kbytes			
		1 Mbyte			1 Mbyte			
Yes	Yes	64 Kbytes	Yes	Yes	64 Kbytes	DDR1 and DDR2 used with memory controller cache line or page interleaving enabled		
		256 Kbytes			256 Kbytes			
		1 Mbyte			1 Mbyte			
Yes	No ⁴	64 Kbytes	Yes	No ⁴	64 Kbytes	DDR1 and DDR2 used with memory controller bank or super-bank interleaving enabled ⁴		
		256 Kbytes			256 Kbytes			
		1 Mbyte			1 Mbyte			

1. When both CPC1 and CPC2 are enabled as SRAM, the SRAM sizes must be the same.
2. If DDR2 is not used, CPC2 may be configured as all SRAM (that is, 1 Mbyte), even though CPC1 SRAM mode is disabled. However, in this case, disable CPC2 speculative requests by setting CPCHDBCR0[SPEC_DIS]. See [CPCHDBCR0](#) for more information.
3. If DDR1 is not used, CPC1 may be configured as all SRAM (that is, 1 Mbyte), even though CPC2 SRAM mode is disabled. However, in this case, disable CPC1 speculative requests by setting CPCHDBCR0[SPEC_DIS]. See [CPCHDBCR0](#), for more information.
4. If DDR1/DDR2 memory controller interleaving is set for bank or super-bank interleaving, the CPC SRAMs must not be interleaved.

8.4.2 Programming Examples

The following section gives examples of how software should configure the CPC control registers for various scenarios.

8.4.2.1 Modifying CPC Control and Status Registers

The sequence for modifying a CPC Control and Status Register is as follows:

1. Ensure that all prior instructions and memory accesses are complete and performed before modifying the CPC register.
2. Perform a store (WIMG = 01xx) to the CPC register to set its desired value.
3. Perform a load (WIMG = 01xx) to the CPC register to push the store to the CPC and ensure it has been performed.
4. Ensure that all subsequent instruction and memory accesses wait for the load to complete, ensuring that the CPC register update has been performed before any subsequent memory accesses.

An example Power Architecture instruction sequence to update CPCCSR0 from a core follows:

1. **mbar**
2. **isync**
3. **stw (WIMG = 01xx) CCSRBAR+CPC n block base address+0x000**
4. **lwz (WIMG = 01xx) CCSRBAR+CPC n block base address+0x000**
5. **mbar**

8.4.2.2 Enabling the CPC after Power-On Reset

The CPC registers have the reset values defined in the "[LAW](#)". Based on these initial register values, the following procedure is performed to invalidate, configure and enable the CPC after power-on reset:

1. Set CPCCSR0[CPCFI], CPCCSR0[CPCLFC] using the same write operation
2. Wait for hardware to clear CPCCSR0[CPCFI] and CPCCSR0[CPCLFC] using a polling loop
3. Set CPC configuration registers, including CPCSRCR1, CPCSRCR0, CPCPIR n , CPCPAR n , CPCPWR n , CPCERRDIS, CPCERRINTEN, and CPCERRCTL
4. Set CPCCSR0 bits to desired values (notably CPCPE, CPCWT, CPCREP and CPCLOA). Note this operation can be combined with step 6.
5. Configure the LAWs and DDR controller; this must be done before enabling the CPC.
6. Set CPCCSR0[CPCE] to enable the CPC.

This procedure needs to be performed for each CPC in the system.

8.4.2.3 Changing the Configuration of an Enabled CPC

In order to change the configuration registers of a CPC that has already been enabled, it is generally necessary to flush and disable the cache first using the following procedure:

1. Clear all bits of $CPCPAR_n$ (this prevents any new transactions from allocating into the CPC) using write operation.
2. Flush the CPC using one of the two following methods:
 - Hardware flush: Set $CPCCSR_0[CPCFL]$ and wait for hardware to clear $CPCCSR_0[CPCFL]$ by polling
 - Software flush: perform a **dcbf** operation to each coherency granule mapped to the memory target (that is, one **dcbf** for each 64 bytes).
3. Clear all lock bits by setting $CPCCSR_0[CPCLFC]$.
4. Wait for hardware to clear $CPCCSR_0[CPCLFC]$ using a polling loop.
5. Disable the CPC by clearing $CPCCSR_0[CPCE]$.
6. Reconfigure CPC as desired.
7. Enable CPC by setting $CPCCSR_0[CPCE]$.

Common operations that require this procedure include changing $CPCCSR_0[CPCWT]$, $CPCCPSR_0[CPCPE]$, $CPCPIR_n$, $CPCPAR_n$, and $CPCPWR_n$.

8.4.2.4 Disabling the CPC

To disable the CPC once it has been enabled, follow steps 1-5 of the procedure described in [Changing the Configuration of an Enabled CPC](#).

Chapter 9

CoreNet Coherency Fabric (CCF)

9.1 CCF Introduction

This chapter provides an overview of the CoreNet coherency fabric (CCF). It describes the components of the CCF and how they are interconnected. It also describes the salient features of the platform and its components.

The CoreNet coherency fabric (CCF) is a fabric-oriented, connectivity infrastructure that enables the implementation of coherent, multicore systems. The CCF acts as a central interconnect for cores, platform-level caches, memory subsystems, peripheral devices, and I/O host bridges in the system. The CCF natively supports Power Architecture® coherency semantics.

NOTE

Please adhere to all power architecture rules. WIMG bits in the system across a physical address must be identical.

The figure below shows organization of a multicore system designed around the CCF.

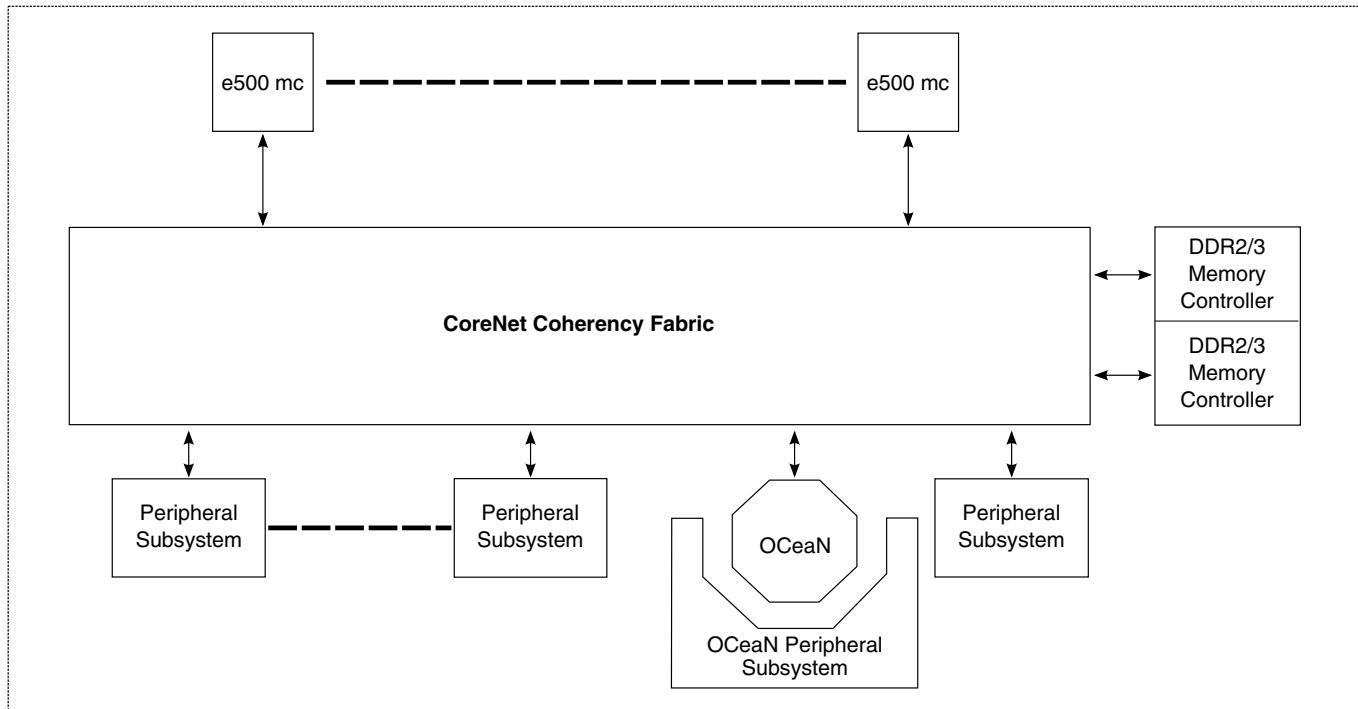


Figure 9-1. CoreNet Coherency Fabric Connectivity

9.1.1 CCF Features Summary

The CCF includes the following distinctive features:

- Multiple in-flight transactions with:
 - Concurrency of transactional progress through the system
 - Out of order completion
- Retry-free operation
 - Zero-loss utilization of system transaction processing bandwidth
- Sustainable bandwidth: 4 transactions/cycle
- Low latency data path for platform cache data
- Sustainable read bandwidth: 128 bytes (or two coherence granules - CGs) per cycle
- Power Architecture coherency semantics
 - Accelerated operation for non-coherent accesses
- 64 byte coherency granules
 - 1-8 bytes within an aligned double word, and aligned 16-byte, 32-byte, and 64-byte single copy atomic access semantics
- Rich transaction types and semantics to support:
 - All Power Architecture storage and synchronization operations
 - Inter-processor low-latency message send operations
 - RapidIO atomic operations
 - Reads and writes with semantic extensions for in-place atomic update of storage

- Transaction ordering support
 - Native support for Power Architecture ordering semantics
 - On-demand strong ordering for all "cache-inhibited" and "guarded" accesses from processors
 - Non-Power Architecture ordering support through host bridges
- Address map support
 - 32 local access windows (LAWs)
 - Boot window and CCSR spaces
 - DDR memory and SRAM interleaving support
- Logical partitioning support
 - Address-based, secure isolation of partitions and their resources
 - Coherency subdomain assignment and snoop limiting per LAW
 - Inter-partition sharing of address ranges
 - Peripheral access management unit (PAMU)
 - Capability to assign peripheral devices to logical partitions and limiting direct storage accesses (DSA) to owner partitions
 - Support for virtualized peripheral devices serving multiple partitions
 - Multiple windows per device
 - Window translation for accesses by devices
 - Transparent cache-coherent operation with PAMU tables in main memory
 - Transaction translation capability
 - Error isolation between partitions
- Two high performance memory complexes
 - Non-inclusive, in-line platform cache
 - Write-back semantics
 - Read latency optimization by keeping track of owner processors
 - I/O stashing
 - Line and way locking
 - Data delivery capability of one CG per cycle
 - Semantic extension support for cache-based atomic update of data
 - Support for stashing
 - Processor cache stashing
 - Address- and attribute-based I/O stashing in platform cache
 - Debug support
 - Source tracking from source to destination
 - Access triggering, filtering, and tracing
 - Event counts for performance monitoring

9.2 CoreNet Coherency Fabric (CCF) Memory Map

CCF memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_8200	Snoop ID 0 Port Mapping Register (CCF_SIDMR0)	32	R	FFF8_0000h	9.2.1/388
1_8204	Snoop ID n Port Mapping Register (CCF_SIDMR1)	32	R/W	0000_0000h	9.2.2/389
1_8208	Snoop ID n Port Mapping Register (CCF_SIDMR2)	32	R/W	0000_0000h	9.2.2/389
1_820C	Snoop ID n Port Mapping Register (CCF_SIDMR3)	32	R/W	0000_0000h	9.2.2/389
1_8210	Snoop ID n Port Mapping Register (CCF_SIDMR4)	32	R/W	0000_0000h	9.2.2/389
1_8214	Snoop ID n Port Mapping Register (CCF_SIDMR5)	32	R/W	0000_0000h	9.2.2/389
1_8218	Snoop ID n Port Mapping Register (CCF_SIDMR6)	32	R/W	0000_0000h	9.2.2/389
1_821C	Snoop ID n Port Mapping Register (CCF_SIDMR7)	32	R/W	0000_0000h	9.2.2/389
1_8220	Snoop ID n Port Mapping Register (CCF_SIDMR8)	32	R/W	0000_0000h	9.2.2/389
1_8224	Snoop ID n Port Mapping Register (CCF_SIDMR9)	32	R/W	0000_0000h	9.2.2/389
1_8228	Snoop ID n Port Mapping Register (CCF_SIDMR10)	32	R/W	0000_0000h	9.2.2/389
1_822C	Snoop ID n Port Mapping Register (CCF_SIDMR11)	32	R/W	0000_0000h	9.2.2/389
1_8230	Snoop ID n Port Mapping Register (CCF_SIDMR12)	32	R/W	0000_0000h	9.2.2/389
1_8234	Snoop ID n Port Mapping Register (CCF_SIDMR13)	32	R/W	0000_0000h	9.2.2/389
1_8238	Snoop ID n Port Mapping Register (CCF_SIDMR14)	32	R/W	0000_0000h	9.2.2/389
1_823C	Snoop ID n Port Mapping Register (CCF_SIDMR15)	32	R/W	0000_0000h	9.2.2/389
1_8240	Snoop ID n Port Mapping Register (CCF_SIDMR16)	32	R/W	0000_0000h	9.2.2/389
1_8244	Snoop ID n Port Mapping Register (CCF_SIDMR17)	32	R/W	0000_0000h	9.2.2/389
1_8248	Snoop ID n Port Mapping Register (CCF_SIDMR18)	32	R/W	0000_0000h	9.2.2/389
1_824C	Snoop ID n Port Mapping Register (CCF_SIDMR19)	32	R/W	0000_0000h	9.2.2/389
1_8250	Snoop ID n Port Mapping Register (CCF_SIDMR20)	32	R/W	0000_0000h	9.2.2/389
1_8254	Snoop ID n Port Mapping Register (CCF_SIDMR21)	32	R/W	0000_0000h	9.2.2/389
1_8258	Snoop ID n Port Mapping Register (CCF_SIDMR22)	32	R/W	0000_0000h	9.2.2/389
1_825C	Snoop ID n Port Mapping Register (CCF_SIDMR23)	32	R/W	0000_0000h	9.2.2/389
1_8260	Snoop ID n Port Mapping Register (CCF_SIDMR24)	32	R/W	0000_0000h	9.2.2/389
1_8264	Snoop ID n Port Mapping Register (CCF_SIDMR25)	32	R/W	0000_0000h	9.2.2/389
1_8268	Snoop ID n Port Mapping Register (CCF_SIDMR26)	32	R/W	0000_0000h	9.2.2/389
1_826C	Snoop ID n Port Mapping Register (CCF_SIDMR27)	32	R/W	0000_0000h	9.2.2/389
1_8270	Snoop ID n Port Mapping Register (CCF_SIDMR28)	32	R/W	0000_0000h	9.2.2/389
1_8274	Snoop ID n Port Mapping Register (CCF_SIDMR29)	32	R/W	0000_0000h	9.2.2/389
1_8278	Snoop ID n Port Mapping Register (CCF_SIDMR30)	32	R/W	0000_0000h	9.2.2/389
1_827C	Snoop ID n Port Mapping Register (CCF_SIDMR31)	32	R/W	0000_0000h	9.2.2/389
1_8600	CSDID 0 Port Mapping Register (CCF_CIDMR0)	32	R	FFF8_0000h	9.2.3/390
1_8604	CSDID n Port Mapping Register (CCF_CIDMR1)	32	R/W	0000_0000h	9.2.4/391
1_8608	CSDID n Port Mapping Register (CCF_CIDMR2)	32	R/W	0000_0000h	9.2.4/391
1_860C	CSDID n Port Mapping Register (CCF_CIDMR3)	32	R/W	0000_0000h	9.2.4/391
1_8610	CSDID n Port Mapping Register (CCF_CIDMR4)	32	R/W	0000_0000h	9.2.4/391
1_8614	CSDID n Port Mapping Register (CCF_CIDMR5)	32	R/W	0000_0000h	9.2.4/391

Table continues on the next page...

CCF memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_8618	CSDID n Port Mapping Register (CCF_CIDMR6)	32	R/W	0000_0000h	9.2.4/391
1_861C	CSDID n Port Mapping Register (CCF_CIDMR7)	32	R/W	0000_0000h	9.2.4/391
1_8620	CSDID n Port Mapping Register (CCF_CIDMR8)	32	R/W	0000_0000h	9.2.4/391
1_8624	CSDID n Port Mapping Register (CCF_CIDMR9)	32	R/W	0000_0000h	9.2.4/391
1_8628	CSDID n Port Mapping Register (CCF_CIDMR10)	32	R/W	0000_0000h	9.2.4/391
1_862C	CSDID n Port Mapping Register (CCF_CIDMR11)	32	R/W	0000_0000h	9.2.4/391
1_8630	CSDID n Port Mapping Register (CCF_CIDMR12)	32	R/W	0000_0000h	9.2.4/391
1_8634	CSDID n Port Mapping Register (CCF_CIDMR13)	32	R/W	0000_0000h	9.2.4/391
1_8638	CSDID n Port Mapping Register (CCF_CIDMR14)	32	R/W	0000_0000h	9.2.4/391
1_863C	CSDID n Port Mapping Register (CCF_CIDMR15)	32	R/W	0000_0000h	9.2.4/391
1_8640	CSDID n Port Mapping Register (CCF_CIDMR16)	32	R/W	0000_0000h	9.2.4/391
1_8644	CSDID n Port Mapping Register (CCF_CIDMR17)	32	R/W	0000_0000h	9.2.4/391
1_8648	CSDID n Port Mapping Register (CCF_CIDMR18)	32	R/W	0000_0000h	9.2.4/391
1_864C	CSDID n Port Mapping Register (CCF_CIDMR19)	32	R/W	0000_0000h	9.2.4/391
1_8650	CSDID n Port Mapping Register (CCF_CIDMR20)	32	R/W	0000_0000h	9.2.4/391
1_8654	CSDID n Port Mapping Register (CCF_CIDMR21)	32	R/W	0000_0000h	9.2.4/391
1_8658	CSDID n Port Mapping Register (CCF_CIDMR22)	32	R/W	0000_0000h	9.2.4/391
1_865C	CSDID n Port Mapping Register (CCF_CIDMR23)	32	R/W	0000_0000h	9.2.4/391
1_8660	CSDID n Port Mapping Register (CCF_CIDMR24)	32	R/W	0000_0000h	9.2.4/391
1_8664	CSDID n Port Mapping Register (CCF_CIDMR25)	32	R/W	0000_0000h	9.2.4/391
1_8668	CSDID n Port Mapping Register (CCF_CIDMR26)	32	R/W	0000_0000h	9.2.4/391
1_866C	CSDID n Port Mapping Register (CCF_CIDMR27)	32	R/W	0000_0000h	9.2.4/391
1_8670	CSDID n Port Mapping Register (CCF_CIDMR28)	32	R/W	0000_0000h	9.2.4/391
1_8674	CSDID n Port Mapping Register (CCF_CIDMR29)	32	R/W	0000_0000h	9.2.4/391
1_8678	CSDID n Port Mapping Register (CCF_CIDMR30)	32	R/W	0000_0000h	9.2.4/391
1_867C	CSDID n Port Mapping Register (CCF_CIDMR31)	32	R/W	0000_0000h	9.2.4/391
1_8A00	CCF Error Detect Register (CCF_CEDR)	32	w1c	0000_0000h	9.2.5/392
1_8A04	CCF Error Enable Register (CCF_CEER)	32	R/W	0000_0000h	9.2.6/393
1_8A0C	CCF Error Capture Attribute Register (CCF_CECAR)	32	w1c	0000_0000h	9.2.7/394
1_8A10	CCF Error Capture Address Register High (CCF_CECADRH)	32	w1c	0000_0000h	9.2.8/394
1_8A14	CCF Error Capture Address Register Low (CCF_CECADRL)	32	w1c	0000_0000h	9.2.9/395
1_8A18	CCF Error Capture Attribute Register 2 (CCF_CECAR2)	32	R/W	0000_0000h	9.2.10/395

9.2.1 Snoop ID 0 Port Mapping Register (CCF_SIDMR0)

The Snoop ID n Port Mapping Registers provide the mechanism for mapping CoreNet Snoop IDs to CoreNet ports that will receive snoop traffic. CoreNet transactions include an 8-bit Snoop ID [0:7]. A Snoop ID of zero indicates an inactive Snoop ID and maps to the default SIDMR0. SIDMR0 is a read-only register. The reset value of SIDMR0 is coherent, but considered lower performing. A non-zero Snoop ID indicates a lookup into the snoop ID port mapping registers (SIDMR1-31). The Snoop ID acts as an index into these registers and extracts a 32-bit PORT_ID field indicating the active snooper ports for this transaction. An active Snoop ID (non-zero) overrides all other implementation-specific determinations for snoopers, and supersedes the coherency subdomain settings in the CCF_CIDMRn registers.

NOTE

Although there are 256 possible Snoop IDs, the CCF only supports bits [3:7] and thus implements 32 snoop ID port mapping registers. If any of the upper three bits (Snoop ID[0:2]) are set, the snoop ID will alias to the 32 SIDMRs defined by Snoop ID[3:7]. If the lower bits [3:7] = 0b00000 then the default SIDMR0 is used.

Address: 1_8000h base + 200h offset = 1_8200h

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

CCF_SIDMR0 field descriptions

Field	Description
0-31 PORT_ID	Port ID. When set, a bit indicates which corresponding core to snoop for this transaction. Bit 0 corresponds to Core 0. Bit 1 corresponds to Core 1. Bit 2 corresponds to Core 2. Bit 3 corresponds to Core 3. Bit 4 corresponds to Core 4. Bit 5 corresponds to Core 5. Bit 6 corresponds to Core 6. Bit 7 corresponds to Core 7. Bits 8-12 correspond to the PAMUs. The PAMUs should only need to be snooped when updating the PAACTs. In this case, all PAMU bits should be set.

CCF_SIDMR0 field descriptions (continued)

Field	Description
	All other bits reserved.

9.2.2 Snoop ID n Port Mapping Register (CCF_SIDMRn)

The Snoop ID n Port Mapping Registers provide the mechanism for mapping CoreNet Snoop IDs to CoreNet ports that will receive snoop traffic. CoreNet transactions include an 8-bit Snoop ID [0:7]. A Snoop ID of zero indicates an inactive Snoop ID and maps to the default SIDMR0. SIDMR0 is a read-only register. The reset value of SIDMR0 is coherent, but considered lower performing. A non-zero Snoop ID indicates a lookup into the snoop ID port mapping registers (SIDMR1-31). The Snoop ID acts as an index into these registers and extracts a 32-bit PORT_ID field indicating the active snooper ports for this transaction. An active Snoop ID (non-zero) overrides all other implementation-specific determinations for snoopers, and supersedes the coherency subdomain settings in the CCF_CIDMRn registers.

NOTE

Although there are 256 possible Snoop IDs, the CCF only supports bits [3:7] and thus implements 32 snoop ID port mapping registers. If any of the upper three bits (Snoop ID[0:2]) are set, the snoop ID will alias to the 32 SIDMRs defined by Snoop ID[3:7]. If the lower bits [3:7] = 0b00000 then the default SIDMR0 is used.

Address: 1_8000h base + 204h offset + (4d × i), where i=0d to 30d

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

CCF_SIDMRn field descriptions

Field	Description
0–31 PORT_ID	Port ID. When set, a bit indicates which corresponding core to snoop for this transaction Bit 0 corresponds to Core 0. Bit 1 corresponds to Core 1. Bit 2 corresponds to Core 2. Bit 3 corresponds to Core 3. Bit 4 corresponds to Core 4.

CCF_SIDMRn field descriptions (continued)

Field	Description
	<p>Bit 5 corresponds to Core 5.</p> <p>Bit 6 corresponds to Core 6.</p> <p>Bit 7 corresponds to Core 7.</p> <p>Bits 8-12 correspond to the PAMUs. The PAMUs should only need to be snooped when updating the PAACTs. In this case, all PAMU bits should be set.</p> <p>All other bits reserved.</p>

9.2.3 CSDID 0 Port Mapping Register (CCF_CIDMR0)

The Coherency Subdomain ID n Port Mapping Registers provide the mechanism for mapping CoreNet Coherency Subdomain ID to CoreNet ports that will receive snoop traffic. The LAW attribute registers include an 8-bit coherency subdomain ID field (LAWARn[CSD_ID]). If the transaction Snoop ID is zero, the CSD_ID obtained from the LAW hit is used to determine the valid snoopers for the transaction. The CSD_ID acts as an index into the CSDIDMR0-31 registers and extracts a 32-bit PORT_ID field indicating the active snooper ports for this transaction.

NOTE

Although there are 256 possible CSD_IDs, the CCF only supports 32 CIDMRs by using CSD_ID[3:7]; CSD_ID[0:2] are assumed to be inactive/zero.

NOTE

CCSR, alternate configuration, and boot window space accesses all carry a CSDID=0xFF and point to the appropriate port as an independent function.

NOTE

A non-zero Snoop ID value overrides the CIDMRn settings; in this case, the SIDMRn settings determine the valid snoopers.

Address: 1_8000h base + 600h offset = 1_8600h

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

CCF_CIDMR0 field descriptions

Field	Description
0-31 PORT_ID	<p>Port ID. When set, a bit indicates which corresponding core to snoop for this transaction</p> <p>Bit 0 corresponds to Core 0.</p> <p>Bit 1 corresponds to Core 1.</p> <p>Bit 2 corresponds to Core 2.</p> <p>Bit 3 corresponds to Core 3.</p> <p>Bit 4 corresponds to Core 4.</p> <p>Bit 5 corresponds to Core 5.</p> <p>Bit 6 corresponds to Core 6.</p> <p>Bit 7 corresponds to Core 7.</p> <p>Bits 8-12 correspond to the PAMUs. The PAMUs should only need to be snooped when updating the PAACRs. In this case, all PAMU bits should be set.</p> <p>All other bits reserved.</p>

9.2.4 CSDID n Port Mapping Register (CCF_CIDMRn)

The Coherency Subdomain ID n Port Mapping Registers provide the mechanism for mapping CoreNet Coherency Subdomain ID to CoreNet ports that will receive snoop traffic. The LAW attribute registers include an 8-bit coherency subdomain ID field (LAWARn[CSD_ID]). If the transaction Snoop ID is zero, the CSD_ID obtained from the LAW hit is used to determine the valid snoopers for the transaction. The CSD_ID acts as an index into the CSDIDMR0-31 registers and extracts a 32-bit PORT_ID field indicating the active snooper ports for this transaction.

NOTE

Although there are 256 possible CSD_IDs, the CCF only supports 32 CIDMRs by using CSD_ID[3:7]; CSD_ID[0:2] are assumed to be inactive/zero.

NOTE

CCSR, alternate configuration, and boot window space accesses all carry a CSDID=0xFF and point to the appropriate port as an independent function.

NOTE

A non-zero Snoop ID value overrides the CIDMRn settings; in this case, the SIDMRn settings determine the valid snoopers.

CoreNet Coherency Fabric (CCF) Memory Map

Address: 1_8000h base + 604h offset + (4d × i), where i=0d to 30d

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

CCF_CIDMRn field descriptions

Field	Description
0-31 PORT_ID	<p>Port ID. When set, a bit indicates which corresponding core to snoop for this transaction</p> <p>Bit 0 corresponds to Core 0.</p> <p>Bit 1 corresponds to Core 1.</p> <p>Bit 2 corresponds to Core 2.</p> <p>Bit 3 corresponds to Core 3.</p> <p>Bit 4 corresponds to Core 4.</p> <p>Bit 5 corresponds to Core 5.</p> <p>Bit 6 corresponds to Core 6.</p> <p>Bit 7 corresponds to Core 7.</p> <p>Bits 8-12 correspond to the PAMUs. The PAMUs should only need to be snooped when updating the PAACTs. In this case, all PAMU bits should be set.</p> <p>All other bits reserved.</p>

9.2.5 CCF Error Detect Register (CCF_CEDR)

Address: 1_8000h base + A00h offset = 1_8A00h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reserved																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R															CV	LAE
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCF_CEDR field descriptions

Field	Description
0-29 -	This field is reserved. Reserved
30 CV	Coherency violation. This indicates a states of the coherency granule were found to be in violation of the coherency protocol.

Table continues on the next page...

CCF_CEDR field descriptions (continued)

Field	Description
31 LAE	<p>Local Access Error Cases that can generate an LAE</p> <ul style="list-style-type: none"> • Local Access Window Miss. An incoming transaction misses all LAWs (except when the PBL is active). • Unavailable target ID programmed in LAW attribute register. See LAWn attribute register (LAW_LAWARn), for more information. <p>0 Local Access Error has not occurred 1 Local Access Error has occurred</p>

9.2.6 CCF Error Enable Register (CCF_CEER)

Address: 1_8000h base + A04h offset = 1_8A04h

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

CCF_CEER field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30 CVE	Coherency violation error enable 0 Coherency violation errors are disabled (default) 1 Coherency violation errors are enabled
31 LAEE	Local access error enable 0 Disable reporting of local access errors as interrupts (default) 1 Enable reporting of local access errors as interrupts

9.2.7 CCF Error Capture Attribute Register (CCF_CECAR)

Address: 1_8000h base + A0Ch offset = 1_8A0Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SRC_ID								Reserved							
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UVT	Reserved											VAL			
W	w1c												w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCF_CECAR field descriptions

Field	Description
0–7 SRC_ID	The transaction system source ID[0:7]. See Global Source and Target IDs for valid encodings.
8–15 -	This field is reserved. Reserved
16 UVT	Unavailable target ID. This indicates that the transaction hit in a LAW window that has an unavailable target ID.
17–30 -	This field is reserved. Reserved
31 VAL	Valid 0 The CCF error capture attribute register is not valid 1 The CCF error capture attribute register is valid

9.2.8 CCF Error Capture Address Register High (CCF_CECADRH)

Address: 1_8000h base + A10h offset = 1_8A10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															ADDRH																
W																w1c																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

CCF_CECADRH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 ADDRH	ADDRH contains the high-order bits[28:31]of the captured error address.

9.2.9 CCF Error Capture Address Register Low (CCF_CECADRL)

Address: 1_8000h base + A14h offset = 1_8A14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ADDRL																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

CCF_CECADRL field descriptions

Field	Description
0–31 ADDRL	ADDRL contains the low-order bits [32:63] of the captured error address.

9.2.10 CCF Error Capture Attribute Register 2 (CCF_CECAR2)

Address: 1_8000h base + A18h offset = 1_8A18h

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

CCF_CECAR2 field descriptions

Field	Description
0–7 CCCCV	Core cache complex coherency violation. Bit 0 corresponds to core cache complex 0, bit 1 corresponds to core cache complex 1, and so on. A coherency violation causes 2 or more bits to be set among CCCCV and CPCCV.
8–22 -	This field is reserved. Reserved
23 CPCCV	CPC coherency violation. A coherency violation causes 2 or more bits to be set among CCCCV and CPCCV.
24–31 -	This field is reserved. Reserved

Chapter 10

Peripheral Access Management Unit (PAMU)

10.1 PAMU Introduction

This section provides an overview and features list for the peripheral access management unit (PAMU).

10.1.1 PAMU Overview

A multicore system, illustrated in [Figure 10-1](#), consists of one or more Power Architecture processors, a system memory separate from other subsystems, and a number of I/O devices that may initiate transactions to system memory. The processors are linked over a host domain interconnect to each other, to the system memory and to one or more I/O devices or domains. Typically, a PAMU resides astride the boundary between the coherency and I/O domains, as shown in [Figure 10-1](#) and enforces authorization and access control over DSA operations into the coherency domain.

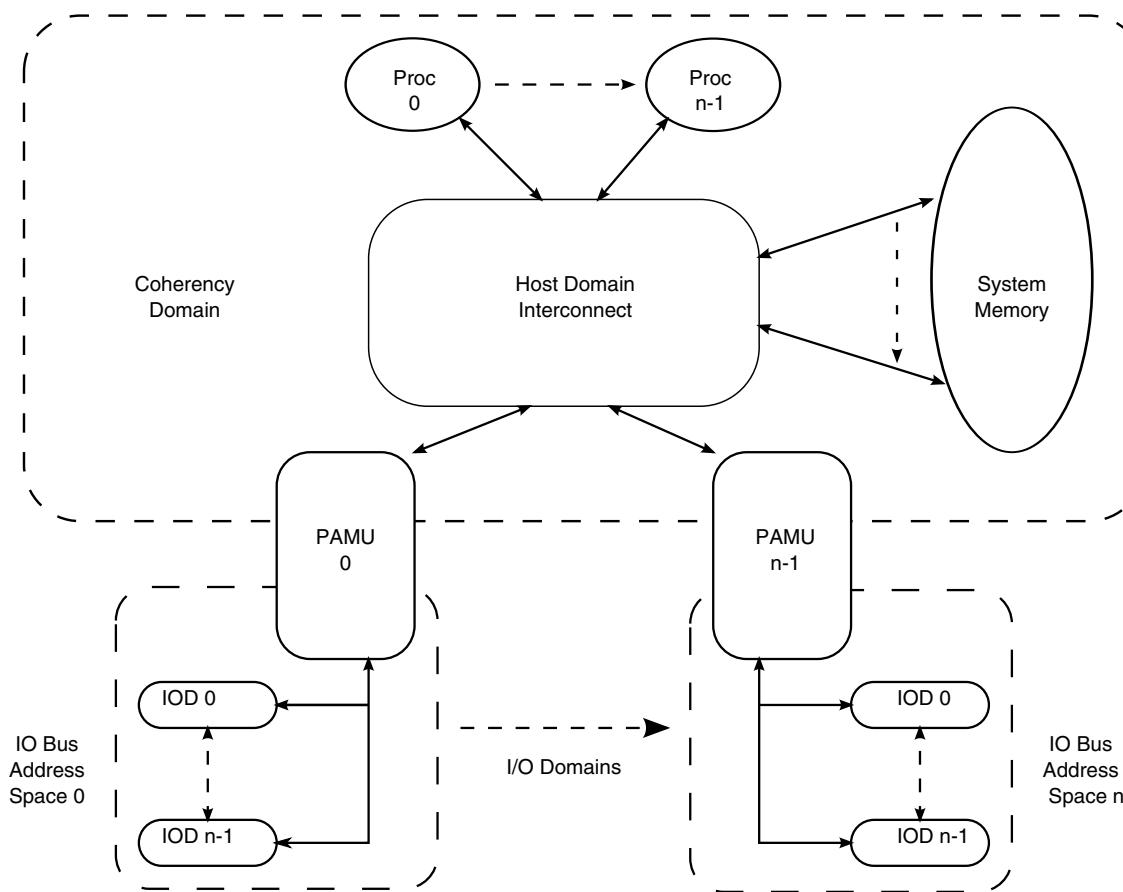


Figure 10-1. Illustration of a Multicore System

To accelerate the authorization and translation of operations, PAMU contains caches for quick lookup of memory resident data structures containing the authorization and translation attributes.

10.1.2 PAMU Features Summary

PAMU includes the following features:

- Support for authorizing DMA and peer-to-peer operations, that is, checking that the operation is originated by a valid I/O adapter, authorized to issue these operations, and that the operation is to a valid address window provided to that I/O adapter.
- Support for the access control of DMA and peer-to-peer operations, that is, allowing only the types of operations that fall within the access privileges provided to that I/O adapter.

- Support for address translation of DMA and peer-to-peer operations, that is, redirecting operations with an address in the I/O bus address space to a corresponding address in system memory space. The following address translation modes are supported:
 - No address translation
 - Window translation
- Support for operation type translation of DMA and peer-to-peer operations, that is, converting the ingress operation encoding to a corresponding egress operation encoding. The following operation translation modes are supported:
 - No Operation Translation
 - Immediate Operation Translation
 - Indexed Operation Translation for up to sixteen indexes, that is, the field is limited to OMI[0:3]
- Support for up to 4-Kbyte LIODNs, that is, signaling is limited to LIODN[4:15]
- Support for up to 256 DSA sub-windows when multiple windows are enabled, that is, the field is limited to WCE[1:3]
- Hardware coherent PAMU Look-aside caches to improve performance
 - A -entry, direct-mapped primary PAACT cache
 - A -entry, 2-way, set-associative secondary PAACT cache
 - A 4-entry, direct-mapped OMT cache
 - Support for receiving invalidating snoop operations for the PAMU look-aside caches (PLCs) from the coherency domain
 - Support for issuing PAMU fetch operations when the relevant entry is not present in the PLC
- Support for a maximum 36-bit system address space
- Support for a maximum 40-bit I/O address space
- Capability to generate interrupts
 - Access violations
 - PAMU operational errors

10.2 Data Structures Used by PAMU

The following data structures are used by the PAMU. The data structures are located in system memory and are accessed by the PAMU as necessary. These data structures are programmed by the hypervisor controlling the system.

10.2.1 Overview of Data Structures

10.2.1.1 Peripheral Access Authorization and Control Tables (PAACTs)

PAACTs are the primary data structures used by PAMU . A PAACT is a table of peripheral access authorization and control entries (PAACE). Each PAACE defines the range of I/O bus address space that is accessible by the LIOD and the associated access capabilities.

There are two types of PAACTs: primary PAACT (PPAACT) and secondary PAACT (SPAACT).

A given physical I/O device may be able to act as one or more independent logical I/O devices (LIODs). Each such logical I/O device is assigned an identifier called logical I/O device number (LIODN).

A LIOD is allocated a contiguous portion of the I/O bus address space called the DSA window for performing DSA operations. The DSA window may optionally be divided into multiple sub-windows, each of which may be used to map to a region in system storage space. The first sub-window is referred to as the primary sub-window and the remaining are called secondary sub-windows.

A DSA window is 2^n bytes in size. Sub-windows are declared in a two-level hierarchy: a DSA window and 2^m ($n > m > 0$) sub-windows within it. There are means available to recover unused secondary windows and unused address sub-ranges (See [Recovering Address Space](#), for more information.). The sub-windows are labeled 0 through 2^F-1 . Each sub-window is of size 2^{n-m} bytes. A sub-range of size 2^k within the sub-window may be specified for mapping. The sub-range can be narrower than the default sub-window size (2^{n-m} bytes).

The access attributes of the DSA window and the primary sub-window are declared using the primary PAACE (PPAACE) and those for the secondary sub-windows are declared using secondary PAACES (SPAACES). Only the primary PAACE is placed in the PPAACT. The secondary PAACES are placed consecutively in the SPAACT in the order of their labels. A pointer in the PPAACE points to the placement of the first of the SPAACES in the SPAACT. See [Figure 10-2](#) and [Figure 10-3](#).

DSA operations from a LIOD may be redirected to different locations in system storage space via the mechanism of address translation via a PAACE. The attributes that define whether address translation is enabled and the type of address translation, if enabled, are contained in the PAACE.

A LIOD may have restrictions as to the type of DSA operation it is allowed to perform. The attributes that define the access permissions for that LIOD are contained in the PAACE.

Furthermore, DSA operations from a LIOD may be translated to different operation types in system storage space via the mechanism of operation translation via a PAACE. The attributes that define whether operation translation is enabled and the type of operation translation, if enabled, are contained in the PAACE. The translation may optionally refer to a data structure called operation mapping table (See [Operation Mapping Table](#), for more information.).

A LIOD presents its LIODN along with every DSA operation issued by it. The PAMU uses the LIODN as a means to identify the relevant PPAACE and then if necessary the SPAACES and other data structures that specify the access capabilities provided to that LIOD.

10.2.1.2 Operation Mapping Table

DSA operations from a LIOD may optionally be converted to different operation types via the mechanism of operation translation. The conversion of operation types may be due to differences between the source and destination interconnect protocols of the DSA operation. The attributes that define operation translation is also contained in PAACE. Additional operation mapping attributes for that LIOD may be optionally contained in a operation mapping entry (OME) in the operation mapping table (OMT).

10.2.1.3 Access Capabilities Across LIODs

LIODs may be provided with a subset of the access capabilities described above as necessary. Certain LIODs may utilize a narrower set of the mechanisms provided while other LIODs simultaneously utilize a broader or the full set of mechanisms. Hypervisor may also choose to vary, over time, the provisions for an LIOD, as application requirements and the capabilities of the system vary.

10.2.2 Structure and Contents of PAACTs

[Figure 10-2](#) displays the placement of the PPAACT within system memory space and the arrangement of PPAACEs within the PPAACT.

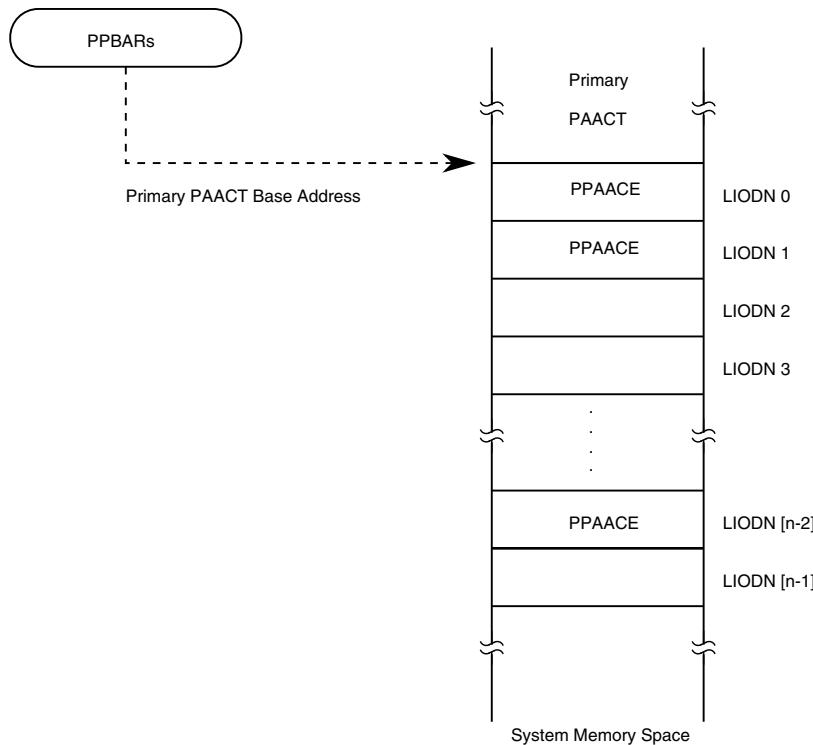


Figure 10-2. Arrangement of Primary PAACEs in System Memory Space

- The LIODN is used to index into the PPAACT to identify the PPAACE corresponding to the LIOD.
- The PPAACE located at the LIODN index of PPAACT contains fields indicating the base address and size of the DSA window for that LIOD (see [Table 10-1](#)). If the DSA window has properties that cannot be expressed in a single PAACE, multiple PAACEs are created. Fields in the PPAACE located at the LIODN index of PPAACT indicate the presence and number of multiple PAACEs for that LIOD:
 - The DSA window is divided into non-overlapping sub-windows of equal size with a PAACE containing the attributes of a sub-window.
 - The entry located at the LIODN index is called the primary PAACE. The primary PAACE contains the attributes describing the entire DSA window as well as the primary sub-window.
 - The SPAACEs are contained in the SPAACT starting at the first secondary PAACE index (FSPI) provided in the PPAACE and are arranged in linear order corresponding to the linear address order of the sub-window (see example in [Figure 10-3](#)).
 - A SPAACE describes the properties of a secondary DSA sub-window (see [Table 10-2](#)).

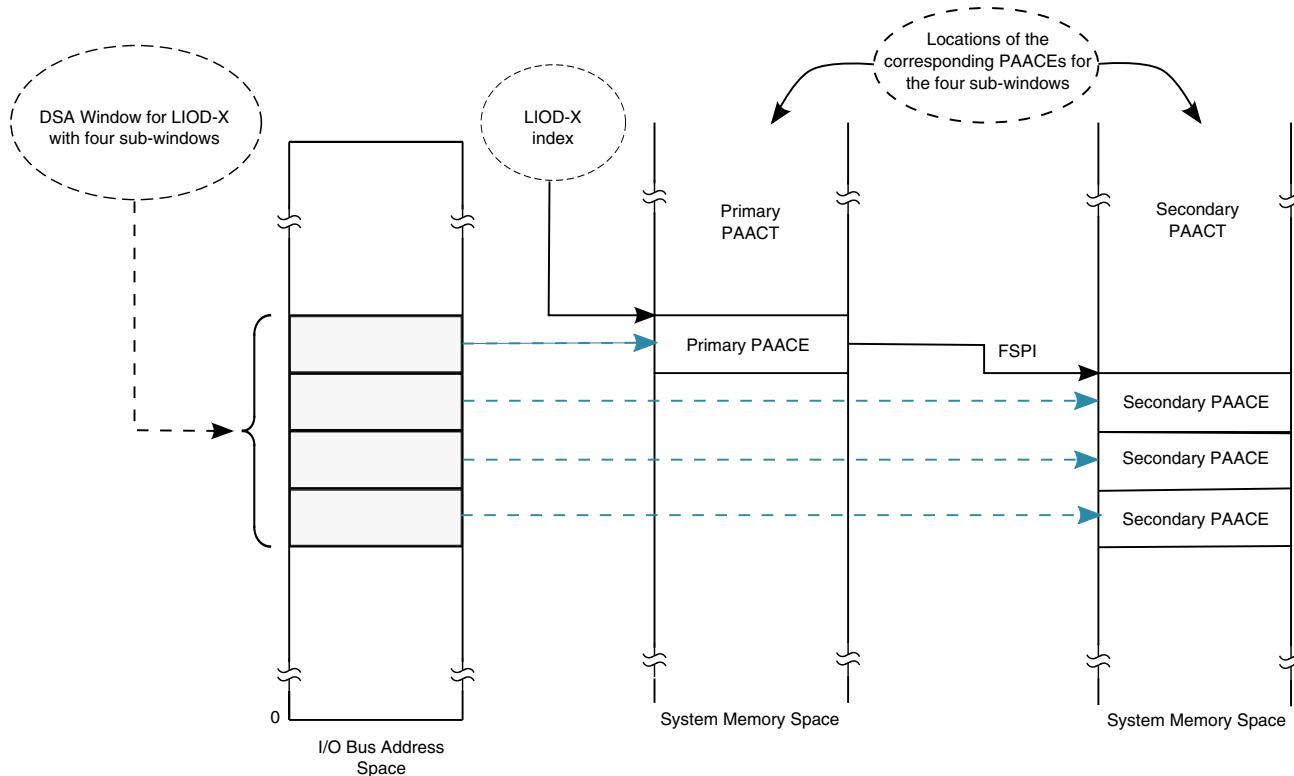


Figure 10-3. Example of DSA Sub-Windows and their Corresponding PAACEs

10.2.3 Peripheral Access Authorization and Control Entry (PAACE)

This section describes the structure and contents of a PAACE. The format of the primary PAACE data structure is shown in [Table 10-1](#). Fields valid in only the primary PAACE apply to the entire DSA window defined by the primary PAACE. Fields valid in both the primary and secondary PAACE apply to the DSA sub-window defined by that PAACE.

Table 10-1. Primary PAACE Format

Index Offset (hex)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
00	WBA				WSE		MW	AP	-	PT	V	
08	DA	IA				WCE			ATM		OTM	
10	TWBA				SWSE				reserved			

Table continues on the next page...

Table 10-1. Primary PAACE Format (continued)

Ind ex Offs et (he x)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
18	FSPI	IOEA	MOEA	IOEB	MOEB							
		reserved		OMI								
20-3 8	reserved											

The format of the secondary PAACE data structure is shown in [Table 10-2](#).

Table 10-2. Secondary PAACE Format

Ind ex Offs et (he x)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
00	reserved	LIODN		reserved			AP	-	PT	V		
08	DA	IA			reserved				ATM		OTM	
10	TWBA			SWSE		reserved						
18	reserved	IOEA	MOEA	IOEB	MOEB							
	reserved			OMI								
20-3 8	reserved											

10.2.3.1 PAACE Offset 0x00

Table 10-3. PAACE Offset 0x00 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-51	See Notes		Notes: If primary PAACE: bits 0-51 of PAACE offset 0x00 contain the WBA. If secondary PAACE: bits 32-47 of PAACE offset 0x00 contain the LIODN. The rest of the bits are reserved.
0-51	WBA	P	Window Base Address

Table continues on the next page...

Table 10-3. PAACE Offset 0x00 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			<p>These bits describe the base address of the DSA window in I/O address space. The WBA is the 52 most significant address bits of a 64-bit address aligned to the window size as described in the WSE field. The WBA field is 52 bits wide to allow for a 64-bit address aligned to the minimum 4 Kbyte window size. These bits are valid in the primary PAACE.</p> <p>In certain platforms, all these bits may not be required. However, enough bits must be implemented to match the largest system storage address in the platform. Reads to unimplemented high order bits must return 0.</p>
32-47	LIODN	S	<p>Logical I/O Device Number</p> <p>These bits define the LIODN for which the DSA sub-window is being described. These bits are valid in the secondary PAACE.</p>
52-57	WSE	P	<p>Window Size Encoding</p> <p>These bits identify the size of the DSA window starting from the window base address. These bits are valid in the primary PAACE and are reserved in a secondary PAACE. DSA window size is $2^{(WSE+1)}$ bytes:</p> <ul style="list-style-type: none"> 0x00-0x0A Reserved (Write reserved, read returns 0) 0x0B 4 Kbytes 0x0C 8 Kbytes ... $2^{(WSE+1)}$ bytes 0x1F 4 Gbytes 0x20 8 Gbytes ... $2^{(WSE+1)}$ bytes 0x3F 16 Ebytes
58	MW	P	<p>Multiple Windows</p> <p>This bit indicates whether the LIOD has multiple DSA sub-windows. These bits are valid in the primary PAACE and are reserved in a secondary PAACE. It is coded as follows:</p> <ul style="list-style-type: none"> 0 A single DSA window exists for the LIOD. 1 Multiple DSA sub-windows exist for the LIOD. The number of sub-windows is indicated in the WCE field of the PPAACE.
59-60	AP	P and S	<p>Access Permissions. These bits define whether access to the window described by this PAACE is permitted and if permitted, the type of permitted access. In the field description, a Query Access typically refers to read type operations where the effect of the operation does not change the values of locations or state of the system; an Update Access typically refers to write type operations where the effect of the operation changes the values of locations or state of the system.</p> <p>00 Access denied</p>

Table continues on the next page...

Table 10-3. PAACE Offset 0x00 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			01 Query access only 10 Update access only 11 Access permitted for all operation types NOTE: If MW = 0, then AP applies to the DSA window described by this PAACE; if MW = 1, then AP applies to the DSA sub-window described by this PAACE.
61	-	P and S	Reserved
62	PT	P and S	PAACE Type This bit indicates whether it is a primary or secondary PAACE. 0 Primary PAACE 1 Secondary PAACE
63	V	P and S	Valid PAACE This bit defines whether this PAACE is valid. 0 Invalid PAACE. Other PAACE fields do not contain valid information. 1 Valid PAACE. Other PAACE fields may be referenced.

10.2.3.2 PAACE Offset 0x08

Table 10-4. PAACE Offset 0x08 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-31	DA	P and S	Domain Attributes These bits define operation attributes specific to the destination domain of the operation. The contents of the DA field are further described in PAACE Domain Attributes .
32-55	IA	P and S	Implementation Attributes These bits define operation attributes that are implementation specific. For details see Implementation Attributes
56-59	WCE	P	Window Count Encoding These bits identify the number of DSA sub-windows for this LIOD. These bits are valid in the primary PAACE when MW is enabled and are reserved in a secondary PAACE. Window count is $2^{(WCE+1)}$ windows: 0000 2 DSA sub-windows 0001 4 DSA sub-windows ...

Table continues on the next page...

Table 10-4. PAACE Offset 0x08 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			1111 65536 DSA sub-windows
60-61	ATM	P and S	<p>Address Translation Mode</p> <p>These bits define whether address translation is enabled for the window described by this PAACE and if enabled, the type of translation to perform.</p> <p>00 No address translation. The ingress and egress operation address are the same.</p> <p>01 Window address translation</p> <p>10 Reserved</p> <p>11 Reserved</p>
62-63	OTM	P and S	<p>Operation Translation Mode</p> <p>These bits define whether operation type translation is enabled for the window described by this PAACE and if enabled, the type of translation to perform.</p> <p>00 No operation translation. The ingress and egress operation type are the same.</p> <p>01 Operation translation enabled with immediate operation translation.</p> <p>10 Operation translation enabled with indexed operation translation.</p> <p>11 Reserved</p>

10.2.3.2.1 PAACE Domain Attributes

Table 10-5 describes the domain attribute (DA) field of PAACE at offset 0x08-0x0B.

Table 10-5. PAACE Domain Attributes

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-15	-	-	Reserved (Write reserved; read returns 0)
16-23	SnpID	P and S	<p>Snoop ID</p> <p>This field may be used to identify a specific snooper or set of snoopers required for the maintenance of hardware coherency for the window described by this PAACE. This field is valid if the M bit for the PAACE is 1.</p> <p>If the SnpID field contains a pattern of all 0s, then all snoopers in the coherency domain must participate in the maintenance of hardware coherency. A SnpID pattern of all 0s indicates no specific snooper or set of snoopers is being specified. If the SnpID pattern is non-zero, only the snoopers</p>

Table continues on the next page...

Table 10-5. PAACE Domain Attributes (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			defined by the SnpID field must participate in the maintenance of hardware coherency. The snoopers indicated by the non-zero SnpID pattern is implementation dependent.
24	M	P and S	<p>Memory Coherence Required</p> <p>This bit defines if hardware cache coherency is to be maintained in the coherency domain for the window described by this PAACE.</p> <p>0 Non-coherent memory, that is, hardware cache coherency mechanisms not required.</p> <p>1 Coherent memory, that is, hardware cache coherency mechanisms required.</p>
25-31	-	-	Reserved (Write reserved; read returns 0)

10.2.3.3 PAACE Offset 0x10

Table 10-6. PAACE Offset 0x10 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-51	TWBA	P and S	<p>Translated Window Base Address</p> <p>These bits describe the translated base address in system storage space of the window described by this PAACE. The TWBA is the 52 most significant address bits of the 64-bit address, aligned to the size of the window described by this PAACE. The TWBA field is 52 bits wide to allow for a 64-bit address aligned to the minimum 4 Kbyte size of a window described by this PAACE. This field is referenced when ATM[0:1] = 01 for this PAACE.</p>
52-57	SWSE	P and S	<p>Sub-Window Sub-Range Encoding</p> <p>These bits identify the size of the sub-range of addresses within the DSA sub-window described by this PAACE, that is eligible for access and translation. This eligible sub-range starts from the starting address of this DSA sub-window. This field is only referenced when the MW bit is 1. The sub-range size is $2^{(SWSE+1)}$ bytes:</p> <ul style="list-style-type: none"> 0x00-0x0A Reserved (Write reserved; read returns 0) 0x0B 4 Kbyte 0x0C 8 Kbyte ... $2^{(SWSE+1)}$ bytes 0x1F 4 Gbyte 0x20 8 Gbyte ... $2^{(SWSE+1)}$ bytes

Table continues on the next page...

Table 10-6. PAACE Offset 0x10 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			0x3F 16Ebytes
58-63	-	-	Reserved (Write reserved; read returns 0)

10.2.3.4 PAACE Offset 0x18**Table 10-7. PAACE Offset 0x18 Field Definitions**

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-31	FSPI	P	<p>First Secondary PAACE Index</p> <p>These bits identify the index location in the SPAACT of the secondary PAACE that describes the second DSA sub-window for this LIOD. These bits are valid in the primary PAACE when MW is set and are reserved in a secondary PAACE.</p>
32-63	See Notes		<p>Notes:</p> <ol style="list-style-type: none"> If OTM[0:1] == 01, bits 32-63 of PAACE Offset 0x18 contain IOEA, MOEA, IOEB, and MOEB respectively. If OTM[0:1] == 10, bits 48-63 of PAACE Offset 0x18 contain the OMI, bits 32-47 are reserved. The following apply to ingress operation encodings: bit[0] is a qualifier indicating the operation type. bit[0] = 0 is the qualifier value reserved for all query operation types. bit[0] = 1 is the qualifier value reserved for all update operation types. The operation encoding in IOE[1:7] is implementation dependent. The following apply to egress operation encodings: bit[0] is a qualifier indicating a valid operation type. The operation encoding in EOE[1:7] is implementation dependent.
48-63	OMI	P and S	<p>Operation Mapping Index</p> <p>If OTM[0:1] == 10, these bits are used to determine the relevant operation mapping entry (OME) of the OMT to be referenced for operation translation.</p>
32-39	IOEA	P and S	<p>Ingress Operation Encoding A</p> <p>If OTM[0:1] == 01, these bits define the ingress operation encoding A for which the corresponding egress encoding will be provided.</p>
40-47	MOEA	P and S	Mapped Operation Encoding A

Table continues on the next page...

Table 10-7. PAACE Offset 0x18 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			These bits define the mapped operation encoding, including a portion of the translated egress operation encoding A (EOEA) for the ingress operation encoding A. MOEA[0] is a valid bit MOEA[1:7] contains the corresponding bits of the egress operation encoding, that is, EOE[1:7].
48-55	IOEB	P and S	Ingress Operation Encoding B If OTM[0:1] == 01, these bits define the ingress operation encoding B for which the corresponding egress encoding will be provided.
56-63	MOEB	P and S	Mapped Operation Encoding B These bits define the mapped operation encoding, including a portion of the translated egress operation encoding B (EOEB) for the ingress operation encoding B. MOEB[0] is a valid bit MOEB[1:7] contains the corresponding bits of the egress operation encoding, that is, EOEB[1:7].

10.3 PAMU Memory Map/Register Definitions

PAMU registers reside in a 4-Kbyte block of CCSR space starting at 0x02_n000, where n is the instance of PAMU being referenced.

10.4 PAMU Memory Map

PAMU memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_0000	Primary PAACT Base Address High register (PAMU1_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_0004	Primary PAACT Base Address Low register (PAMU1_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_0008	Primary PAACT Limit Address High register (PAMU1_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_000C	Primary PAACT Limit Address Low register (PAMU1_PPLAL)	32	R/W	0000_0000h	10.4.4/432

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
2_0010	Secondary PAACT Base Address High register (PAMU1_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_0014	Secondary PAACT Base Address Low register (PAMU1_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_0018	Secondary PAACT Limit Address High register (PAMU1_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_001C	Secondary PAACT Limit Address Low register (PAMU1_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_0020	OMT Base Address High register (PAMU1_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_0024	OMT Base Address Low register (PAMU1_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_0028	OMT Limit Address High register (PAMU1_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_002C	OMT Limit Address Low register (PAMU1_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_0030	PAMU Address Capabilities Register 1 (PAMU1_PAC1)	32	R	0000_000Fh	10.4.13/436
2_0034	PAMU Address Capabilities Register 2 (PAMU1_PAC2)	32	R	0000_00FFh	10.4.14/436
2_0040	PAMU Operation Error Status register 1 (PAMU1_POES1)	32	R/W	0000_0000h	10.4.15/437
2_0044	PAMU Operation Error Status register 2 (PAMU1_POES2)	32	R/W	0000_0000h	10.4.16/438
2_0048	PAMU Operation Error Address High register (PAMU1_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_004C	PAMU Operation Error Address Low register (PAMU1_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_0050	Access Violation Status register 1 (PAMU1_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_0054	Access Violation Status register 2 (PAMU1_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_0058	Access Violation Address High register (PAMU1_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_005C	Access Violation Address Low register (PAMU1_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_0060	ECC Error Control Register (PAMU1_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_0068	ECC Error Interrupt Enable Register (PAMU1_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_006C	ECC Error Detect Register (PAMU1_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_0070	ECC Error Attributes Register (PAMU1_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_0074	ECC Error Address High (PAMU1_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_0078	ECC Error Address Low (PAMU1_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_007C	ECC Error Data High (PAMU1_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_0080	ECC Error Data Low (PAMU1_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_0090	Unauthorized device access detection register (PAMU1_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_0BF8	PAMU Revision register 1 (PAMU1_PR1)	32	R	0901_0100h	10.4.32/449
2_0BFC	PAMU Revision register 2 (PAMU1_PR2)	32	R	0000_0000h	10.4.33/450
2_0C00	PAMU Capabilities register 1 (PAMU1_PC1)	32	R	0000_000Fh	10.4.34/450
2_0C04	PAMU Capabilities register 2 (PAMU1_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_0C08	PAMU Capabilities register 3 (PAMU1_PC3)	32	R	007E_1E00h	10.4.36/452
2_0C0C	PAMU Capabilities register 4 (PAMU1_PC4)	32	R	0000_0CE1h	10.4.37/454

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_0C10	PAMU Control register (PAMU1_PC)	32	R/W	0000_1111h	10.4.38/455
2_0C1C	PAMU Interrupt Control and Status register (PAMU1_PICS)	32	R/W	0000_0000h	10.4.39/456
2_1000	Primary PAACT Base Address High register (PAMU2_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_1004	Primary PAACT Base Address Low register (PAMU2_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_1008	Primary PAACT Limit Address High register (PAMU2_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_100C	Primary PAACT Limit Address Low register (PAMU2_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_1010	Secondary PAACT Base Address High register (PAMU2_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_1014	Secondary PAACT Base Address Low register (PAMU2_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_1018	Secondary PAACT Limit Address High register (PAMU2_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_101C	Secondary PAACT Limit Address Low register (PAMU2_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_1020	OMT Base Address High register (PAMU2_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_1024	OMT Base Address Low register (PAMU2_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_1028	OMT Limit Address High register (PAMU2_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_102C	OMT Limit Address Low register (PAMU2_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_1030	PAMU Address Capabilities Register 1 (PAMU2_PAC1)	32	R	0000_000Fh	10.4.13/436
2_1034	PAMU Address Capabilities Register 2 (PAMU2_PAC2)	32	R	0000_00FFh	10.4.14/436
2_1040	PAMU Operation Error Status register 1 (PAMU2_POES1)	32	R/W	0000_0000h	10.4.15/437
2_1044	PAMU Operation Error Status register 2 (PAMU2_POES2)	32	R/W	0000_0000h	10.4.16/438
2_1048	PAMU Operation Error Address High register (PAMU2_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_104C	PAMU Operation Error Address Low register (PAMU2_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_1050	Access Violation Status register 1 (PAMU2_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_1054	Access Violation Status register 2 (PAMU2_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_1058	Access Violation Address High register (PAMU2_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_105C	Access Violation Address Low register (PAMU2_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_1060	ECC Error Control Register (PAMU2_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_1068	ECC Error Interrupt Enable Register (PAMU2_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_106C	ECC Error Detect Register (PAMU2_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_1070	ECC Error Attributes Register (PAMU2_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_1074	ECC Error Address High (PAMU2_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_1078	ECC Error Address Low (PAMU2_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_107C	ECC Error Data High (PAMU2_EEDHI)	32	R/W	0000_0000h	10.4.29/448

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_1080	ECC Error Data Low (PAMU2_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_1090	Unauthorized device access detection register (PAMU2_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_1BF8	PAMU Revision register 1 (PAMU2_PR1)	32	R	0901_0100h	10.4.32/449
2_1BFC	PAMU Revision register 2 (PAMU2_PR2)	32	R	0000_0000h	10.4.33/450
2_1C00	PAMU Capabilities register 1 (PAMU2_PC1)	32	R	0000_000Fh	10.4.34/450
2_1C04	PAMU Capabilities register 2 (PAMU2_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_1C08	PAMU Capabilities register 3 (PAMU2_PC3)	32	R	007E_1E00h	10.4.36/452
2_1C0C	PAMU Capabilities register 4 (PAMU2_PC4)	32	R	0000_0CE1h	10.4.37/454
2_1C10	PAMU Control register (PAMU2_PC)	32	R/W	0000_1111h	10.4.38/455
2_1C1C	PAMU Interrupt Control and Status register (PAMU2_PICS)	32	R/W	0000_0000h	10.4.39/456
2_2000	Primary PAACT Base Address High register (PAMU3_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_2004	Primary PAACT Base Address Low register (PAMU3_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_2008	Primary PAACT Limit Address High register (PAMU3_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_200C	Primary PAACT Limit Address Low register (PAMU3_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_2010	Secondary PAACT Base Address High register (PAMU3_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_2014	Secondary PAACT Base Address Low register (PAMU3_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_2018	Secondary PAACT Limit Address High register (PAMU3_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_201C	Secondary PAACT Limit Address Low register (PAMU3_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_2020	OMT Base Address High register (PAMU3_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_2024	OMT Base Address Low register (PAMU3_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_2028	OMT Limit Address High register (PAMU3_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_202C	OMT Limit Address Low register (PAMU3_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_2030	PAMU Address Capabilities Register 1 (PAMU3_PAC1)	32	R	0000_000Fh	10.4.13/436
2_2034	PAMU Address Capabilities Register 2 (PAMU3_PAC2)	32	R	0000_00FFh	10.4.14/436
2_2040	PAMU Operation Error Status register 1 (PAMU3_POES1)	32	R/W	0000_0000h	10.4.15/437
2_2044	PAMU Operation Error Status register 2 (PAMU3_POES2)	32	R/W	0000_0000h	10.4.16/438
2_2048	PAMU Operation Error Address High register (PAMU3_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_204C	PAMU Operation Error Address Low register (PAMU3_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_2050	Access Violation Status register 1 (PAMU3_AVS1)	32	R/W	0000_0000h	10.4.19/440
2_2054	Access Violation Status register 2 (PAMU3_AVS2)	32	R/W	0000_0000h	10.4.20/442

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_2058	Access Violation Address High register (PAMU3_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_205C	Access Violation Address Low register (PAMU3_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_2060	ECC Error Control Register (PAMU3_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_2068	ECC Error Interrupt Enable Register (PAMU3_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_206C	ECC Error Detect Register (PAMU3_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_2070	ECC Error Attributes Register (PAMU3_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_2074	ECC Error Address High (PAMU3_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_2078	ECC Error Address Low (PAMU3_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_207C	ECC Error Data High (PAMU3_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_2080	ECC Error Data Low (PAMU3_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_2090	Unauthorized device access detection register (PAMU3_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_2BF8	PAMU Revision register 1 (PAMU3_PR1)	32	R	0901_0100h	10.4.32/449
2_2BFC	PAMU Revision register 2 (PAMU3_PR2)	32	R	0000_0000h	10.4.33/450
2_2C00	PAMU Capabilities register 1 (PAMU3_PC1)	32	R	0000_000Fh	10.4.34/450
2_2C04	PAMU Capabilities register 2 (PAMU3_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_2C08	PAMU Capabilities register 3 (PAMU3_PC3)	32	R	007E_1E00h	10.4.36/452
2_2C0C	PAMU Capabilities register 4 (PAMU3_PC4)	32	R	0000_0CE1h	10.4.37/454
2_2C10	PAMU Control register (PAMU3_PC)	32	R/W	0000_1111h	10.4.38/455
2_2C1C	PAMU Interrupt Control and Status register (PAMU3_PICS)	32	R/W	0000_0000h	10.4.39/456
2_3000	Primary PAACT Base Address High register (PAMU4_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_3004	Primary PAACT Base Address Low register (PAMU4_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_3008	Primary PAACT Limit Address High register (PAMU4_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_300C	Primary PAACT Limit Address Low register (PAMU4_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_3010	Secondary PAACT Base Address High register (PAMU4_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_3014	Secondary PAACT Base Address Low register (PAMU4_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_3018	Secondary PAACT Limit Address High register (PAMU4_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_301C	Secondary PAACT Limit Address Low register (PAMU4_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_3020	OMT Base Address High register (PAMU4_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_3024	OMT Base Address Low register (PAMU4_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_3028	OMT Limit Address High register (PAMU4_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_302C	OMT Limit Address Low register (PAMU4_OLAL)	32	R/W	0000_0000h	10.4.12/435

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_3030	PAMU Address Capabilities Register 1 (PAMU4_PAC1)	32	R	0000_000Fh	10.4.13/436
2_3034	PAMU Address Capabilities Register 2 (PAMU4_PAC2)	32	R	0000_00FFh	10.4.14/436
2_3040	PAMU Operation Error Status register 1 (PAMU4_POES1)	32	R/W	0000_0000h	10.4.15/437
2_3044	PAMU Operation Error Status register 2 (PAMU4_POES2)	32	R/W	0000_0000h	10.4.16/438
2_3048	PAMU Operation Error Address High register (PAMU4_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_304C	PAMU Operation Error Address Low register (PAMU4_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_3050	Access Violation Status register 1 (PAMU4_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_3054	Access Violation Status register 2 (PAMU4_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_3058	Access Violation Address High register (PAMU4_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_305C	Access Violation Address Low register (PAMU4_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_3060	ECC Error Control Register (PAMU4_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_3068	ECC Error Interrupt Enable Register (PAMU4_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_306C	ECC Error Detect Register (PAMU4_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_3070	ECC Error Attributes Register (PAMU4_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_3074	ECC Error Address High (PAMU4_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_3078	ECC Error Address Low (PAMU4_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_307C	ECC Error Data High (PAMU4_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_3080	ECC Error Data Low (PAMU4_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_3090	Unauthorized device access detection register (PAMU4_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_3BF8	PAMU Revision register 1 (PAMU4_PR1)	32	R	0901_0100h	10.4.32/449
2_3BFC	PAMU Revision register 2 (PAMU4_PR2)	32	R	0000_0000h	10.4.33/450
2_3C00	PAMU Capabilities register 1 (PAMU4_PC1)	32	R	0000_000Fh	10.4.34/450
2_3C04	PAMU Capabilities register 2 (PAMU4_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_3C08	PAMU Capabilities register 3 (PAMU4_PC3)	32	R	007E_1E00h	10.4.36/452
2_3C0C	PAMU Capabilities register 4 (PAMU4_PC4)	32	R	0000_0CE1h	10.4.37/454
2_3C10	PAMU Control register (PAMU4_PC)	32	R/W	0000_1111h	10.4.38/455
2_3C1C	PAMU Interrupt Control and Status register (PAMU4_PICS)	32	R/W	0000_0000h	10.4.39/456
2_4000	Primary PAACT Base Address High register (PAMU5_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_4004	Primary PAACT Base Address Low register (PAMU5_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_4008	Primary PAACT Limit Address High register (PAMU5_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_400C	Primary PAACT Limit Address Low register (PAMU5_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_4010	Secondary PAACT Base Address High register (PAMU5_SPBAH)	32	R/W	0000_0000h	10.4.5/432

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_4014	Secondary PAACT Base Address Low register (PAMU5_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_4018	Secondary PAACT Limit Address High register (PAMU5_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_401C	Secondary PAACT Limit Address Low register (PAMU5_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_4020	OMT Base Address High register (PAMU5_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_4024	OMT Base Address Low register (PAMU5_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_4028	OMT Limit Address High register (PAMU5_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_402C	OMT Limit Address Low register (PAMU5_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_4030	PAMU Address Capabilities Register 1 (PAMU5_PAC1)	32	R	0000_000Fh	10.4.13/436
2_4034	PAMU Address Capabilities Register 2 (PAMU5_PAC2)	32	R	0000_00FFh	10.4.14/436
2_4040	PAMU Operation Error Status register 1 (PAMU5_POES1)	32	R/W	0000_0000h	10.4.15/437
2_4044	PAMU Operation Error Status register 2 (PAMU5_POES2)	32	R/W	0000_0000h	10.4.16/438
2_4048	PAMU Operation Error Address High register (PAMU5_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_404C	PAMU Operation Error Address Low register (PAMU5_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_4050	Access Violation Status register 1 (PAMU5_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_4054	Access Violation Status register 2 (PAMU5_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_4058	Access Violation Address High register (PAMU5_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_405C	Access Violation Address Low register (PAMU5_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_4060	ECC Error Control Register (PAMU5_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_4068	ECC Error Interrupt Enable Register (PAMU5_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_406C	ECC Error Detect Register (PAMU5_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_4070	ECC Error Attributes Register (PAMU5_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_4074	ECC Error Address High (PAMU5_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_4078	ECC Error Address Low (PAMU5_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_407C	ECC Error Data High (PAMU5_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_4080	ECC Error Data Low (PAMU5_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_4090	Unauthorized device access detection register (PAMU5_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_4BF8	PAMU Revision register 1 (PAMU5_PR1)	32	R	0901_0100h	10.4.32/449
2_4BFC	PAMU Revision register 2 (PAMU5_PR2)	32	R	0000_0000h	10.4.33/450
2_4C00	PAMU Capabilities register 1 (PAMU5_PC1)	32	R	0000_000Fh	10.4.34/450
2_4C04	PAMU Capabilities register 2 (PAMU5_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_4C08	PAMU Capabilities register 3 (PAMU5_PC3)	32	R	007E_1E00h	10.4.36/452
2_4C0C	PAMU Capabilities register 4 (PAMU5_PC4)	32	R	0000_0CE1h	10.4.37/454
2_4C10	PAMU Control register (PAMU5_PC)	32	R/W	0000_1111h	10.4.38/455
2_4C1C	PAMU Interrupt Control and Status register (PAMU5_PICS)	32	R/W	0000_0000h	10.4.39/456

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_5000	Primary PAACT Base Address High register (PAMU6_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_5004	Primary PAACT Base Address Low register (PAMU6_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_5008	Primary PAACT Limit Address High register (PAMU6_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_500C	Primary PAACT Limit Address Low register (PAMU6_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_5010	Secondary PAACT Base Address High register (PAMU6_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_5014	Secondary PAACT Base Address Low register (PAMU6_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_5018	Secondary PAACT Limit Address High register (PAMU6_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_501C	Secondary PAACT Limit Address Low register (PAMU6_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_5020	OMT Base Address High register (PAMU6_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_5024	OMT Base Address Low register (PAMU6_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_5028	OMT Limit Address High register (PAMU6_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_502C	OMT Limit Address Low register (PAMU6_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_5030	PAMU Address Capabilities Register 1 (PAMU6_PAC1)	32	R	0000_000Fh	10.4.13/436
2_5034	PAMU Address Capabilities Register 2 (PAMU6_PAC2)	32	R	0000_00FFh	10.4.14/436
2_5040	PAMU Operation Error Status register 1 (PAMU6_POES1)	32	R/W	0000_0000h	10.4.15/437
2_5044	PAMU Operation Error Status register 2 (PAMU6_POES2)	32	R/W	0000_0000h	10.4.16/438
2_5048	PAMU Operation Error Address High register (PAMU6_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_504C	PAMU Operation Error Address Low register (PAMU6_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_5050	Access Violation Status register 1 (PAMU6_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_5054	Access Violation Status register 2 (PAMU6_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_5058	Access Violation Address High register (PAMU6_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_505C	Access Violation Address Low register (PAMU6_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_5060	ECC Error Control Register (PAMU6_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_5068	ECC Error Interrupt Enable Register (PAMU6_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_506C	ECC Error Detect Register (PAMU6_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_5070	ECC Error Attributes Register (PAMU6_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_5074	ECC Error Address High (PAMU6_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_5078	ECC Error Address Low (PAMU6_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_507C	ECC Error Data High (PAMU6_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_5080	ECC Error Data Low (PAMU6_EEDLO)	32	R/W	0000_0000h	10.4.30/448

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_5090	Unauthorized device access detection register (PAMU6_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_5BF8	PAMU Revision register 1 (PAMU6_PR1)	32	R	0901_0100h	10.4.32/449
2_5BFC	PAMU Revision register 2 (PAMU6_PR2)	32	R	0000_0000h	10.4.33/450
2_5C00	PAMU Capabilities register 1 (PAMU6_PC1)	32	R	0000_000Fh	10.4.34/450
2_5C04	PAMU Capabilities register 2 (PAMU6_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_5C08	PAMU Capabilities register 3 (PAMU6_PC3)	32	R	007E_1E00h	10.4.36/452
2_5C0C	PAMU Capabilities register 4 (PAMU6_PC4)	32	R	0000_0CE1h	10.4.37/454
2_5C10	PAMU Control register (PAMU6_PC)	32	R/W	0000_1111h	10.4.38/455
2_5C1C	PAMU Interrupt Control and Status register (PAMU6_PICS)	32	R/W	0000_0000h	10.4.39/456
2_6000	Primary PAACT Base Address High register (PAMU7_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_6004	Primary PAACT Base Address Low register (PAMU7_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_6008	Primary PAACT Limit Address High register (PAMU7_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_600C	Primary PAACT Limit Address Low register (PAMU7_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_6010	Secondary PAACT Base Address High register (PAMU7_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_6014	Secondary PAACT Base Address Low register (PAMU7_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_6018	Secondary PAACT Limit Address High register (PAMU7_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_601C	Secondary PAACT Limit Address Low register (PAMU7_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_6020	OMT Base Address High register (PAMU7_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_6024	OMT Base Address Low register (PAMU7_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_6028	OMT Limit Address High register (PAMU7_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_602C	OMT Limit Address Low register (PAMU7_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_6030	PAMU Address Capabilities Register 1 (PAMU7_PAC1)	32	R	0000_000Fh	10.4.13/436
2_6034	PAMU Address Capabilities Register 2 (PAMU7_PAC2)	32	R	0000_00FFh	10.4.14/436
2_6040	PAMU Operation Error Status register 1 (PAMU7_POES1)	32	R/W	0000_0000h	10.4.15/437
2_6044	PAMU Operation Error Status register 2 (PAMU7_POES2)	32	R/W	0000_0000h	10.4.16/438
2_6048	PAMU Operation Error Address High register (PAMU7_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_604C	PAMU Operation Error Address Low register (PAMU7_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_6050	Access Violation Status register 1 (PAMU7_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_6054	Access Violation Status register 2 (PAMU7_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_6058	Access Violation Address High register (PAMU7_AVAH)	32	R/W	0000_0000h	10.4.21/443

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_605C	Access Violation Address Low register (PAMU7_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_6060	ECC Error Control Register (PAMU7_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_6068	ECC Error Interrupt Enable Register (PAMU7_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_606C	ECC Error Detect Register (PAMU7_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_6070	ECC Error Attributes Register (PAMU7_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_6074	ECC Error Address High (PAMU7_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_6078	ECC Error Address Low (PAMU7_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_607C	ECC Error Data High (PAMU7_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_6080	ECC Error Data Low (PAMU7_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_6090	Unauthorized device access detection register (PAMU7_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_6BF8	PAMU Revision register 1 (PAMU7_PR1)	32	R	0901_0100h	10.4.32/449
2_6BFC	PAMU Revision register 2 (PAMU7_PR2)	32	R	0000_0000h	10.4.33/450
2_6C00	PAMU Capabilities register 1 (PAMU7_PC1)	32	R	0000_000Fh	10.4.34/450
2_6C04	PAMU Capabilities register 2 (PAMU7_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_6C08	PAMU Capabilities register 3 (PAMU7_PC3)	32	R	007E_1E00h	10.4.36/452
2_6C0C	PAMU Capabilities register 4 (PAMU7_PC4)	32	R	0000_0CE1h	10.4.37/454
2_6C10	PAMU Control register (PAMU7_PC)	32	R/W	0000_1111h	10.4.38/455
2_6C1C	PAMU Interrupt Control and Status register (PAMU7_PICS)	32	R/W	0000_0000h	10.4.39/456
2_7000	Primary PAACT Base Address High register (PAMU8_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_7004	Primary PAACT Base Address Low register (PAMU8_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_7008	Primary PAACT Limit Address High register (PAMU8_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_700C	Primary PAACT Limit Address Low register (PAMU8_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_7010	Secondary PAACT Base Address High register (PAMU8_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_7014	Secondary PAACT Base Address Low register (PAMU8_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_7018	Secondary PAACT Limit Address High register (PAMU8_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_701C	Secondary PAACT Limit Address Low register (PAMU8 SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_7020	OMT Base Address High register (PAMU8_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_7024	OMT Base Address Low register (PAMU8_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_7028	OMT Limit Address High register (PAMU8_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_702C	OMT Limit Address Low register (PAMU8_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_7030	PAMU Address Capabilities Register 1 (PAMU8_PAC1)	32	R	0000_000Fh	10.4.13/436

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_7034	PAMU Address Capabilities Register 2 (PAMU8_PAC2)	32	R	0000_00FFh	10.4.14/436
2_7040	PAMU Operation Error Status register 1 (PAMU8_POES1)	32	R/W	0000_0000h	10.4.15/437
2_7044	PAMU Operation Error Status register 2 (PAMU8_POES2)	32	R/W	0000_0000h	10.4.16/438
2_7048	PAMU Operation Error Address High register (PAMU8_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_704C	PAMU Operation Error Address Low register (PAMU8_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_7050	Access Violation Status register 1 (PAMU8_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_7054	Access Violation Status register 2 (PAMU8_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_7058	Access Violation Address High register (PAMU8_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_705C	Access Violation Address Low register (PAMU8_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_7060	ECC Error Control Register (PAMU8_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_7068	ECC Error Interrupt Enable Register (PAMU8_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_706C	ECC Error Detect Register (PAMU8_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_7070	ECC Error Attributes Register (PAMU8_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_7074	ECC Error Address High (PAMU8_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_7078	ECC Error Address Low (PAMU8_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_707C	ECC Error Data High (PAMU8_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_7080	ECC Error Data Low (PAMU8_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_7090	Unauthorized device access detection register (PAMU8_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_7BF8	PAMU Revision register 1 (PAMU8_PR1)	32	R	0901_0100h	10.4.32/449
2_7BFC	PAMU Revision register 2 (PAMU8_PR2)	32	R	0000_0000h	10.4.33/450
2_7C00	PAMU Capabilities register 1 (PAMU8_PC1)	32	R	0000_000Fh	10.4.34/450
2_7C04	PAMU Capabilities register 2 (PAMU8_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_7C08	PAMU Capabilities register 3 (PAMU8_PC3)	32	R	007E_1E00h	10.4.36/452
2_7C0C	PAMU Capabilities register 4 (PAMU8_PC4)	32	R	0000_0CE1h	10.4.37/454
2_7C10	PAMU Control register (PAMU8_PC)	32	R/W	0000_1111h	10.4.38/455
2_7C1C	PAMU Interrupt Control and Status register (PAMU8_PICS)	32	R/W	0000_0000h	10.4.39/456
2_8000	Primary PAACT Base Address High register (PAMU9_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_8004	Primary PAACT Base Address Low register (PAMU9_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_8008	Primary PAACT Limit Address High register (PAMU9_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_800C	Primary PAACT Limit Address Low register (PAMU9_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_8010	Secondary PAACT Base Address High register (PAMU9_SPBAH)	32	R/W	0000_0000h	10.4.5/432

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_8014	Secondary PAACT Base Address Low register (PAMU9_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_8018	Secondary PAACT Limit Address High register (PAMU9_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_801C	Secondary PAACT Limit Address Low register (PAMU9_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_8020	OMT Base Address High register (PAMU9_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_8024	OMT Base Address Low register (PAMU9_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_8028	OMT Limit Address High register (PAMU9_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_802C	OMT Limit Address Low register (PAMU9_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_8030	PAMU Address Capabilities Register 1 (PAMU9_PAC1)	32	R	0000_000Fh	10.4.13/436
2_8034	PAMU Address Capabilities Register 2 (PAMU9_PAC2)	32	R	0000_00FFh	10.4.14/436
2_8040	PAMU Operation Error Status register 1 (PAMU9_POES1)	32	R/W	0000_0000h	10.4.15/437
2_8044	PAMU Operation Error Status register 2 (PAMU9_POES2)	32	R/W	0000_0000h	10.4.16/438
2_8048	PAMU Operation Error Address High register (PAMU9_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_804C	PAMU Operation Error Address Low register (PAMU9_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_8050	Access Violation Status register 1 (PAMU9_AVS1)	32	R/W	0000_0000h	10.4.19/440
2_8054	Access Violation Status register 2 (PAMU9_AVS2)	32	R/W	0000_0000h	10.4.20/442
2_8058	Access Violation Address High register (PAMU9_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_805C	Access Violation Address Low register (PAMU9_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_8060	ECC Error Control Register (PAMU9_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_8068	ECC Error Interrupt Enable Register (PAMU9_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_806C	ECC Error Detect Register (PAMU9_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_8070	ECC Error Attributes Register (PAMU9_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_8074	ECC Error Address High (PAMU9_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_8078	ECC Error Address Low (PAMU9_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_807C	ECC Error Data High (PAMU9_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_8080	ECC Error Data Low (PAMU9_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_8090	Unauthorized device access detection register (PAMU9_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_8BF8	PAMU Revision register 1 (PAMU9_PR1)	32	R	0901_0100h	10.4.32/449
2_8BFC	PAMU Revision register 2 (PAMU9_PR2)	32	R	0000_0000h	10.4.33/450
2_8C00	PAMU Capabilities register 1 (PAMU9_PC1)	32	R	0000_000Fh	10.4.34/450
2_8C04	PAMU Capabilities register 2 (PAMU9_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_8C08	PAMU Capabilities register 3 (PAMU9_PC3)	32	R	007E_1E00h	10.4.36/452
2_8C0C	PAMU Capabilities register 4 (PAMU9_PC4)	32	R	0000_0CE1h	10.4.37/454
2_8C10	PAMU Control register (PAMU9_PC)	32	R/W	0000_1111h	10.4.38/455
2_8C1C	PAMU Interrupt Control and Status register (PAMU9_PICS)	32	R/W	0000_0000h	10.4.39/456

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
2_9000	Primary PAACT Base Address High register (PAMU10_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_9004	Primary PAACT Base Address Low register (PAMU10_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_9008	Primary PAACT Limit Address High register (PAMU10_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_900C	Primary PAACT Limit Address Low register (PAMU10_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_9010	Secondary PAACT Base Address High register (PAMU10_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_9014	Secondary PAACT Base Address Low register (PAMU10_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_9018	Secondary PAACT Limit Address High register (PAMU10_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_901C	Secondary PAACT Limit Address Low register (PAMU10_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_9020	OMT Base Address High register (PAMU10_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_9024	OMT Base Address Low register (PAMU10_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_9028	OMT Limit Address High register (PAMU10_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_902C	OMT Limit Address Low register (PAMU10_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_9030	PAMU Address Capabilities Register 1 (PAMU10_PAC1)	32	R	0000_000Fh	10.4.13/436
2_9034	PAMU Address Capabilities Register 2 (PAMU10_PAC2)	32	R	0000_00FFh	10.4.14/436
2_9040	PAMU Operation Error Status register 1 (PAMU10_POES1)	32	R/W	0000_0000h	10.4.15/437
2_9044	PAMU Operation Error Status register 2 (PAMU10_POES2)	32	R/W	0000_0000h	10.4.16/438
2_9048	PAMU Operation Error Address High register (PAMU10_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_904C	PAMU Operation Error Address Low register (PAMU10_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_9050	Access Violation Status register 1 (PAMU10_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_9054	Access Violation Status register 2 (PAMU10_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_9058	Access Violation Address High register (PAMU10_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_905C	Access Violation Address Low register (PAMU10_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_9060	ECC Error Control Register (PAMU10_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_9068	ECC Error Interrupt Enable Register (PAMU10_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_906C	ECC Error Detect Register (PAMU10_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_9070	ECC Error Attributes Register (PAMU10_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_9074	ECC Error Address High (PAMU10_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_9078	ECC Error Address Low (PAMU10_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_907C	ECC Error Data High (PAMU10_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_9080	ECC Error Data Low (PAMU10_EEDLO)	32	R/W	0000_0000h	10.4.30/448

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_9090	Unauthorized device access detection register (PAMU10_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_9BF8	PAMU Revision register 1 (PAMU10_PR1)	32	R	0901_0100h	10.4.32/449
2_9BFC	PAMU Revision register 2 (PAMU10_PR2)	32	R	0000_0000h	10.4.33/450
2_9C00	PAMU Capabilities register 1 (PAMU10_PC1)	32	R	0000_000Fh	10.4.34/450
2_9C04	PAMU Capabilities register 2 (PAMU10_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_9C08	PAMU Capabilities register 3 (PAMU10_PC3)	32	R	007E_1E00h	10.4.36/452
2_9C0C	PAMU Capabilities register 4 (PAMU10_PC4)	32	R	0000_0CE1h	10.4.37/454
2_9C10	PAMU Control register (PAMU10_PC)	32	R/W	0000_1111h	10.4.38/455
2_9C1C	PAMU Interrupt Control and Status register (PAMU10_PICS)	32	R/W	0000_0000h	10.4.39/456
2_A000	Primary PAACT Base Address High register (PAMU11_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_A004	Primary PAACT Base Address Low register (PAMU11_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_A008	Primary PAACT Limit Address High register (PAMU11_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_A00C	Primary PAACT Limit Address Low register (PAMU11_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_A010	Secondary PAACT Base Address High register (PAMU11_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_A014	Secondary PAACT Base Address Low register (PAMU11_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_A018	Secondary PAACT Limit Address High register (PAMU11_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_A01C	Secondary PAACT Limit Address Low register (PAMU11_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_A020	OMT Base Address High register (PAMU11_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_A024	OMT Base Address Low register (PAMU11_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_A028	OMT Limit Address High register (PAMU11_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_A02C	OMT Limit Address Low register (PAMU11_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_A030	PAMU Address Capabilities Register 1 (PAMU11_PAC1)	32	R	0000_000Fh	10.4.13/436
2_A034	PAMU Address Capabilities Register 2 (PAMU11_PAC2)	32	R	0000_00FFh	10.4.14/436
2_A040	PAMU Operation Error Status register 1 (PAMU11_POES1)	32	R/W	0000_0000h	10.4.15/437
2_A044	PAMU Operation Error Status register 2 (PAMU11_POES2)	32	R/W	0000_0000h	10.4.16/438
2_A048	PAMU Operation Error Address High register (PAMU11_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_A04C	PAMU Operation Error Address Low register (PAMU11_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_A050	Access Violation Status register 1 (PAMU11_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_A054	Access Violation Status register 2 (PAMU11_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_A058	Access Violation Address High register (PAMU11_AVAH)	32	R/W	0000_0000h	10.4.21/443

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_A05C	Access Violation Address Low register (PAMU11_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_A060	ECC Error Control Register (PAMU11_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_A068	ECC Error Interrupt Enable Register (PAMU11_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_A06C	ECC Error Detect Register (PAMU11_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_A070	ECC Error Attributes Register (PAMU11_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_A074	ECC Error Address High (PAMU11_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_A078	ECC Error Address Low (PAMU11_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_A07C	ECC Error Data High (PAMU11_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_A080	ECC Error Data Low (PAMU11_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_A090	Unauthorized device access detection register (PAMU11_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_ABF8	PAMU Revision register 1 (PAMU11_PR1)	32	R	0901_0100h	10.4.32/449
2_ABFC	PAMU Revision register 2 (PAMU11_PR2)	32	R	0000_0000h	10.4.33/450
2_AC00	PAMU Capabilities register 1 (PAMU11_PC1)	32	R	0000_000Fh	10.4.34/450
2_AC04	PAMU Capabilities register 2 (PAMU11_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_AC08	PAMU Capabilities register 3 (PAMU11_PC3)	32	R	007E_1E00h	10.4.36/452
2_AC0C	PAMU Capabilities register 4 (PAMU11_PC4)	32	R	0000_0CE1h	10.4.37/454
2_AC10	PAMU Control register (PAMU11_PC)	32	R/W	0000_1111h	10.4.38/455
2_AC1C	PAMU Interrupt Control and Status register (PAMU11_PICS)	32	R/W	0000_0000h	10.4.39/456
2_B000	Primary PAACT Base Address High register (PAMU12_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_B004	Primary PAACT Base Address Low register (PAMU12_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_B008	Primary PAACT Limit Address High register (PAMU12_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_B00C	Primary PAACT Limit Address Low register (PAMU12_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_B010	Secondary PAACT Base Address High register (PAMU12_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_B014	Secondary PAACT Base Address Low register (PAMU12_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_B018	Secondary PAACT Limit Address High register (PAMU12_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_B01C	Secondary PAACT Limit Address Low register (PAMU12_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_B020	OMT Base Address High register (PAMU12_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_B024	OMT Base Address Low register (PAMU12_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_B028	OMT Limit Address High register (PAMU12_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_B02C	OMT Limit Address Low register (PAMU12_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_B030	PAMU Address Capabilities Register 1 (PAMU12_PAC1)	32	R	0000_000Fh	10.4.13/436

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_B034	PAMU Address Capabilities Register 2 (PAMU12_PAC2)	32	R	0000_00FFh	10.4.14/436
2_B040	PAMU Operation Error Status register 1 (PAMU12_POES1)	32	R/W	0000_0000h	10.4.15/437
2_B044	PAMU Operation Error Status register 2 (PAMU12_POES2)	32	R/W	0000_0000h	10.4.16/438
2_B048	PAMU Operation Error Address High register (PAMU12_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_B04C	PAMU Operation Error Address Low register (PAMU12_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_B050	Access Violation Status register 1 (PAMU12_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_B054	Access Violation Status register 2 (PAMU12_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_B058	Access Violation Address High register (PAMU12_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_B05C	Access Violation Address Low register (PAMU12_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_B060	ECC Error Control Register (PAMU12_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_B068	ECC Error Interrupt Enable Register (PAMU12_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_B06C	ECC Error Detect Register (PAMU12_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_B070	ECC Error Attributes Register (PAMU12_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_B074	ECC Error Address High (PAMU12_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_B078	ECC Error Address Low (PAMU12_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_B07C	ECC Error Data High (PAMU12_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_B080	ECC Error Data Low (PAMU12_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_B090	Unauthorized device access detection register (PAMU12_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_BBF8	PAMU Revision register 1 (PAMU12_PR1)	32	R	0901_0100h	10.4.32/449
2_BBFC	PAMU Revision register 2 (PAMU12_PR2)	32	R	0000_0000h	10.4.33/450
2_BC00	PAMU Capabilities register 1 (PAMU12_PC1)	32	R	0000_000Fh	10.4.34/450
2_BC04	PAMU Capabilities register 2 (PAMU12_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_BC08	PAMU Capabilities register 3 (PAMU12_PC3)	32	R	007E_1E00h	10.4.36/452
2_BC0C	PAMU Capabilities register 4 (PAMU12_PC4)	32	R	0000_0CE1h	10.4.37/454
2_BC10	PAMU Control register (PAMU12_PC)	32	R/W	0000_1111h	10.4.38/455
2_BC1C	PAMU Interrupt Control and Status register (PAMU12_PICS)	32	R/W	0000_0000h	10.4.39/456
2_C000	Primary PAACT Base Address High register (PAMU13_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_C004	Primary PAACT Base Address Low register (PAMU13_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_C008	Primary PAACT Limit Address High register (PAMU13_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_C00C	Primary PAACT Limit Address Low register (PAMU13_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_C010	Secondary PAACT Base Address High register (PAMU13_SPBAH)	32	R/W	0000_0000h	10.4.5/432

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_C014	Secondary PAACT Base Address Low register (PAMU13_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_C018	Secondary PAACT Limit Address High register (PAMU13_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_C01C	Secondary PAACT Limit Address Low register (PAMU13_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_C020	OMT Base Address High register (PAMU13_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_C024	OMT Base Address Low register (PAMU13_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_C028	OMT Limit Address High register (PAMU13_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_C02C	OMT Limit Address Low register (PAMU13_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_C030	PAMU Address Capabilities Register 1 (PAMU13_PAC1)	32	R	0000_000Fh	10.4.13/436
2_C034	PAMU Address Capabilities Register 2 (PAMU13_PAC2)	32	R	0000_00FFh	10.4.14/436
2_C040	PAMU Operation Error Status register 1 (PAMU13_POES1)	32	R/W	0000_0000h	10.4.15/437
2_C044	PAMU Operation Error Status register 2 (PAMU13_POES2)	32	R/W	0000_0000h	10.4.16/438
2_C048	PAMU Operation Error Address High register (PAMU13_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_C04C	PAMU Operation Error Address Low register (PAMU13_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_C050	Access Violation Status register 1 (PAMU13_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_C054	Access Violation Status register 2 (PAMU13_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_C058	Access Violation Address High register (PAMU13_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_C05C	Access Violation Address Low register (PAMU13_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_C060	ECC Error Control Register (PAMU13_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_C068	ECC Error Interrupt Enable Register (PAMU13_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_C06C	ECC Error Detect Register (PAMU13_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_C070	ECC Error Attributes Register (PAMU13_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_C074	ECC Error Address High (PAMU13_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_C078	ECC Error Address Low (PAMU13_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_C07C	ECC Error Data High (PAMU13_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_C080	ECC Error Data Low (PAMU13_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_C090	Unauthorized device access detection register (PAMU13_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_CBF8	PAMU Revision register 1 (PAMU13_PR1)	32	R	0901_0100h	10.4.32/449
2_CBFC	PAMU Revision register 2 (PAMU13_PR2)	32	R	0000_0000h	10.4.33/450
2_CC00	PAMU Capabilities register 1 (PAMU13_PC1)	32	R	0000_000Fh	10.4.34/450
2_CC04	PAMU Capabilities register 2 (PAMU13_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_CC08	PAMU Capabilities register 3 (PAMU13_PC3)	32	R	007E_1E00h	10.4.36/452
2_CC0C	PAMU Capabilities register 4 (PAMU13_PC4)	32	R	0000_0CE1h	10.4.37/454
2_CC10	PAMU Control register (PAMU13_PC)	32	R/W	0000_1111h	10.4.38/455
2_CC1C	PAMU Interrupt Control and Status register (PAMU13_PICS)	32	R/W	0000_0000h	10.4.39/456

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_D000	Primary PAACT Base Address High register (PAMU14_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_D004	Primary PAACT Base Address Low register (PAMU14_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_D008	Primary PAACT Limit Address High register (PAMU14_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_D00C	Primary PAACT Limit Address Low register (PAMU14_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_D010	Secondary PAACT Base Address High register (PAMU14_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_D014	Secondary PAACT Base Address Low register (PAMU14_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_D018	Secondary PAACT Limit Address High register (PAMU14_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_D01C	Secondary PAACT Limit Address Low register (PAMU14_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_D020	OMT Base Address High register (PAMU14_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_D024	OMT Base Address Low register (PAMU14_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_D028	OMT Limit Address High register (PAMU14_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_D02C	OMT Limit Address Low register (PAMU14_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_D030	PAMU Address Capabilities Register 1 (PAMU14_PAC1)	32	R	0000_000Fh	10.4.13/436
2_D034	PAMU Address Capabilities Register 2 (PAMU14_PAC2)	32	R	0000_00FFh	10.4.14/436
2_D040	PAMU Operation Error Status register 1 (PAMU14_POES1)	32	R/W	0000_0000h	10.4.15/437
2_D044	PAMU Operation Error Status register 2 (PAMU14_POES2)	32	R/W	0000_0000h	10.4.16/438
2_D048	PAMU Operation Error Address High register (PAMU14_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_D04C	PAMU Operation Error Address Low register (PAMU14_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_D050	Access Violation Status register 1 (PAMU14_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_D054	Access Violation Status register 2 (PAMU14_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_D058	Access Violation Address High register (PAMU14_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_D05C	Access Violation Address Low register (PAMU14_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_D060	ECC Error Control Register (PAMU14_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_D068	ECC Error Interrupt Enable Register (PAMU14_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_D06C	ECC Error Detect Register (PAMU14_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_D070	ECC Error Attributes Register (PAMU14_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_D074	ECC Error Address High (PAMU14_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_D078	ECC Error Address Low (PAMU14_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_D07C	ECC Error Data High (PAMU14_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_D080	ECC Error Data Low (PAMU14_EEDLO)	32	R/W	0000_0000h	10.4.30/448

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_D090	Unauthorized device access detection register (PAMU14_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_DBF8	PAMU Revision register 1 (PAMU14_PR1)	32	R	0901_0100h	10.4.32/449
2_DBFC	PAMU Revision register 2 (PAMU14_PR2)	32	R	0000_0000h	10.4.33/450
2_DC00	PAMU Capabilities register 1 (PAMU14_PC1)	32	R	0000_000Fh	10.4.34/450
2_DC04	PAMU Capabilities register 2 (PAMU14_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_DC08	PAMU Capabilities register 3 (PAMU14_PC3)	32	R	007E_1E00h	10.4.36/452
2_DC0C	PAMU Capabilities register 4 (PAMU14_PC4)	32	R	0000_0CE1h	10.4.37/454
2_DC10	PAMU Control register (PAMU14_PC)	32	R/W	0000_1111h	10.4.38/455
2_DC1C	PAMU Interrupt Control and Status register (PAMU14_PICS)	32	R/W	0000_0000h	10.4.39/456
2_E000	Primary PAACT Base Address High register (PAMU15_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_E004	Primary PAACT Base Address Low register (PAMU15_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_E008	Primary PAACT Limit Address High register (PAMU15_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_E00C	Primary PAACT Limit Address Low register (PAMU15_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_E010	Secondary PAACT Base Address High register (PAMU15_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_E014	Secondary PAACT Base Address Low register (PAMU15_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_E018	Secondary PAACT Limit Address High register (PAMU15_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_E01C	Secondary PAACT Limit Address Low register (PAMU15 SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_E020	OMT Base Address High register (PAMU15_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_E024	OMT Base Address Low register (PAMU15_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_E028	OMT Limit Address High register (PAMU15_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_E02C	OMT Limit Address Low register (PAMU15_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_E030	PAMU Address Capabilities Register 1 (PAMU15_PAC1)	32	R	0000_000Fh	10.4.13/436
2_E034	PAMU Address Capabilities Register 2 (PAMU15_PAC2)	32	R	0000_00FFh	10.4.14/436
2_E040	PAMU Operation Error Status register 1 (PAMU15_POES1)	32	R/W	0000_0000h	10.4.15/437
2_E044	PAMU Operation Error Status register 2 (PAMU15_POES2)	32	R/W	0000_0000h	10.4.16/438
2_E048	PAMU Operation Error Address High register (PAMU15_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_E04C	PAMU Operation Error Address Low register (PAMU15_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_E050	Access Violation Status register 1 (PAMU15_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_E054	Access Violation Status register 2 (PAMU15_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_E058	Access Violation Address High register (PAMU15_AVAH)	32	R/W	0000_0000h	10.4.21/443

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_E05C	Access Violation Address Low register (PAMU15_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_E060	ECC Error Control Register (PAMU15_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_E068	ECC Error Interrupt Enable Register (PAMU15_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_E06C	ECC Error Detect Register (PAMU15_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_E070	ECC Error Attributes Register (PAMU15_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_E074	ECC Error Address High (PAMU15_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_E078	ECC Error Address Low (PAMU15_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_E07C	ECC Error Data High (PAMU15_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_E080	ECC Error Data Low (PAMU15_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_E090	Unauthorized device access detection register (PAMU15_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_EBF8	PAMU Revision register 1 (PAMU15_PR1)	32	R	0901_0100h	10.4.32/449
2_EBFC	PAMU Revision register 2 (PAMU15_PR2)	32	R	0000_0000h	10.4.33/450
2_EC00	PAMU Capabilities register 1 (PAMU15_PC1)	32	R	0000_000Fh	10.4.34/450
2_EC04	PAMU Capabilities register 2 (PAMU15_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_EC08	PAMU Capabilities register 3 (PAMU15_PC3)	32	R	007E_1E00h	10.4.36/452
2_EC0C	PAMU Capabilities register 4 (PAMU15_PC4)	32	R	0000_0CE1h	10.4.37/454
2_EC10	PAMU Control register (PAMU15_PC)	32	R/W	0000_1111h	10.4.38/455
2_EC1C	PAMU Interrupt Control and Status register (PAMU15_PICS)	32	R/W	0000_0000h	10.4.39/456
2_F000	Primary PAACT Base Address High register (PAMU16_PPBAH)	32	R/W	0000_0000h	10.4.1/430
2_F004	Primary PAACT Base Address Low register (PAMU16_PPBAL)	32	R/W	0000_0000h	10.4.2/431
2_F008	Primary PAACT Limit Address High register (PAMU16_PPLAH)	32	R/W	0000_0000h	10.4.3/431
2_F00C	Primary PAACT Limit Address Low register (PAMU16_PPLAL)	32	R/W	0000_0000h	10.4.4/432
2_F010	Secondary PAACT Base Address High register (PAMU16_SPBAH)	32	R/W	0000_0000h	10.4.5/432
2_F014	Secondary PAACT Base Address Low register (PAMU16_SPBAL)	32	R/W	0000_0000h	10.4.6/433
2_F018	Secondary PAACT Limit Address High register (PAMU16_SPLAH)	32	R/W	0000_0000h	10.4.7/433
2_F01C	Secondary PAACT Limit Address Low register (PAMU16_SPLAL)	32	R/W	0000_0000h	10.4.8/433
2_F020	OMT Base Address High register (PAMU16_OBAH)	32	R/W	0000_0000h	10.4.9/434
2_F024	OMT Base Address Low register (PAMU16_OBAL)	32	R/W	0000_0000h	10.4.10/434
2_F028	OMT Limit Address High register (PAMU16_OLAH)	32	R/W	0000_0000h	10.4.11/435
2_F02C	OMT Limit Address Low register (PAMU16_OLAL)	32	R/W	0000_0000h	10.4.12/435
2_F030	PAMU Address Capabilities Register 1 (PAMU16_PAC1)	32	R	0000_000Fh	10.4.13/436

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_F034	PAMU Address Capabilities Register 2 (PAMU16_PAC2)	32	R	0000_00FFh	10.4.14/436
2_F040	PAMU Operation Error Status register 1 (PAMU16_POES1)	32	R/W	0000_0000h	10.4.15/437
2_F044	PAMU Operation Error Status register 2 (PAMU16_POES2)	32	R/W	0000_0000h	10.4.16/438
2_F048	PAMU Operation Error Address High register (PAMU16_POEAH)	32	R/W	0000_0000h	10.4.17/438
2_F04C	PAMU Operation Error Address Low register (PAMU16_POEAL)	32	R/W	0000_0000h	10.4.18/439
2_F050	Access Violation Status register 1 (PAMU16_AVIS1)	32	R/W	0000_0000h	10.4.19/440
2_F054	Access Violation Status register 2 (PAMU16_AVIS2)	32	R/W	0000_0000h	10.4.20/442
2_F058	Access Violation Address High register (PAMU16_AVAH)	32	R/W	0000_0000h	10.4.21/443
2_F05C	Access Violation Address Low register (PAMU16_AVAL)	32	R/W	0000_0000h	10.4.22/443
2_F060	ECC Error Control Register (PAMU16_EECTL)	32	R/W	0000_0000h	10.4.23/443
2_F068	ECC Error Interrupt Enable Register (PAMU16_EEINTEN)	32	R/W	0000_0000h	10.4.24/444
2_F06C	ECC Error Detect Register (PAMU16_EEDET)	32	w1c	0000_0000h	10.4.25/445
2_F070	ECC Error Attributes Register (PAMU16_EEATTR)	32	R/W	0000_0000h	10.4.26/446
2_F074	ECC Error Address High (PAMU16_EEAHI)	32	R/W	0000_0000h	10.4.27/447
2_F078	ECC Error Address Low (PAMU16_EEALO)	32	R/W	0000_0000h	10.4.28/447
2_F07C	ECC Error Data High (PAMU16_EEDHI)	32	R/W	0000_0000h	10.4.29/448
2_F080	ECC Error Data Low (PAMU16_EEDLO)	32	R/W	0000_0000h	10.4.30/448
2_F090	Unauthorized device access detection register (PAMU16_UDAD)	32	R/W	0000_0000h	10.4.31/449
2_FBF8	PAMU Revision register 1 (PAMU16_PR1)	32	R	0901_0100h	10.4.32/449
2_FBFC	PAMU Revision register 2 (PAMU16_PR2)	32	R	0000_0000h	10.4.33/450
2_FC00	PAMU Capabilities register 1 (PAMU16_PC1)	32	R	0000_000Fh	10.4.34/450
2_FC04	PAMU Capabilities register 2 (PAMU16_PC2)	32	R	0FFF_00FFh	10.4.35/451
2_FC08	PAMU Capabilities register 3 (PAMU16_PC3)	32	R	007E_1E00h	10.4.36/452
2_FC0C	PAMU Capabilities register 4 (PAMU16_PC4)	32	R	0000_0CE1h	10.4.37/454
2_FC10	PAMU Control register (PAMU16_PC)	32	R/W	0000_1111h	10.4.38/455
2_FC1C	PAMU Interrupt Control and Status register (PAMU16_PICS)	32	R/W	0000_0000h	10.4.39/456

10.4.1 Primary PAACT Base Address High register (PAMUx_PPBAH)

A write to the primary PAACT Base Address High register invalidates the contents of Primary PAACT cache.

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

PAMUx_PPBAH field descriptions

Field	Description
0–31 PPBAH	<p>Primary PAACT Base Address High</p> <p>The PPBAH is the 32 most significant bits of the 64-bit base address of the primary PAACT.</p> <p>NOTE: The most significant bits of PPBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).</p>

10.4.2 Primary PAACT Base Address Low register (PAMUx_PPBAL)

A write to the primary PAACT Base Address Low register invalidates the contents of Primary PAACT cache.

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

Reset 0

PAMUx_PPBAL field descriptions

Field	Description
0–19 PPBAL	<p>Primary PAACT Base Address Low</p> <p>The PPBAL is the 20 next-significant of the 64 bits that identify the primary PAACT base address.</p>
20–31 -	<p>This field is reserved.</p> <p>Reserved. These bits are the 12 least significant base address bits of the primary PAACT base address which must be aligned to a 4-Kbyte page boundary.</p>

10.4.3 Primary PAACT Limit Address High register (PAMUx_PPLAH)

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

Reset 0

PAMUx_PPLAH field descriptions

Field	Description
0–31 PPLAH	<p>Primary PAACT Limit Address High</p> <p>The PPLAH is the 32 most significant of the 64 address bits that identify the limit of the primary PAACT.</p> <p>NOTE: The most significant bits of PPLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).</p>

10.4.4 Primary PAACT Limit Address Low register (PAMUx_PPLAL)

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

Reset 0

PAMUx_PPLAL field descriptions

Field	Description
0–25 PPLAL	Primary PAACT Limit Address Low The PPLAL is the 26 next-significant of the 64 bits that identify the limit of the primary PAACT. NOTE: The primary PAACT limit is the address of the byte following the last byte of the PPAACT.
26–31 -	This field is reserved. Reserved. These bits are the 6 least significant address bits representing the 64-byte PAACE size.

10.4.5 Secondary PAACT Base Address High register (PAMUx_SPBAH)

A write to the secondary PAACT Base Address High register invalidates the contents of secondary PAACT cache.

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

Reset 0

PAMUx_SPBAH field descriptions

Field	Description
0–31 SPBAH	Secondary PAACT Base Address High The SPBAH is the 32 most-significant of the 64 bits that identify the secondary PAACT base address. NOTE: The most significant bits of SPBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1) .

10.4.6 Secondary PAACT Base Address Low register (PAMUx_SPBAL)

A write to the secondary PAACT Base Address Low register invalidates the contents of secondary PAACT cache.

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

PAMUx_SPBAL field descriptions

Field	Description
0–19 SPBAL	Secondary PAACT Base Address Low The SPBAL is the 20 next-significant of the 64 bits that identify the secondary PAACT base address.
20–31 -	This field is reserved. Reserved. These bits are the 12 least significant base address bits of the secondary PAACT which must be aligned to a 4-Kbyte page boundary.

10.4.7 Secondary PAACT Limit Address High register (PAMUx_SPLAH)

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

PAMUx_SPLAH field descriptions

Field	Description
0–31 SPLAH	Secondary PAACT Limit Address High The SPLAH is the 32 most-significant of the 64 bits that identify the limit of the secondary PAACT. NOTE: The most significant bits of SPLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1) .

10.4.8 Secondary PAACT Limit Address Low register (PAMUx_SPLAL)

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

PAMUx_SPLAL field descriptions

Field	Description
0–25 SPLAL	Secondary PAACT Limit Address Low The SPLAL is the 26 next-significant of the 64 bits that identify the limit of the secondary PAACT. NOTE: The secondary PAACT limit is the address of the byte following the last byte of the SPAACT.
26–31 -	This field is reserved. Reserved. These bits are the 6 least significant address bits representing the 64-byte PAACE size.

10.4.9 OMT Base Address High register (PAMUx_OBAH)

A write to OMT Base Address High register invalidates the contents of OMT cache.

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

PAMUx_OBAH field descriptions

Field	Description
0–31 OBAH	OMT Base Address High The OBAH is the 32 most significant address bits that identify the base of the OMT window. NOTE: The most significant bits of OBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities 1 .

10.4.10 OMT Base Address Low register (PAMUx_OBAL)

A write to OMT Base Address Low register invalidates the contents of OMT cache.

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | Reserved

PAMUx_OBAL field descriptions

Field	Description
0–19 OBAL	OMT Base Address Low The OBAL is the next 20 least significant address bits that identify the base of the OMT window above the 12 least significant address bits representing the minimum 4-Kbyte page boundary for the base address.

Table continues on the next page...

PAMUX_OBAL field descriptions (continued)

Field	Description
20-31 -	This field is reserved. Reserved. These bits are the 12 least significant address bits a 4-Kbyte page boundary.

10.4.11 OMT Limit Address High register (PAMUx_OLAH)

PAMUx OLAH field descriptions

Field	Description
0-31 OLAH	<p>OMT Limit Address High</p> <p>The OLAH is the 32 most significant address bits that identify the limit of the OMT window.</p> <p>NOTE: The most significant bits of OLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities 1.</p>

10.4.12 OMT Limit Address Low register (PAMUx_OLAL)

PAMUx OLAL field descriptions

Field	Description
0–24 OLAL	<p>OMT Limit Address Low</p> <p>The OLAL is the next 25 least significant address bits that identify the limit of the OMT window above the 7 least significant address bits representing the 128-byte size of an OME.</p>
25–31 -	<p>This field is reserved.</p> <p>Reserved. These bits are the 7 least significant address bits representing the 128-byte size of an OME.</p>

10.4.13 PAMU Address Capabilities Register 1 (PAMUx_PAC1)

The PAMU address capabilities registers denote implementation parameters for a given hardware instance of the PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

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

PAMUx_PAC1 field descriptions

Field	Description
0–31 MSA	<p>Maximum System Storage Address Capability. These bits define the maximum system storage address capability of this PAMU implementation. The MSA identifies the maximum value of the 32 most-significant address bits above the 32 least-significant address bits representing 4 Gbytes. This PAMU implements only the least significant MSA number of bits of the TWBA field within a PAACE.</p> <p>NOTE: These bits do not define the maximum address capability of the system. A PAMU implementation can be used in a system where system address capability is less than or equal to maximum system-storage address capability of the PAMU.</p> <p>0x0000_000F Maximum 36-bit System Address Space</p>

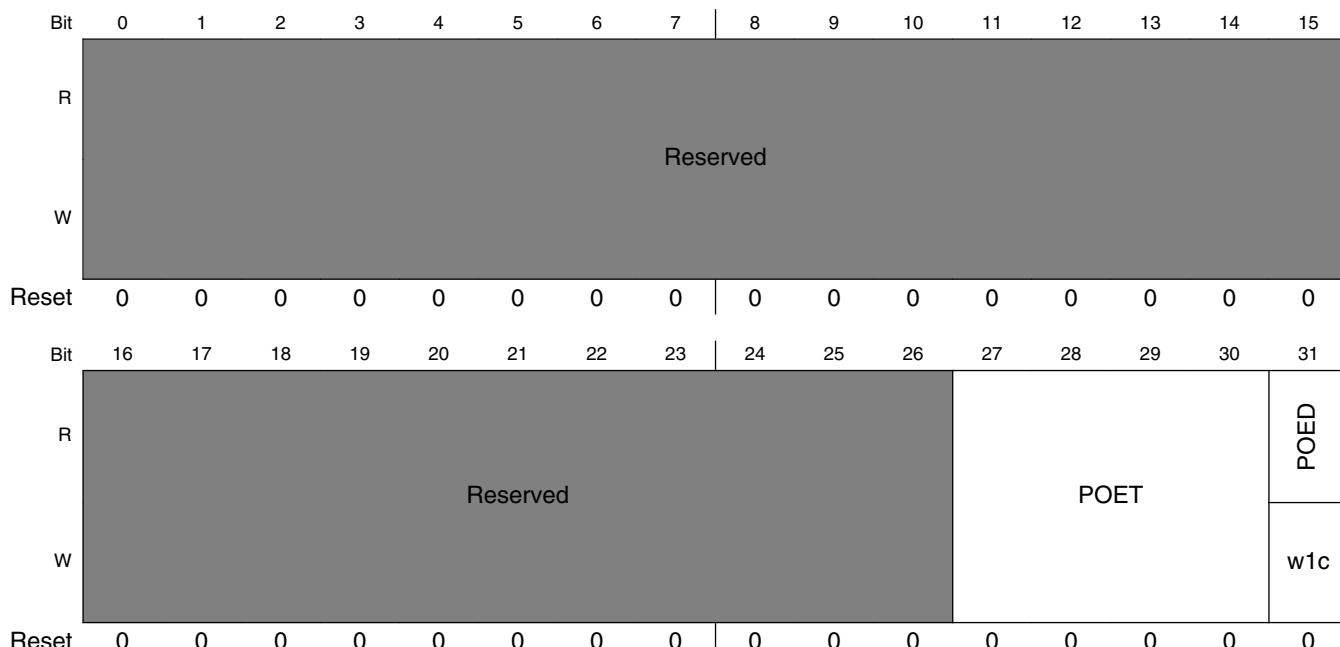
10.4.14 PAMU Address Capabilities Register 2 (PAMUx_PAC2)

The PAMU address capabilities registers denote implementation parameters for a given hardware instance of the PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

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

PAMUx_PAC2 field descriptions

Field	Description
0–31 MIA	<p>Maximum I/O Address Capability. These bits define the maximum I/O bus address capability of this PAMU implementation. The MIA identifies the maximum value of the 32 most-significant address bits above the 32 least-significant address bits representing 4 Gbytes. This PAMU implements only the least significant MIA number of bits of the WBA field within a PAACE.</p> <p>NOTE: In no-address-translation mode, the I/O bus address of the DSA operation is also the system storage address. Since there is no address translation, the DSA Window must be contained within the system storage space.</p> <p>0x0000_00FF Maximum 40-bit I/O bus address space</p>

10.4.15 PAMU Operation Error Status register 1 (PAMUx_POES1)**PAMUx_POES1 field descriptions**

Field	Description										
0–26 -	This field is reserved. Reserved										
27–30 POET	<p>PAMU Operation Error Type</p> <p>These bits identify the type of error encountered by the PAMU. The bits are coded as follows:</p> <table> <tr> <td>0000</td> <td>PAMU fetch: bad data received</td> </tr> <tr> <td>0001-0111</td> <td>Reserved</td> </tr> <tr> <td>1000</td> <td>Unsupported ATM code requested in PAACE</td> </tr> <tr> <td>1001</td> <td>Unsupported OTM code requested in PAACE</td> </tr> <tr> <td>1010-1111</td> <td>Reserved</td> </tr> </table>	0000	PAMU fetch: bad data received	0001-0111	Reserved	1000	Unsupported ATM code requested in PAACE	1001	Unsupported OTM code requested in PAACE	1010-1111	Reserved
0000	PAMU fetch: bad data received										
0001-0111	Reserved										
1000	Unsupported ATM code requested in PAACE										
1001	Unsupported OTM code requested in PAACE										
1010-1111	Reserved										

Table continues on the next page...

PAMUx_POES1 field descriptions (continued)

Field	Description				
31 POED	<p>PAMU Operation Error Detected</p> <p>This bit is set when a hardware error is encountered by PAMU.</p> <p>If the bit is set, the rest of the contents of the Error Status register and the contents of the Error Status and Error Address registers are valid. It is coded as follows:</p> <table> <tr> <td>0</td> <td>No PAMU error</td> </tr> <tr> <td>1</td> <td>PAMU error detected. Software should write a 1 to clear this bit.</td> </tr> </table>	0	No PAMU error	1	PAMU error detected. Software should write a 1 to clear this bit.
0	No PAMU error				
1	PAMU error detected. Software should write a 1 to clear this bit.				

10.4.16 PAMU Operation Error Status register 2 (PAMUx_POES2)

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

Reset 0

PAMUx_POES2 field descriptions

Field	Description
0–31 -	This field is reserved. Reserved

10.4.17 PAMU Operation Error Address High register (PAMUx_POEAH)

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

Reset 0

PAMUx_POEAH field descriptions

Field	Description
0–31 POEAH	<p>PAMU Operation Error Address High</p> <p>The POEAH is the 32 most significant address bits that identify the address, in system memory space, of the PAACT, OMT location that encountered the error condition.</p> <p>NOTE: POEAH[0:27] has a value of 0x000_0000 and is consistent with the MSA capability field as described in PAMU Address Capabilities Register 1.</p>

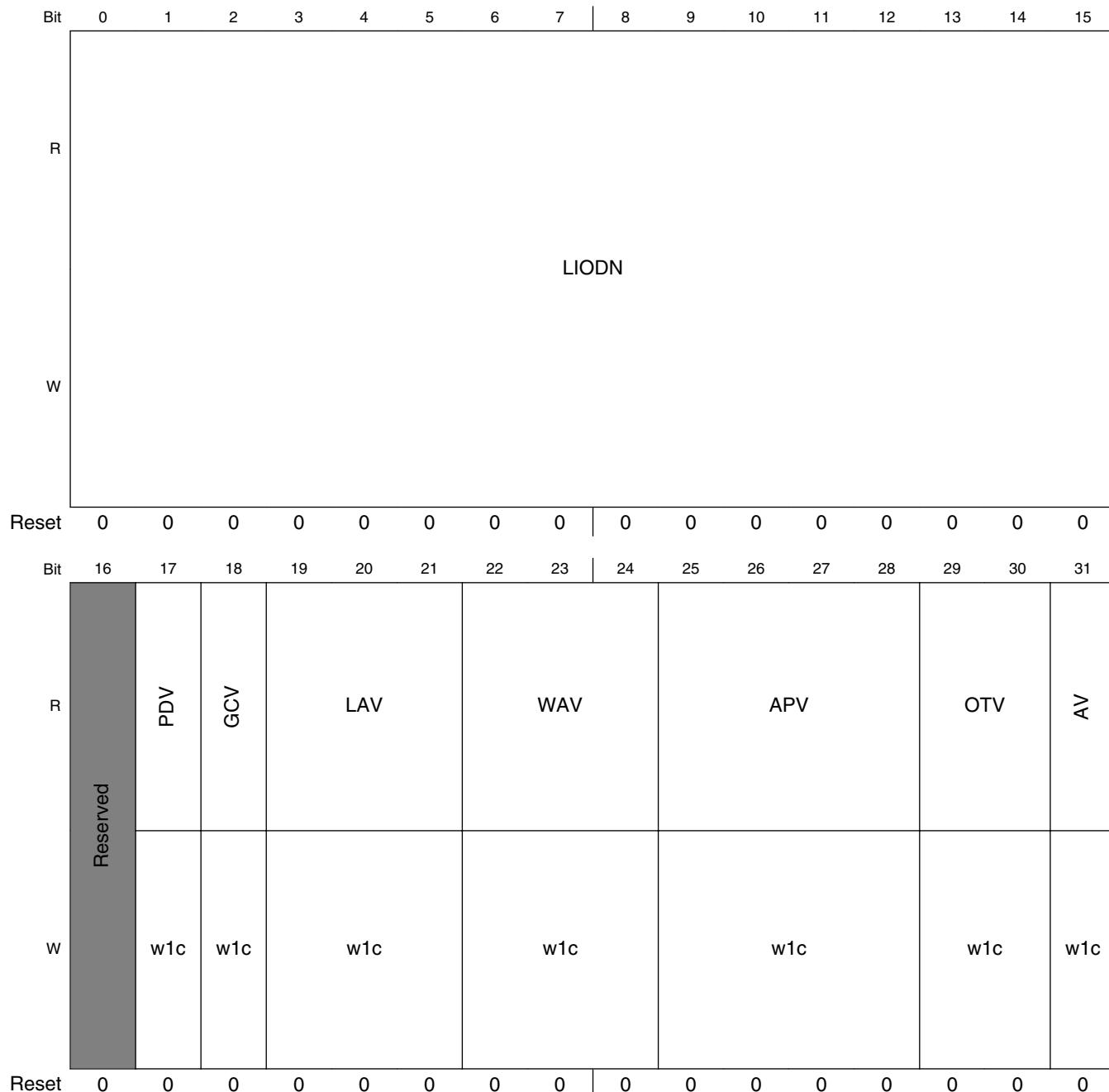
10.4.18 PAMU Operation Error Address Low register (PAMUx_POEAL)

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

PAMUx_POEAL field descriptions

Field	Description
0–31 POEAL	<p>PAMU Operation Error Address Low</p> <p>These bits define the address, in system memory space, of the PAACT or OMT location that encountered the error condition.</p> <p>The POEAL is the 32 least significant address bits of the address, in system memory space, that encountered the PAMU error. The granularity of the POEAL varies depending on the type of data structure that encountered the error:</p> <ul style="list-style-type: none"> For a PAACT location, the 6 least significant address bits of the POEAL has a value of 0x00, representing the PAACE size.

10.4.19 Access Violation Status register 1 (PAMUx_AVs1)



PAMUx_AVs1 field descriptions

Field	Description
0–15 LIODN	Logical I/O Device Number These bits identify the LIODN of the operation that encountered the error.

Table continues on the next page...

PAMUx_AVSI field descriptions (continued)

Field	Description
	NOTE: LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2 .
16 -	This field is reserved. Reserved
17 PDV	<p>PAMU Disable Violation</p> <p>This bit identifies an access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted).</p> <p>0 No access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted)</p> <p>1 Access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted)</p>
18 GCV	<p>Gate Closed Violation</p> <p>This bit identifies an access violation during assertion of PAMU gate closed bit.</p> <p>0 No access violation during PAMU gate closed</p> <p>1 Access violation during PAMU gate closed</p>
19–21 LAV	<p>LIODN Access Violation</p> <p>These bits identify if a LIODN access violation has occurred or the type of LIODN access violation encountered.</p> <p>000 No LIODN access violation</p> <p>001 LIODN access violation-LIODN index not within PPAACT</p> <p>010 LIODN access violation-Secondary PAACE index not within SPAACT</p> <p>011-111 reserved</p>
22–24 WAV	<p>Window Access Violation</p> <p>These bits identify if a window access violation has occurred or the type of window access violation encountered by the logical device.</p> <p>000 No window access violation</p> <p>001 Window access violation-Address not within valid window</p> <p>010 Sub-window access violation-Address not within valid sub-range of the sub-window</p> <p>011 Sub-window access violation-LIODN does not match secondary PAACE</p> <p>100-111 Reserved</p>
25–28 APV	<p>Access Permission Violation</p> <p>These bits identify if an access permission violation has occurred or the type of access permission violation encountered by the logical device.</p> <p>NOTE: The APV encoding of 0001 can only be used when the MW field of the relevant PAACE is not set. Similarly, the APV encoding of 0010 can only be used when the MW field of the relevant PAACE is set.</p> <p>0000 No access permission violation</p> <p>0001 Window access permission violation-access denied to the DSA window</p> <p>0010 Sub-window access permission violation-access denied to the DSA sub-window</p> <p>0011 Reserved</p> <p>0100 Window access type violation-illegal access type to the DSA window</p> <p>0101 Sub-window access type violation-illegal access type to the DSA sub-window</p>

Table continues on the next page...

PAMUx_AVs1 field descriptions (continued)

Field	Description
	0110-0111 Reserved 1000 Sub-window Access Type Violation - Access to a DSA sub-window with invalid SPAACE 1001-1111 Reserved
29-30 OTV	Operation Type Violation These bits identify if an operation type violation has occurred or the type of operation type violation encountered by the logical device. 00 No operation type violation 01 Immediate operation translation violation-IOE does not match IOA or IOB 10 Immediate operation translation violation-matched MOE valid bit not set (MOEA[0]=0 or MOEB[0]=0) 11 Indexed operation translation violation (MOEn[0]=0)
31 AV	Access Violation This bit is set when an access violation has been detected by PAMU. If the bit is set, the rest of the contents of the access violation register and the contents of the access violation address registers are valid. 0 No access violation 1 Access violation detected.

10.4.20 Access Violation Status register 2 (PAMUx_AVs2)

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

PAMUx_AVs2 field descriptions

Field	Description
0-22 -	This field is reserved. Reserved
23 OIV	Operation Index Violation OMI in PAACE refers to Operation Mapping Entry (OME) beyond the OMT limit address
24-31 IOE	Ingress Operation Encoding These bits identify the ingress operation encoding of the operation that encountered the error.

10.4.21 Access Violation Address High register (PAMUx_AVAH)

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

Reset 0

PAMUx_AVAH field descriptions

Field	Description
0–31 AVAH	Access Violation Address High The AVAH is the 32 most significant address bits that identify the address, in I/O address space, of the operation that encountered the error condition.

10.4.22 Access Violation Address Low register (PAMUx_AVAL)

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

Reset 0

PAMUx_AVAL field descriptions

Field	Description
0–19 AVAL	Access Violation Address Low AVAL is the 20 next-significant of the 64 bits that identify the address, in I/O address space, of the operation that encountered the error condition.
20–31 -	This field is reserved. Reserved. The 12 of the 64 bits that identify the address, in I/O address space, of the operation that encountered the error condition are not reported.

10.4.23 ECC Error Control Register (PAMUx_EECTL)

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

Reset 0

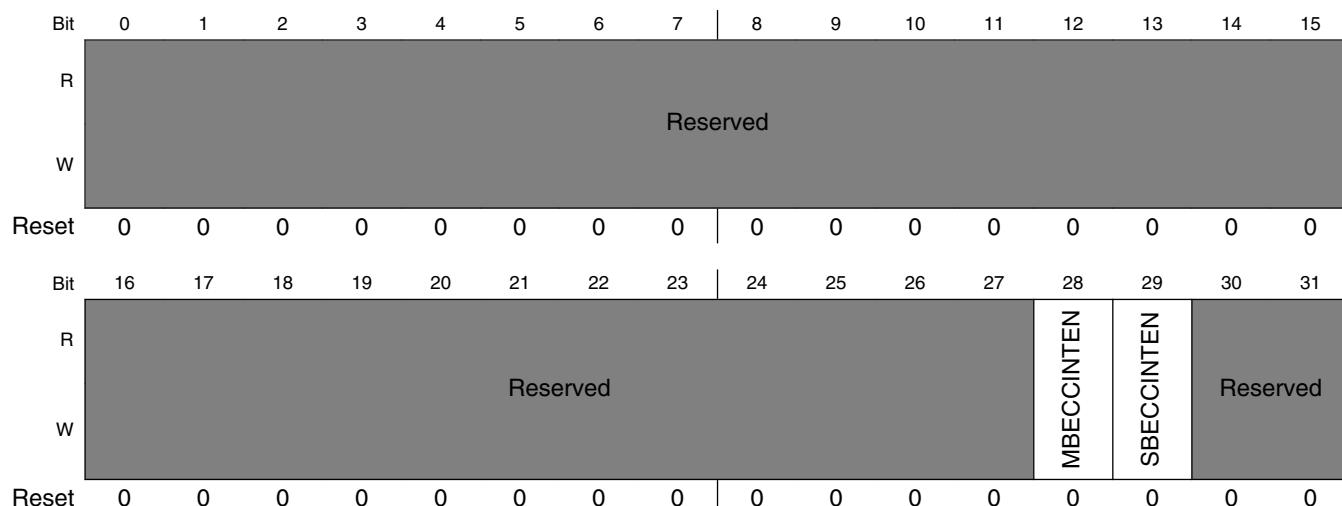
PAMUx_EECTL field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 THRESH	PAMU Threshold

Table continues on the next page...

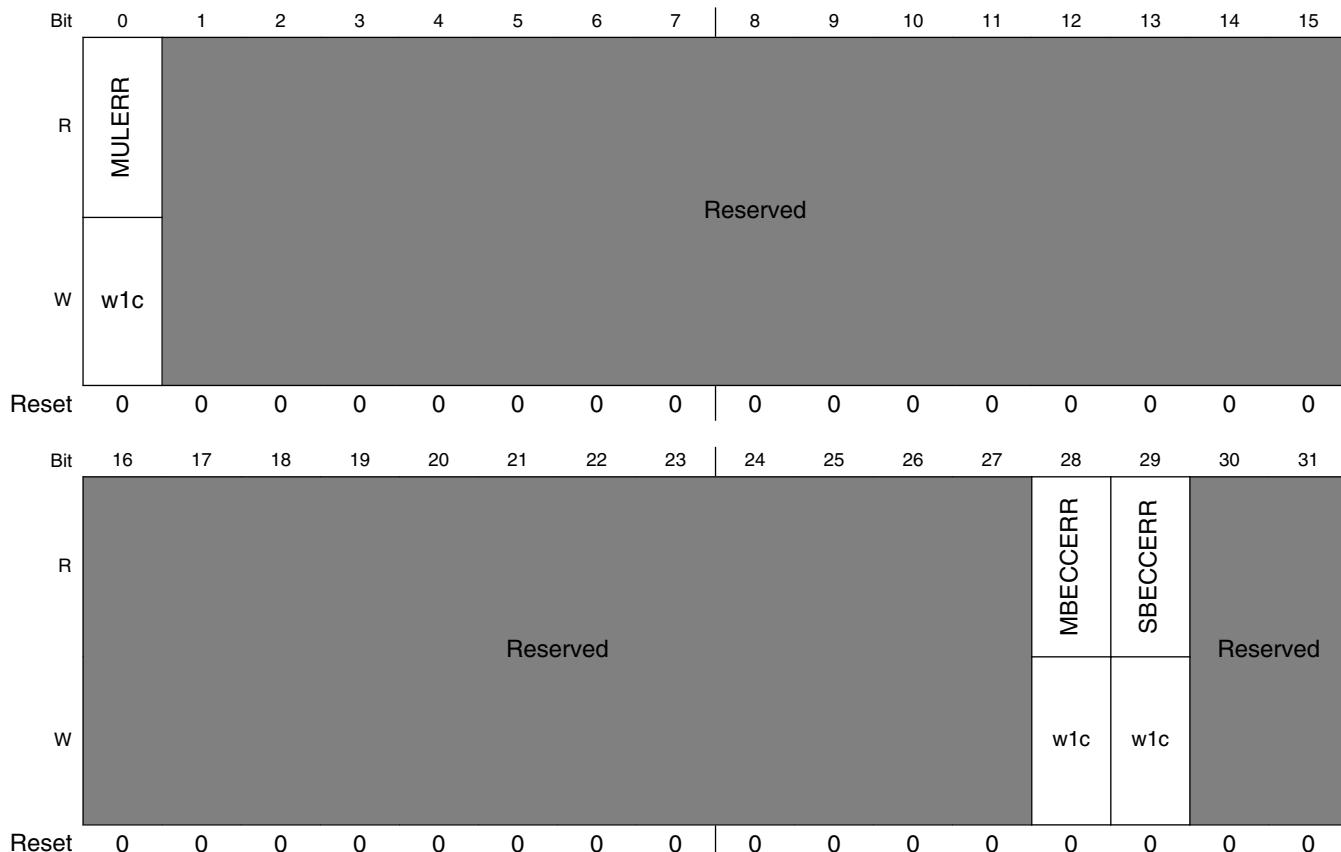
PAMUx_EECTL field descriptions (continued)

Field	Description
	Threshold value for the number of ECC single-bit error that are detected before reporting an error condition. After setting the threshold value, single bit ECC error reporting has to be enabled to report an error condition. If single bit ECC error reporting is enabled with a threshold value of 0, an error condition is reported on the first occurrence of single bit ECC error.
16–23 -	This field is reserved. Reserved
24–31 COUNT	PAMU Count Counts ECC single-bit errors detected. If COUNT is greater than or equal to THRESH then an error is reported, provided single-bit error reporting is enabled. Single bit ECC error is counted across a cache line, as a result one cache access can lead to only one single bit ECC error. Primary PAACT and secondary PAACT accesses in the same cycle can lead to a COUNT update by more than 1. Software must write zeros in bits 24–31 in order to clear the COUNT bits.

10.4.24 ECC Error Interrupt Enable Register (PAMUx_EEINTEN)**PAMUx_EEINTEN field descriptions**

Field	Description
0–27 -	This field is reserved. Reserved
28 MBECCINTEN	Multiple-bit ECC error reporting enable. 0 Multiple-bit ECC error reporting disabled. 1 Multiple-bit ECC error reporting enabled.
29 SBECCINTEN	Single-bit ECC error reporting enable. 0 Single-bit ECC error reporting disabled. 1 Single-bit ECC error reporting enabled.
30–31 -	This field is reserved. Reserved

10.4.25 ECC Error Detect Register (PAMUx_EEDET)

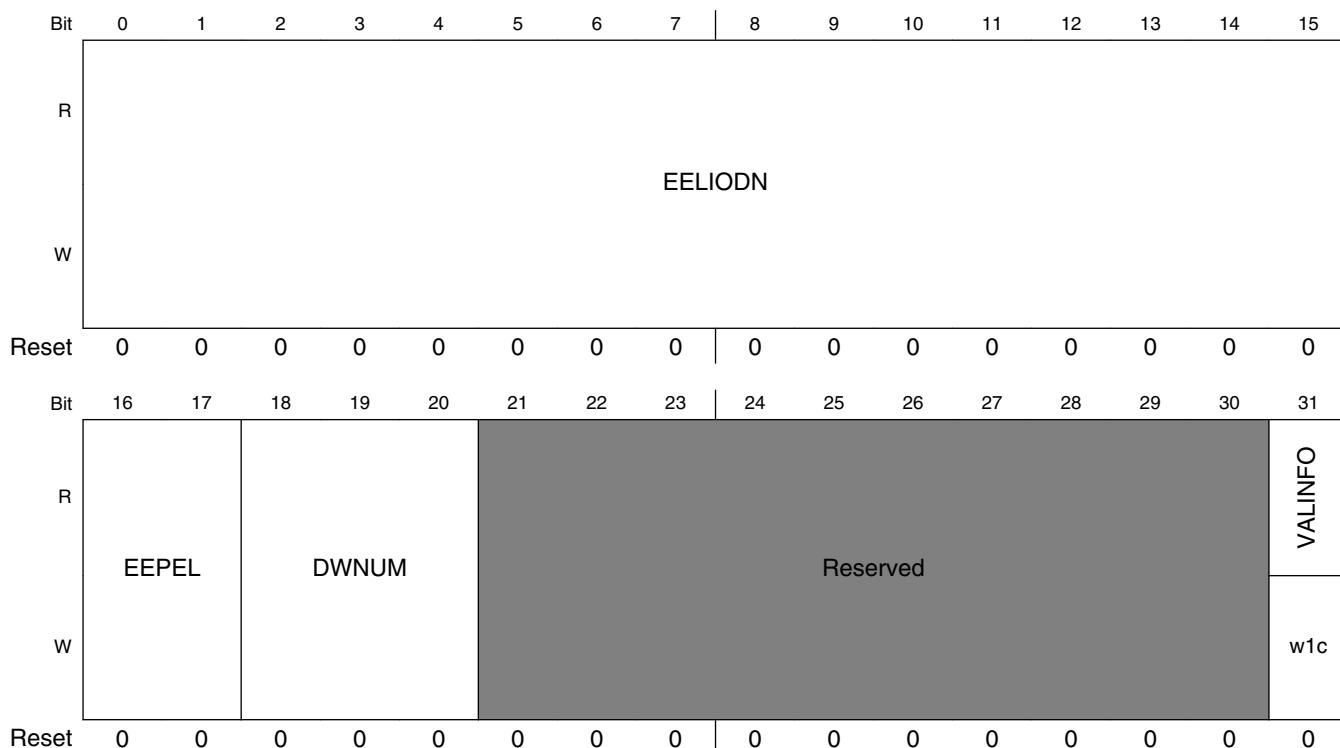


PAMUx_EEDET field descriptions

Field	Description
0 MULERR	Multiple errors. This bit is set, when multiple-bit ECC error is set and PAMU gets another multiple-bit ECC error or single-bit ECC error is set and PAMU gets another single-bit ECC error. The bit does not get set for a Primary PAACT and secondary PAACT access in the same cycle and both accesses have same type of errors. 0 Multiple errors of the same type (Multiple-bit ECC/single-bit ECC) were not detected 1 Multiple errors of the same type (Multiple-bit ECC/single-bit ECC) were detected.
1–27 -	This field is reserved. Reserved
28 MBECCERR	Multiple-bit ECC error 0 Multiple-bit ECC error was not detected. 1 Multiple-bit ECC error was detected.
29 SBECCERR	Single-bit ECC error 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected.
30–31 -	This field is reserved. Reserved

10.4.26 ECC Error Attributes Register (PAMUx_EEATTR)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.



PAMUx_EEATTR field descriptions

Field	Description
0–15 EEIODN	<p>Logical I/O Device Number</p> <p>These bits identify the LIODN of the operation that encountered the ECC error.</p> <p>NOTE: LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2.</p>
16–17 EEPEL	<p>PAMU Error Location. These bits identify the location of the error encountered by the PAMU. For cases, where an error is detected in the primary PAACT cache and secondary PAACT cache in the same cycle, EEPEL will be set to 00.</p> <ul style="list-style-type: none"> 00 Error detected in primary PAACT cache 01 Error detected in secondary PAACT cache 10 Reserved 11 Error detected in OMT cache
18–20 DWNUM	<p>Double word number. Indicates the double-word number within the cache line where the ECC error was detected.</p> <p>Example: 0x1: Indicates double word 1</p>

Table continues on the next page...

PAMUX_EEATTR field descriptions (continued)

Field	Description
21–30 -	This field is reserved. Reserved
31 VALINFO	PAMU ECC error attribute register valid 0 PAMU ECC error attribute register contains no valid information or no enabled errors were detected. 1 PAMU ECC error attribute register contains information of the first detected error that has reporting enabled. Software must clear this bit to unfreeze error capture so error detection hardware can overwrite the capture address/data/attributes for a newly detected error.

10.4.27 ECC Error Address High (PAMUx_EEAHI)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

PAMUx EEAHI field descriptions

Field	Description
0-31 ECCEAH	ECC Error Address High

The ECCEAH is the 32 most significant address bits that identify the address that encountered an ECC error while reading from the cache.

10.4.28 ECC Error Address Low (PAMUX_EEALO)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

PAMUx_EEALO field descriptions

Field	Description
0-25 ECCEAL	ECC Error Address Low

Table continues on the next page...

PAMUx_EEALO field descriptions (continued)

Field	Description
	ECCEAL is the next 26 least significant address bits that identify the address that encountered an ECC error while reading from the cache.
26–31 -	This field is reserved. Reserved

10.4.29 ECC Error Data High (PAMUx_EEDHI)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

PAMUx_EEDHI field descriptions

Field	Description
0–31 DATAHI	ECC Error Data High High-order bits for data presenting an ECC error. Data[0:31].

10.4.30 ECC Error Data Low (PAMUx_EEDLO)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

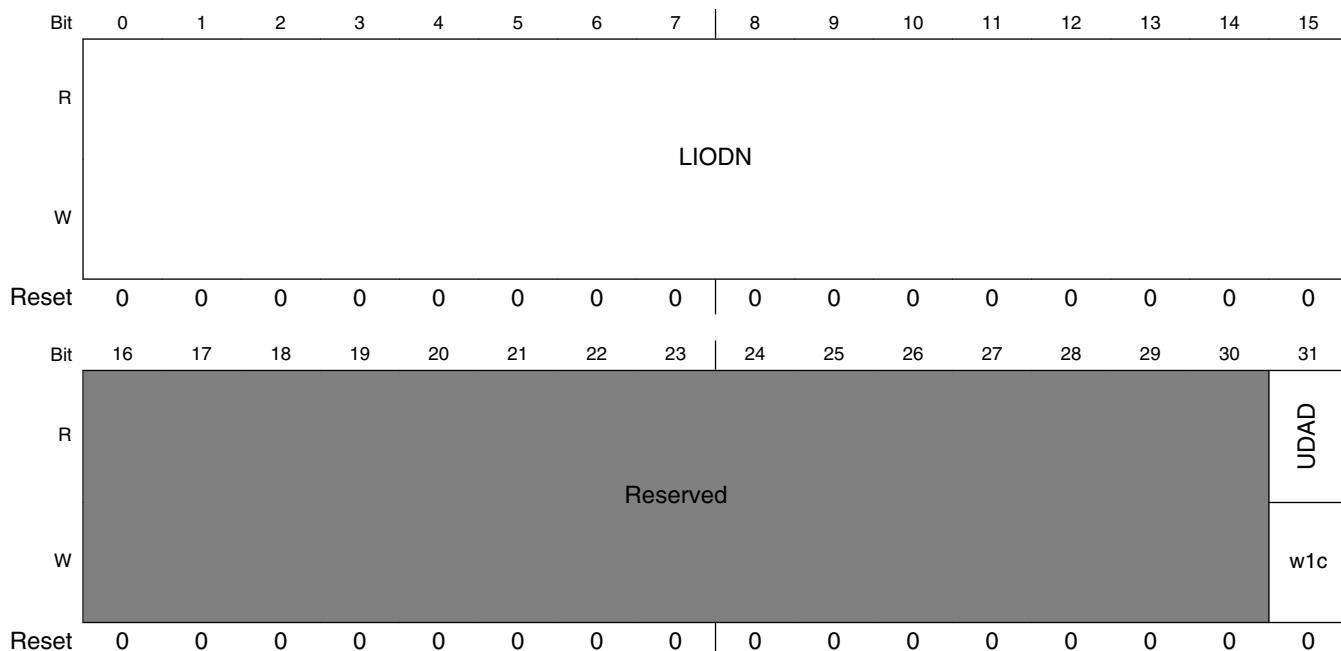
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

PAMUx_EEDLO field descriptions

Field	Description
0–31 DATALO	ECC Error Data Low Low-order bits for data presenting an ECC error. Data[32:63].

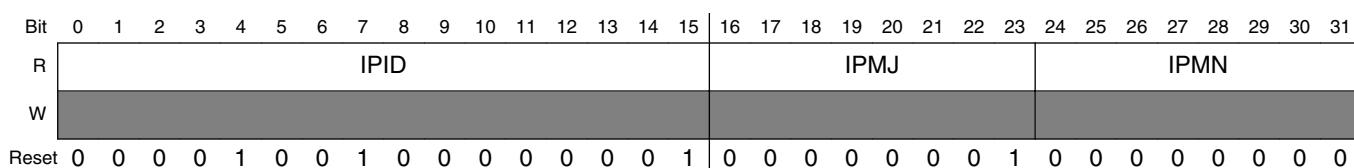
10.4.31 Unauthorized device access detection register (PAMUx_UDAD)



PAMUx_UDAD field descriptions

Field	Description
0–15 LIODN	Logical I/O device number. These bits identify the LIODN of the operation that encountered the unauthorized device access. NOTE: LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2 .
16–30 -	This field is reserved. Reserved
31 UDAD	Unauthorized device access detected. This bit is set when an unauthorized device access has been detected by the PAMU (LIODN index not pointing to a valid PPAACE). If the bit is set, the rest of the contents of the Unauthorized Device Access Detection register are valid. 0 No Unauthorized Device Access Detected 1 Unauthorized Device Access Detected. Software must write 1 in order to clear the pin.

10.4.32 PAMU Revision register 1 (PAMUx_PR1)



PAMUx_PR1 field descriptions

Field	Description
0–15 IPID	IP block ID. These bits provide the IP block ID. For PAMU, the value is 0x0901.
16–23 IPMJ	Major revision. These bits provide the major revision number for PAMU.
24–31 IPMN	Minor revision. These bits provide the minor revision number for PAMU.

10.4.33 PAMU Revision register 2 (PAMUx_PR2)

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

PAMUx_PR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 PIO	PAMU Integration Options. These bits provide information on the integration options for PAMU.
16–23 PER	PAMU ECO Revision. These bits provide information on the ECO Revision Number for PAMU.
24–31 PCO	PAMU Configuration Options. These bits provide information on the configuration options for PAMU.

10.4.34 PAMU Capabilities register 1 (PAMUx_PC1)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

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

PAMUx_PC1 field descriptions

Field	Description
0–31 -	Reserved

10.4.35 PAMU Capabilities register 2 (PAMUx_PC2)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

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

PAMUx_PC2 field descriptions

Field	Description
0–15 ML	Maximum LIODN. These bits define the maximum number of bits of LIODN supported by this PAMU instance. This field is specified as a 16-bit vector, with the sub-vector of the supported number of bits set to 1. Examples: 0x00FF Maximum 8-bit LIODN capability 0xFFFF Maximum 12-bit LIODN capability
16–23 -	This field is reserved. Reserved
24–31 MSnPID	Maximum Snoop ID. These bits define the maximum Snoop ID capability of the system or the maximum Snoop ID that PAMU can signal to the Host/Coherency Domain. This field is specified as a 8-bit vector, with the sub-vector of the supported number of bits set to 1. The bits are coded as follows: 0x0F Maximum 4-bit Snoop ID or the capability for up to 16 unique Snoop IDs 0x1F Maximum 5-bit Snoop ID or the capability for up to 32 unique Snoop IDs 0xFF Maximum 8-bit Snoop ID or the capability for up to 256 unique Snoop IDs

10.4.36 PAMU Capabilities register 3 (PAMUx_PC3)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. The software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming PAMU, its data structures, and related system resources.

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

PAMUx_PC3 field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7–10 MWCE	Maximum Window Count Encoding. These bits define the maximum number of DSA sub-windows available to a LIODN. The window count identifies the maximum number of secondary PAACEs that the PAMU accesses to perform authorization and access control services. The maximum window count is $2^{(MWCE+1)}$ windows. NOTE: MWCE affects the maximum value of the WCE field in a PAACE that can be handled by this PAMU instance. Example: 0x3 Maximum 16 window count capability
11–14 MOMI	Maximum Operation Mapping Index. These bits define the number of least significant bits of the OMI field in a PAACE that are accessed and used as OMT index by this instance of PAMU. This field is specified as a 4-bit vector, with the sub-vector of the supported number of bits set to 1. The field thus defines the maximum number of operation mapping entries available to a LIODN for indexed operation type translation. NOTE: MOMI affects the maximum index value of the OMI field of a PAACE that can be handled by this PAMU instance. The bits are coded as follows: 0001 Indicates a maximum 2 OME capability 0011 Indicates a maximum 4 OME capability 0111 Indicates a maximum 8 OME capability 1111 Indicates a maximum 16 OME capability
15 -	This field is reserved. Reserved
16–22 MIOE	Maximum Ingress Operation Encoding. These bits define the number of least significant bits of an ingress operation encoding (IOE) that are used as an index into an OME to obtain an EOE mapping by this instance of PAMU. This field is specified as a 7-bit vector, with the sub-vector of the supported number of bits set to 1. NOTE: The field defines the maximum number of IOEs available to an LIODN for operation type translation through this PAMU instance. Example: 0x0F Indicates a maximum 16 IOE capability
23–31 -	This field is reserved. Reserved

10.4.37 PAMU Capabilities register 4 (PAMUx_PC4)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. The software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming PAMU, its data structures, and related system resources.

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

PAMUx_PC4 field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–23 ATMS	ATM Support. These bits identify the address translation mode(s) supported by this instance of PAMU. The bits are coded as follows: Bit 20, when set, indicates that no address translation mode is supported by this PAMU. Bit 21, when set, indicates that window address translation mode is supported by this PAMU. Bits 22-23 are reserved. Example: 1100 Indicates PAMU support for the following address translation modes: No Address Translation. Window Address Translation.
24–27 OTMS	OTM Support. These bits identify the operation translation mode(s) supported by this instance of PAMU. The bits are coded as follows: Bit 24, when set, indicates that no operation translation mode is supported by this PAMU. Bit 25, when set, indicates that immediate operation translation mode is supported by this PAMU. Bit 26, when set, indicates that indexed operation translation mode is supported by this PAMU. Bit 27 is reserved. Example: 1110 Indicates PAMU support for the following operation translation modes: No Operation Translation Immediate Operation Translation Indexed Operation Translation
28–31 ALS	Architecture Level Support. These bits identify the version(s) of the PAMU architecture supported by this revision of the PAMU implementation. 0001 Indicates PAMU support for PAMU architecture Version 1.0 only

10.4.38 PAMU Control register (PAMUx_PC)

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGC		PE													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R				OCE									SPCC			PPCC
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

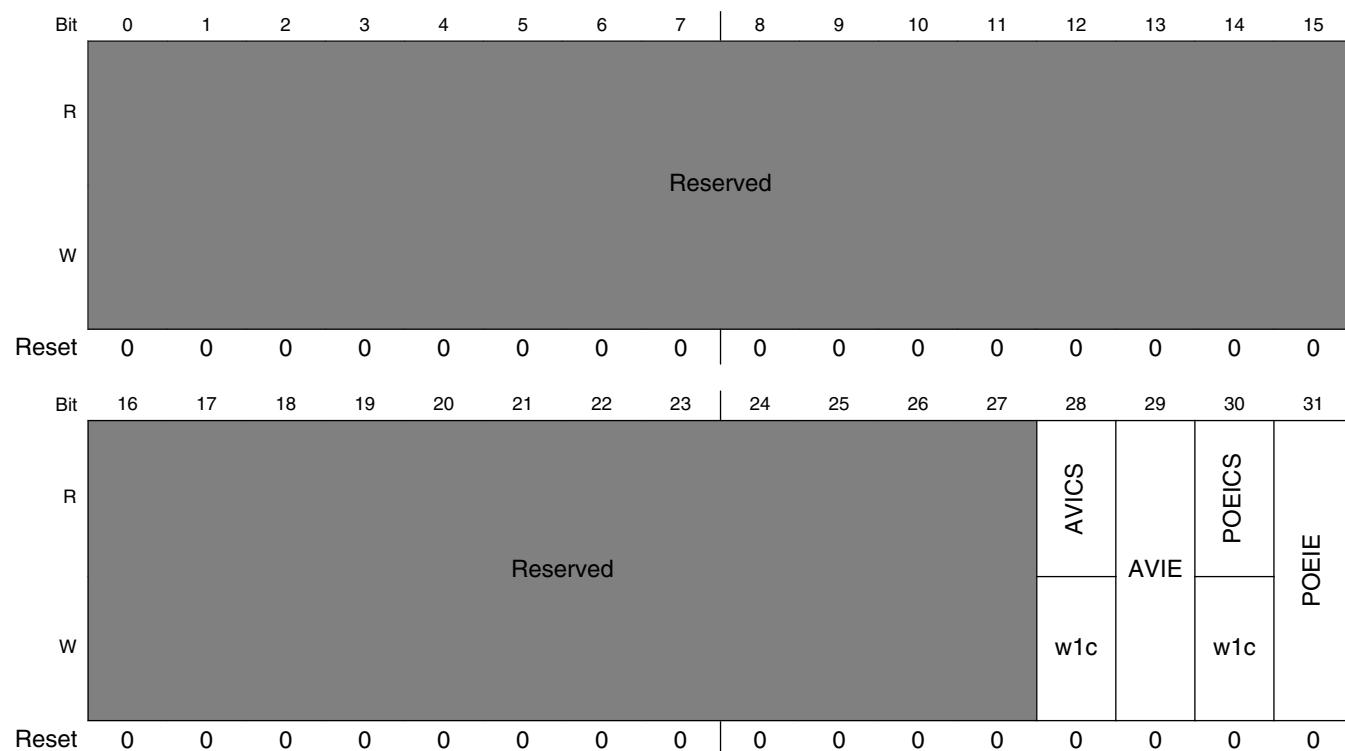
PAMUx_PC field descriptions

Field	Description
0 PGC	PAMU Gate Closed 0 PAMU gate open. PAMU may allow peripheral accesses, subject to authorization and access control. 1 PAMU gate closed. PAMU blocks all peripheral accesses.
1 PE	PAMU Enable This bit controls whether PAMU performs authorization and translation functions. This is to be set by software after setting up the authorization and translation data structures, PAMU and devices (including the setting of LIODNs). NOTE: <ul style="list-style-type: none">If PGC is set, or if PE is cleared, all peripheral accesses are blocked.If PGC is cleared and PE is set, peripheral accesses are allowed based on the LIODN signaled with the access capabilities permitted for the LIOD. 0 PAMU Disabled 1 PAMU Enabled
2–18 -	This field is reserved. Reserved
19 OCE	OMT Cache Enable These bits define the OMT cache enable bit. 0 OMT cache disabled 1 OMT cache enabled
20–23 -	This field is reserved. Reserved
24–27 SPCC	Secondary PAACT Cache Control These bits, SPCC[0:3], define the properties of the secondary PAACT cache. SPCC[0:2] are reserved. SPCC[3] is the cache enable bit. 0 Secondary PAACT cache disabled 1 Secondary PAACT cache enabled

Table continues on the next page...

PAMUx_PC field descriptions (continued)

Field	Description				
28–31 PPCC	<p>Primary PAACT Cache Control</p> <p>These bits, PPCC[0:3], define the properties of the primary PAACT cache.</p> <p>PPCC[0:2] are reserved.</p> <p>PPCC[3] is the cache enable bit.</p> <table> <tr> <td>0</td> <td>Primary PAACT cache disabled</td> </tr> <tr> <td>1</td> <td>Primary PAACT cache enabled</td> </tr> </table>	0	Primary PAACT cache disabled	1	Primary PAACT cache enabled
0	Primary PAACT cache disabled				
1	Primary PAACT cache enabled				

10.4.39 PAMU Interrupt Control and Status register (PAMUx_PICS)**PAMUx_PICS field descriptions**

Field	Description				
0–27 -	This field is reserved. Reserved				
28 AVICS	<p>Access Violation Interrupt Control and Status</p> <p>This bit provides status on whether the access violation interrupt pin is asserted and controls whether the access violation interrupt pin is to be deasserted.</p> <table> <tr> <td>0</td> <td>Access violation interrupt pin is deasserted</td> </tr> <tr> <td>1</td> <td>Access violation interrupt pin is asserted.</td> </tr> </table>	0	Access violation interrupt pin is deasserted	1	Access violation interrupt pin is asserted.
0	Access violation interrupt pin is deasserted				
1	Access violation interrupt pin is asserted.				

Table continues on the next page...

PAMUx_PICS field descriptions (continued)

Field	Description
29 AVIE	<p>Access Violation Interrupt Enable</p> <p>This bit identifies if an access violation interrupt is to be asserted.</p> <p>0 No interrupt asserted if an access violation is detected 1 Access violation interrupt pin asserted if an access violation is detected</p>
30 POEICS	<p>PAMU Operation Error Interrupt Control and Status</p> <p>This bit provides status on whether the operation error interrupt pin is asserted and controls whether the operation error interrupt pin is to be deasserted.</p> <p>0 Operation error interrupt pin is deasserted 1 Operation error interrupt pin is asserted.</p>
31 POEIE	<p>PAMU Operation Error Interrupt Enable</p> <p>This bit identifies if an operation error interrupt is to be asserted.</p> <p>0 No interrupt asserted if an operation error is detected 1 Operation error interrupt pin asserted if an operation error is detected</p>

10.5 PAMU Functional Description

The following is a brief description of the set-up requirements and sequence of events involving processing of DSA operations by PAMU.

10.5.1 System Set-Up for PAMU Operation

System software, for example, a Hypervisor, enumerates these physical I/O devices with one or more Logical I/O Device Numbers (LIODN).

Hypervisor then creates PPAACT and if necessary SPAACT and OMT in system memory (see [Data Structures Used by PAMU](#)).

After setting up the data structures, PAMU CCSR_s are programmed, including the locations of the above data structures. After a PAMU is programmed, it is enabled to process DSA operations.

NOTE

Depending on reset configuration, the PAMUs may default to PAMU bypass mode. The PAMU bypass enable register in the device configuration module (DCFG_PAMUBYPENR) controls enabling and disabling PAMU bypass mode for each PAMU. PAMU bypass mode must be disabled for PAMU to authorize and translate transactions. See [PAMU Bypass Mode](#)

and [No Operation Translation Mode](#) for more information about PAMU bypass mode and [PAMU bypass enable register \(DCFG_PAMUBYPENR\)](#) for more information about disabling PAMU bypass mode.

10.5.2 Steps in Processing of DSA Operations by PAMU

[Figure 10-667](#) illustrates the data structures and actions associated with processing of a DSA operation by PAMU.

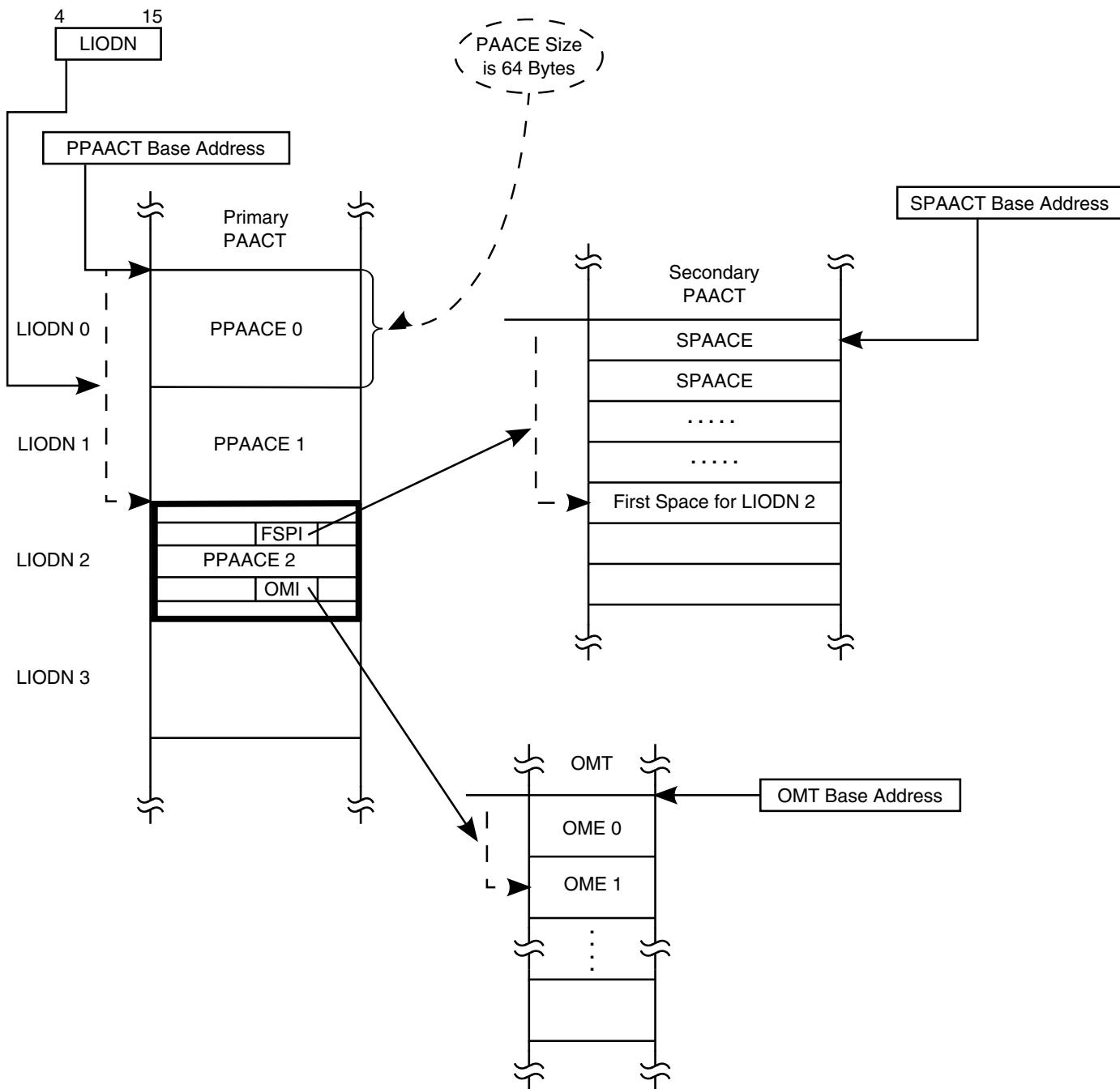


Figure 10-667. PAMU Operation flow overview

- An operation from a logical I/O device arrives at the entrance of the coherency domain. The operation carries with it the LIODN representing that logical device.
- PAMU uses the LIODN to look up the relevant PPAACE in the PAACT cache.
- In the case of a cache-miss, PAMU will fetch the necessary data structure from the PAACT in system memory.

- The contents of the PPAACE indicate if additional data structures have to be referenced to obtain all the necessary authorization and translation attributes for the operation by that LIODN to a particular DSA window.
- A secondary PAACE is accessed if the address accessed is beyond the primary sub-window.
- If the access falls beyond the windows assigned to the LIOD, it is considered an access violation.
- An OME is accessed optionally based on the value of the OTM. The base address of the operation mapping table (OMT) and the operation mapping index (OMI) contained in the PAACE are used to access the entry.
- The translated transaction type and operation is returned for forwarding to the destination domain.

The next section describes the above steps in greater detail.

10.5.3 Detailed Description of PAMU Actions

This section describes in detail the handling of DSA operations by PAMU.

10.5.3.1 PAMU Gate Closed and PAMU Enable Check

In secure boot mode, or if the PAMU Gate Closed (PGC) bit is set to 1, the PAMU will block all peripheral accesses through assertion of access violation regardless of the PAMU Enable (PE) bit setting and regardless of the signaled LIODN.

- If a PAMU lookup request were to be issued while the PGC bit is set to 1, an access violation has been detected and bits are set in the relevant CCSR (see [PAMU Operation Error Address High register \(PAMU_POEAH\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)).
 - The access violation (AV) detected bit is set.
 - The Gate Closed Violation (GCV) bit is set to indicate the peripheral access is not allowed.
 - The LIODN and DSA Address encountering the access violation is logged.

If the PGC bit is not asserted and the PE bit is not set, the PAMU will block all peripheral accesses regardless of the signaled LIODN.

If the PGC bit is not asserted and the PE bit is set, the PAMU will provide authorization and translation services for all peripheral accesses based on the signaled LIODN. Description of this flow is detailed in subsequent sections, starting with section [PPAACT Request Range Check](#). This setting is recommended once all authorization and translation data structures have been set up in system memory space.

If secure boot and trust architecture is not enabled, the PAMU Gate Closed (PGC) bit can be cleared.

Table 10-671. PAMU Gate Closed and PAMU Enable Check

PAMU Gate Closed	PAMU Enable	PAMU Authorizing	Comments
0	0	Yes	Access Violation
0	1	Yes	Normal Operation
1	X	Yes	Access Violation

10.5.3.2 PPAACT Request Range Check

The following describes the manner in which the PAMU lookup request is checked to see if it falls within a valid range:

- During a PAMU lookup request, the LIODN of the operation is signaled along with the rest of the information that describes the DSA operation (address, operation type, and so on).
- The PPAACT base address programmed in the PPBARs (see [Primary PAACt Base Address High register \(PAMU_PPBAH\)](#), and [Primary PAACt Base Address Low register \(PAMU_PPBAL\)](#)) along with the LIODN as offset is used to generate the PPAACE address.
- The PPAACE address is checked against the PPAACT limit address programmed in the PPLARs (see sections [Primary PAACt Limit Address High register \(PAMU_PPLAH\)](#), and [Primary PAACt Limit Address Low register \(PAMU_PPLAL\)](#)).
- A PPAACE address higher than the PPAACT limit address is due to an erroneously or illegally signaled LIODN. An access violation has been detected and bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVIS1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.
 - The LIODN access violation (LAV) bits are set to indicate the LIODN index is not within the PPAACT.
 - The LIODN and DSA address encountering the access violation is logged.

10.5.3.3 PPAACT Cache Lookup

The following describes the flow of a PPAACT cache lookup and the manner in which the information is used by the cache:

- During a PAMU lookup request, the LIODN of the operation is signaled along with the rest of the information that describes the DSA operation (address, operation type, and so on).
 - The LIODN is used to index into the PPAACT cache.
- The PPAACT base address programmed in the PPBARs (see [Primary PAACT Base Address High register \(PAMU_PPBAH\)](#), and [Primary PAACT Base Address Low register \(PAMU_PPBAL\)](#)) along with the LIODN offset is used to generate the PPAACE address.
 - A match of the PPAACE address against the stored tag of a valid entry indicates a cache hit while a mismatch indicates a cache miss. If the tag does not match, a cache miss has occurred.
- If a cache-hit occurs, ECC bits are computed for the contents of the PAACE and these ECC bits are compared against the ECC bits stored along with the PAACE. Mismatched ECC bits, indicates an ECC error.

10.5.3.4 PAMU Fetch Request

Fetch requests are issued to retrieve the relevant PAACE or OME from system memory. These requests are also issued when cache misses occur.

10.5.3.5 Primary PAACE Processing

The primary PAACE is processed for authorization, access control and optionally, translation:

- The WBA and WSE fields of the PPAACE are used to determine if the address of the DSA operation is within a valid window in the I/O bus address space.
 - If the operation is not within a valid window, an access violation has been detected and several bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVSI\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.

- The window access violation (WAV) bits are set to indicate a window access violation has occurred.
- The LIODN and DSA address encountering the access violation is logged.
- The MW field of the PPAACE is used to determine if DSA sub-windows exist.
 - The WCE field indicates the number of DSA sub-windows if DSA sub-windows exist.
- The least significant WSE+1 bits of the DSA address represent its offset (DSA window offset or DWO) within the LIOD's DSA window.
- The Sub-window index (SWI) is the most significant WCE+1 bits of the DWO.
- If the MW bit is set and if SWI = 0, then PPAACE is being accessed. In this case the SWSE field of the PPAACE is used to check if the DSA address is valid, that is, it falls within the defined sub-range.
 - If the address of the DSA operation is not within the sub-range, an access violation has been detected and bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVIS1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.
 - The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
 - The LIODN and DSA address encountering the access violation is logged.
 - If the address of a DSA operation is valid, the access permission (AP) field of the PPAACE is used to determine if the DSA operation is from a valid LIOD that has permission to issue transactions to system storage space. If the AP indicates access is denied or if the specific type of access is not permitted, bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVIS1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.
 - The LIODN and DSA address encountering the access violation is logged.
 - The access permission violation (APV) bits are set to indicate the type of violation that has occurred:
 - If AP indicates that access is denied, either the DSA operation has signaled an invalid LIODN or this LIOD does not have access permissions to this system storage space, that is, the DSA window or sub-window has not been provided to the LIOD assigned the LIODN. The APV bits are set to indicate access has been denied to either the DSA window or sub-window.
 - The AP field of the entry is also used to determine if the type of operation is within the access privileges for that LIOD. Bit 0 of the IOE of the operation is compared against the permitted operation type(s) identified in the AP field. If the type of DSA operation is not permitted,

the APV bits are set to indicate an illegal access type to either the DSA window or sub-window.

- If SWI != 0, the PAACE being referenced is a SPAACE. See [SPAACE Access and Processing](#).
- Details on how address translation is performed is detailed in [Address Translation Service](#).
- Details on how the DSA operation type is processed is detailed in [OMT Access and Service](#).

10.5.3.6 SPAACE Access and Processing

If SWI != 0, the DSA address indicates that the relevant PAACE is a SPAACE, and the PAMU will perform a lookup on the SPAACT as follows:

If the PPAACE indicates multiple window capability and the window count is n, then:

- The attributes for the DSA sub-window defined by addresses where SWI = 0 are contained in the PPAACE.
- The attributes for the DSA sub-window defined by addresses where SWI = 1 are contained in the SPAACE located at the FSPI of SPAACT.
- The attributes for the DSA sub-window defined by addresses where SWI = 2 are contained in the SPAACE located at the FSPI + 1.
- The attributes for the DSA sub-window defined by addresses where SWI = n - 1 (n > 1) are contained in the SPAACE located at the FSPI + (n-2).

See [Figure 10-3](#) for an example illustration for n = 4.

The PAMU also performs an SPAACT range check based on the address of the SPAACE being accessed (see [SPAACT Request Range Check](#)).

10.5.3.6.1 SPAACT Request Range Check

The following describes the manner in which the SPAACE address is checked to see if it falls within a valid range:

- The SPAACE address is checked against the SPAACT limit address programmed in the SPLARs (see [Secondary PAACT Limit Address High register \(PAMU_SPLAH\)](#), and [Secondary PAACT Limit Address Low register \(PAMU_SPLAL\)](#)).
- A SPAACE address higher than the SPAACT limit address is due to an erroneous or illegal DSA address. An access violation has been detected and bits are set in the relevant CCRs (see [Access Violation Status register 1 \(PAMU_AVSI\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.

- The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
- The LIODN and DSA address encountering the access violation is logged.

10.5.3.6.2 SPAACT Cache Lookup Request

SPAACT cache lookup requests are issued to retrieve the relevant SPAACE from the SPAACT cache.

- If the lookup results in a cache hit, the ECC status is checked.
 - If the ECC status indicates an error, bits are set in the relevant CCSR (see [ECC Error Control Register \(PAMU_EECTL\)](#), through [ECC Error Data Low \(PAMU_EEDLO\)](#)).
- If the lookup response indicates a cache-miss, the SPAACE fetch request is issued.

10.5.3.6.3 Secondary PAACE Processing

- The SWSE field of the SPAACE is used to determine if the address of the DSA operation is within a valid sub-range in the I/O bus address space.
 - If the operation is not within the valid sub-range within the DSA sub-window, an access violation has been detected and bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVIS1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.
 - The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
 - The LIODN and DSA Address encountering the access violation is logged.
- If the address of a DSA operation is within the valid sub-range within the DSA sub-window, the Access Permission (AP) field of the SPAACE is used to determine if the DSA operation is from a valid LIOD that has permission to issue transactions to system storage space. If the AP indicates access is denied or if the specific type of access is not permitted, bits are set in the relevant CCSR (see [Access Violation Status register 1 \(PAMU_AVIS1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)):
 - The access violation (AV) detected bit is set.
 - The LIODN and DSA Address encountering the access violation is logged.
 - The Access Permission Violation (APV) bits are set to indicate the type of violation that has occurred:

- If AP indicates that access is denied, this LIOD does not have access permissions to the DSA sub-window. The APV bits are set to indicate access has been denied to the DSA sub-window.
- The AP field of the entry is also used to determine if the type of operation is within the access privileges for that LIOD. IOE[0] of the operation is compared against the permitted operation type(s) identified in the AP field. If the type of DSA operation is not permitted, the APV bits are set to indicate an illegal access type to the DSA sub-window.
- Details on how address translation is performed is detailed in [Address Translation Service](#).
- Details on how the DSA operation type is processed is detailed in [OMT Access and Service](#).

10.5.3.7 Address Translation Service

The address of the DSA operation issued by the LIOD falls within the I/O address space. This section describes the mechanisms to direct DSA operations in I/O address space to the corresponding locations in system storage space.

When a valid PAACE is referenced for a DSA operation from a LIOD, the DSA address translation mode field of the PAACE is used to determine if the additional step of address translation is needed for that DSA operation. There can be two types of encodings indicated in the ATM field, as follows:

- No address translation mode
- Window address translation mode

10.5.3.7.1 No Address Translation Mode

If the address range of system storage space is equal to or less than the address range of the I/O address space, and a particular LIOD has the addressing capability to access the address range of system storage space, then the no address translation Mode may be enabled because the LIOD does not have any addressability limitations to locations in system storage space.

The following describes how no address translation mode is achieved for DSA operations:

- The I/O bus address of the DSA operation is also the system storage address.
- Because there is no address translation, the DSA window must be contained within the system storage space.
- [Figure 10-668](#) shows an example of DSA operations with no address translation.

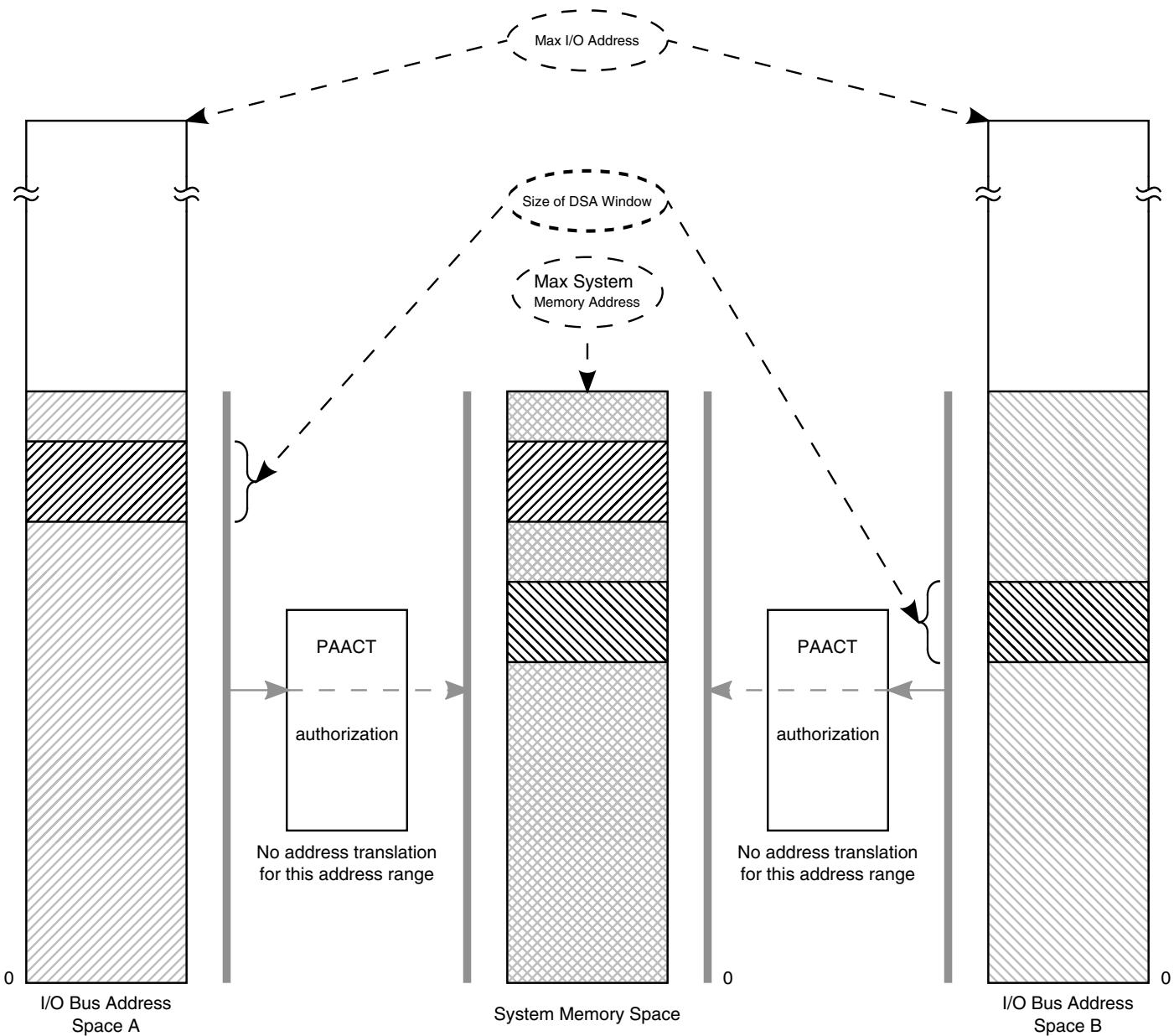


Figure 10-668. DSA Operations with No Address Translation

10.5.3.7.2 Window Address Translation Mode

Window address translation allows the capability for a LIOD to direct DSA operations to a contiguous DSA window in I/O bus address space that get translated to a contiguous window of the same size in system storage space. While no address translation mode implies that the system address is identical to the I/O bus address, with window address translation mode the DSA window can be located anywhere in the I/O bus address space including an address range where the addressing width of the I/O bus address space is larger than that defined for system storage space.

The following describes how DSA operations are achieved when window address translation is enabled:

- The TWBA field of the PAACE defines the base address of the DSA window in system storage space (see [Figure 10-670](#)). As mentioned in [Table 10-6](#), TWBA is aligned to the size to the window described by the PAACE:
- If MW is disabled, the size of the window is indicated by the WSE field of the PAACE
- If MW is enabled, the size of the window is indicated by the SWSE field of the PAACE
- The window size is also used to determine which bits of the I/O bus address used in the DSA operation will be retained in the system storage address (also shown in [Figure 10-670](#)). because the minimum window size is 4 Kbytes, the 4-Kbyte address offset is always retained.
- [Figure 10-670](#) shows an illustration of DSA operations with window address translation.

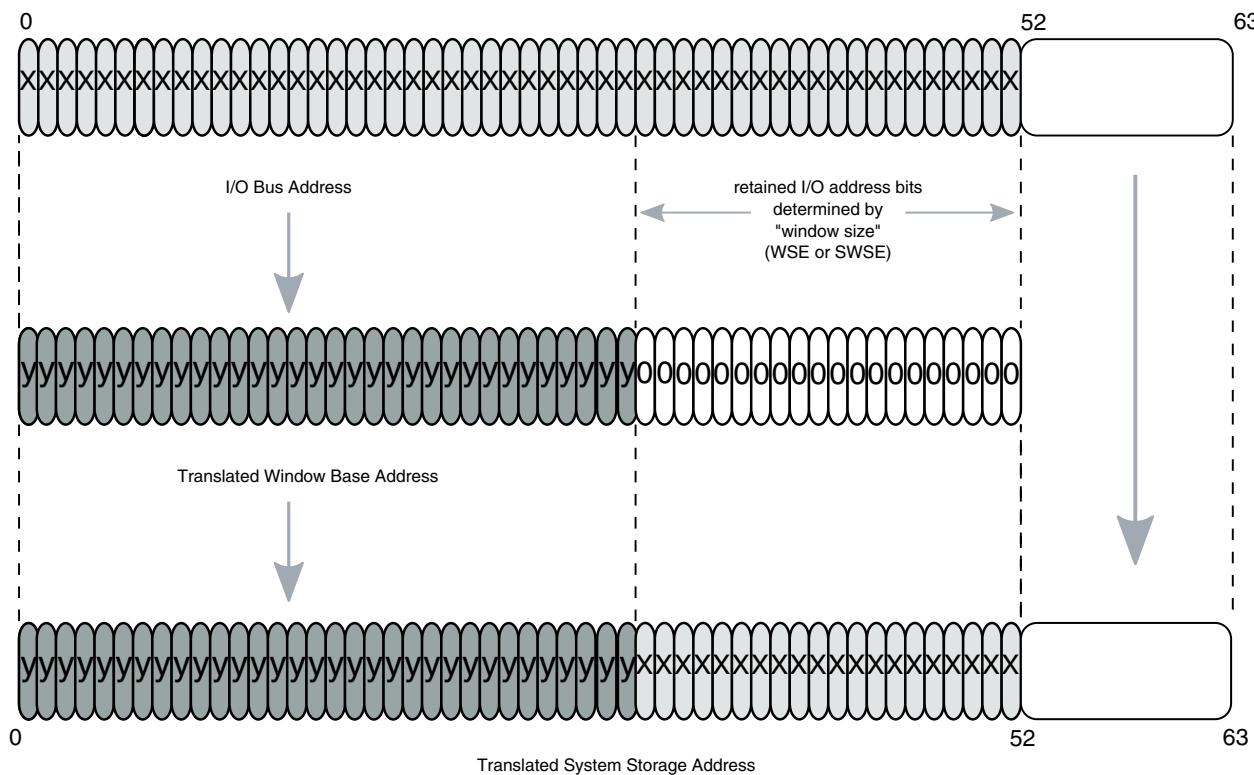


Figure 10-669. Window Address Translation

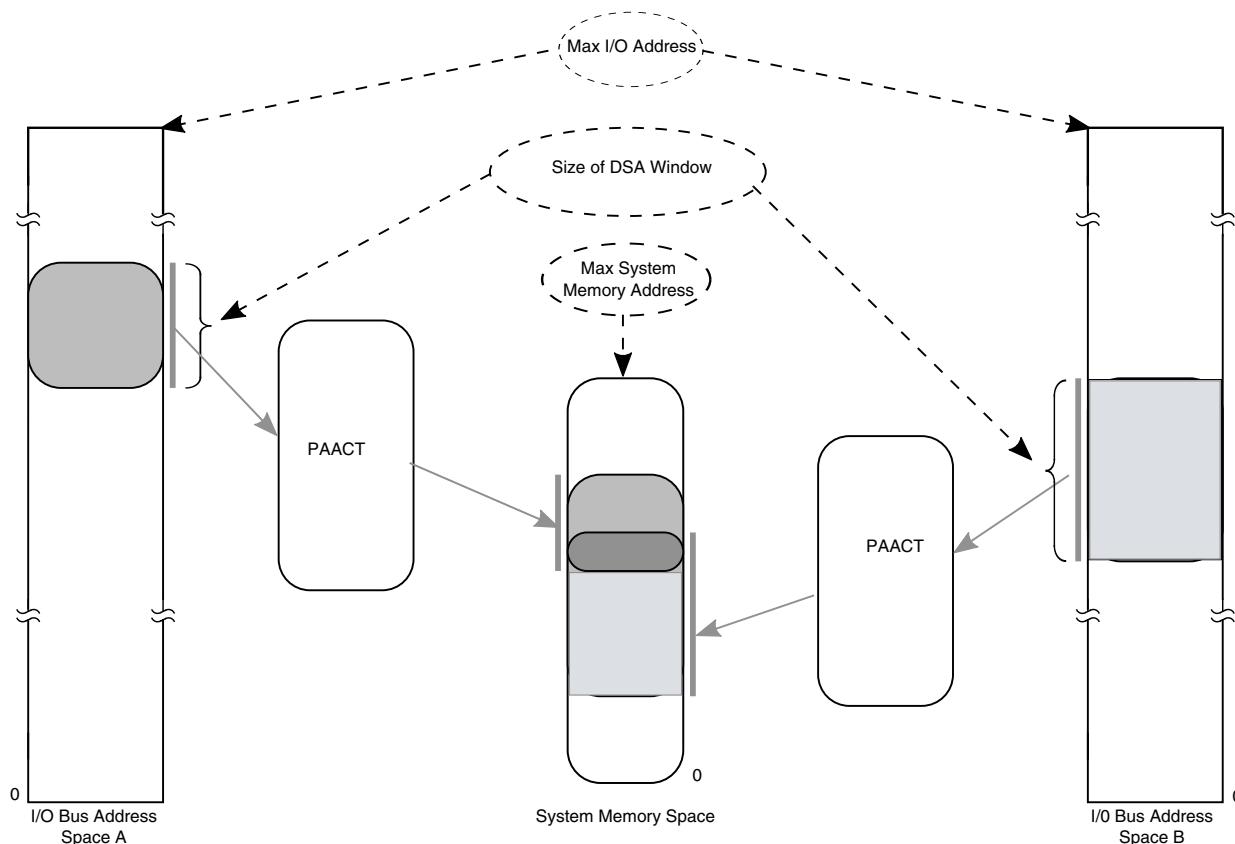


Figure 10-670. DSA Operations with Window Address Translation

10.5.3.8 OMT Access and Service

A DSA operation issued by a LIOD has an ingress operation encoding. This section describes the mechanisms to check a DSA operation's ingress operation encoding and optionally remap the encoding to a corresponding egress operation encoding.

When a valid PAACE is referenced for a DSA operation from a LIOD, the Operation Translation Mode field of the entry is used to determine the type of operation translation needed for that DSA operation. There can be three types of encodings indicated in the OTM field, as follows:

- No operation translation
- Immediate translation mode
- Indexed translation mode

10.5.3.8.1 No Operation Translation

If the source and destination protocols of the DSA operation are the same or the same ingress and egress encodings can be used to express the operation types, No operation translation may be required.

The following describes how no address translation mode is achieved for DSA operations:

- The I/O operation type of the DSA operation is also the operation type of the interconnect protocol in system storage space, or the I/O operation encoding can also be used to express the operation encoding in system storage space.

10.5.3.8.2 Operation Type Translation

The translation process starts by the PAMU-resident hardware encoding an incoming DSA operation type. The output of the encode function is a 8-bit code called the ingress operation encoding. As mentioned in [PAACE Offset 0x18](#), bit 0 of the ingress operation encoding is the operation type qualifier:

- If IOE[0] is a 0, the DSA operation's ingress transaction is a query type of operation.
- If IOE[0] is a 1, the DSA operation's ingress transaction is a update type of operation.

The ingress operation types and their corresponding encodings to the least significant 7-bits of the IOE is I/O domain specific.

Immediate Translation Mode

The following describes DSA operation mapping when OTM indicates immediate translation mode:

- If the IOE matches the encoding contained in the IOEA field of the PAACE:
 - bit-0 of the MOEA field of the PAACE indicates whether the field contains a valid egress operation encoding.
 - If bit-0 of the MOEA field is set, MOEA[1:7] contains the corresponding egress operation encoding A (EOEA).
 - If bit-0 of the MOEA field is not set, an access violation has been detected.
- If the IOE matches the encoding contained in the IOEB field of the PAACE:
 - bit-0 of the MOEB field of the PAACE indicates whether the field contains a valid egress operation encoding.
 - If bit-0 of the MOEB field is set, MOEB[1:7] contains the corresponding egress operation encoding B (EOEB).
 - If bit-0 of the MOEB field is not set, an access violation has been detected.
- If the IOE does not match both IOEA and IOEB, an access violation has been detected.

Indexed Translation Mode

The following describes DSA operation mapping when OTM indicates indexed translation mode:

- The contents of the OMI field of the PAACE are used to index into the relevant OME off the OMT base address¹ (see example in [Figure 10-671](#)).
- The least significant 7-bits of the IOE is used to index into the OME to obtain the relevant MOE to be used².
- The selected MOE contains the egress operation encoding³
 - bit-0 of the MOE field indicates whether field contains a valid egress operation encoding.
 - If bit-0 of the MOE field is set, MOE[1:7] contains the corresponding egress operation encoding (EOE).
 - If bit-0 of the MOE field is not set, an access violation has been detected.

1. PAMU caches the OMT to improve translation performance.

2. Thus, in the limit, up to 128 operation types may be translated.

3. A system could consolidate all operation mappings into a single global OMT with various PAACEs using appropriate indices into the table via their OMI fields.

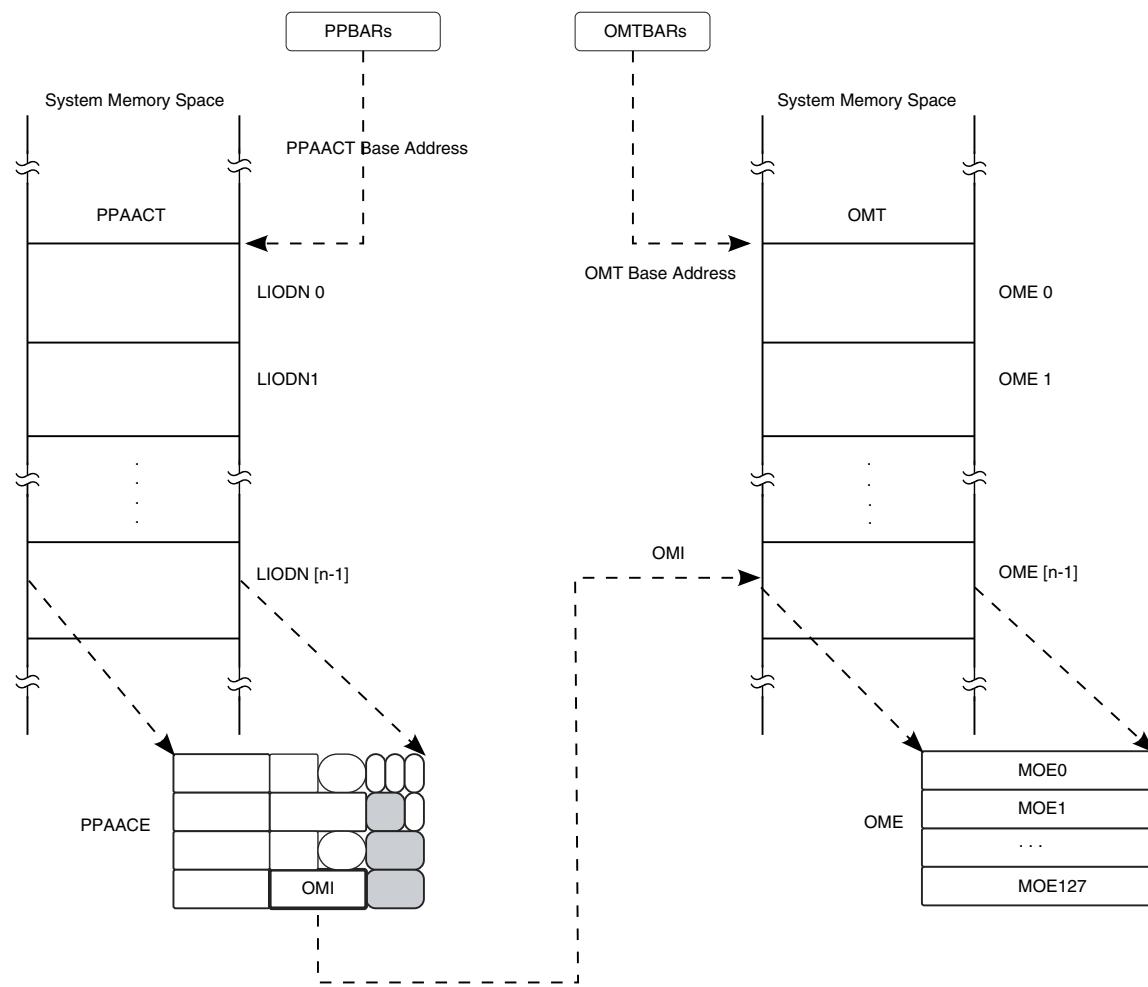


Figure 10-671. PPAACE to OME mapping in System Memory Space

10.5.3.8.3 OMT Cache Access

If the operation translation mode indicates that accessed location requires indexed operation translation performed, the PAMU will perform a lookup on the OMT cache:

- The OMT base address programmed in the OMTBARs see [OMT Base Address Low register \(PAMU_OBAL\)](#) and [OMT Base Address High register \(PAMU_OBAH\)](#) along with the OMI offset contained in the PAACE is used to determine the OME address.
- The MOMI-indicated least significant bits of the OMI are used to index into the OMT cache.
- The valid bit at the OMI index location is used to determine whether the relevant OME is in the cache.

- On a cache miss, the PAMU accesses the OME address from the system memory.
- The MIOE-indicated least significant bits of the IOE are used to index into the OME to determine the 8-bit MOE and the EOE within it.

10.5.3.9 Access Violation

As described in previous sections, during the process of authorizing and checking access permissions, PAMU may detect access violations. In addition to logging relevant error information in the access violation CCSR (see [Access Violation Status register 1 \(PAMU_AV1\)](#), through [Access Violation Address Low register \(PAMU_AVAL\)](#)), the following actions may also be incurred.

- An interrupt is optionally asserted based on the setup in the interrupt mapping CCSR (see [PAMU Interrupt Control and Status register \(PAMU_PICS\)](#)).
- For DSA operations encountering access violations, the system terminates write operations and return 0s for read operations.

10.6 PAMU Initialization/Application Information

This section describes system/software configuration and set-up of the data structures and PAMU for system implementations. This section also describes system/software interactions of the data structures and PAMU during run-time operation.

10.6.1 System Set-Up

Before the LIODs can initiate DSA operations in the system, the Hypervisor must do the following:

- Create the necessary data structures: PAACTs and OMTs in system memory.
- Set up the PAMU (see [Setting Up PAMU](#)).
- Program the corresponding LIODNs for the LIODs. Hypervisor assigns a unique LIODN to a given LIOD of a given I/O domain.

10.6.1.1 Setting Up PAMU

Each PAMU has CCSR (see [PAMU Memory Map/Register Definitions](#)) that Hypervisor must either enable or program with the relevant locations of the data structures in system memory space:

- The base and limit addresses of the memory-resident tables accessed by PAMU are programmed. Note that a PAMU instance can access at most one instance of each of these tables at a time.
 - The primary PAACT base and limit address registers, PPBARs and PPLARs, are programmed with the relevant locations of the PAACT.
 - The secondary PAACT base and limit address registers, SPBARs and SPLARs, are programmed with the relevant locations of the SPAACT.
 - The OMT base and limit address registers, OMTBARs and OMTLARs, are programmed with the relevant locations of the PAACT.
- The PAMU control register is programmed to enable the necessary caches resident in the PAMU hardware.
- ~~The PAMU fetch attributes register is programmed with the necessary information.~~
- As the last step, the PAMU enable bit in PAMU CCSR space must be set by Hypervisor. PAMU will now enforce authorization, translation, or access control based on the signaled LIODN.



10.6.1.2 Power-On Reset

Coming out of reset, Hypervisor must program PAMU configuration registers (PPBARs, PPLARs, SPBARs, SPLARs, OMTBARs, OMTLARs, PC) before enabling PAMU. After programming PAMU configuration registers, Hypervisor must enable PAMU by setting PE bit in PAMU control register.

10.6.2 System with Multiple PAMUs

Each PAMU may have access to authorization and access control data structures that are unique to that PAMU. Alternatively, each PAMU may have access to a common set of authorization and access control data structures. These considerations influence how each PAMU is set up.

While multiple PAMUs may share common settings in terms of accessing data structures in system memory space, the control and status registers are unique to each PAMU. For example, the error control and status registers log error information unique to each PAMU.

10.6.2.1 PAACT Locations

If the desired setup is a single set of peripheral access, authorization, and control tables (PAACTs) that are accessed by all PAMUs in the system, each PAMU would be programmed with the same primary base address and limit locations and the same secondary base address and limit locations in the corresponding CCSR.

If the desired setup is a unique set of PAACTs for access by each PAMU in the system, each PAMU would be programmed with unique primary base address and limit locations and unique secondary base address and limit locations corresponding to the unique primary and secondary PAACT locations in system address space.

10.6.2.2 OMT Locations

Similar to the PAACT setup described in [PAACT Locations](#), either the desired setup is a single operation mapping table (OMT) that is accessed by all PAMUs in the system, or a unique table for access by each PAMU in the system. Accordingly, all PAMUs would either be programmed with the same base address and limit locations or each PAMU would be programmed with a unique base address and limit locations corresponding to the unique OMT location in system address space.

Note that it is possible for multiple PAMUs to have unique OMT base and limit addresses programmed but have the same PPAACT base and limit address and SPAACT base and limit address programmed. This would allow for protocol specific operation translation capabilities unique to the protocol being bridged while accessing a single set of PAACT and associated data structures that contain the authorization, access control, and address translation properties unique to each LIODN.

10.6.2.3 Location of PAACT and OMT Data Structures

Regardless of whether there is a single set of PAACTs or OMT common to all PAMUs or PAACTs or OMT unique to each PAMU, these structures must be placed in one contiguous window for efficiency in the hardware coherency mechanisms. The single window is defined in a LAW that has the coherency sub-domain identifier (CSDId) field of the LAW set up in a manner that includes the relevant PAMUs in the coherency sub-domain. [PAMU Cache Coherency](#), provides details on the setup and use of CSDId towards maintaining coherency.

10.6.3 Peer-to-Peer I/O Operations

The memory map of the system must be set up in a particular manner for authorization, access control and translation services to be made available for peer-to-peer I/O operations:

- The source peer I/O device issues an operation to I/O address space.
- CCSR_s across the hierarchy relating to address decode and/or routing are set up such that the operation is sent upstream from the I/O to the host/coherency domain.
- System interface on receiving the operation performs a PAMU lookup.
- PAMU authorizes the operation and checks the access permissions.
- The translated address provided by PAMU is the address of the operation in system memory space.
- The operation with the translated system address is forwarded to the system.
- System address of the operation decodes to a LAW that identifies the peer I/O device as the destination of the operation.
- The system forwards the operation to the relevant target device.
- CCSR_s across the hierarchy relating to address decode and/or routing, including the target ID designation, are set up such that the operation is sent downstream from the host/coherency domain to the destination peer I/O device.
- As a result, the destination peer I/O device receives an operation that has had authorization, access control and translation services performed.

10.6.4 PAMU Cache Coherency

The presence of caches in PAMU is transparent to application software. However, hardware coherency mechanisms are applied in order for PAMU to maintain coherent copies of the PAACT or OMT data structures in system memory. There are system setup considerations for maintaining coherent caches:

- A mastering agent, for example, a CPU, accesses either the PAACT or OMT locations in system memory.
- The system performs address decode to determine the particular LAW to be referenced for the access.
- The relevant LAW contains the coherency sub-domain identifier (CSDId) which in turn identifies all ports that are to be snooped by that operation.
- Because this access is to a PAACT or OMT location, CSDId must be set up to include ports that have PAMUs present.

10.6.5 Quiescing I/O Devices

10.6.5.1 Quiescing I/O Devices for Table/Entry Updates

While PAMU maintains coherent copies of the system memory resident data structures, a race condition still exists when device activity causes PAMU to reference in the PAMU cache a stale copy of the data structure. This is because the cache invalidating operations, related to the data structure update activity, is still in flight in the system/platform and has not reached PAMU before PAMU references the cache.

Therefore, when system software, for example, a Hypervisor, updates or moves either the PAACT or OMT locations, the software should ensure that there are no peripheral devices performing operations that require the referencing of those data structures. It may do so by quiescing the activities of the relevant I/O devices.

10.6.5.2 Quiescing I/O Devices for Enabling PAMU and its Caches

When system software, for example, a Hypervisor, enables PAMU (see [PAMU Control register \(PAMU_PC\)](#)) or enables one or more of its caches, the software should ensure that there are no peripheral devices performing operations that require the referencing of PAMU and its caches. Therefore, software should quiesce activity by I/O devices that would cause a PAMU lookup to occur.

10.6.6 Locality of References

The efficiency of the PAACT, SPAACT, and OMT caches depends on how the LIODNs are enumerated, the placement of entries in system memory space, and the size of the DSA window.

10.6.6.1 Spatial Locality

The benefits associated with spatial locality of references apply to both the locality of LIODNs that accompany the different transactions flowing toward the same PAMU as well as the locality of the addresses contained within the transactions themselves:

- Enumerating LIODNs under a particular PAMU in linear order reduces the likelihood of thrashing of PAAACES stored in the PAACT cache.
- Placing the secondary PAACES for one LIODN in linear order with respect to other LIODNs under the same PAMU reduces the likelihood of thrashing of SPAACES stored in the SPAACT cache.

10.6.6.2 Temporal Locality

Achieving temporal locality of references in the caches is more a function of the size of the DSA window described by the PAACE. With the maximum transfer size in the coherency domain being 64-bytes, even the minimum 4 Kbyte granularity allowed by the architecture for a PAACE window size results in 64 references to the same PAACE for the case of a 4 Kbyte DSA access. Assigning window sizes larger than 4 Kbyte to a PAACE will result in a correspondingly higher temporal locality of reference for PAACEs stored in the PAMU cache.

10.6.7 Recovering Address Space

The architecture has several fields and settings defined as a 2^n (power-of-two) value. These values may lead to consumption of a larger address space than desired, either I/O address space or system memory space. However, it is possible to recover unallocated address space.

10.6.7.1 Primary PAACT Address Window

While LIODN is defined as a 2^n (power-of-two) value, implementations that have less than 2^n LIODNs can have system software set up the primary PAACT limit address registers to the location in system memory space following the 64-byte index location indicated by the last valid LIODN value.

10.6.7.2 Secondary PAACT Address Window

While the locations of secondary PAACEs are determined by the FSPI index in each PPAACE, system software can set up the secondary PAACT limit address registers to the location in system memory space following the last SPAACE index location of the last valid LIODN that has multiple sub-window capability.

10.6.7.3 Secondary Sub-Windows and PAACEs

The number of sub-windows for a particular LIODN is a function of the window count encoding (WCE) field in the primary PAACE, defined as a 2^n (power-of-two) value. Correspondingly, the number of secondary sub-windows and SPAACES to be accessed for that LIODN is $(2^n - 1)$. The secondary sub-windows described by the SPAACES start

with the SPAACE at the FSPI index location in the SPAACT that describes the first secondary sub-window. Subsequent sub-windows are described by subsequent SPAACES after the SPAACE at the FSPI index location.

Implementations that have less than $(2^n - 1)$ secondary sub-windows, and therefore less than $(2^n - 1)$ SPAACES for a particular LIODN can have that LIODN programmed into the LIODN field for only the required number of SPAACES. This allows for the recoverability of the I/O bus address space for unallocated sub-windows for that LIODN. This also allows for the recoverability of SPAACE index locations in the SPAACT.

10.6.8 Data Structure Size and Alignment

While the Architecture describes the size of a PAACE as being 64-bytes and OME as being 128-bytes, the PAMU implementation only references the defined 32-bytes of the PAACE data structure and 16-bytes of the OME data structure. However, in order to maintain programming model consistency with the architecture across multiple generations of PAMU implementations, software must maintain the 64-byte and 128-byte address alignment for the PAACE and OME data structures respectively.

10.7 PAMU Setup

10.7.1 PAMU Operation Encoding

The specification allows for several methods to achieve the ingress and egress operation encoding (IOE & EOE) from the source and destination interconnect protocols. For this chip, an implementation-specific decode table will be used to achieve both the ingress and egress operation types (see [Figure 10-672](#)):

- The IOE is acquired from an SoC-specific Ingress Decode Table (IDT) that identifies the operation type that generate a particular IOE.
- The EOE is an index into an SoC-specific Egress Decode Table (EDT) that elaborates the CoreNet transaction type to be generated.

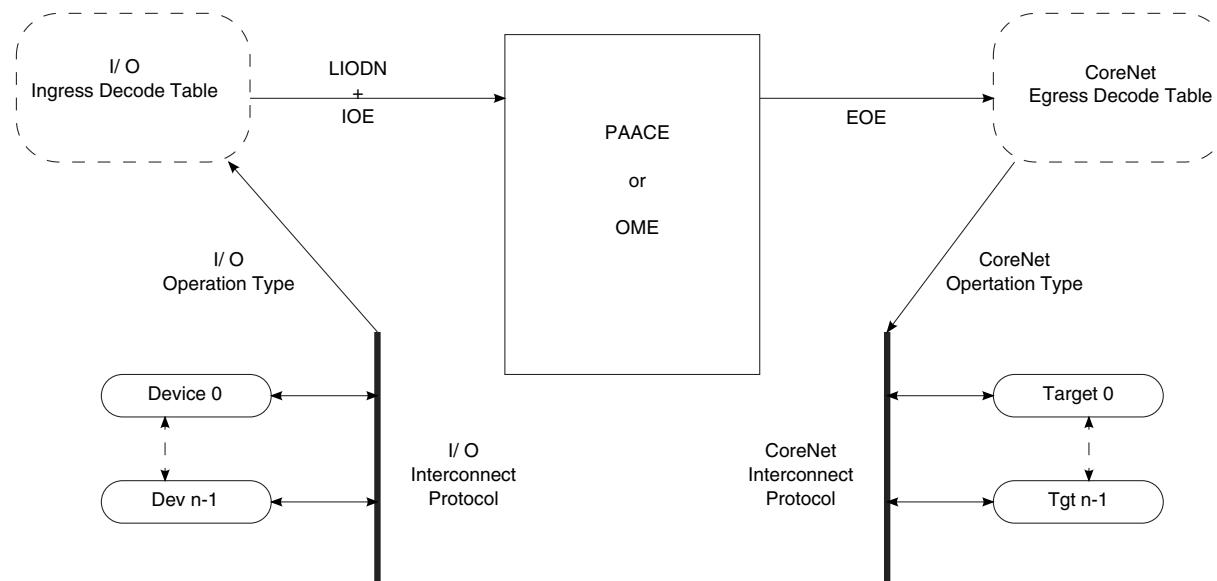


Figure 10-672. Illustration: Generation of Ingress and Egress Operation Encodings

10.7.1.1 Ingress Operation Encoding (IOE)

[Table 10-672](#) describes the ingress operation encodings that are issued on the PAMU Lookup Interface.

Table 10-672. PAMU Ingress Operation Encodings

Ingress Operation Encoding [0:7]	Name	Description
0_000_0000	READ	Read
1_000_0001	WRITE	Write
1_000_0010	EREAD0	Enhanced Read Type 0
1_000_0011	EWRITE0	Enhanced Write Type 0
1_000_0100	DIRECT0	Directive Type 0
1_000_0101	EREAD1	Enhanced Read Type 1
1_000_0110	EWRITE1	Enhanced Write Type 1
1_000_0111	DIRECT1	Directive Type 1
1_000_1100	RAC	Read with Atomic Clear
1_000_1101	RAS	Read with Atomic Set
1_000_1110	RAD	Read with Atomic Decrement
1_000_1111	RAI	Read with Atomic Increment
All other encodings are reserved		

10.7.1.2 Egress Operation Encodings

[Table 10-673](#) describes the egress operation encodings that are issued on the PAMU Lookup Response Interface. The EOEs are indexes into CoreNet commands.

NOTE

EOE[0] is not referenced for the Egress Decode Table. This is because EOE[0] == MOE[0] which determines if the EOE is valid in either the PAACE or OME.

Table 10-673. PAMU Egress Decode Table¹

EOE [1:7]	Description
READ (000_0000)	Read
WRITE (000_0001)	Write
RAC (000_1100)	Read with Atomic Clear
RAS (000_1101)	Read with Atomic Set
RAD (000_1110)	Read with Atomic Decrement
RAI (000_1111)	Read with Atomic Increment
LDEC (001_0000)	Load External Cache
LDECL (001_0001)	Load External Cache with Stash Lock
LDECPE (001_0010)	Load External Cache with Preferred Exclusive
LDECPEL (001_0011)	Load External Cache with Preferred Exclusive and Lock
LDECFE (001_0100)	Load External Cache with Forced Exclusive
LDECFEL (001_0101)	Load External Cache with Forced Exclusive and Lock
RSA (001_0110)	Read with Stash Allocate
RSAU (001_0111)	Read with Stash Allocate and Unlock
READI (001_1000)	Read with Invalidate
RWNITC (001_1001)	Read with No Intention to Cache
WCI (001_1010)	Write Cache Inhibited
WWSA (001_1011)	Write with Stash Allocate
WWSAL (001_1100)	Write with Stash Allocate and Lock
WWSOT (001_1101)	Write with Stash or Target
WWSOTL (001_1110)	Write with Stash Allocate and Lock or Target
<i>All other encodings are reserved</i>	

1. Stashing capability is restricted to 64-byte operations in the chip. Operations < 64 bytes will not have the stash portion of the operation performed.
2. While CoreNet supports all sizes for Atomic Operations, the chip will only support 1,2,4, and 8 byte sizes. See [CoreNet Platform Cache \(CPC\)](#) for resultant behavior when EOE mappings do not fall within these size constraints.

10.7.1.3 IOE to EOE Translations

Due to operation semantics at the source interconnect protocol, only certain ingress operation encodings can map in a seamless manner to an egress operation encoding in order to achieve the same operation semantics at the destination interconnect protocol (CoreNet for PAMU).

10.7.1.3.1 PAMU Bypass Mode and No Operation Translation Mode

Only the basic command set is supported when PAMU is in Bypass Mode. Thus basic Read and Write operations are supported. In the case of SRIO, the Read Atomic Operations are supported as well.

The same basic command set is supported when PAMU is not bypassed but the PAACE entry for that LIODN indicates No Operation Translation has been enabled. Unsupported enhanced IOEs map to reserved EOEs that are used by hardware to detect that an unsupported IOE was encountered while No Operation Translation Mode was enabled.

[Table 10-674](#) shows the default IOE to EOE mappings that occur when PAMU is in bypass mode or when the PAACE entry for an LIODN indicates No Operation Translation has been enabled.

Table 10-674. Default IOE to EOE mappings: PAMU Bypass Mode and No Operation Translation Mode

IOE	EOE
Basic Commands	
READ	READ
WRITE	WRITE
RAC	RAC
RAS	RAS
RAD	RAD
RAI	RAI
Enhanced Commands- Unsupported	
EREAD0	EREAD0
EWRITE0	EWRITE0
DIRECT0	DIRECT0
EREAD1	EREAD1
EWRITE1	EWRITE1
DIRECT1	DIRECT1

[Table 10-675](#) shows the CoreNet operations that can be achieved when PAMU is in bypass mode or when the PAACE entry for an LIODN indicates No Operation Translation has been enabled. Issuance of operations by an LIODN outside of that listed in [Table 10-675](#) can be achieved only when an OMT has been created with the relevant mappings and the relevant PAACE entry for that LIODN has the Immediate or Indexed Operation Translation Mode enabled (see [Immediate and Indexed Operation Translation Modes](#)).

Table 10-675. Default CoreNet operations in PAMU Bypass Mode and No Operation Translation Mode

Description
Read
Write
Read with Atomic Clear
Read with Atomic Set
Read with Atomic Decrement
Read with Atomic Increment

10.7.1.3.2 Immediate and Indexed Operation Translation Modes

[Table 10-676](#) shows the permitted EOEs for particular IOEs for Immediate and Indexed Operation Translation.

Table 10-676. Permitted IOE to EOE translations for Immediate and Index Operation Translation

IOE	EOE
READ, EREAD0, EREAD1	READ, RAC, RAD, RAS, RAI, RSA, RSAU, READI, RWNITC
WRITE, EWRITE0, EWRITE1	WRITE, WCI, WWSA, WWSAL, WWSOT, WWSOTL
DIRECT0, DIRECT1	LDEC, LDECL, LDECPE, LDECPEL, LDECFC, LDECFCF
RAC	RAC
RAS	RAS
RAD	RAD
RAI	RAI

10.7.2 Domain Attributes

10.7.2.1 Constrained Domain Attribute

[Table 10-677](#) describes a domain attribute field constraint in this chip.

Table 10-677. Domain Attributes constrained

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
16-23	SnpID	P & S	Snoop ID While PAMU in this chip can access an 8-bit Snoop ID Domain Attribute from the PAACE, the CoreNet Coherency Fabric (CCF) in the chip supports up to 32 Snoop IDs Therefore, only SnpID[3:7] is supported.

10.7.3 Implementation Attributes

Table 10-678 describes the SoC definition of Implementation Attributes fields of PAACE Offset 0x0C through 0x0E.

Table 10-678. Implementation Attributes

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-7	-	-	Reserved
8-15	CID	P and S	Cache ID These bits define the cache identifier. This field may be used to identify a specific cache entity for the window described by this PAACE. These bits are used for explicit stashing where CID identifies the specific destination.
16-23	-	-	Reserved

10.7.4 Operation Mapping Table (OMT)

For this chip, OMI can be up to 4 bits supporting 16 OMEs. Each OME can contain up to 16 valid bytes supporting 16 MOEs. Because an OME can have at most 16 valid bytes in the 128-byte OME, PAMU fetches 16 bytes of data after a OMT cache miss. Invalidation will only invalidate one entry out of the 4 entries in the OMT cache. Each entry will hold 128 bits of data.

Chapter 11

DDR Memory Controller

11.1 DDR Introduction

The two fully programmable DDR SDRAM controllers support most JEDEC standard x8, x16, or x32 DDR2 and DDR3 memories available. Only x32 DRAMs that use 1 data strobe per data byte are supported. In addition, unbuffered and registered DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

NOTE

In this chapter, the word 'bank' refers to a physical bank specified by a chip select; 'logical bank' refers to one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the 2 or 3 bits on the bank address (MBA) pins during a memory access.

The figure below is a high-level block diagram of the DDR memory controller with its associated interfaces. [DDR Functional Description](#) contains detailed figures of the controller.

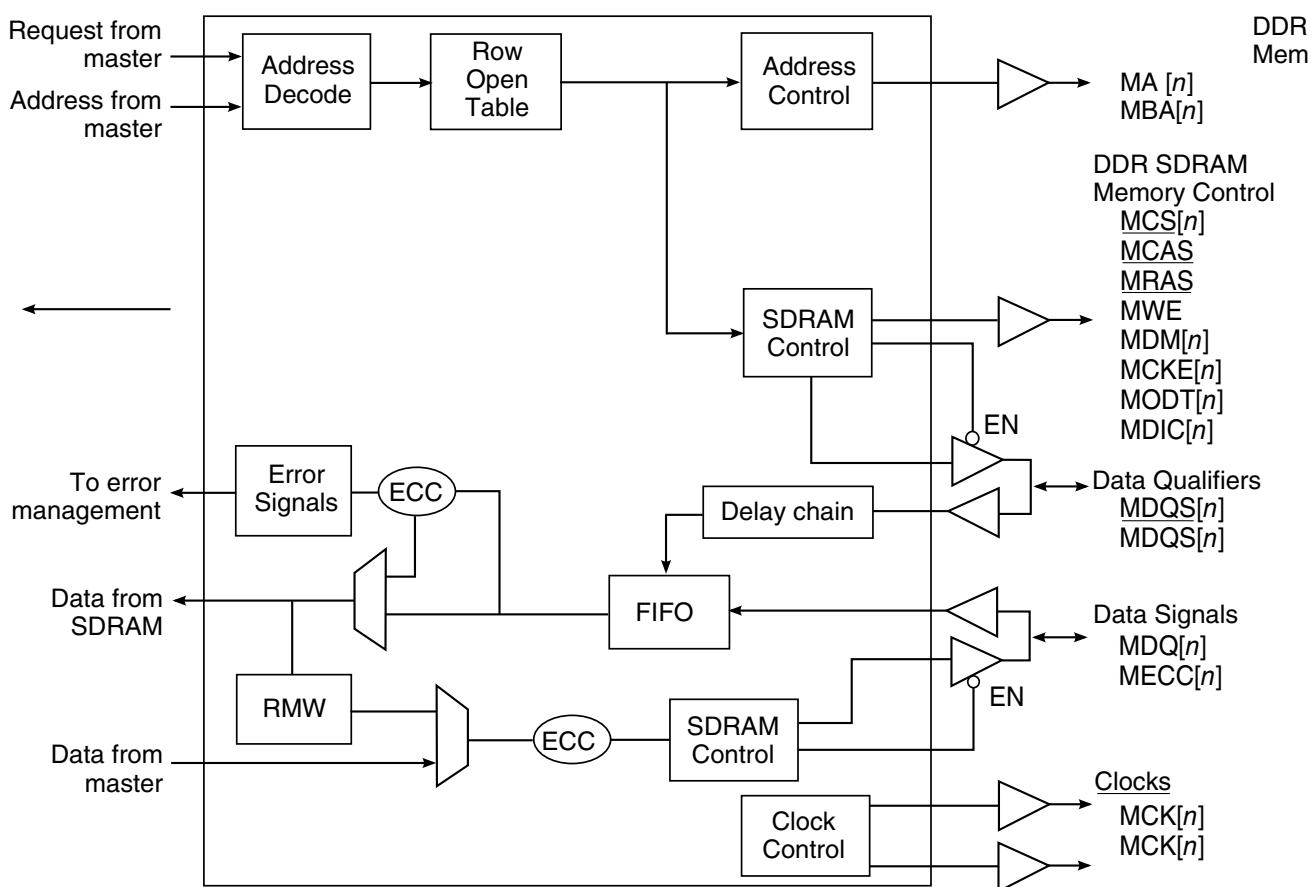


Figure 11-1. DDR Memory Controller Simplified Block Diagram

11.2 DDR Features Summary

The DDR memory controller includes the following features:

- Support for DDR2 and DDR3 SDRAM
- Dual independent controllers
- 64-/72-bit SDRAM or 32-/40-bit data bus for DDR2 and DDR3
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
 - As many as four physical banks (chip selects), each bank independently addressable
 - 64-Mbit to 8 -Gbit devices depending on internal device configuration with x8/x16/x32 data ports (no direct x4 support)
 - Unbuffered and registered DIMMs
- Chip-select interleaving support
- Partial array self refresh support
- Controller interleaving support

- Support for data mask signals and read-modify-write for sub-double-word writes.
Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)
- Support for address parity for registered DIMMs
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Interrupt driven rapid clear of memory
- Write leveling supported for DDR3 memories
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

11.2.1 DDR Modes of Operation

The DDR memory controller supports the following modes:

- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing DDR_SDRAM_INTERVAL[BSTOPRE] causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting CS n _CONFIG[AP_ n _EN].

11.3 DDR External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

11.3.1 DDR Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals

Table 11-1 shows the memory address signal mappings.

Table 11-1. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA15	A15
	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR)
	MA9	A9
	MA8	A8 (alternate AP for DDR)
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

11.3.2 DDR Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

Note that this device has two DDR memory controllers. The same set of signals exist on both DDR controllers. The signals are differentiated by the *Dn_* in front of the signal names. When a DDR signal is discussed specifically elsewhere in this book it is referred to without the *Dn_* prefix. For example, the *D1_Mdq[0]* and *D2_Mdq[0]* signals are both referred to as just *MDQ[0]*.

11.3.2.1 Memory Interface Signals

This table describes the DDR controller memory interface signals.

Table 11-2. Memory Interface Signals-Detailed Signal Descriptions

Signal	I/O	Description	
Dn_MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated - Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation - Driven coincident with corresponding data strobes (MDQS) signal. High impedance - No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated - Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation - The DDR SDRAM drives data during a READ transaction. High impedance - No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.
	I/O	Data strobes. Inputs with read data, outputs with write data. The data strobes must be differential.	
Dn_MDQS [0:8] / Dn_MQDS_B[0:8]	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		State Meaning	Asserted/Negated - Driven high when positive capture data is transmitted and driven low when negative capture data is transmitted. Centered in the data "eye" for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 11-235 for byte lane assignments.
		Timing	Assertion/Negation - If a WRITE command is registered at clock edge n, data strobes at the DRAM assert centered in the data eye on clock edge n + 1. See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated - Driven high when positive capture data is received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 11-235 for byte lane assignments.

Table continues on the next page...

Table 11-2. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		Timing	Assertion/Negation - If a READ command is registered at clock edge n , and the latency is programmed in TIMING_CFG_1[CASLAT] to be m clocks, data strobes at the DRAM assert coincident with the data on clock edge $n + m$. See the JEDEC DDR SDRAM specification for more information.
Dn_MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller's bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. See Error Checking and Correcting (ECC) ," for more details.	
		State Meaning	Asserted/Negated - Represents the state of ECC being driven by the DDR controller on writes.
		Timing	Assertion/Negation - Same timing as MDQ High impedance - Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		State Meaning	Asserted/Negated - Represents the state of ECC being driven by the DDR SDRAMs on reads.
		Timing	Assertion/Negation - Same timing as MDQ High impedance - Same timing as MDQ
Dn_MA[15:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[15:0] carry 16 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated - Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 11-238 for a complete description of the mapping of these signals.
		Timing	Assertion/Negation - The address is always driven when the memory controller is enabled. It is valid when a transaction is driven to DRAM (when MCS_B n is active). High impedance - When the memory controller is disabled
Dn_MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	
		State Meaning	Asserted/Negated - Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 11-238 describes the mapping of these signals in all cases.

Table continues on the next page...

Table 11-2. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		Timing	Assertion/Negation - Same timing as MAn High impedance - Same timing as MAn
Dn_ MCAS_B	O	Column address strobe. Active-low SDRAM address multiplexing signal. MCAS_B is asserted for read or write transactions and for mode register set, refresh, and precharge commands.	
		State Meaning	Asserted - Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 11-244 for more information on the states required on MCAS_B for various other SDRAM commands. Negated - The column address is not guaranteed to be valid.
		Timing	Assertion/Negation - Assertion and negation timing is directed by the values described in DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0) , DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1) , DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2) , and DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3) . High impedance - MCAS_B is always driven unless the memory controller is disabled.
D n_ MRAS_B	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.	
		State Meaning	Asserted - Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 11-244 for more information on the states required on MRAS_B for various other SDRAM commands. Negated - The row address is not guaranteed to be valid.
		Timing	Assertion/Negation - Assertion and negation timing is directed by the values described in DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0) , DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1) , DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2) , and DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3) . High impedance - MRAS_B is always driven unless the memory controller is disabled.
D n_MCS_B[0:3]	O	Chip selects. Four chip selects supported by the memory controller.	
		State Meaning	Asserted - Selects a physical SDRAM bank to perform a memory operation as described in Chip select n memory bounds (DDR_CSn_BNDS) , and Chip select n configuration (DDR_CSn_CONFIG) . The DDR controller asserts one of the MCS_B [0:3]signals to begin a memory cycle. Negated - Indicates no SDRAM action during the current cycle.
		Timing	Assertion/Negation - Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_0-TIMING_CFG_3. High impedance - Always driven unless the memory controller is disabled.

Table continues on the next page...

Table 11-2. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D _n _MWE_B	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.	
		State Meaning	Asserted - Indicates a memory write operation. See Table 11-244 for more information on the states required on MWE_B for various other SDRAM commands. Negated - Indicates a memory read operation.
		Timing	Assertion/Negation - Similar timing as MRAS_B and MCAS_B. Used for write commands. High impedance - MWE_B is always driven unless the memory controller is disabled.
D _n _MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB) and MDM7 corresponds to the LSB, while MDM8 corresponds to the ECC byte. Table 11-235 shows byte lane encodings.	
		State Meaning	Asserted - Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDM _n signals are active-high for the DDR controller. MDM _n is part of the DDR command encoding. Negated - Allows the corresponding byte to be read from or written to the SDRAM.
		Timing	Assertion/Negation - Same timing as MDQx as outputs. High impedance - Always driven unless the memory controller is disabled.
D _n _MODT[0:3]	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT[0:3] represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		State Meaning	Asserted/Negated - Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation - Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS _n _CONFIG[ODT_RD_CFG] and CS _n _CONFIG[ODT_WR_CFG] fields. High impedance - Always driven.
D _n _MDIC[0:1]	I/O	Driver impedance calibration. Note that the MDIC signals require the use of precision resistors (see the chip's hardware specifications for specific values); MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV _{DD} . See DDR Control Driver Register 1 (DDR_DDRCDR_1) , for more information on these signals.	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These will be driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.
D _n _MAPAR_ERR_B	I	Address parity error. Reflects whether an address parity error has been detected by the DRAM. This signal is active low.	

Table continues on the next page...

Table 11-2. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		State Meaning	Asserted - An error has been detected. Negated - An error has not been detected.
		Timing	Assertion/Negation - Driven by registered DDR3 DIMMs 3 DRAM cycles after the parity bit has been driven by the memory controller (for DDR2 DIMMs, driven 1 cycle after the parity bit has been driven by the memory controller). This error signal should be held valid for 2 DRAM cycles.
D _n _MAPAR_OUT_B	O	Address parity out. Driven by the memory controller as the parity bit calculated across the address and command bits. Even parity is used, and parity is not calculated for the MCKE _n , MODT _n , or MCS_B signals.	
		State Meaning	Asserted - The parity bit is high. Negated - The parity bit is low.
		Timing	Assertion/Negation - Will be issued one DRAM cycle after the chip select for each command.

11.3.2.2 Clock Interface Signals

The following table contains the detailed descriptions of the clock signals of the DDR controller.

Table 11-3. Clock Signals-Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:5] , MCK_B [0:5]	O	DRAM clock outputs and their complements. See Clock Distribution .	
		State Meaning	Asserted/Negated-The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation-Timing is controlled by the DDR_SDRAM_CLK_CNTL register at offset 0x130.
MCKE[0:3]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0:3] can be negated to stop clocking the DDR SDRAM. The MCKE signals should be connected to the same rank of memory as the corresponding MCS_B and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as MCS0_B and MODT[0].	
		State Meaning	Asserted-Clocking to the SDRAM is enabled. Negated-Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or MCK_B. MCK/ MCK_B are don't cares while MCKE[0:3] are negated.
		Timing	Assertion/Negation-Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Will be asserted again when exiting dynamic power management or self refresh. High impedance-Always driven.

11.4 DDR Memory Controller Memory Map

The following table shows the register memory map for the DDR memory controller .

DDR memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000	Chip select n memory bounds (DDR1_CS0_BNDS)	32	R/W	0000_0000h	11.4.1/499
8008	Chip select n memory bounds (DDR1_CS1_BNDS)	32	R/W	0000_0000h	11.4.1/499
8010	Chip select n memory bounds (DDR1_CS2_BNDS)	32	R/W	0000_0000h	11.4.1/499
8018	Chip select n memory bounds (DDR1_CS3_BNDS)	32	R/W	0000_0000h	11.4.1/499
8080	Chip select n configuration (DDR1_CS0_CONFIG)	32	R/W	0000_0000h	11.4.2/500
8084	Chip select n configuration (DDR1_CS1_CONFIG)	32	R/W	0000_0000h	11.4.2/500
8088	Chip select n configuration (DDR1_CS2_CONFIG)	32	R/W	0000_0000h	11.4.2/500
808C	Chip select n configuration (DDR1_CS3_CONFIG)	32	R/W	0000_0000h	11.4.2/500
80C0	Chip select n configuration 2 (DDR1_CS0_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
80C4	Chip select n configuration 2 (DDR1_CS1_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
80C8	Chip select n configuration 2 (DDR1_CS2_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
80CC	Chip select n configuration 2 (DDR1_CS3_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
8100	DDR SDRAM timing configuration 3 (DDR1_TIMING_CFG_3)	32	R/W	0000_0000h	11.4.4/503
8104	DDR SDRAM timing configuration 0 (DDR1_TIMING_CFG_0)	32	R/W	0011_0105h	11.4.5/505
8108	DDR SDRAM timing configuration 1 (DDR1_TIMING_CFG_1)	32	R/W	0000_0000h	11.4.6/508
810C	DDR SDRAM timing configuration 2 (DDR1_TIMING_CFG_2)	32	R/W	0000_0000h	11.4.7/512
8110	DDR SDRAM control configuration (DDR1_DDR_SDRAM_CFG)	32	R/W	0300_0000h	11.4.8/516
8114	DDR SDRAM control configuration 2 (DDR1_DDR_SDRAM_CFG_2)	32	R/W	0000_0000h	11.4.9/519
8118	DDR SDRAM mode configuration (DDR1_DDR_SDRAM_MODE)	32	R/W	0000_0000h	11.4.10/522
811C	DDR SDRAM mode configuration 2 (DDR1_DDR_SDRAM_MODE_2)	32	R/W	0000_0000h	11.4.11/522
8120	DDR SDRAM mode control (DDR1_DDR_SDRAM_MD_CNTL)	32	R/W	0000_0000h	11.4.12/523
8124	DDR SDRAM interval configuration (DDR1_DDR_SDRAM_INTERVAL)	32	R/W	0000_0000h	11.4.13/526
8128	DDR SDRAM data initialization (DDR1_DDR_DATA_INIT)	32	R/W	0000_0000h	11.4.14/527

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8130	DDR SDRAM clock control (DDR1_DDR_SDRAM_CLK_CNTL)	32	R/W	0200_0000h	11.4.15/527
8148	DDR training initialization address (DDR1_DDR_INIT_ADDR)	32	R/W	0000_0000h	11.4.16/528
814C	DDR training initialization extended address (DDR1_DDR_INIT_EXT_ADDRESS)	32	R/W	0000_0000h	11.4.17/529
8160	DDR SDRAM timing configuration 4 (DDR1_TIMING_CFG_4)	32	R/W	0000_0000h	11.4.18/530
8164	DDR SDRAM timing configuration 5 (DDR1_TIMING_CFG_5)	32	R/W	0000_0000h	11.4.19/533
8170	DDR ZQ calibration control (DDR1_DDR_ZQ_CNTL)	32	R/W	0000_0000h	11.4.20/535
8174	DDR write leveling control (DDR1_DDR_WRLVL_CNTL)	32	R/W	0000_0000h	11.4.21/537
817C	DDR Self Refresh Counter (DDR1_DDR_SR_CNTR)	32	R/W	0000_0000h	11.4.22/540
8180	DDR Register Control Words 1 (DDR1_DDR_SDRAM_RCW_1)	32	R/W	0000_0000h	11.4.23/541
8184	DDR Register Control Words 2 (DDR1_DDR_SDRAM_RCW_2)	32	R/W	0000_0000h	11.4.24/542
8190	DDR write leveling control 2 (DDR1_DDR_WRLVL_CNTL_2)	32	R/W	0000_0000h	11.4.25/543
8194	DDR write leveling control 3 (DDR1_DDR_WRLVL_CNTL_3)	32	R/W	0000_0000h	11.4.26/546
8200	DDR SDRAM mode configuration 3 (DDR1_DDR_SDRAM_MODE_3)	32	R/W	0000_0000h	11.4.27/548
8204	DDR SDRAM mode configuration 4 (DDR1_DDR_SDRAM_MODE_4)	32	R/W	0000_0000h	11.4.28/549
8208	DDR SDRAM mode configuration 5 (DDR1_DDR_SDRAM_MODE_5)	32	R/W	0000_0000h	11.4.29/550
820C	DDR SDRAM mode configuration 6 (DDR1_DDR_SDRAM_MODE_6)	32	R/W	0000_0000h	11.4.30/551
8210	DDR SDRAM mode configuration 7 (DDR1_DDR_SDRAM_MODE_7)	32	R/W	0000_0000h	11.4.31/551
8214	DDR SDRAM mode configuration 8 (DDR1_DDR_SDRAM_MODE_8)	32	R/W	0000_0000h	11.4.32/552
8B20	DDR Debug Status Register 1 (DDR1_DDRDSR_1)	32	R	0000_0000h	11.4.33/553
8B24	DDR Debug Status Register 2 (DDR1_DDRDSR_2)	32	R/W	0000_0000h	11.4.34/554
8B28	DDR Control Driver Register 1 (DDR1_DDRCDR_1)	32	R/W	0000_0000h	11.4.35/555
8B2C	DDR Control Driver Register 2 (DDR1_DDRCDR_2)	32	R/W	0000_0000h	11.4.36/558
8BF8	DDR IP block revision 1 (DDR1_DDR_IP_REV1)	32	R	See section	11.4.37/559
8BFC	DDR IP block revision 2 (DDR1_DDR_IP_REV2)	32	R	See section	11.4.38/560
8D00	DDR Memory Test Control Register (DDR1_DDR_MTCSR)	32	R/W	0000_0000h	11.4.39/561
8D20	DDR Memory Test Pattern n Register (DDR1_DDR_MTP1)	32	R/W	0000_0000h	11.4.40/562
8D24	DDR Memory Test Pattern n Register (DDR1_DDR_MTP2)	32	R/W	0000_0000h	11.4.40/562

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8D28	DDR Memory Test Pattern n Register (DDR1_DDR_MTP3)	32	R/W	0000_0000h	11.4.40/562
8D2C	DDR Memory Test Pattern n Register (DDR1_DDR_MTP4)	32	R/W	0000_0000h	11.4.40/562
8D30	DDR Memory Test Pattern n Register (DDR1_DDR_MTP5)	32	R/W	0000_0000h	11.4.40/562
8D34	DDR Memory Test Pattern n Register (DDR1_DDR_MTP6)	32	R/W	0000_0000h	11.4.40/562
8D38	DDR Memory Test Pattern n Register (DDR1_DDR_MTP7)	32	R/W	0000_0000h	11.4.40/562
8D3C	DDR Memory Test Pattern n Register (DDR1_DDR_MTP8)	32	R/W	0000_0000h	11.4.40/562
8D40	DDR Memory Test Pattern n Register (DDR1_DDR_MTP9)	32	R/W	0000_0000h	11.4.40/562
8D44	DDR Memory Test Pattern n Register (DDR1_DDR_MTP10)	32	R/W	0000_0000h	11.4.40/562
8E00	Memory data path error injection mask high (DDR1_DATA_ERR_INJECT_HI)	32	R/W	0000_0000h	11.4.41/563
8E04	Memory data path error injection mask low (DDR1_DATA_ERR_INJECT_LO)	32	R/W	0000_0000h	11.4.42/563
8E08	Memory data path error injection mask ECC (DDR1_ECC_ERR_INJECT)	32	R/W	0000_0000h	11.4.43/564
8E20	Memory data path read capture high (DDR1_CAPTURE_DATA_HI)	32	R/W	0000_0000h	11.4.44/565
8E24	Memory data path read capture low (DDR1_CAPTURE_DATA_LO)	32	R/W	0000_0000h	11.4.45/565
8E28	Memory data path read capture ECC (DDR1_CAPTURE_ECC)	32	R/W	0000_0000h	11.4.46/566
8E40	Memory error detect (DDR1_ERR_DETECT)	32	w1c	0000_0000h	11.4.47/567
8E44	Memory error disable (DDR1_ERR_DISABLE)	32	R/W	0000_0000h	11.4.48/569
8E48	Memory error interrupt enable (DDR1_ERR_INT_EN)	32	R/W	0000_0000h	11.4.49/571
8E4C	Memory error attributes capture (DDR1_CAPTURE_ATTRIBUTES)	32	R/W	0000_0000h	11.4.50/573
8E50	Memory error address capture (DDR1_CAPTURE_ADDRESS)	32	R/W	0000_0000h	11.4.51/574
8E54	Memory error extended address capture (DDR1_CAPTURE_EXT_ADDRESS)	32	R/W	0000_0000h	11.4.52/575
8E58	Single-Bit ECC memory error management (DDR1_ERR_SBE)	32	R/W	0000_0000h	11.4.53/575
9000	Chip select n memory bounds (DDR2_CS0_BNDS)	32	R/W	0000_0000h	11.4.1/499
9008	Chip select n memory bounds (DDR2_CS1_BNDS)	32	R/W	0000_0000h	11.4.1/499
9010	Chip select n memory bounds (DDR2_CS2_BNDS)	32	R/W	0000_0000h	11.4.1/499
9018	Chip select n memory bounds (DDR2_CS3_BNDS)	32	R/W	0000_0000h	11.4.1/499
9080	Chip select n configuration (DDR2_CS0_CONFIG)	32	R/W	0000_0000h	11.4.2/500
9084	Chip select n configuration (DDR2_CS1_CONFIG)	32	R/W	0000_0000h	11.4.2/500
9088	Chip select n configuration (DDR2_CS2_CONFIG)	32	R/W	0000_0000h	11.4.2/500
908C	Chip select n configuration (DDR2_CS3_CONFIG)	32	R/W	0000_0000h	11.4.2/500
90C0	Chip select n configuration 2 (DDR2_CS0_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
90C4	Chip select n configuration 2 (DDR2_CS1_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
90C8	Chip select n configuration 2 (DDR2_CS2_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
90CC	Chip select n configuration 2 (DDR2_CS3_CONFIG_2)	32	R/W	0000_0000h	11.4.3/502
9100	DDR SDRAM timing configuration 3 (DDR2_TIMING_CFG_3)	32	R/W	0000_0000h	11.4.4/503
9104	DDR SDRAM timing configuration 0 (DDR2_TIMING_CFG_0)	32	R/W	0011_0105h	11.4.5/505
9108	DDR SDRAM timing configuration 1 (DDR2_TIMING_CFG_1)	32	R/W	0000_0000h	11.4.6/508
910C	DDR SDRAM timing configuration 2 (DDR2_TIMING_CFG_2)	32	R/W	0000_0000h	11.4.7/512
9110	DDR SDRAM control configuration (DDR2_DDR_SDRAM_CFG)	32	R/W	0300_0000h	11.4.8/516
9114	DDR SDRAM control configuration 2 (DDR2_DDR_SDRAM_CFG_2)	32	R/W	0000_0000h	11.4.9/519
9118	DDR SDRAM mode configuration (DDR2_DDR_SDRAM_MODE)	32	R/W	0000_0000h	11.4.10/522
911C	DDR SDRAM mode configuration 2 (DDR2_DDR_SDRAM_MODE_2)	32	R/W	0000_0000h	11.4.11/522
9120	DDR SDRAM mode control (DDR2_DDR_SDRAM_MD_CNTL)	32	R/W	0000_0000h	11.4.12/523
9124	DDR SDRAM interval configuration (DDR2_DDR_SDRAM_INTERVAL)	32	R/W	0000_0000h	11.4.13/526
9128	DDR SDRAM data initialization (DDR2_DDR_DATA_INIT)	32	R/W	0000_0000h	11.4.14/527
9130	DDR SDRAM clock control (DDR2_DDR_SDRAM_CLK_CNTL)	32	R/W	0200_0000h	11.4.15/527
9148	DDR training initialization address (DDR2_DDR_INIT_ADDR)	32	R/W	0000_0000h	11.4.16/528
914C	DDR training initialization extended address (DDR2_DDR_INIT_EXT_ADDRESS)	32	R/W	0000_0000h	11.4.17/529
9160	DDR SDRAM timing configuration 4 (DDR2_TIMING_CFG_4)	32	R/W	0000_0000h	11.4.18/530
9164	DDR SDRAM timing configuration 5 (DDR2_TIMING_CFG_5)	32	R/W	0000_0000h	11.4.19/533
9170	DDR ZQ calibration control (DDR2_DDR_ZQ_CNTL)	32	R/W	0000_0000h	11.4.20/535
9174	DDR write leveling control (DDR2_DDR_WRLVL_CNTL)	32	R/W	0000_0000h	11.4.21/537
917C	DDR Self Refresh Counter (DDR2_DDR_SR_CNTR)	32	R/W	0000_0000h	11.4.22/540
9180	DDR Register Control Words 1 (DDR2_DDR_SDRAM_RCW_1)	32	R/W	0000_0000h	11.4.23/541
9184	DDR Register Control Words 2 (DDR2_DDR_SDRAM_RCW_2)	32	R/W	0000_0000h	11.4.24/542
9190	DDR write leveling control 2 (DDR2_DDR_WRLVL_CNTL_2)	32	R/W	0000_0000h	11.4.25/543

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9194	DDR write leveling control 3 (DDR2_DDR_WRLVL_CNTL_3)	32	R/W	0000_0000h	11.4.26/546
9200	DDR SDRAM mode configuration 3 (DDR2_DDR_SDRAM_MODE_3)	32	R/W	0000_0000h	11.4.27/548
9204	DDR SDRAM mode configuration 4 (DDR2_DDR_SDRAM_MODE_4)	32	R/W	0000_0000h	11.4.28/549
9208	DDR SDRAM mode configuration 5 (DDR2_DDR_SDRAM_MODE_5)	32	R/W	0000_0000h	11.4.29/550
920C	DDR SDRAM mode configuration 6 (DDR2_DDR_SDRAM_MODE_6)	32	R/W	0000_0000h	11.4.30/551
9210	DDR SDRAM mode configuration 7 (DDR2_DDR_SDRAM_MODE_7)	32	R/W	0000_0000h	11.4.31/551
9214	DDR SDRAM mode configuration 8 (DDR2_DDR_SDRAM_MODE_8)	32	R/W	0000_0000h	11.4.32/552
9B20	DDR Debug Status Register 1 (DDR2_DDRDSR_1)	32	R	0000_0000h	11.4.33/553
9B24	DDR Debug Status Register 2 (DDR2_DDRDSR_2)	32	R/W	0000_0000h	11.4.34/554
9B28	DDR Control Driver Register 1 (DDR2_DDRCDR_1)	32	R/W	0000_0000h	11.4.35/555
9B2C	DDR Control Driver Register 2 (DDR2_DDRCDR_2)	32	R/W	0000_0000h	11.4.36/558
9BF8	DDR IP block revision 1 (DDR2_DDR_IP_REV1)	32	R	See section	11.4.37/559
9BFC	DDR IP block revision 2 (DDR2_DDR_IP_REV2)	32	R	See section	11.4.38/560
9D00	DDR Memory Test Control Register (DDR2_DDR_MTCSR)	32	R/W	0000_0000h	11.4.39/561
9D20	DDR Memory Test Pattern n Register (DDR2_DDR_MTP1)	32	R/W	0000_0000h	11.4.40/562
9D24	DDR Memory Test Pattern n Register (DDR2_DDR_MTP2)	32	R/W	0000_0000h	11.4.40/562
9D28	DDR Memory Test Pattern n Register (DDR2_DDR_MTP3)	32	R/W	0000_0000h	11.4.40/562
9D2C	DDR Memory Test Pattern n Register (DDR2_DDR_MTP4)	32	R/W	0000_0000h	11.4.40/562
9D30	DDR Memory Test Pattern n Register (DDR2_DDR_MTP5)	32	R/W	0000_0000h	11.4.40/562
9D34	DDR Memory Test Pattern n Register (DDR2_DDR_MTP6)	32	R/W	0000_0000h	11.4.40/562
9D38	DDR Memory Test Pattern n Register (DDR2_DDR_MTP7)	32	R/W	0000_0000h	11.4.40/562
9D3C	DDR Memory Test Pattern n Register (DDR2_DDR_MTP8)	32	R/W	0000_0000h	11.4.40/562
9D40	DDR Memory Test Pattern n Register (DDR2_DDR_MTP9)	32	R/W	0000_0000h	11.4.40/562
9D44	DDR Memory Test Pattern n Register (DDR2_DDR_MTP10)	32	R/W	0000_0000h	11.4.40/562
9E00	Memory data path error injection mask high (DDR2_DATA_ERR_INJECT_HI)	32	R/W	0000_0000h	11.4.41/563
9E04	Memory data path error injection mask low (DDR2_DATA_ERR_INJECT_LO)	32	R/W	0000_0000h	11.4.42/563
9E08	Memory data path error injection mask ECC (DDR2_ECC_ERR_INJECT)	32	R/W	0000_0000h	11.4.43/564
9E20	Memory data path read capture high (DDR2_CAPTURE_DATA_HI)	32	R/W	0000_0000h	11.4.44/565
9E24	Memory data path read capture low (DDR2_CAPTURE_DATA_LO)	32	R/W	0000_0000h	11.4.45/565

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
9E28	Memory data path read capture ECC (DDR2_CAPTURE_ECC)	32	R/W	0000_0000h	11.4.46/566
9E40	Memory error detect (DDR2_ERR_DETECT)	32	w1c	0000_0000h	11.4.47/567
9E44	Memory error disable (DDR2_ERR_DISABLE)	32	R/W	0000_0000h	11.4.48/569
9E48	Memory error interrupt enable (DDR2_ERR_INT_EN)	32	R/W	0000_0000h	11.4.49/571
9E4C	Memory error attributes capture (DDR2_CAPTURE_ATTRIBUTES)	32	R/W	0000_0000h	11.4.50/573
9E50	Memory error address capture (DDR2_CAPTURE_ADDRESS)	32	R/W	0000_0000h	11.4.51/574
9E54	Memory error extended address capture (DDR2_CAPTURE_EXT_ADDRESS)	32	R/W	0000_0000h	11.4.52/575
9E58	Single-Bit ECC memory error management (DDR2_ERR_SBE)	32	R/W	0000_0000h	11.4.53/575

11.4.1 Chip select n memory bounds (DDRx_CS_n_BNDS)

The chip select bounds register s (CS_n_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip select s . Note that the size specified in CS_n_BNDS should equal the size of physical DRAM. Also, note that EA_n must be greater than or equal to SA_n .

If chip select interleaving is enabled, all fields in the lower interleaved chip select will be used, and the other chip selects' bounds registers will be unused. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS will be used, and all fields in CS1_BNDS will be unused.

Address: Base address + 0h offset + (8d × i), where i=0d to 3d

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

DDRx_CS_n_BNDS field descriptions

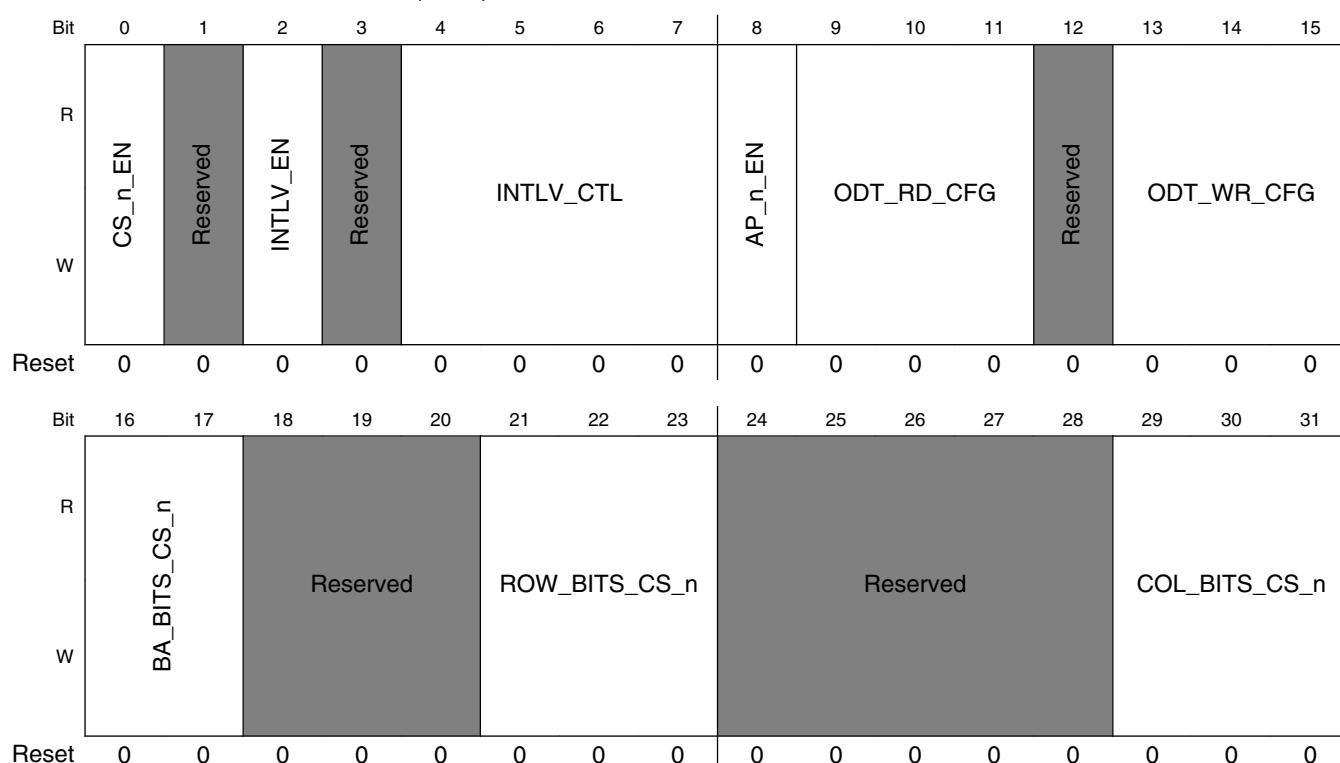
Field	Description
0–3 -	This field is reserved. Reserved
4–15 SAn	Starting address for chip select (bank) n. This value is compared against the 12 msbs of the 36-bit address.
16–19 -	This field is reserved. Reserved
20–31 EAn	Ending address for chip select (bank) n. This value is compared against the 12 msbs of the 36-bit address.

11.4.2 Chip select n configuration (DDR_x_CS_n_CONFIG)

The chip select configuration (CS_n_CONFIG) registers shown in the figure below enable the DDR chip select s and set the number of row and column bits used for each chip select. These register s should be loaded with the correct number of row and column bits for each SDRAM. Because CS_n_CONFIG[ROW_BITS_CS_n, COL_BITS_CS_n] establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused, with the exception of the ODT_RD_CFG and ODT_WR_CFG fields. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_CONFIG will be used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG will be used.

Address: Base address + 80h offset + (4d × i), where i=0d to 3d



DDR_x_CS_n_CONFIG field descriptions

Field	Description
0 CS _n _EN	Chip select n enable 0 Chip select n is not active 1 Chip select n is active and assumes the state set in CS _n _BNDS.

Table continues on the next page...

DDRx_CS_n_CONFIG field descriptions (continued)

Field	Description
1 -	This field is reserved. Reserved
2 INTLV_EN	Memory controller interleave enable Note: This field is only available in CS0_CONFIG. If memory controller interleaving is enabled, then the data bus widths must be programmed identically for the 2 memory controllers. If this bit is set, INTLV_CTL must be set to select the bit(s) used to control interleaving. 0 No interleaving 1 Interleaving between 2 memory controllers
3 -	This field is reserved. Reserved
4–7 INTLV_CTL	Interleaving control-Controls the bit(s) used to select the memory controller when memory controller interleaving is enabled (INTLV_EN is set) Note: This field is only available in CS0_CONFIG. 0000 Cache line interleaving. In this mode, bit 29 of the 36-bit physical address is used to select the memory controller. 0001 Page interleaving. In this mode, the bit used to select the controller is the bit to the left of the highest-order column bits. 0010 Bank interleaving. In this mode, the bit used to select the controller is the bit to the left of the bank select bits, which are to the left of the highest-order column bits. 0011 Super-bank interleaving. Super-bank interleaving should only be used if chip select interleaving is enabled. In this mode, the decoded address bit to the left of the chip select interleaved bit(s) is used for the memory controller interleaving. 0100-1111 Reserved
8 AP _n _EN	Chip select <i>n</i> auto-precharge enable 0 Chip select <i>n</i> will only be auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select <i>n</i> will always issue an auto-precharge for read and write transactions.
9–11 ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. ODT should only be used with DDR2andDDR3 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS F 010 Assert ODT only during reads to other chip selects — 011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads — 101 Assert ODT only during transactions to same DIMM 110 Assert ODT only during transactions to own CS and other DIMM. 111 Assert ODT only during transactions to other CS in same DIMM.
12 -	This field is reserved. Reserved
13–15 ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT _WR_CFG to be enabled. ODT should only be used with DDR2andDDR3 memories.

Table continues on the next page...

DDR_x_CS_n_CONFIG field descriptions (continued)

Field	Description
	000 Never assert ODT for writes 001 Assert ODT only during writes to CS <i>n</i> 010 Assert ODT only during writes to other chip selects — 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes — 101 Assert ODT only during transactions to same DIMM 110 Assert ODT only during transactions to own CS and other DIMM. 111 Assert ODT only during transactions to other CS in same DIMM.
16–17 BA_BITS_CS_ <i>n</i>	Number of bank bits for SDRAM on chip select <i>n</i> . These bits correspond to the sub-bank bits driven on MBA <i>n</i> DDR SDRAM Address Multiplexing . 00 2 logical bank bits 01 3 logical bank bits 10-11 Reserved
18–20 -	This field is reserved. Reserved
21–23 ROW_BITS_CS_ <i>n</i>	Number of row bits for SDRAM on chip select <i>n</i> . See DDR SDRAM Address Multiplexing for details. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits — — 101-111 Reserved
24–28 -	This field is reserved. Reserved
29–31 COL_BITS_CS_ <i>n</i>	Number of column bits for SDRAM on chip select <i>n</i> . For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100-111 Reserved

11.4.3 Chip select *n* configuration 2 (DDR_x_CS_n_CONFIG_2)

The chip select configuration (CS *n* _CONFIG_2) registers enable the partial array self refresh address decode in each chip select.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused.

Address: Base address + C0h offset + (4d × i), where i=0d to 3d

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

Reset 0

DDRx_CS_n_CONFIG_2 field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–7 PASR_CFG	Partial array self refresh config. Controls the bits that will be placed on MA[2:0] during the write to the EMRS(2) register when the automatic hardware DRAM initialization is used (DDR_SDRAM_CFG[B1] is cleared when DDR_SDRAM_CFG[MEM_EN] is set). If this field is a non-zero value, then it will override the least significant 3 bits in DDR_SDRAM_MODE_2[ESDMODE2] during the automatic initialization for chip select n . In addition, if a non-zero value is programmed in this field, then the address decode for chip select n will be optimized for partial array self refresh, as shown in DDR SDRAM Address Multiplexing . 000 Partial array self refresh is disabled 001-111 Partial array self refresh is enabled per JEDEC specifications. Overriding the least significant 3 bits of EMRS or EMRS(2) is only supported for DDR2 and DDR3 memory types.
8–31 -	This field is reserved. Reserved

11.4.4 DDR SDRAM timing configuration 3 (DDRx_TIMING_CFG_3)

DDR SDRAM timing configuration register 3 sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

Address: Base address + 100h offset

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

DDR_x_TIMING_CFG_3 field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 EXT_ACTTOPRE	Extended Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. This field is concatenated with TIMING_CFG_1[ACTTOPRE] to obtain a 5-bit value for the total activate to precharge. Note that a 5-bit value of 0_0000 is the same as a 5-bit value of 1_0000. Both values represent 16 cycles. 0 0 clocks 1 16 clocks
8–10 -	This field is reserved. Reserved
11–15 EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain an 9-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 9-bit value of the refresh recovery. $t_{RFC} = \{EXT_REFREC REFREC\} + 8$, such that t_{RFC} is calculated as follows: 00000 0 clocks 00001 16 clocks 00010 32 clocks 00011 48 clocks 00100 64 clocks 00101 80 clocks 00110 96 clocks 00111 112 clocks 01000 128 clocks 01001 144 clocks 01010 160 clocks 01011 176 clocks 01100 192 clocks 01101 208 clocks 01110 224 clocks 01111 240 clocks
16–18 -	This field is reserved. Reserved
19 EXT_CASLAT	Extended MCAS_B latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n+m$. This field is concatenated with TIMING_CFG_1[CASLAT] to obtain a 5-bit value for the total CAS latency. Note that if this bit is set, then 8 clocks are added to the programmed value in TIMING_CFG_1[CASLAT]. 0 0 clocks 1 8 clocks
20–28 -	This field is reserved. Reserved
29–31 CNTL_ADJ	Control Adjust. Controls the amount of delay to add to the lightly loaded control signals w/ respect to all other DRAM address and command signals. The signals affected by this field are MODT[0:3], MCS_B [0:3], and MCKE[0:3]

Table continues on the next page...

DDRx_TIMING_CFG_3 field descriptions (continued)

Field	Description	
	000	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched aligned with the other DRAM address and control signals.
	001	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched 1/2 platform cycle later than the other DRAM address and control signals.
	010	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched 1 platform cycle later than the other DRAM address and control signals.
	011	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched 3/2 platform cycles later than the other DRAM address and control signals.
	100	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched 2 platform cycles later than the other DRAM address and control signals.
	101	MODT[0:3], MCS_B [0:3], and MCKE[0:3] will be launched 5/2 platform cycles later than the other DRAM address and control signals.
	110-111	Reserved

11.4.5 DDR SDRAM timing configuration 0 (DDRx_TIMING_CFG_0)

DDR SDRAM timing configuration register 0 sets the number of clock cycles between various SDRAM control commands.

Address: Base address + 104h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	RWT	WRT	RRT	WWT	ACT_PD_EXIT	PRE_PD_EXIT	Reserved	ODT_PD_EXIT	Reserved	MRS_CYC																								
W	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1

DDRx_TIMING_CFG_0 field descriptions

Field	Description	
0-1 RWT	Read-to-write turnaround (t_{RTW_B}). Specifies how many extra cycles will be added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller will determine the read-to-write turnaround as $CL - WL + BL/2 + 2$. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length.	
	00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks	
2-3 WRT	Write-to-read turnaround. Specifies how many extra cycles will be added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller will determine the write-to-read turnaround as $WL - CL + BL/2 + 1$. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length.	

Table continues on the next page...

DDR_x_TIMING_CFG_0 field descriptions (continued)

Field	Description
	00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
4–5 RRT	<p>Read-to-read turnaround. Specifies how many extra cycles will be added between reads to different chip selects. As a default, 3 cycles will be required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles will be the default. Note that DDR2 does not support 8-beat bursts.</p> <p>00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks</p>
6–7 WWT	<p>Write-to-write turnaround. Specifies how many extra cycles will be added between writes to different chip selects. As a default, 2 cycles will be required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles will be the default. Note that DDR2 does not support 8-beat bursts.</p> <p>00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks</p>
8–11 ACT_PD_EXIT	<p>Active powerdown exit timing (DDR2:t_{XARD} and t_{XARDS}, DDR3:t_{XP}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command.</p> <p>0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1100 8 clocks 1101 9 clocks 1110 10 clocks 1111 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks</p>
12–16 PRE_PD_EXIT	<p>Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. Note the decoding for this field requires the least significant bit to be set to add 16 clocks.</p> <p>00000 Reserved 00010 1 clock 00100 2 clocks 00110 3 clocks</p>

Table continues on the next page...

DDRx_TIMING_CFG_0 field descriptions (continued)

Field	Description
	01000 4 clocks 01010 5 clocks 01100 6 clocks 01110 7 clocks 10000 8 clocks 10010 9 clocks 10100 10 clocks 10110 11 clocks 11000 12 clocks 11010 13 clocks 11100 14 clocks 11110 15 clocks 00001 16 clocks 00011 17 clocks 00101 18 clocks 00111 19 clocks 01001 20 clocks 01011 21 clocks 01101 22 clocks 01111 23 clocks 10001 24 clocks 10011 25 clocks 10101 26 clocks 10111 27 clocks 11001 28 clocks 11011 29 clocks 11101 30 clocks 11111 31 clocks
17–19 -	This field is reserved. Reserved
20–23 ODT_PD_EXIT	ODT powerdown exit timing (DDR2: t_{AXPD} , DDR3: set to 1). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. ODT_PD_EXIT must be greater than TIMING_CFG_5[RODT_ON] when using RODT_ON overrides and must be greater than TIMING_CFG_5[WODT_ON] when using WODT_ON overrides. 0000 0 clock 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks

Table continues on the next page...

DDR_x_TIMING_CFG_0 field descriptions (continued)

Field	Description
	1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27 -	This field is reserved. Reserved
28–31 MRS_CYC	Mode register set cycle time (t_{MRD}). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks

11.4.6 DDR SDRAM timing configuration 1 (DDR_x_TIMING_CFG_1)

DDR SDRAM timing configuration register 1 sets the number of clock cycles between various SDRAM control commands.

Address: Base address + 108h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRETOACT	ACTTOPRE	ACTTORW	CASLAT	REFREC	WRREC	ACTTOACT	WRTORD																								
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DDR_x_TIMING_CFG_1 field descriptions

Field	Description
0–3 PRETOACT	Precharge-to-activate interval (t_{RP}). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed. 0000 Reserved 0001 1 clock

Table continues on the next page...

DDRx_TIMING_CFG_1 field descriptions (continued)

Field	Description
	0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
4–7 ACTTOPRE	Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. This field is concatenated with TIMING_CFG_3[EXT_ACTTOPRE] to obtain a 5-bit value for the total activate to precharge time. Note that the decode of 0000-0011 is equal to 16-19 clocks when TIMING_CFG_3[EXT_ACTTOPRE] = 0, but it is equal to 0-3 clocks when TIMING_CFG_3[EXT_ACTTOPRE] = 1. 0000 16 clocks 0001 17 clocks 0010 18 clocks 0011 19 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks ... 1111 15 clocks
8–11 ACTTORW	Activate to read/write interval for SDRAM (t_{RCD}). Controls the number of clock cycles from an activate command until a read or write command is allowed. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks

Table continues on the next page...

DDR_x_TIMING_CFG_1 field descriptions (continued)

Field	Description
	1110 14 clocks 1111 15 clocks
12–15 CASLAT	MCAS_B latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n+m$. This field is concatenated with TIMING_CFG_3[EXT_CASLAT] to obtain a 5-bit value for the total CAS latency. This value must be programmed at initialization as described in DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2) .
	0000 Reserved 0001 1 clock 0010 1 . 5 clocks 0011 2 clocks 0100 2 . 5 clocks 0101 3 clocks 0110 3 . 5 clocks 0111 4 clocks 1000 4 . 5 clocks 1001 5 clocks 1010 5 . 5 clocks 1011 6 clocks 1100 6 . 5 clocks 1101 7 clocks 1110 7 . 5 clocks 1111 8 clocks
16–19 REFREC	Refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 9-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 9-bit value of the refresh recovery, such that t_{RFC} is calculated as follows: $t_{RFC} = \{EXT_REFREC \parallel REFREC\} + 8$. 0000 8 clocks 0001 9 clocks 0010 10 clocks 0011 11 clocks ... 1111 23 clocks
20–23 WRREC	Last data to precharge minimum interval (t_{WR}). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks

Table continues on the next page...

DDR_x_TIMING_CFG_1 field descriptions (continued)

Field	Description
	1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27 ACTTOACT	<p>Activate-to-activate interval (t_{RRD}). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select).</p> 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
28–31 WRTORD	<p>Last write data pair to read command issue interval (t_{WTR}). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank.</p> 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks

11.4.7 DDR SDRAM timing configuration 2 (DDR_x_TIMING_CFG_2)

DDR SDRAM timing configuration 2 sets the clock delay to data for writes.

Address: Base address + 10Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ADD_LAT				CPO				WR_LAT				Reserved		RD_TO_PRE	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RD_TO_PRE		WR_DATA_DELAY			CKE_PLS			FOUR_ACT							
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_x_TIMING_CFG_2 field descriptions

Field	Description
0–3 ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTOWR]. 0000 0 clocks 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 Reserved
4–8 CPO	MCAS_B -to-preamble override . Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. For these decodings, "READ_LAT" is equal to the CAS latency plus the additive latency. 00000 READ_LAT + 1 00001 Reserved 00010 READ_LAT 00011 READ_LAT + 1/4 00100 READ_LAT + 1/2 00101 READ_LAT + 3/4 00110 READ_LAT + 1

Table continues on the next page...

DDRx_TIMING_CFG_2 field descriptions (continued)

Field	Description
	00111 READ_LAT + 5/4 01000 READ_LAT + 3/2 01001 READ_LAT + 7/4 01010 READ_LAT + 2 01011 READ_LAT + 9/4 01100 READ_LAT + 5/2 01101 READ_LAT + 11/4 01110 READ_LAT + 3 01111 READ_LAT + 13/4 10000 READ_LAT + 7/2 10001 READ_LAT + 15/4 10010 READ_LAT + 4 10011 READ_LAT + 17/4 10100 READ_LAT + 9/2 10101 READ_LAT + 19/4 10110 READ_LAT + 5 10111 READ_LAT + 21/4 11000 READ_LAT + 11/2 11001 READ_LAT + 23/4 11010 READ_LAT + 6 11011 READ_LAT + 25/4 11100 READ_LAT + 13/2 11101 READ_LAT + 27/4 11110 READ_LAT + 7 11111 Automatic calibration (recommended)
9–12 WR_LAT	Write latency. Note that the total write latency for DDR2/ DDR3 is equal to WR_LAT + ADD_LAT; If a write latency of 1 is desired, then the additive latency must also be set to at least 1 cycle. <p> 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks </p>
13–14 -	This field is reserved. Reserved

Table continues on the next page...

DDRx_TIMING_CFG_2 field descriptions (continued)

Field	Description
15–18 RD_TO_PRE	Read to precharge (t_{RTP}). For DDR2, with a non-zero ADD_LAT value, takes a minimum of ADD_LAT + t_{RTP} cycles between read and precharge. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
19–22 WR_DATA_DELAY	Write command to write data strobe timing adjustment. Controls the amount of delay applied to the data and data strobes for writes. See DDR SDRAM Write Timing Adjustments for details. The write preamble will typically be driven high for 1/2 DRAM cycle, and then it will be driven low for 1/2 DRAM cycle. However, for WR_DATA_DELAY settings of 0 clocks and 1/4 clocks, the write preamble will be driven low for the entire DRAM cycle. If the preamble needs to switch high first (to meet DDR3 specifications), then these values should not be used. 0000 0 clock delay 0001 2 clock delay 0010 1/4 clock delay 0011 9/4 clock delay 0100 1/2 clock delay 0101 5/2 clock delay 0110 3/4 clock delay 0111 Reserved 1000 1 clock delay 1001 Reserved 1010 5/4 clock delay 1011 Reserved 1100 3/2 clock delay 1101 Reserved 1110 7/4 clock delay 1111 Reserved
23–25 CKE_PLS	Minimum CKE pulse width (t_{CKE}). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks

Table continues on the next page...

DDR_x_TIMING_CFG_2 field descriptions (continued)

Field	Description	
	100	4 clocks
	101	5 clocks
	110	6 clocks
	111	7 clocks
26–31 FOUR_ACT	Window for four activates (t_{FAW}). This is applied to DDR2/ DDR3 with eight logical banks only.	
	000000	Reserved
	000001	1 cycle
	000010	2 cycles
	000011	3 cycles
	000100	4 cycles
	...	
	011110	30 cycles
	011111	31 cycles
	100000	32 cycles
	100001-111111	Reserved

- For CPO decodings other than 00000 and 11111, 'READ_LAT' is rounded up to the next integer value.

11.4.8 DDR SDRAM control configuration (DDR_x_DDR_SDRAM_CFG)

The DDR SDRAM control configuration register enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Address: Base address + 110h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MEM_EN	SREN	ECC_EN	RD_EN	Reserved	SDRAM_TYPE	Reserved	DYN_PWR	DBW	8_BE	Reserved	3T_EN				
W																
Reset	0	0	0	0	0			0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	2T_EN												HSE	Reserved	MEM_HALT	BI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_x_DDR_SDRAM_CFG field descriptions

Field	Description
0 MEM_EN	DDR SDRAM interface logic enable. 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1 SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep . Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep . 1 SDRAM self refresh is enabled during sleep .
2 ECC_EN	ECC enable. Note that uncorrectable read errors may cause an interrupt. 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.

Table continues on the next page...

DDRx_DDR_SDRAM_CFG field descriptions (continued)

Field	Description
3 RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. Note that RD_EN and 2T_EN must not both be set at the same time. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs.
4 -	This field is reserved. Reserved
5–7 SDRAM_TYPE	Type of SDRAM device to be used. This field will be used when issuing the automatic hardware initialization sequence to DRAM via Mode Register Set and Extended Mode Register Set commands. Default value is 011 designating DDR2 SDRAM. 000-010 Reserved 011 DDR2 SDRAM 100-110 Reserved 111 DDR3 SDRAM
8–9 -	This field is reserved. Reserved
10 DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11–12 DBW	DRAM data bus width. — 00 64-bit bus is used. 01 32-bit bus is used. 10 Reserved — 11 Reserved
13 8_BE	8-beat burst enable. NOTE: DDR2 (SDRAM_TYPE = 011) must use 4-beat bursts, even when using 32-bit bus mode; DDR3 (SDRAM_TYPE = 111) must use 8-beat bursts when using 32-bit 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface.
14 -	This field is reserved. Reserved.
15 3T_EN	Enable 3T timing. This field cannot be set if DDR_SDRAM_CFG[2T_EN] is also set. This field cannot be used with a 32-bit bus if 4-beat bursts are used. NOTE: 3T timing may not be used with 4-beat bursts . 0 1T timing is enabled if 2T_EN is cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 3T timing is enabled. The DRAM command/address are held for 3 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the third cycle.
16 2T_EN	Enable 2T timing. This field should not be set if DDR_SDRAM_CFG[3T_EN] is set.

Table continues on the next page...

DDRx_DDR_SDRAM_CFG field descriptions (continued)

Field	Description
	<p>Note that RD_EN and 2T_EN must not both be set at the same time.</p> <p>0 1T timing is enabled if 3T_EN is cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus.</p> <p>1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle.</p>
17–23 BA_INTLV_CTL	<p>Bank (chip select) interleaving control. Set this field to use bank interleaving. ('x' denotes a don't care bit value. All unlisted field values are reserved.)</p> <p>0000000 No external memory banks are interleaved</p> <p>1000000 External memory banks 0 and 1 are interleaved</p> <p>0100000 External memory banks 2 and 3 are interleaved</p> <p>1100000 External memory banks 0 and 1 are interleaved together and banks 2 and 3 are interleaved together</p> <p>xx00100 External memory banks 0 through 3 are all interleaved together</p>
24–27 -	This field is reserved. Reserved
28 HSE	<p>Global half-strength override</p> <p>S</p> <p>Sets I/O driver impedance to half strength. This impedance will be used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in DDR Control Driver Register 1 (DDR_DDRCDR_1) and DDR Control Driver Register 2 (DDR_DDRCDR_2).</p> <p>This bit should be cleared if using automatic hardware calibration.</p> <p>0 I/O driver impedance will be configured to full strength.</p> <p>1 I/O driver impedance will be configured to half strength.</p>
29 -	This field is reserved. Reserved
30 MEM_HALT	<p>DDR memory controller halt. When this bit is set, the memory controller will not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software.</p> <p>0 DDR controller will accept new transactions.</p> <p>1 DDR controller will finish any remaining transactions, and then it will remain halted until this bit is cleared by software.</p>
31 BI	<p>Bypass initialization</p> <p>See DDR training initialization address (DDR_DDR_INIT_ADDR) for details on avoiding ECC errors in this mode.</p> <p>0 DDR controller will cycle through initialization routine based on SDRAM_TYPE</p> <p>1 Initialization routine will be bypassed. Software is responsible for initializing memory through DDR_SDRAM_MD_CTRL register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller will not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled.</p>

11.4.9 DDR SDRAM control configuration 2 (DDR_x_DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2 provides more control configuration for the DDR controller.

Address: Base address + 114h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FRC_SR	SR_IE	DLL_RST_DIS	Reserved	DQS_CFG		Reserved		ODT_CFG		Reserved		Reserved		Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_x_DDR_SDRAM_CFG_2 field descriptions

Field	Description
0 FRC_SR	Force self refresh 0 DDR controller will operate in normal mode. 1 DDR controller will enter self-refresh mode.
1 SR_IE	Self-refresh interrupt enable. The DDR controller can be placed into self refresh mode by forcing the PIC to assert <code>sie0</code> . This is considered a panic interrupt by the DDR controller, and it will enter self refresh as soon as possible. DDR_SDRAM_CFG[SREN] must also be set if the panic interrupt will be used. 0 DDR controller will not enter self-refresh mode if panic interrupt is asserted. 1 DDR controller will enter self-refresh mode if panic interrupt is asserted.
2 DLL_RST_DIS	DLL reset disable. If self-refresh is to be used with DDR3, this bit should be set. The DDR controller will typically issue a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization.

Table continues on the next page...

DDR_x_DDR_SDRAM_CFG_2 field descriptions (continued)

Field	Description
	0 DDR controller will issue a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller will not issue a DLL reset to the DRAMs when exiting self refresh.
3 -	This field is reserved. Reserved
4–5 DQS_CFG	DQS configuration 00 Reserved. 01 Differential DQS signals are used. 10 Reserved 11 Reserved
6–8 -	This field is reserved. Reserved
9–10 ODT_CFG	ODT configuration. This field defines how ODT will be driven to the on-chip IOs. See DDR Control Driver Register 1 and DDR Control Driver Register 2 (DDR_DDRCDR_2) which define the termination value that will be used. 00 Never assert ODT to internal IOs 01 Assert ODT to internal IOs only during writes to DRAM 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15 -	This field is reserved. Reserved
16–19 NUM_PR	Number of posted refreshes. This will determine how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum t _{ras} specification cannot be violated. 0000 Reserved 0001 1 refresh will be issued at a time 0010 2 refreshes will be issued at a time 0011 3 refreshes will be issued at a time 1000 8 refreshes will be issued at a time 1001-1111 Reserved
20–21 -	This field is reserved. Reserved
22 QD_EN	Quad-rank enable. Determines if a quad-ranked DIMM is used. This bit should also be set if quad-stacked discrete memory chips are used. 0 Quad-ranked DIMMs are not used. 1 Quad-ranked DIMMs are used. The controller will internally swap CKE[1] and CKE[2], along with MODT[1] and MODT[2].
23 UNQ_MRS_EN	Unique MRS Enable. Determines if the DDR_SDRAM_MODE_{3..8} registers will be used when initializing the memories for chip selects 1, 2, and 3. These can be used to provide unique values to the Mode Registers of the DRAM to allow different termination values for each rank.
24–25 -	This field is reserved. Reserved
26 AP_EN	Address Parity Enable. Determines if address parity will be generated and checked for the address and control signals when using registered DIMMs. If address parity is used, the MAPAR_OUT and MAPAR_ERR_B pins will be used to drive the parity bit and to receive errors from the open-drain parity

Table continues on the next page...

DDR_x_DDR_SDRAM_CFG_2 field descriptions (continued)

Field	Description
	<p>error signal. Even parity will be used, and parity will be generated for the MA[15:0], MBA[2:0], MRAS_B , MCAS_B , MWE_B signals. Parity will not be generated for the MCKE[0:3], MODT[0:3], or MCS_B [0:3] signals. Note that address parity should not be used for non-zero values of TIMING_CFG_3[CNTL_ADJ].</p> <ul style="list-style-type: none"> 0 Address parity will not be used 1 Address parity will be used
27 D_INIT	<p>DRAM data initialization This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller will automatically initialize DRAM after it is enabled. This bit will be automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle.</p> <ul style="list-style-type: none"> 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller will initialize memory once it is enabled. This bit will remain asserted until the initialization is complete. The value in DDR_DATA_INIT register will be used to initialize memory.
28 -	This field is reserved. Reserved
29 RCW_EN	<p>Register Control Word Enable. If DDR3 registered DIMMs are used, it may be necessary to write the register control words before issuing commands to DRAM. If this bit is set, the controller will write the register control words after DDR_SDRAM_CFG[MEM_EN] is set, unless DDR_SDRAM_CFG[B1] is set. The register control words will be written with the values in DDR_SDRAM_RCW_1 and DDR_SDRAM_RCW_2.</p> <ul style="list-style-type: none"> 0 Register control words will not be automatically written during DRAM initialization 1 Register control words will be automatically written during DRAM initialization. This bit should only be set if DDR3 registered DIMMs are used, and the default settings need to be modified.
30 CD_DIS	<p>Corrupted Data Disable. If this bit is set, then the corrupted data feature will be disabled. When the corrupted data feature is enabled, the DDR controller will invert the generated ECC code for any beat of data which is known to have corrupted data. When a read to the corrupted data is later generated, the <i>ERR_DETECT[CDE]</i> error will be set if error reporting is enabled.</p> <ul style="list-style-type: none"> 0 Corrupted data is enabled 1 Corrupted data is disabled
31 MD_EN	<p>Mirrored DIMM Enable. Some DDR3 DIMMs will be mirrored, where certain MA and MBA pins are mirrored on one side of the DIMM. When this bit is set, the controller will know to swap these signals before transmitting to the DRAM. The controller will assume that CS1 and CS3 are the 'mirrored' ranks of memory. The following signals are mirrored (MBA[0] vs. MBA[1]; MA[3] vs. MA[4]; MA[5] vs. MA[6]; MA[7] vs. MA[8]).</p> <ul style="list-style-type: none"> 0 Mirrored DIMMs are not used 1 Mirrored DIMMs are used

11.4.10 DDR SDRAM mode configuration (DDRx_DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register sets the values loaded into the DDR's mode registers.

Address: Base address + 118h offset

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

Reset 0

DDRx_DDR_SDRAM_MODE field descriptions

Field	Description
0–15 ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsbit of ESDMODE, which, in the big-endian convention shown in DDR SDRAM Mode Configuration , corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[B1] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.
16–31 SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsbit of SDMODE, which, in the big-endian convention shown in DDR SDRAM Mode Configuration , corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

11.4.11 DDR SDRAM mode configuration 2 (DDRx_DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register sets the values loaded into the DDR's extended mode 2 and 3 registers.

Address: Base address + 11Ch offset

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

Reset 0

DDR_x_DDR_SDRAM_MODE_2 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in DDR SDRAM Mode 2 Configuration , corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in DDR SDRAM Mode 2 Configuration , corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

11.4.12 DDR SDRAM mode control (DDR_x_DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signal s to a specific value

Before issuing a command via the DDR_SDRAM_MD_CNTL register, the DDR interface should be idle. This can be done by setting DDR_SDRAM_CFG[MEM_HALT] and disabling refreshes by clearing DDR_INTERVAL[REFINT]. If there are memory contents that need to be preserved during this time, then software should also force any required refresh commands while DDR_INTERVAL[REFINT] is cleared.

NOTE

Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields can be set at a time.

Table 11-105. Settings of DDR_SDRAM_MD_CNTL Fields

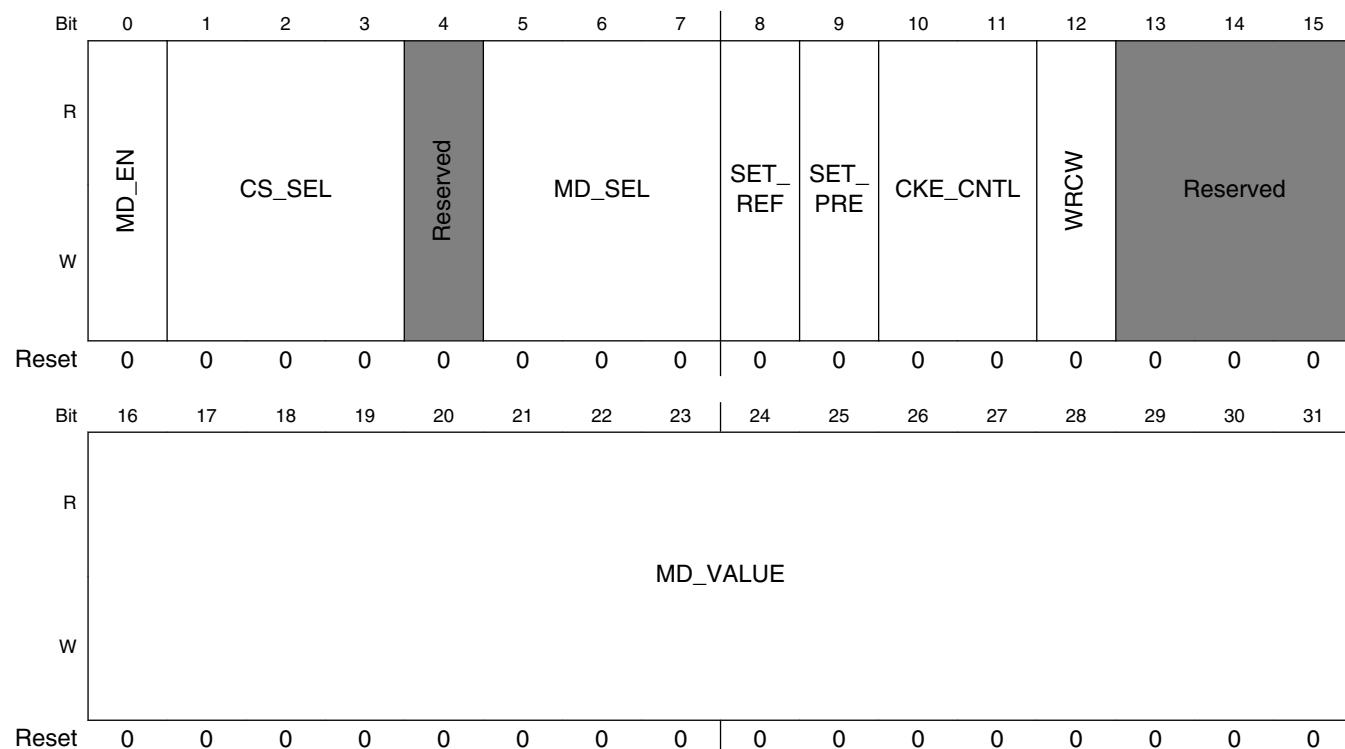
Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	-
SET_REF	0	1	0	-
SET_PRE	0	0	1	-
CS_SEL	Chooses chip select (CS)			-
MD_SEL	Select mode register. See DDR SDRAM Mode Control Register .	-	Selects logical bank	-

Table continues on the next page...

**Table 11-105. Settings of DDR_SDRAM_MD_CNTL Fields
(continued)**

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_VALUE	Value written to mode register	-	Only bit 5 is significant. See DDR SDRAM Mode Control Register .	-
CKE_CNTL	0	0	0	See DDR SDRAM Mode Control Register .

Address: Base address + 120h offset



DDRx_DDR_SDRAM_MD_CNTL field descriptions

Field	Description
0 MD_EN	<p>Mode enable. Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands:</p> <ul style="list-style-type: none"> • MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 <p>The specific command to be executed is selected by setting MD_SEL. In addition, the chip select must be chosen by setting CS_SEL. MD_EN is set by software and cleared by hardware once the command has been issued.</p>

Table continues on the next page...

DDR_x_DDR_SDRAM_MD_CNTL field descriptions (continued)

Field	Description																								
	<p>0 Indicates that no mode register set command needs to be issued.</p> <p>1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.</p>																								
1–3 CS_SEL	<p>Select chip select. Specifies the chip select that will be driven active due to any command forced by software in DDR_SDRAM_MD_CNTL.</p> <table> <tr><td>000</td><td>Chip select 0 is active</td></tr> <tr><td>001</td><td>Chip select 1 is active</td></tr> <tr><td>—</td><td>—</td></tr> <tr><td>010</td><td>Chip select 2 is active</td></tr> <tr><td>—</td><td>—</td></tr> <tr><td>011</td><td>Chip select 3 is active</td></tr> <tr><td>—</td><td>—</td></tr> <tr><td>100</td><td>Chip select 0 and chip select 1 are active</td></tr> <tr><td>—</td><td>—</td></tr> <tr><td>101</td><td>Chip select 2 and chip select 3 are active</td></tr> <tr><td>—</td><td>—</td></tr> <tr><td>110-111</td><td>Reserved</td></tr> </table>	000	Chip select 0 is active	001	Chip select 1 is active	—	—	010	Chip select 2 is active	—	—	011	Chip select 3 is active	—	—	100	Chip select 0 and chip select 1 are active	—	—	101	Chip select 2 and chip select 3 are active	—	—	110-111	Reserved
000	Chip select 0 is active																								
001	Chip select 1 is active																								
—	—																								
010	Chip select 2 is active																								
—	—																								
011	Chip select 3 is active																								
—	—																								
100	Chip select 0 and chip select 1 are active																								
—	—																								
101	Chip select 2 and chip select 3 are active																								
—	—																								
110-111	Reserved																								
4 -	This field is reserved. Reserved																								
5–7 MD_SEL	<p>Mode register select. MD_SEL specifies one of the following:</p> <ul style="list-style-type: none"> During a mode select command, selects the SDRAM mode register to be changed During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field. During a refresh command, this field is ignored. <p>Note that MD_SEL contains the value that will be presented onto the memory bank address pins (MBA <i>n</i>) of the DDR controller.</p> <table> <tr><td>000</td><td>MR</td></tr> <tr><td>001</td><td>EMR</td></tr> <tr><td>010</td><td>EMR2</td></tr> <tr><td>011</td><td>EMR3</td></tr> </table>	000	MR	001	EMR	010	EMR2	011	EMR3																
000	MR																								
001	EMR																								
010	EMR2																								
011	EMR3																								
8 SET_REF	<p>Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued.</p> <p>0 Indicates that no refresh command needs to be issued.</p> <p>1 Indicates that a refresh command is ready to be issued.</p>																								
9 SET_PRE	<p>Set precharge. Forces a precharge or precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued.</p> <p>0 Indicates that no precharge all command needs to be issued.</p> <p>1 Indicates that a precharge all command is ready to be issued.</p>																								
10–11 CKE_CNTL	Clock enable control. Allows software to globally clear or set the all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value will continue to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller will continue to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (that is, self refresh entry/exit, power down entry/exit).																								

Table continues on the next page...

DDR_x_DDR_SDRAM_MD_CNTL field descriptions (continued)

Field	Description
	00 CKE signals are not forced by software. 01 CKE signals are forced to a low value by software. 10 CKE signals are forced to a high value by software. 11 Reserved
12 WRCW	Write register control word. If software sets this bit, then a register control word will be written by asserting the selected chip selects while providing the programmed data on the MA and MBA signals. The RAS, CAS, and WE will remain deasserted during this write. The MD_EN field should also be set to force a register control word write. This should only be set if DDR3 registered DIMM's are used, and the register needs to be configured. If DDR_SDRAM_MD_CNTL is used to write RCW2 specifically, then software must guarantee that the timing parameter, <i>t-STAB</i> , is met before future accesses to the controller are allowed. In addition, DDR_SDRAM_MD_CNTL register cannot be used to write the RCWs if write leveling will be used, since write leveling is run automatically before DDR_SDRAM_MD_CNTL can be used to force RCW writes. 0 Indicates that a register control word write will not be issued if MD_EN is set. 1 Indicates that a register control word write will be issued if MD_EN is set.
13–15 -	This field is reserved. Reserved
16–31 MD_VALUE	Mode register value. This field, which specifies the value that will be presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

11.4.13 DDR SDRAM interval configuration (DDR_x_DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.

Address: Base address + 124h offset

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

DDR_x_DDR_SDRAM_INTERVAL field descriptions

Field	Description
0–15 REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes will not be issued when the REFINT is set to all 0s.
16–17 -	This field is reserved. Reserved
18–31 BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

11.4.14 DDR SDRAM data initialization (DDR_x_DDR_DATA_INIT)

The DDR SDRAM data initialization register provides the value that will be used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

Address: Base address + 128h offset

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

Reset 0

DDR_x_DDR_DATA_INIT field descriptions

Field	Description
0–31 INIT_VALUE	Initialization value. Represents the value that DRAM will be initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

11.4.15 DDR SDRAM clock control (DDR_x_DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register provides a 1/8-cycle clock adjustment.

Address: Base address + 130h offset

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

Reset 0 0 0 0 0 0 0 1 0

DDRx DDR SDRAM CLK CNTL field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–8 CLK_ADJUST	Clock adjust
0000	Clock is launched aligned with address/command
0001	Clock is launched 1/8 applied cycle after address/command
0010	Clock is launched 1/4 applied cycle after address/command
0011	Clock is launched 3/8 applied cycle after address/command
0100	Clock is launched 1/2 applied cycle after address/command
0101	Clock is launched 5/8 applied cycle after address/command
0110	Clock is launched 3/4 applied cycle after address/command
0111	Clock is launched 7/8 applied cycle after address/command
1000	Clock is launched 1 applied cycle after address/command
1001-1111	Reserved
9–31 -	This field is reserved. Reserved

11.4.16 DDR training initialization address (DDRx_DDR_INIT_ADDR)

The DDR SDRAM initialization address register provides the address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration after POR.

NOTE

After the skew adjustment, this address will contain bad ECC data. This is not important at POR, as all of memory should be subsequently initialized if ECC is enabled (either by software or through the use of DDR_SDRAM_CFG_2[D_INIT]).

NOTE

If an HRESET_B has been issued after the DRAM is in self-refresh mode, however, memory is not initialized, so this address should be written to using a 32-byte transaction to avoid possible ECC errors if this address could later be accessed.

Address: Base address + 148h offset

DDRx_DDR_INIT_ADDR field descriptions

Field	Description
0–31 INIT_ADDR	Initialization address. Represents the address that will be used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at POR. This address will be written to during the initialization sequence.

11.4.17 DDR training initialization extended address (DDRx_DDR_INIT_EXT_ADDRESS)

The DDR SDRAM initialization extended address register provides the extended address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration after POR.

Address: Base address + 14Ch offset

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

DDRx_DDR_INIT_EXT_ADDRESS field descriptions

Field	Description
0 UIA	Use initialization address. 0 Use the default address for training sequence as calculated by the controller. This will be the first valid address in each enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR and DDR_INIT_EXT_ADDR.
1–27 -	This field is reserved. Reserved
28–31 INIT_EXT_ADDR	Initialization extended address. Represents the extended address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration at POR. This extended address will be written to during the initialization sequence.

11.4.18 DDR SDRAM timing configuration 4 (DDRx_TIMING_CFG_4)

The DDR SDRAM timing configuration 4 register provides additional timing fields required to support DDR3 memories.

Address: Base address + 160h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RWT				WRT				RRT				WWT			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved												DLL_LOCK			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRx_TIMING_CFG_4 field descriptions

Field	Description																																
0–3 RWT	<p>Read-to-write turnaround for same chip select. Specifies how many cycles will be added between a read to write turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[RWT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[RWT] will also be met before issuing a write command.</p> <table> <tr><td>0000</td><td>Default</td></tr> <tr><td>0001</td><td>1 clock</td></tr> <tr><td>0010</td><td>2 clocks</td></tr> <tr><td>0011</td><td>3 clocks</td></tr> <tr><td>0100</td><td>4 clocks</td></tr> <tr><td>0101</td><td>5 clocks</td></tr> <tr><td>0110</td><td>6 clocks</td></tr> <tr><td>0111</td><td>7 clocks</td></tr> <tr><td>1000</td><td>8 clocks</td></tr> <tr><td>1001</td><td>9 clocks</td></tr> <tr><td>1010</td><td>10 clocks</td></tr> <tr><td>1011</td><td>11 clocks</td></tr> <tr><td>1100</td><td>12 clocks</td></tr> <tr><td>1101</td><td>13 clocks</td></tr> <tr><td>1110</td><td>14 clocks</td></tr> <tr><td>1111</td><td>15 clocks</td></tr> </table>	0000	Default	0001	1 clock	0010	2 clocks	0011	3 clocks	0100	4 clocks	0101	5 clocks	0110	6 clocks	0111	7 clocks	1000	8 clocks	1001	9 clocks	1010	10 clocks	1011	11 clocks	1100	12 clocks	1101	13 clocks	1110	14 clocks	1111	15 clocks
0000	Default																																
0001	1 clock																																
0010	2 clocks																																
0011	3 clocks																																
0100	4 clocks																																
0101	5 clocks																																
0110	6 clocks																																
0111	7 clocks																																
1000	8 clocks																																
1001	9 clocks																																
1010	10 clocks																																
1011	11 clocks																																
1100	12 clocks																																
1101	13 clocks																																
1110	14 clocks																																
1111	15 clocks																																
4–7 WRT	<p>Write-to-read turnaround for same chip select. Specifies how many cycles will be added between a write to read turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[WRT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip</p>																																

Table continues on the next page...

DDRx_TIMING_CFG_4 field descriptions (continued)

Field	Description																																
	<p>selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[WRT] will also be met before issuing a read command.</p> <table> <tr><td>0000</td><td>Default</td></tr> <tr><td>0001</td><td>1 clock</td></tr> <tr><td>0010</td><td>2 clocks</td></tr> <tr><td>0011</td><td>3 clocks</td></tr> <tr><td>0100</td><td>4 clocks</td></tr> <tr><td>0101</td><td>5 clocks</td></tr> <tr><td>0110</td><td>6 clocks</td></tr> <tr><td>0111</td><td>7 clocks</td></tr> <tr><td>1000</td><td>8 clocks</td></tr> <tr><td>1001</td><td>9 clocks</td></tr> <tr><td>1010</td><td>10 clocks</td></tr> <tr><td>1011</td><td>11 clocks</td></tr> <tr><td>1100</td><td>12 clocks</td></tr> <tr><td>1101</td><td>13 clocks</td></tr> <tr><td>1110</td><td>14 clocks</td></tr> <tr><td>1111</td><td>15 clocks</td></tr> </table>	0000	Default	0001	1 clock	0010	2 clocks	0011	3 clocks	0100	4 clocks	0101	5 clocks	0110	6 clocks	0111	7 clocks	1000	8 clocks	1001	9 clocks	1010	10 clocks	1011	11 clocks	1100	12 clocks	1101	13 clocks	1110	14 clocks	1111	15 clocks
0000	Default																																
0001	1 clock																																
0010	2 clocks																																
0011	3 clocks																																
0100	4 clocks																																
0101	5 clocks																																
0110	6 clocks																																
0111	7 clocks																																
1000	8 clocks																																
1001	9 clocks																																
1010	10 clocks																																
1011	11 clocks																																
1100	12 clocks																																
1101	13 clocks																																
1110	14 clocks																																
1111	15 clocks																																
8–11 RRT	<p>Read-to-read turnaround for same chip select. Specifies how many cycles will be added between reads to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between read commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for read-to-read transactions to the same chip select.</p> <table> <tr><td>0000</td><td>BL/2 clocks</td></tr> <tr><td>0001</td><td>BL/2 + 1 clock</td></tr> <tr><td>0010</td><td>BL/2 + 2 clocks</td></tr> <tr><td>0011</td><td>BL/2 + 3 clocks</td></tr> <tr><td>0100</td><td>BL/2 + 4 clocks</td></tr> <tr><td>0101</td><td>BL/2 + 5 clocks</td></tr> <tr><td>0110</td><td>BL/2 + 6 clocks</td></tr> <tr><td>0111</td><td>BL/2 + 7 clocks</td></tr> <tr><td>1000</td><td>BL/2 + 8 clocks</td></tr> <tr><td>1001</td><td>BL/2 + 9 clocks</td></tr> <tr><td>1010</td><td>BL/2 + 10 clocks</td></tr> <tr><td>1011</td><td>BL/2 + 11 clocks</td></tr> <tr><td>1100</td><td>BL/2 + 12 clocks</td></tr> <tr><td>1101</td><td>BL/2 + 13 clocks</td></tr> <tr><td>1110</td><td>BL/2 + 14 clocks</td></tr> <tr><td>1111</td><td>BL/2 + 15 clocks</td></tr> </table>	0000	BL/2 clocks	0001	BL/2 + 1 clock	0010	BL/2 + 2 clocks	0011	BL/2 + 3 clocks	0100	BL/2 + 4 clocks	0101	BL/2 + 5 clocks	0110	BL/2 + 6 clocks	0111	BL/2 + 7 clocks	1000	BL/2 + 8 clocks	1001	BL/2 + 9 clocks	1010	BL/2 + 10 clocks	1011	BL/2 + 11 clocks	1100	BL/2 + 12 clocks	1101	BL/2 + 13 clocks	1110	BL/2 + 14 clocks	1111	BL/2 + 15 clocks
0000	BL/2 clocks																																
0001	BL/2 + 1 clock																																
0010	BL/2 + 2 clocks																																
0011	BL/2 + 3 clocks																																
0100	BL/2 + 4 clocks																																
0101	BL/2 + 5 clocks																																
0110	BL/2 + 6 clocks																																
0111	BL/2 + 7 clocks																																
1000	BL/2 + 8 clocks																																
1001	BL/2 + 9 clocks																																
1010	BL/2 + 10 clocks																																
1011	BL/2 + 11 clocks																																
1100	BL/2 + 12 clocks																																
1101	BL/2 + 13 clocks																																
1110	BL/2 + 14 clocks																																
1111	BL/2 + 15 clocks																																
12–15 WWT	<p>Write-to-write turnaround for same chip select. Specifies how many cycles will be added between writes to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between write commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for write-to-write transactions to the same chip select.</p>																																

Table continues on the next page...

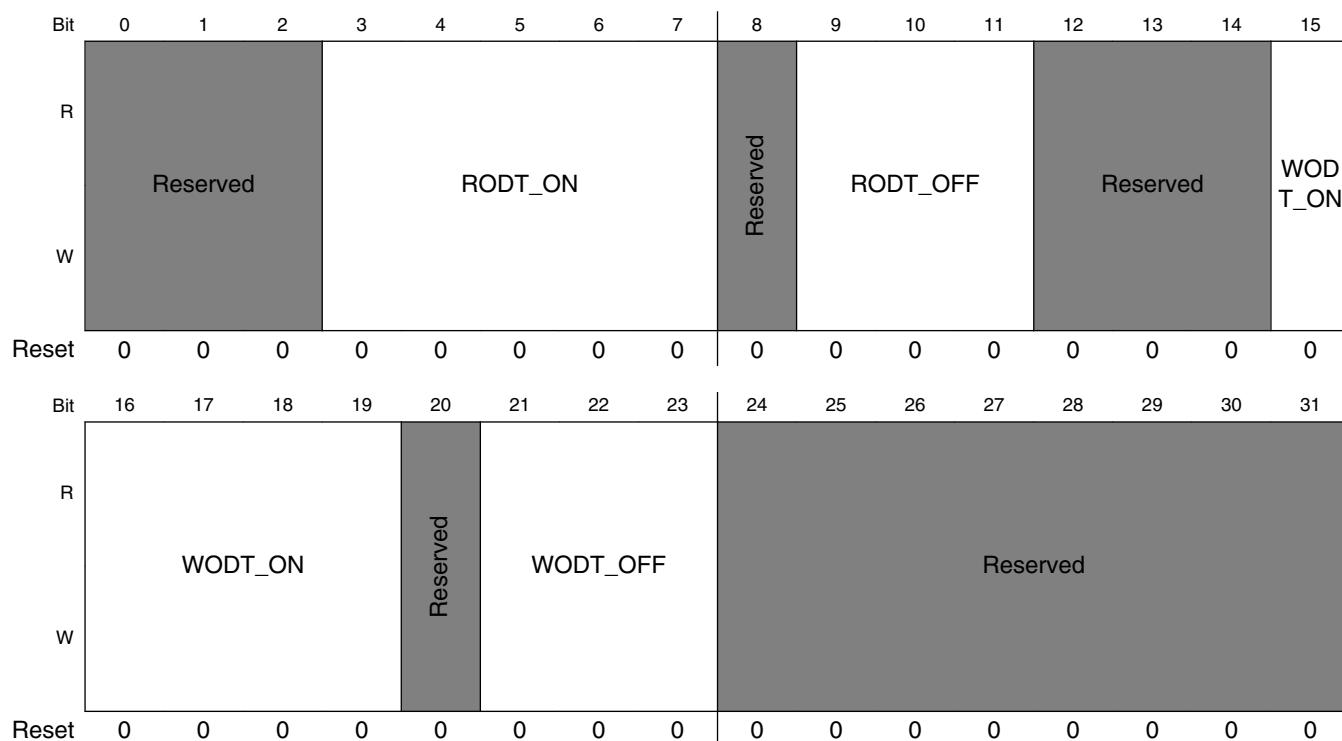
DDR_x_TIMING_CFG_4 field descriptions (continued)

Field	Description
	0000 BL/2 clocks 0001 BL/2 + 1 clock 0010 BL/2 + 2 clocks 0011 BL/2 + 3 clocks 0100 BL/2 + 4 clocks 0101 BL/2 + 5 clocks 0110 BL/2 + 6 clocks 0111 BL/2 + 7 clocks 1000 BL/2 + 8 clocks 1001 BL/2 + 9 clocks 1010 BL/2 + 10 clocks 1011 BL/2 + 11 clocks 1100 BL/2 + 12 clocks 1101 BL/2 + 13 clocks 1110 BL/2 + 14 clocks 1111 BL/2 + 15 clocks
16–29 -	This field is reserved. Reserved
30–31 DLL_LOCK	DDR SDRAM DLL Lock Time. This provides the number of cycles that it will take for the DRAMs DLL to lock at POR and after exiting self refresh. The controller will wait the specified number of cycles before issuing any commands after exiting POR or self refresh. 00 200 clocks 01 512 clocks 10 Reserved 11 Reserved

11.4.19 DDR SDRAM timing configuration 5 (DDR_x_TIMING_CFG_5)

The DDR SDRAM timing configuration 5 register provides additional timing fields required to support DDR3 memories.

Address: Base address + 164h offset



DDR_x_TIMING_CFG_5 field descriptions

Field	Description														
0–2 -	This field is reserved. Reserved														
3–7 RODT_ON	Read to ODT on. Specifies the number of cycles that will pass from when a read command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of RL - 3 cycles to support legacy of past products. RL is the read latency, derived from CAS latency + additive latency. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field. <table border="0"> <tr> <td>00000</td> <td>RL - 3 clocks</td> </tr> <tr> <td>00001</td> <td>0 clocks</td> </tr> <tr> <td>00010</td> <td>1 clocks</td> </tr> <tr> <td>00011</td> <td>2 clocks</td> </tr> <tr> <td>011111</td> <td>14 clocks</td> </tr> <tr> <td>10000</td> <td>15 clocks</td> </tr> <tr> <td>10001</td> <td>16 clocks</td> </tr> </table>	00000	RL - 3 clocks	00001	0 clocks	00010	1 clocks	00011	2 clocks	011111	14 clocks	10000	15 clocks	10001	16 clocks
00000	RL - 3 clocks														
00001	0 clocks														
00010	1 clocks														
00011	2 clocks														
011111	14 clocks														
10000	15 clocks														
10001	16 clocks														

Table continues on the next page...

DDRx_TIMING_CFG_5 field descriptions (continued)

Field	Description	
	10010	17 clocks
	10011	18 clocks
	11111	30 clocks
8 -	This field is reserved. Reserved	
9–11 RODT_OFF	Read to ODT off. Specifies the number of cycles that the relevant ODT signal(s) will remain asserted for each read transaction. The default case (000) will leave the ODT signal(s) asserted for 3 DRAM cycles.	
	000	3 clocks
	001	1 clock
	010	2 clocks
	010	3 clocks
	100	4 clocks
	101	5 clocks
	110	6 clocks
	111	7 clocks
12–14 -	This field is reserved. Reserved	
15–19 WODT_ON	Write to ODT On Specifies the number of cycles that will pass from when a write command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of WL - 3 cycles to support legacy of past products. WL is the write latency, derived from Write Latency + Additive Latency. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field.	
	00000	WL - 3 clocks
	00001	0 clocks
	00010	1 clocks
	00011	2 clocks
	01111	14 clocks
	10000	15 clocks
	10001	16 clocks
	10010	17 clocks
	10011	18 clocks
	11111	30 clocks
20 -	This field is reserved. Reserved	
21–23 WODT_OFF	Write to ODT Off. Specifies the number of cycles that the relevant ODT signal(s) will remain asserted for each write transaction. The default case (000) will leave the ODT signal(s) asserted for 3 DRAM cycles.	
	000	3 clocks
	001	1 clock
	010	2 clocks
	010	3 clocks
	100	4 clocks
	101	5 clocks
	110	6 clocks
	111	7 clocks

Table continues on the next page...

DDRx_TIMING_CFG_5 field descriptions (continued)

Field	Description
24–31 -	This field is reserved. Reserved

11.4.20 DDR ZQ calibration control (DDRx_DDR_ZQ_CNTL)

The DDR ZQ Calibration Control register provides the enable and controls required for ZQ calibration when using DDR3 SDRAM devices.

There is a limitation for various DRAM timing parameters when ZQ calibration is used. The factors involved in this limitation are DDR_ZQ_CNTL[ZQOPER], DDR_ZQ_CNTL[ZQCS], TIMING_CFG_1[PRETOACT], TIMING_CFG_1[REFREC], DDR_SDRAM_INTERVAL[REFINT], and the number of chip selects enabled. If the following condition is true:

$$[((\text{DDR_ZQ_CNTL[ZQOPER]} + \text{DDR_ZQ_CNTL[ZQCS]}) * (\# \text{ enabled chip selects})) + \text{TIMING_CFG_1[PRETOACT]} + \text{TIMING_CFG_1[REFREC]} + 2t_{CK}] > (\text{DDR_SDRAM_INTERVAL[REFINT]}),$$

it is possible that one refresh will be skipped when the controller is exiting self refresh. If this is an issue, then posted refreshes could be used to extend the refresh interval.

Another alternative is to use the DDR_SDRAM_MD_CNTL register to force an extra refresh to each chip select after exiting self refresh mode. However, DDR3 timing parameters for most devices/frequencies will not allow for a refresh to be missed.

Address: Base address + 170h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ZQ_EN	Reserved			ZQINIT			Reserved			ZQOPER					
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				ZQCS				Reserved							
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRx_DDR_ZQ_CNTL field descriptions

Field	Description
0 ZQ_EN	ZQ Calibration Enable. This bit determines if ZQ calibrating will be used. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111).

Table continues on the next page...

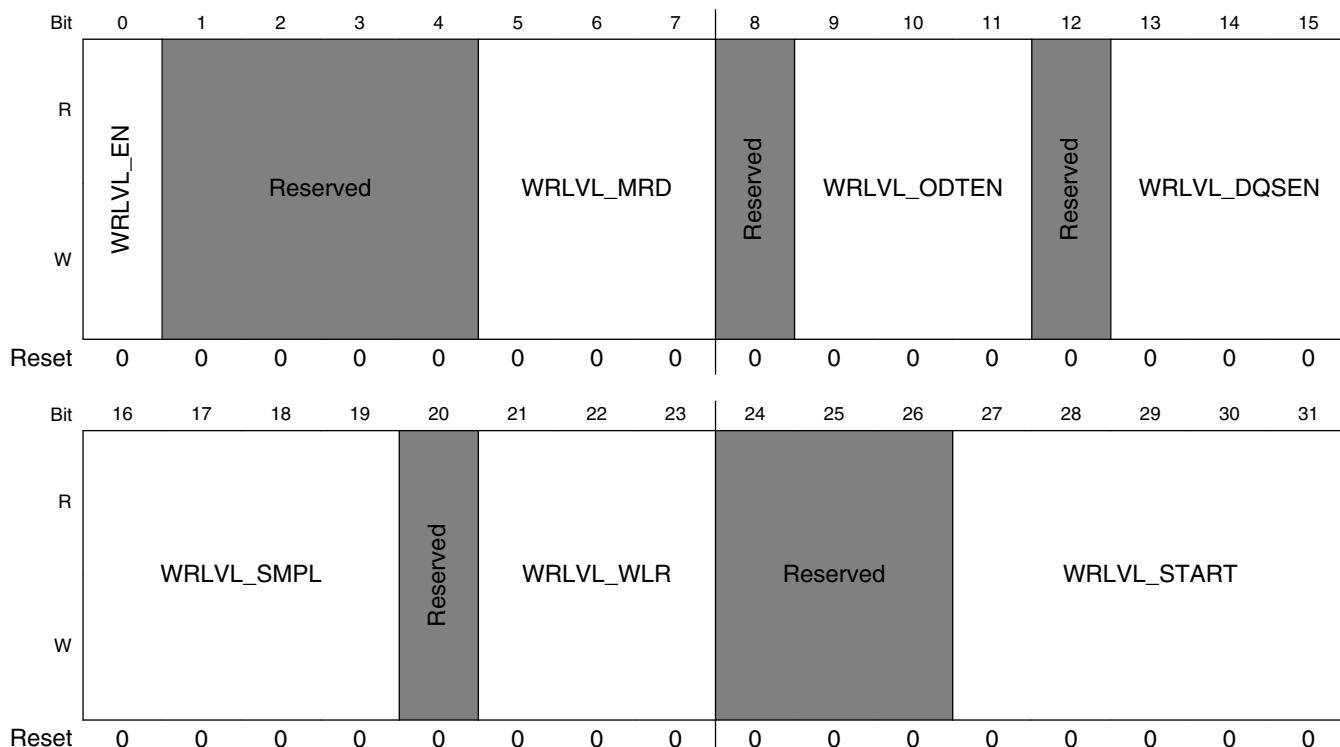
DDRx_DDR_ZQ_CNTL field descriptions (continued)

Field	Description																						
	<p>0 ZQ Calibration will not be used.</p> <p>1 ZQ Calibration will be used. A ZQCL command will be issued by the DDR controller after POR and anytime the DDR controller is exiting self refresh. A ZQCS command will be issued every 32 refresh sequences to account for VT variations.</p>																						
1–3 -	This field is reserved. Reserved																						
4–7 ZQINIT	<p>POR ZQ Calibration Time (t_{zqinit}). Determines the number of cycles that must be allowed for DRAM ZQ calibration at POR. Each chip select will be calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued.</p> <table> <tr><td>0000-0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>128 clocks</td></tr> <tr><td>1000</td><td>256 clocks</td></tr> <tr><td>1001</td><td>512 clocks</td></tr> <tr><td>1010</td><td>1024 clocks</td></tr> <tr><td>1011-1111</td><td>Reserved</td></tr> </table>	0000-0110	Reserved	0111	128 clocks	1000	256 clocks	1001	512 clocks	1010	1024 clocks	1011-1111	Reserved										
0000-0110	Reserved																						
0111	128 clocks																						
1000	256 clocks																						
1001	512 clocks																						
1010	1024 clocks																						
1011-1111	Reserved																						
8–11 -	This field is reserved. Reserved																						
12–15 ZQOPER	<p>Normal Operation Full Calibration Time (t_{zqoper}). Determines the number of cycles that must be allowed for DRAM ZQ calibration when exiting self refresh. Each chip select will be calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued.</p> <table> <tr><td>0000-0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>128 clocks</td></tr> <tr><td>1000</td><td>256 clocks</td></tr> <tr><td>1001</td><td>512 clocks</td></tr> <tr><td>1010</td><td>1024 clocks</td></tr> <tr><td>1011-1111</td><td>Reserved</td></tr> </table>	0000-0110	Reserved	0111	128 clocks	1000	256 clocks	1001	512 clocks	1010	1024 clocks	1011-1111	Reserved										
0000-0110	Reserved																						
0111	128 clocks																						
1000	256 clocks																						
1001	512 clocks																						
1010	1024 clocks																						
1011-1111	Reserved																						
16–19 -	This field is reserved. Reserved																						
20–23 ZQCS	<p>Normal Operation Short Calibration Time (t_{zqcs}). Determines the number of cycles that must be allowed for DRAM ZQ calibration during dynamic calibration which is issued every 32 refresh cycles. Each chip select will be calibrated separately, and this time must elapse after the ZQCS command is issued for each chip select before a separate command may be issued.</p> <table> <tr><td>0000</td><td>1 clocks</td></tr> <tr><td>0001</td><td>2 clocks</td></tr> <tr><td>0010</td><td>4 clocks</td></tr> <tr><td>0011</td><td>8 clocks</td></tr> <tr><td>0100</td><td>16 clocks</td></tr> <tr><td>0101</td><td>32 clocks</td></tr> <tr><td>0110</td><td>64 clocks</td></tr> <tr><td>0111</td><td>128 clocks</td></tr> <tr><td>1000</td><td>256 clocks</td></tr> <tr><td>1001</td><td>512 clocks</td></tr> <tr><td>1010-1111</td><td>Reserved</td></tr> </table>	0000	1 clocks	0001	2 clocks	0010	4 clocks	0011	8 clocks	0100	16 clocks	0101	32 clocks	0110	64 clocks	0111	128 clocks	1000	256 clocks	1001	512 clocks	1010-1111	Reserved
0000	1 clocks																						
0001	2 clocks																						
0010	4 clocks																						
0011	8 clocks																						
0100	16 clocks																						
0101	32 clocks																						
0110	64 clocks																						
0111	128 clocks																						
1000	256 clocks																						
1001	512 clocks																						
1010-1111	Reserved																						
24–31 -	This field is reserved. Reserved																						

11.4.21 DDR write leveling control (DDRx_DDR_WRLVL_CNTL)

The DDR Write Leveling Control register, shown in [DDR Write Leveling Control](#), provides controls for write leveling, as it is supported for DDR3 memory devices.

Address: Base address + 174h offset



DDRx_DDR_WRLVL_CNTL field descriptions

Field	Description
0 WRLVL_EN	Write Leveling Enable. This bit determines if write leveling will be used. If this bit is set, then the DDR controller will perform write leveling immediately after initializing the DRAM. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111). 0 Write leveling will not be used 1 Write leveling will be used
1–4 -	This field is reserved. Reserved
5–7 WRLVL_MRD	First DQS pulse rising edge after margining mode is programmed (t_{WL_MRD}). Determines how many cycles to wait after margining mode has been programmed before the first DQS pulse may be issued. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks

Table continues on the next page...

DDR_x_DDR_WRLVL_CNTL field descriptions (continued)

Field	Description
	011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
8 -	This field is reserved. Reserved
9–11 WRLVL_ODTEN	ODT delay after margining mode is programmed (t_{WL_ODTEN}). Determines how many cycles to wait after margining mode has been programmed.until ODT may be asserted.This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
12 -	This field is reserved. Reserved
13–15 WRLVL_DQSEN	DQS/ DQS_B delay after margining mode is programmed (t_{WL_DQSEN}). Determines how many cycles to wait after margining mode has been programmed.until DQS may be actively driven. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
16–19 WRLVL_SMPL	Write leveling sample time. Determines the number of cycles that must pass before the data signals are sampled after a DQS pulse during margining mode. This field should be programmed at least 6 cycles higher than t_{WLO} to allow enough time for propagation delay and sampling of the prime data bits. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 0000 32 clocks 0001 1 clocks 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks

Table continues on the next page...

DDR_x_DDR_WRLVL_CNTL field descriptions (continued)

Field	Description
	1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
20 -	This field is reserved. Reserved
21–23 WRLVL_WLR	Write leveling repetition time. Determines the number of cycles that must pass between DQS pulses during write leveling. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
24–26 -	This field is reserved. Reserved
27–31 WRLVL_START	Write leveling start time for DQS[0]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 0 clock delay 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 c'lock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101-11111 Reserved

11.4.22 DDR Self Refresh Counter (DDRx_DDR_SR_CNTR)

The DDR Self Refresh Counter register can be programmed to force the DDR controller to enter self refresh after a predefined period of idle time.

Address: Base address + 17Ch offset

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

DDRx_DDR_SR_CNTR field descriptions

Field	Description
0–11 -	This field is reserved. Reserved
12–15 SR_IT	Self Refresh Idle Threshold. Defines the number of DRAM cycles that must pass while the DDR controller is idle before it will enter self refresh. Anytime a transaction is issued to the DDR controller, it will reset its internal counter. When a new transaction is received by the DDR controller, it will exit self refresh and reset its internal counter. If this field is zero, then the described power savings feature will be disabled. In addition, if a non-zero value is programmed into this field, then the DDR controller will exit self refresh anytime a transaction is issued to the DDR controller, regardless of the reason self refresh was initially entered. 0000 Automatic self refresh entry disabled 0001 Reserved ^{2^10} DRAM data rate (DDR2 only) 0010 2 ¹² DRAM data rate 0011 2 ¹⁴ DRAM data rate 0100 2 ¹⁶ DRAM data rate 0101 2 ¹⁸ DRAM data rate 0110 2 ²⁰ DRAM data rate 0111 2 ²² DRAM data rate 1000 2 ²⁴ DRAM data rate 1001 2 ²⁶ DRAM data rate 1010 2 ²⁸ DRAM data rate 1011 2 ³⁰ DRAM data rate 1100-1111 Reserved
16–31 -	This field is reserved. Reserved

11.4.23 DDR Register Control Words 1 (DDR_x_DDR_SDRAM_RCW_1)

The DDR Register Control Word 1 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG[RCW_EN] is set. Each 4-bit field represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Address: Base address + 180h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RCW0	RCW1	RCW2	RCW3		RCW4	RCW5	RCW6	RCW7																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DDR_x_DDR_SDRAM_RCW_1 field descriptions

Field	Description
0–3 RCW0	Register Control Word 0. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 0.
4–7 RCW1	Register Control Word 1. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 1.
8–11 RCW2	Register Control Word 2. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 2.
12–15 RCW3	Register Control Word 3. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 3.
16–19 RCW4	Register Control Word 4. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 4.
20–23 RCW5	Register Control Word 5. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 5.
24–27 RCW6	Register Control Word 6. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 6.
28–31 RCW7	Register Control Word 7. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 7.

11.4.24 DDR Register Control Words 2 (DDR_x_DDR_SDRAM_RCW_2)

The DDR Register Control Word 2 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG[RCW_EN] is set. Each 4-bit field represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Address: Base address + 184h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RCW8	RCW9	RCW10	RCW11	RCW12	RCW13	RCW14	RCW15																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_x_DDR_SDRAM_RCW_2 field descriptions

Field	Description
0–3 RCW8	Register Control Word 8. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 8.
4–7 RCW9	Register Control Word 9. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 9.
8–11 RCW10	Register Control Word 10. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 10.
12–15 RCW11	Register Control Word 11. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 11.
16–19 RCW12	Register Control Word 12. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 12.
20–23 RCW13	Register Control Word 13. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 13.
24–27 RCW14	Register Control Word 14. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 14.
28–31 RCW15	Register Control Word 15. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 15.

11.4.25 DDR write leveling control 2 (DDRx_DDR_WRLVL_CNTL_2)

The DDR Write Leveling Control 2 register provides controls for write leveling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes.

Address: Base address + 190h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	WRLVL_START_1		Reserved	WRLVL_START_2		Reserved	WRLVL_START_3		Reserved	WRLVL_START_4																					
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

DDRx_DDR_WRLVL_CNTL_2 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 WRLVL_START_1	Write leveling start time for DQS[1]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101-11111 Reserved
8–10 -	This field is reserved. Reserved

Table continues on the next page...

DDRx_DDR_WRLVL_CNTL_2 field descriptions (continued)

Field	Description
11–15 WRLVL_START_2	Write leveling start time for DQS[2]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101-11111 Reserved
16–18 -	This field is reserved. Reserved
19–23 WRLVL_START_3	Write leveling start time for DQS[3]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay

Table continues on the next page...

DDRx_DDR_WRLVL_CNTL_2 field descriptions (continued)

Field	Description	
	10001	17/8 clock delay
	10010	9/4 clock delay
	10011	19/8 clock delay
	10100	5/2 clock delay
	10101-11111	Reserved
24–26 -	This field is reserved. Reserved	
27–31 WRLVL_START_4	Write leveling start time for DQS[4]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled.	
	00000	Use value from DDR_WRLVL_CNTL[WRLVL_START]
	00001	1/8 clock delay
	00010	1/4 clock delay
	00011	3/8 clock delay
	00100	1/2 clock delay
	00101	5/8 clock delay
	00110	3/4 clock delay
	00111	7/8 clock delay
	01000	1 clock delay
	01001	9/8 clock delay
	01010	5/4 clock delay
	01011	11/8 clock delay
	01100	3/2 clock delay
	01101	13/8 clock delay
	01110	7/4 clock delay
	01111	15/8 clock delay
	10000	2 clock delay
	10001	17/8 clock delay
	10010	9/4 clock delay
	10011	19/8 clock delay
	10100	5/2 clock delay
	10101-11111	Reserved

11.4.26 DDR write leveling control 3 (DDRx_DDR_WRLVL_CNTL_3)

The DDR Write Leveling Control 3 register provides controls for write leveling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes.

Address: Base address + 194h offset

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

DDRx_DDR_WRLVL_CNTL_3 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 WRLVL_START_5	Write leveling start time for DQS[5]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001 1/8 clock delay 00010 1/4 clock delay 00011 3/8 clock delay 00100 1/2 clock delay 00101 5/8 clock delay 00110 3/4 clock delay 00111 7/8 clock delay 01000 1 clock delay 01001 9/8 clock delay 01010 5/4 clock delay 01011 11/8 clock delay 01100 3/2 clock delay 01101 13/8 clock delay 01110 7/4 clock delay 01111 15/8 clock delay 10000 2 clock delay 10001 17/8 clock delay 10010 9/4 clock delay 10011 19/8 clock delay 10100 5/2 clock delay 10101-11111 Reserved
8–10 -	This field is reserved. Reserved

Table continues on the next page...

DDRx_DDR_WRLVL_CNTL_3 field descriptions (continued)

Field	Description
11–15 WRLVL_START_6	Write leveling start time for DQS[6]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled.
00000	Use value from DDR_WRLVL_CNTL[WRLVL_START]
00001	1/8 clock delay
00010	1/4 clock delay
00011	3/8 clock delay
00100	1/2 clock delay
00101	5/8 clock delay
00110	3/4 clock delay
00111	7/8 clock delay
01000	1 clock delay
01001	9/8 clock delay
01010	5/4 clock delay
01011	11/8 clock delay
01100	3/2 clock delay
01101	13/8 clock delay
01110	7/4 clock delay
01111	15/8 clock delay
10000	2 clock delay
10001	17/8 clock delay
10010	9/4 clock delay
10011	19/8 clock delay
10100	5/2 clock delay
10101-11111	Reserved
16–18 -	This field is reserved. Reserved
19–23 WRLVL_START_7	Write leveling start time for DQS[7]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled.
00000	Use value from DDR_WRLVL_CNTL[WRLVL_START]
00001	1/8 clock delay
00010	1/4 clock delay
00011	3/8 clock delay
00100	1/2 clock delay
00101	5/8 clock delay
00110	3/4 clock delay
00111	7/8 clock delay
01000	1 clock delay
01001	9/8 clock delay
01010	5/4 clock delay
01011	11/8 clock delay
01100	3/2 clock delay
01101	13/8 clock delay
01110	7/4 clock delay
01111	15/8 clock delay
10000	2 clock delay
10001	17/8 clock delay

Table continues on the next page...

DDRx_DDR_WRLVL_CNTL_3 field descriptions (continued)

Field	Description	
	10010	9/4 clock delay
	10011	19/8 clock delay
	10100	5/2 clock delay
	10101-11111	Reserved
24–26 -	This field is reserved. Reserved	
27–31 WRLVL_START_8	Write leveling start time for DQS[8]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled.	
	00000	Use value from DDR_WRLVL_CNTL[WRLVL_START]
	00001	1/8 clock delay
	00010	1/4 clock delay
	00011	3/8 clock delay
	00100	1/2 clock delay
	00101	5/8 clock delay
	00110	3/4 clock delay
	00111	7/8 clock delay
	01000	1 clock delay
	01001	9/8 clock delay
	01010	5/4 clock delay
	01011	11/8 clock delay
	01100	3/2 clock delay
	01101	13/8 clock delay
	01110	7/4 clock delay
	01111	15/8 clock delay
	10000	2 clock delay
	10001	17/8 clock delay
	10010	9/4 clock delay
	10011	19/8 clock delay
	10100	5/2 clock delay
	10101-11111	Reserved

11.4.27 DDR SDRAM mode configuration 3 (DDRx_DDR_SDRAM_MODE_3)

The DDR SDRAM mode configuration register sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: Base address + 200h offset

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

DDRx DDR SDRAM MODE 3 field descriptions

Field	Description
0-15 ESDMODE	<p>Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 3 Configuration, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[B1] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.</p>
16-31 SDMODE	<p>SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 3 Configuration, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].</p>

11.4.28 DDR SDRAM mode configuration 4 (DDR_x_DDR_SDRAM_MODE_4)

The DDR SDRAM mode 4 configuration register sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2 and DDR3). This register is used specifically for chip select 1 if DDR SDRAM CFG 2[UNQ MRS EN] is set.

Address: Base address + 204h offset

DDRx DDR SDRAM MODE 4 field descriptions

Field	Description
0–15 ESDMODE2	<p>Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in DDR SDRAM Mode 4 Configuration, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].</p>

Table continues on the next page...

DDRx DDR SDRAM MODE 4 field descriptions (continued)

Field	Description
16–31 ESDMODE3	<p>Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in DDR SDRAM Mode 4 Configuration, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].</p>

11.4.29 DDR SDRAM mode configuration 5 (DDRx DDR SDRAM MODE 5)

The DDR SDRAM mode configuration register sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 2 if DDR SDRAM CFG 2[UNO MRS EN] is set.

Address: Base address + 208h offset

DDRx DDR SDRAM MODE 5 field descriptions

Field	Description
0–15 ESDMODE	<p>Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 5 Configuration, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[B1] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.</p>
16–31 SDMODE	<p>SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 5 Configuration, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].</p>

11.4.30 DDR SDRAM mode configuration 6 (DDR_x_DDR_SDRAM_MODE_6)

The DDR SDRAM mode 6 configuration register sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2 and DDR3). This register is used specifically for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: Base address + 20Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ESDMODE2															ESDMODE3																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DDR_x_DDR_SDRAM_MODE_6 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in DDR SDRAM Mode 6 Configuration , corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in DDR SDRAM Mode 6 Configuration , corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

11.4.31 DDR SDRAM mode configuration 7 (DDR_x_DDR_SDRAM_MODE_7)

The DDR SDRAM mode configuration register sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: Base address + 210h offset

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

DDRx DDR SDRAM MODE 7 field descriptions

Field	Description
0–15 ESDMODE	<p>Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 7 Configuration, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[B1] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.</p>
16–31 SDMODE	<p>SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in DDR SDRAM Mode 7 Configuration, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].</p>

11.4.32 DDR SDRAM mode configuration 8 (DDR_x_DDR_SDRAM_MODE_8)

The DDR SDRAM mode 8 configuration register sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2 and DDR3). This register is used specifically for chip select 3 if DDR SDRAM CFG 2[UNQ MRS EN] is set.

Address: Base address + 214h offset

DDRx DDR SDRAM MODE 8 field descriptions

Field	Description
0–15 ESDMODE2	<p>Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer.</p> <p>When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in DDR SDRAM Mode 8 Configuration, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].</p>

Table continues on the next page...

DDR_x_DDR_SDRAM_MODE_8 field descriptions (continued)

Field	Description
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in DDR SDRAM Mode 8 Configuration , corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

11.4.33 DDR Debug Status Register 1 (DDR_x_DDRDSR_1)

The DDRDSR_1 register contains the DDR driver compensation input value and the current settings of the P and N FET impedance for MDIC_n, command/control, and data.

Address: Base address + B20h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DDRDC		MDICPZ			MDICNZ										
W																Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R		CPZ		CNZ					DPZ				DNZ			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_x_DDRDSR_1 field descriptions

Field	Description
0–1 DDRDC	DDR driver compensation input value
2–5 MDICPZ	Current setting of PFET driver MDIC impedance
6–9 MDICNZ	Current setting of NFET driver MDIC impedance
10–15 -	This field is reserved. Reserved
16–19 CPZ	Current setting of PFET driver command impedance
20–23 CNZ	Current setting of NFET driver command impedance
24–27 DPZ	Current setting of PFET driver data impedance

Table continues on the next page...

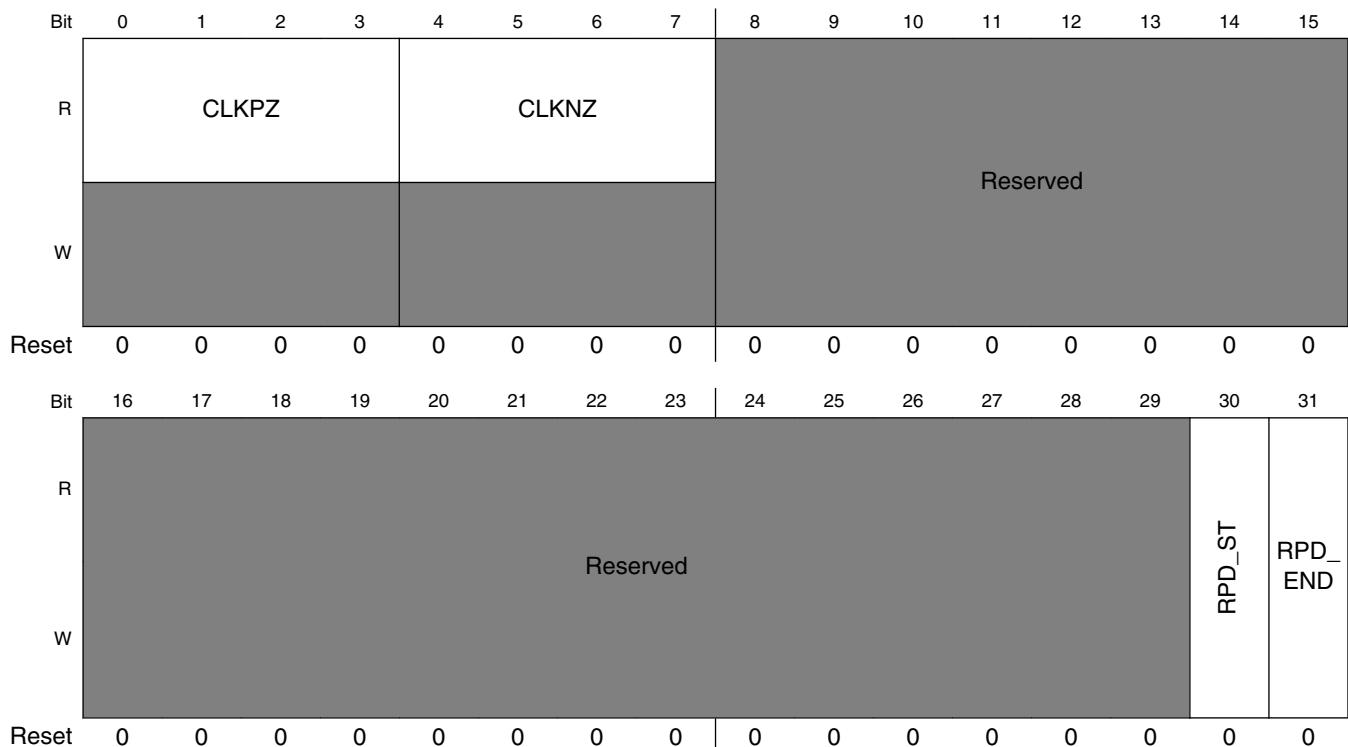
DDR_x_DDRDSR_1 field descriptions (continued)

Field	Description
28–31 DNZ	Current setting of NFET driver data impedance

11.4.34 DDR Debug Status Register 2 (DDR_x_DDRDSR_2)

The DDRDSR_2 register contains the current settings of the P and N FET impedance for the DDR drivers for clocks.

Address: Base address + B24h offset

**DDR_x_DDRDSR_2 field descriptions**

Field	Description
0–3 CLKPZ	Current setting of PFET driver clock impedance
4–7 CLKNZ	Current setting of NFET driver clock impedance
8–29 -	This field is reserved. Reserved
30 RPD_ST	Rapid clear of memory start. See DDR Rapid Clear of Memory for more information.

Table continues on the next page...

DDR_x_DDRDSR_2 field descriptions (continued)

Field	Description
	<p>0 The rapid clear of memory function has not been started.</p> <p>1 The rapid clear of memory function has started. During the rapid clear of memory, writes to the DDR memory mapped registers will not affect the register contents. This bit is cleared by software writing a 1.</p>
31 RPD_END	<p>Rapid clear of memory end. See DDR Rapid Clear of Memory for more information.</p> <p>0 The rapid clear of memory function has not ended.</p> <p>1 The rapid clear of memory function has completed. This bit is cleared by software writing a 1.</p>

11.4.35 DDR Control Driver Register 1 (DDR_x_DDRCDR_1)

DDRCDR_1 sets the driver hardware compensation enable, the DDR MDIC driver P/N impedance, ODT termination value for IOs, driver software override enable for MDIC, driver software override enable for address/command, driver software override enable for data, the DDR address/command driver P/N impedance, and the DDR data driver P/N impedance.

The fields in DDRCDR_1, other than DDRCDR_1[ODT], are used to enable driver calibration with the MDIC[0:1] pins.

Hardware DDR driver calibration is enabled by setting DDRCDR_1[DHC_EN].

NOTE

All driver calibration, whether by software or hardware, should be done before the DDR controller is enabled (before DDR_SDRAM_CFG[MEM_EN] is set).

Software can be used to calibrate the drivers instead of the automatic hardware calibration. If software calibration is used, the following steps should be taken:

1. Set DDRCDR_1[DSO_MDIC_EN] and ensure that DDRCDR_1[DHC_EN] is cleared
2. Set the highest impedance (value 0000) for DDRCDR_1[DSO_MDICPZ]
3. Set DDRCDR_1[DSO_MDIC_PZ_OE] to enable the output enable for MDIC[0]
4. After at least 4 cycles, read DDRDSR_1[0]. If the value is 0, then use the next lowest impedance (see the table below) and read DDRDSR_1[0] again. Once a value of 1 is detected, then leave DDRCDR_1[DSO_MDICPZ] at the calibrated value
5. Clear DDRCDR_1[DSO_MDIC_PZ_OE]
6. After DDRCDR_1[DSO_MDICPZ] is calibrated, set a value of 0000 for DDRCDR_1[DSO_MDICNZ]
7. Set DDRCDR_1[DSO_MDIC_NZ_OE] to enable the output enable for MDIC[1]

8. After at least 4 cycles, read DDRDSR_1[1]. If the value is 1, then use the next lowest impedance (see the table below) and read DDRDSR_1[1] again. Once a value of 0 is detected, then leave DDRCDR_1[DSO_MDICNZ] at the calibrated value
9. Clear DDRCDR_1[DSO_MDIC_NZ_OE]

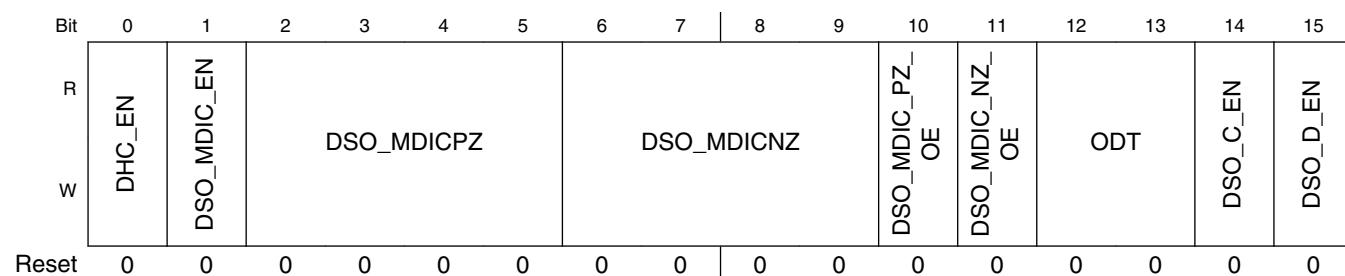
The following table lists the valid impedance override values from highest impedance (lowest drive strength) to lowest impedance (highest drive strength). Note that the drivers may be calibrated to either full-strength or half-strength.

Table 11-129. Valid Impedance Override Values

Driver impedance	Impedance Override Value	Notes
Highest	0000	
	0001	
	0011	
	0010	
	0110	
	0111	¹
	0101	
	0100	
	1100	
	1101	
	1111	
	1110	
	1010	
	1011	²
	1001	
Lowest	1000	³

1. 0111 provides the target for half-strength mode in both DDR2 (1.8 V) and DDR3 (1.5 V) modes when driver calibration is not used.
2. 1011 provides the target for full-strength mode in DDR2 mode (1.8 V) when driver calibration is not used.
3. 1000 provides the target for full-strength mode in DDR3 mode (1.5 V) when driver calibration is not used.

Address: Base address + B28h offset



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

DDRx_DDRCDR_1 field descriptions

Field	Description
0 DHC_EN	DDR driver hardware compensation enable
1 DSO_MDIC_EN	Driver software override enable for MDIC
2–5 DSO_MDICPZ	DDR driver software MDIC p-impedance override
6–9 DSO_MDICNZ	DDR driver software MDIC n-impedance override
10 DSO_MDIC_PZ_OE	Driver software override p-impedance output enable
11 DSO_MDIC_NZ_OE	Driver software override n-impedance output enable
12–13 ODT	ODT termination value for IOs. This is combined with DDRCDR_2[ODT] to determine the termination value. Below is the termination based on concatenating these two fields. Note that the order of concatenation is (from left to right) DDRCDR_1[ODT], DDRCDR_2[ODT] 000 75 Ω 001 55 Ω 010 60 Ω 011 50 Ω 100 150 Ω 101 43 Ω 110 120 Ω 111 Reserved
14 DSO_C_EN	Driver software override enable for address/command
15 DSO_D_EN	Driver software override enable for data
16–19 DSO_CPZ	DDR driver software command p-impedance override
20–23 DSO_CNZ	DDR driver software command n-impedance override
24–27 DSO_DPZ	Driver software data p-impedance override

Table continues on the next page...

DDRx_DDRCDR_1 field descriptions (continued)

Field	Description
28–31 DSO_DNZ	Driver software data n-impedance override

1. 0111 provides the target for half-strength mode in both DDR2 (1.8 V) and DDR3 (1.5 V) modes when driver calibration is not used.
2. 1011 provides the target for full-strength mode in DDR2 mode (1.8 V) when driver calibration is not used.
3. 1000 provides the target for full-strength mode in DDR3 mode (1.5 V) when driver calibration is not used.

11.4.36 DDR Control Driver Register 2 (DDRx_DDRCDR_2)

The DDRCDR_2 sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

Address: Base address + B2Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DSO_CLK_EN	Reserved			DSO_CLKPZ				DSO_CLKNZ				Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									Reserved						ODT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRx_DDRCDR_2 field descriptions

Field	Description
0 DSO_CLK_EN	Driver software override enable for clocks
1–3 -	This field is reserved. Reserved
4–7 DSO_CLKPZ	Driver software clocks p-impedance override
8–11 DSO_CLKNZ	Driver software clocks n-impedance override
12–30 -	This field is reserved. Reserved

Table continues on the next page...

DDRx_DDRCDR_2 field descriptions (continued)

Field	Description																
31 ODT	<p>ODT termination value for IOs. This is combined with DDRCDR_2[ODT] to determine the termination value. Below is the termination based on concatenating these two fields.</p> <p>Note that the order of concatenation is (from left to right)</p> <p>DDRCDR_1[ODT], DDRCDR_2[ODT]</p> <table> <tr><td>000</td><td>75 Ω</td></tr> <tr><td>001</td><td>55 Ω</td></tr> <tr><td>010</td><td>60 Ω</td></tr> <tr><td>011</td><td>50 Ω</td></tr> <tr><td>100</td><td>150 Ω</td></tr> <tr><td>101</td><td>43 Ω</td></tr> <tr><td>110</td><td>120 Ω</td></tr> <tr><td>111</td><td>Reserved</td></tr> </table>	000	75 Ω	001	55 Ω	010	60 Ω	011	50 Ω	100	150 Ω	101	43 Ω	110	120 Ω	111	Reserved
000	75 Ω																
001	55 Ω																
010	60 Ω																
011	50 Ω																
100	150 Ω																
101	43 Ω																
110	120 Ω																
111	Reserved																

11.4.37 DDR IP block revision 1 (DDRx_DDR_IP_REV1)

The DDR IP block revision 1 register provides read-only fields with the IP block ID, along with major and minor revision information.

Address: Base address + BF8h offset

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

DDRx_DDR_IP_REV1 field descriptions

Field	Description
0–15 IP_ID	IP block ID. For the DDR controller, this value is 0x0002.
16–23 IP_MJ	Major revision. This is currently set to 0x04.
24–31 IP_MN	Minor revision. This is currently set to 0x04 .

11.4.38 DDR IP block revision 2 (DDRx_DDR_IP_REV2)

The DDR IP block revision 2 register provides read-only fields with the IP block integration and configuration options.

Address: Base address + BFCh offset

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

DDRx_DDR_IP_REV2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IP_INT	IP block integration options
16–23 -	This field is reserved. Reserved
24–31 IP_CFG	IP block configuration options

11.4.39 DDR Memory Test Control Register (DDRx_DDR_MTCR)

The DDR Memory Test Control Register provides the enable and controls for an automatic memory test.

Address: Base address + D00h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MT_EN	Reserved					MT_TYP	Reserved					MT_TRNARND			
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															MT_STAT
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDRx_DDR_MTCR field descriptions

Field	Description
0 MT_EN	<p>Memory Test Enable. This bit can be set by software to enable the memory test. Based on the value of MT_TYP, the controller will either issue writes only, reads only, or it will issue writes and reads. The memory controller will issue transactions throughout all of memory, as defined by the CS_n_CONFIG and CSn_BNDS registers. The memory controller will check the data during this test. In addition, the ERR_DETECT register should be read by software if ECC is enabled after the memory test to ensure there were no ECC errors. If an ECC error is detected, the error capture registers will hold the address, data, and attributes captured for the first fail. If there is no ECC error and the test failed (for a data miscompare), then the capture registers will hold information for the transaction that caused the first data miscompare. Note that transactions with ECC errors will take priority in the capture registers. Hardware will clear MT_EN after the memory test is complete. This bit can be set before DDR_SDRAM_CFG[MEM_EN] is set, or it can be set again after the memory controller has been enabled.</p> <p>NOTE: The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN].</p> <p>0 Memory test has been disabled. 1 Memory test has been enabled. Hardware will clear this bit when it is complete.</p>
1–5 -	This field is reserved. Reserved
6–7 MT_TYP	<p>Memory Test Type. This field will determine if the memory test will issue writes only, reads only, or both writes and reads. Note that the 'read only' test should not be used unless memory has already been initialized.</p> <p>00 Memory test will issue writes and reads. 01 Memory test will issue writes only.</p>

Table continues on the next page...

DDRx DDR MTCR field descriptions (continued)

Field	Description
	10 Memory test will issue reads only. 11 Reserved
8-11 -	This field is reserved. Reserved
12-15 MT_TRNARND	Memory Test Turnaround. This field determines how many writes will be issued during the memory test before the reads to the same addresses will be issued. This can be used to allow longer streams of writes/reads, and it can be used to test the write->read and read->write turnarounds in a stressful manner. This field is only relevant if MT_TYP is set to 2'b00.
	0000 Entire memory will be written before read transactions are issued.
	0001 Total write/read streams will be 1 transaction each.
	0010 Total write/read streams will be 2 transactions each.
	0011 Total write/read streams will be 4 transactions each.
	0100-1111 Reserved
16-30 -	This field is reserved. Reserved
31 MT_STAT	Memory Test Status. After hardware clears MT_EN, this bit will be set if there was a fail. If there is a fail during the memory test (that is, a data miscompare), then this bit will be set at the same time that MT_EN is cleared. Software can clear this bit after it has been set.
	0 No fail has been detected.
	1 A data miscompare was detected during the memory test.

11.4.40 DDR Memory Test Pattern n Register (DDRx_DDR_MTPn)

The DDR memory test pattern n register provides the data pattern that will be written during the n th set of 32-bits of each 40-byte memory test pattern. This is used when DDR_MTCR[MT_EN] is set to enable the memory write/read test.

NOTE

Memory test read compares data that is written in this register.

Address: Base address + D20h offset + (4d × i), where i=0d to 9d

DDR_x DDR MTP*n* field descriptions

Field	Description
0-31 DDR_PATT	This 32-bit pattern will be used during the memory test if enabled via DDR_MTCR[MT_EN]. This memory test will write/read a programmable 40-byte pattern. The 10 DDR_MTP n registers will create the 40-byte pattern that is used.

11.4.41 Memory data path error injection mask high (DDRx_DATA_ERR_INJECT_HI)

Address: Base address + E00h offset

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

Reset 0

DDRx_DATA_ERR_INJECT_HI field descriptions

Field	Description
0–31 EIMH	Error injection mask high data path. Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

11.4.42 Memory data path error injection mask low (DDRx_DATA_ERR_INJECT_LO)

Address: Base address + E04h offset

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

Reset 0

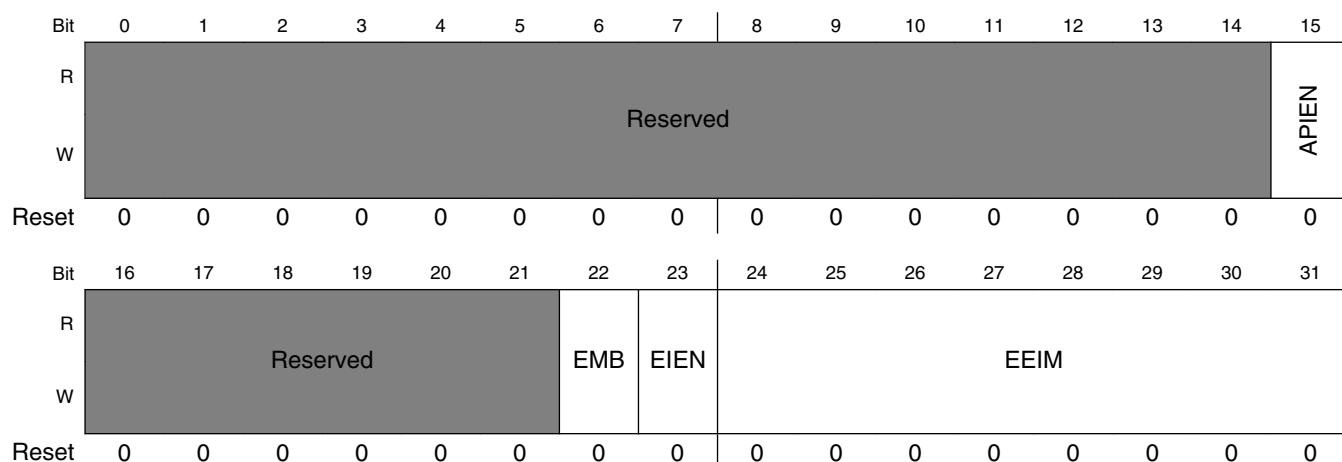
DDRx_DATA_ERR_INJECT_LO field descriptions

Field	Description
0–31 EIML	Error injection mask low data path. Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

11.4.43 Memory data path error injection mask ECC (DDRx_ECC_ERR_INJECT)

The memory data path error injection mask ECC register sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte. In addition, a single address parity error may be injected through this register.

Address: Base address + E08h offset



DDRx_ECC_ERR_INJECT field descriptions

Field	Description
0–14 -	This field is reserved. Reserved
15 APIEN	Address parity error injection enable. This bit will be cleared by hardware after a single address parity error has been injected. 0 Address parity error injection disabled. 1 Address parity error injection enabled.
16–21 -	This field is reserved. Reserved
22 EMB	ECC mirror byte 0 Mirror byte functionality disabled. 1 Mirror the most significant data path byte onto the ECC byte.
23 EIEN	Error injection enable 0 Error injection disabled. 1 Error injection enabled. This applies to the data mask bits, the ECC mask bits, and the ECC mirror bit. Note that error injection should not be enabled until the memory controller has been enabled via DDR_SDRAM_CFG[MEM_EN].
24–31 EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

11.4.44 Memory data path read capture high (DDRx CAPTURE DATA HI)

The memory data path read capture high register stores the high word of the read data path during error capture.

Address: Base address + E20h offset

DDRx_CAPTURE_DATA_HI field descriptions

Field	Description
0-31 EHD	Error capture high data path. Captures the high word of the data path when errors are detected.

11.4.45 Memory data path read capture low (DDR_x_CAPTURE_DATA_LO)

The memory data path read capture low register stores the low word of the read data path during error capture.

Address: Base address + E24h offset

DDRx_CAPTURE_DATA_LO field descriptions

Field	Description
0-31 ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

11.4.46 Memory data path read capture ECC (DDR_x_CAPTURE_ECC)

The memory data path read capture ECC register stores the ECC syndrome bits that were on the data bus when an error was detected.

Address: Base address + E28h offset

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

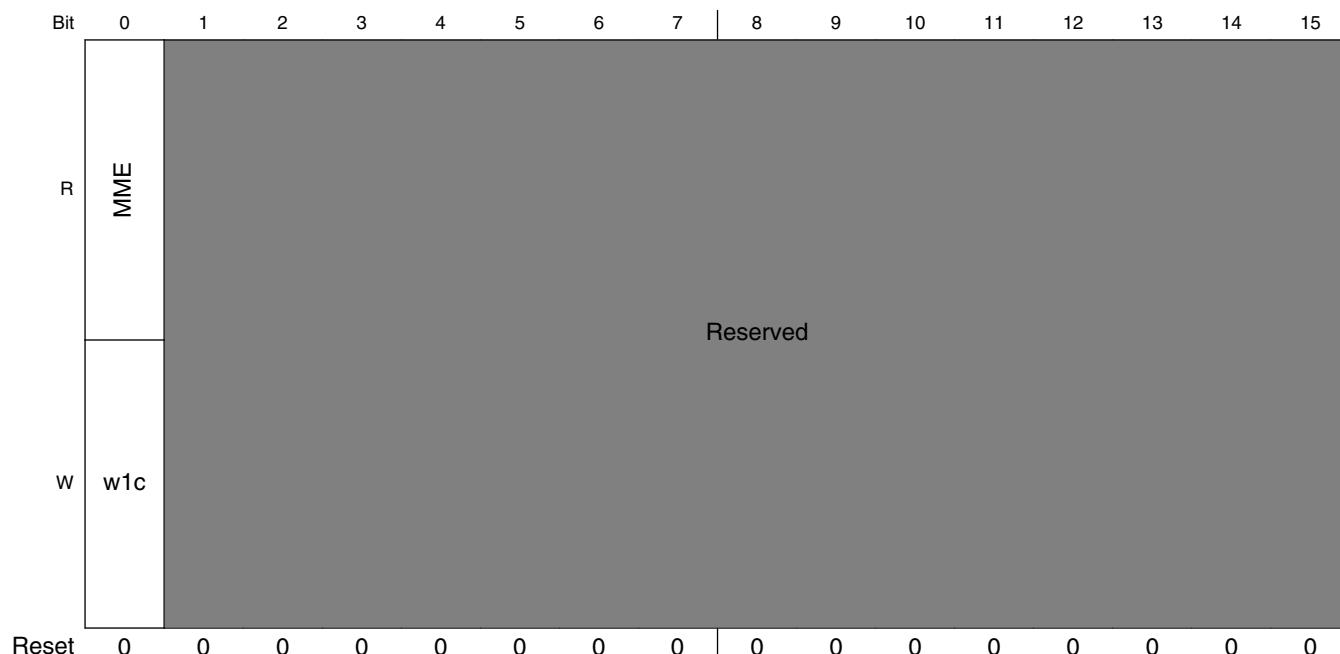
DDR_x_CAPTURE_ECC field descriptions

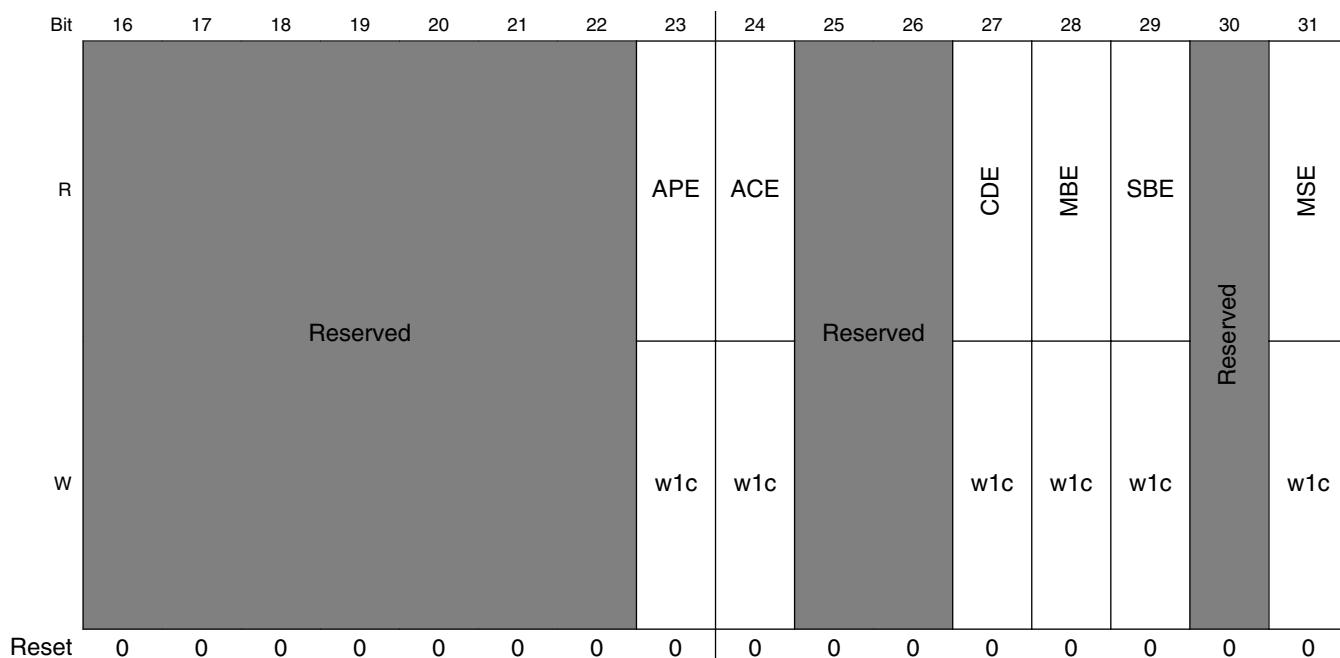
Field	Description
0–31 ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 64-bit mode: In 64-bit mode, only 24:31 should be used, although 0:7, 8:15, and 16:23 show the 8-bit ECC code replicated. <ul style="list-style-type: none"> • 0:7 should be ignored • 8:15 should be ignored • 16:23 should be ignored • 24:31 all 64 bits 32-bit mode: <ul style="list-style-type: none"> • 0:7 should be ignored • 8:15 8-bit ECC for the 32 bits in beats 0, 2, 4, 6 in 32-bit bus mode • 16:23 should be ignored • 24:31 8-bit ECC for the 32 bits in beats 1, 3, 5, 7 in 32-bit bus mode

11.4.47 Memory error detect (DDRx_ERR_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

Address: Base address + E40h offset



**DDR_x_ERR_DETECT field descriptions**

Field	Description
0 MME	Multiple memory errors. This bit is cleared by software writing a 1. 0 Multiple memory errors of the same type were not detected. 1 Multiple memory errors of the same type were detected.
1–22 -	This field is reserved. Reserved
23 APE	Address parity error. This bit is cleared by software writing a 1. 0 An address parity error has not been detected. 1 An address parity error has been detected.
24 ACE	Automatic calibration error. This bit is cleared by software writing a 1. 0 An automatic calibration error has not been detected. 1 An automatic calibration error has been detected.
25–26 -	This field is reserved. Reserved
27 CDE	Corrupted data error. This bit is cleared by software writing a 1. 0 A corrupted data error has not been detected. 1 A corrupted data error has been detected. This bit will be set if the actual ECC is inverted from the expected ECC during a read command. The memory controller will intentionally invert the ECC code if <i>DDR_SDRAM_CFG_2[CD_DIS]</i> is cleared when corrupted data is written to memory. The <i>ERR_DETECT[MBE]</i> bit will also be set when a corrupted data error is detected. Note it is also possible for a 2-bit data error to cause the ECC code to become inverted, which would either mask a corrupted data error or cause a normal multi-bit error to also appear as a corrupted data error.
28 MBE	Multiple-bit error. This bit is cleared by software writing a 1.

Table continues on the next page...

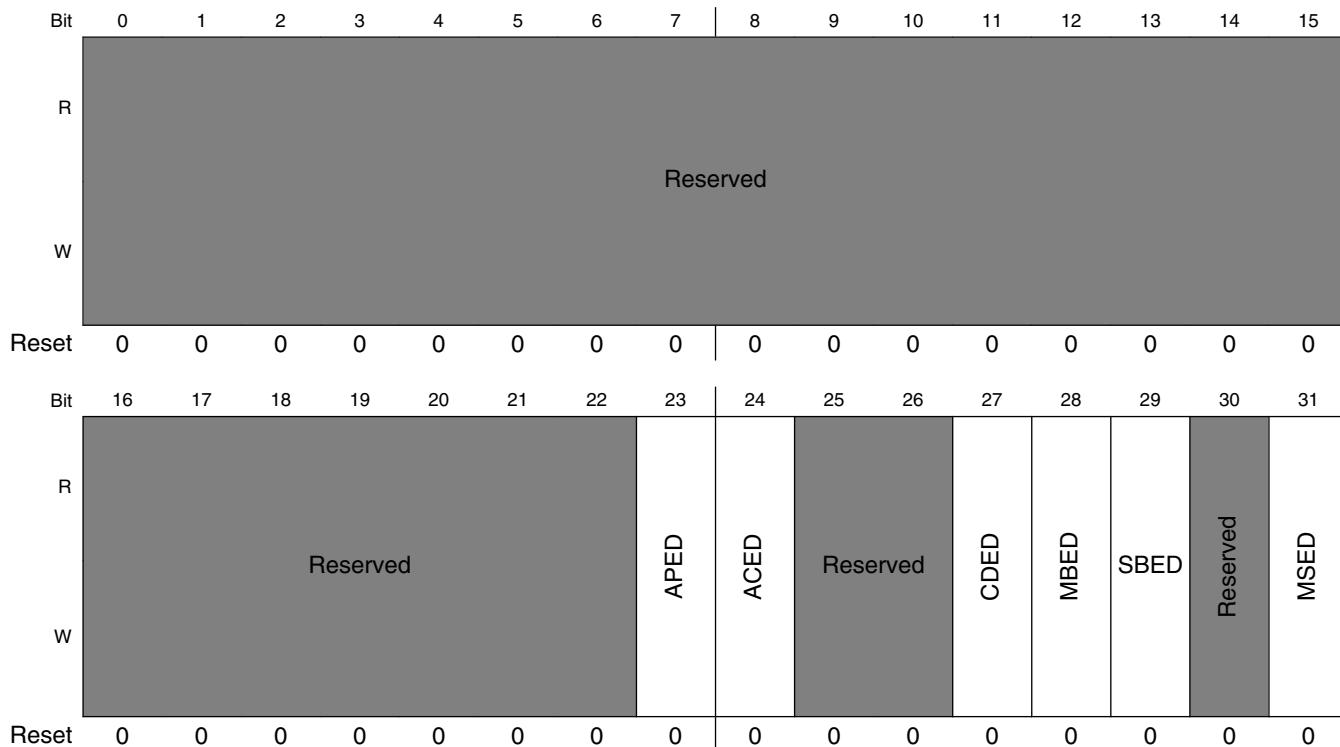
DDR_x_ERR_DETECT field descriptions (continued)

Field	Description
	0 A multiple-bit error has not been detected. 1 A multiple-bit error has been detected.
29 SBE	Single-bit ECC error. This bit is cleared by software writing a 1. 0 The number of single-bit ECC errors detected has not crossed the threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30 -	This field is reserved. Reserved
31 MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

11.4.48 Memory error disable (DDR_x_ERR_DISABLE)

The memory error disable register allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

Address: Base address + E44h offset



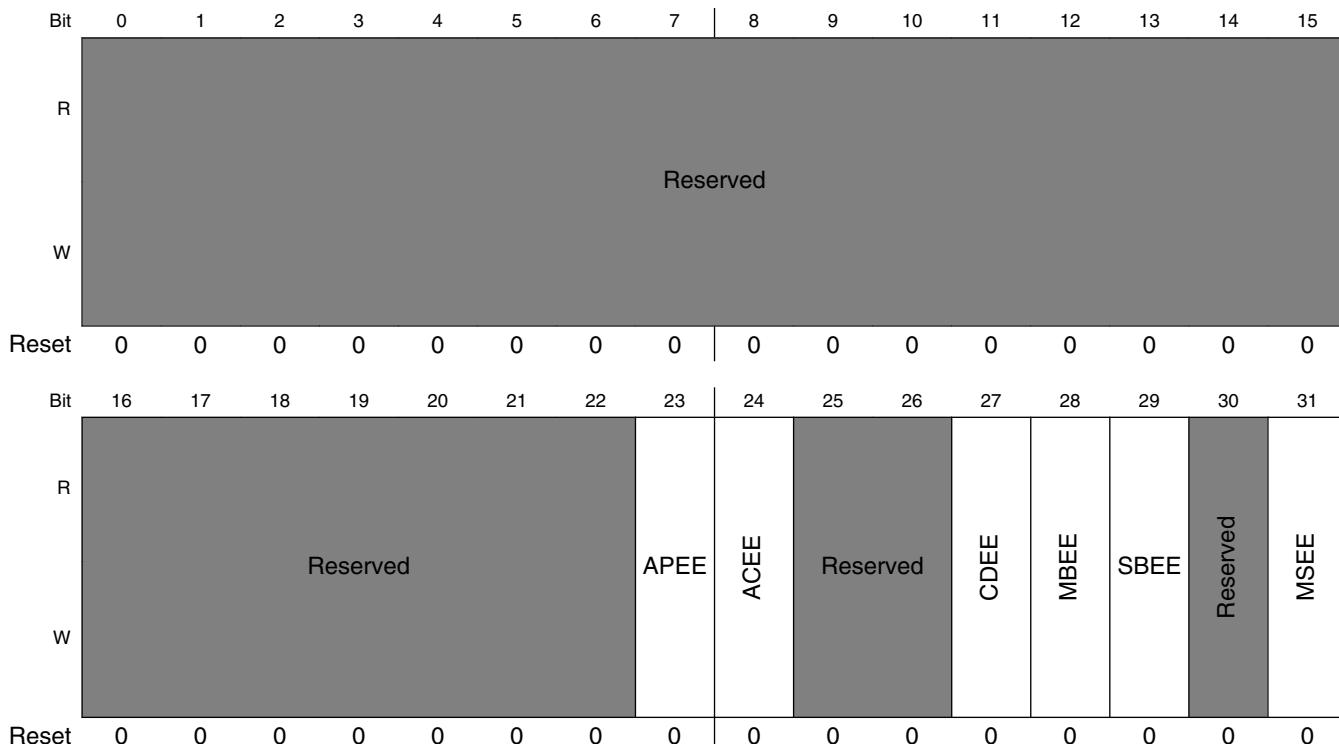
DDR_x_ERR_DISABLE field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 APED	Address parity error disable 0 Address parity errors are detected if (DDR_SDRAM_CFG2[AP_EN] is set. They are reported if ERR_INT_EN[APEE] is set. 1 Address parity errors are not detected or reported.
24 ACED	Automatic calibration error disable 0 Automatic calibration errors are enabled. 1 Automatic calibration errors are disabled.
25–26 -	This field is reserved. Reserved
27 CDED	Corrupted data error disable 0 Corrupted data error checking is enabled. 1 Corrupted data error checking is disabled.
28 MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. See Error Management , for more information. MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
29 SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30 -	This field is reserved. Reserved
31 MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

11.4.49 Memory error interrupt enable (DDRx_ERR_INT_EN)

The memory error interrupt enable register enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal int_B signal is asserted to the programmable interrupt controller (PIC).

Address: Base address + E48h offset



DDRx_ERR_INT_EN field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 APEE	Address parity error interrupt enable 0 Address parity errors cannot generate interrupts. 1 Address parity errors generate interrupts.
24 ACEE	Automatic calibration error interrupt enable 0 Automatic calibration errors cannot generate interrupts. 1 Automatic calibration errors generate interrupts.
25–26 -	This field is reserved. Reserved

Table continues on the next page...

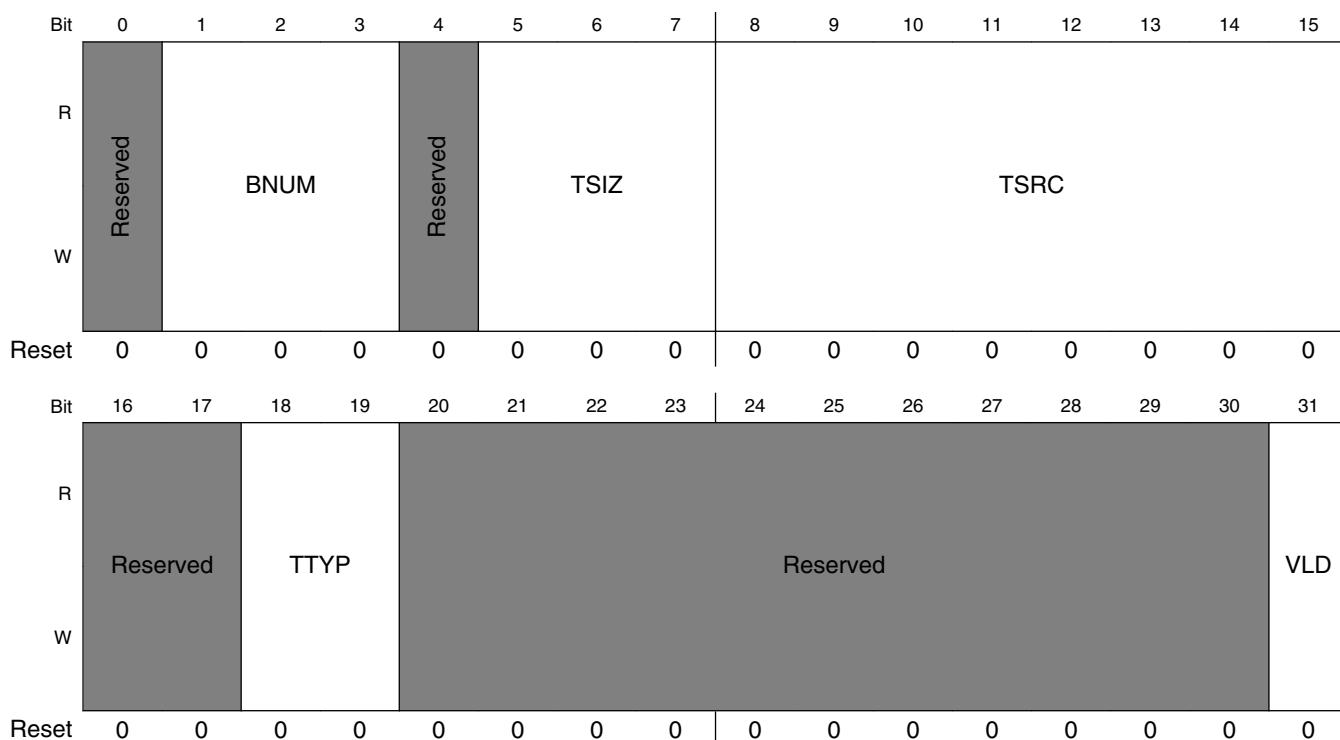
DDR_x_ERR_INT_EN field descriptions (continued)

Field	Description
27 CDEE	Corrupted data error interrupt enable 0 Corrupted data errors cannot generate interrupts. 1 Corrupted data errors generate interrupts.
28 MBEE	Multiple-bit ECC error interrupt enable. See Error Management , for more information. Note that uncorrectable read errors may cause an interrupt. ERR_DISABLE[MBED] must be zero and MBEE and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29 SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30 -	This field is reserved. Reserved
31 MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

11.4.50 Memory error attributes capture (DDRx_CAPTURE_ATTRIBUTES)

The memory error attributes capture register sets attributes for errors including type, size, source, and others.

Address: Base address + E4Ch offset



DDRx_CAPTURE_ATTRIBUTES field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–3 BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4 -	This field is reserved. Reserved
5–7 TSIZ	Transaction size for the error. Captures the transaction size in double words. 000 8 double words 001 1 double word 010 2 double words 011 3 double words 100 4 double words

Table continues on the next page...

DDR_x_CAPTURE_ATTRIBUTES field descriptions (continued)

Field	Description
	101 5 double word 110 6 double words 111 7 double words
8–15 TSRC	Transaction source for the error. See Global Source and Target IDs for the defined encodings.
16–17 -	This field is reserved. Reserved
18–19 TTYP	Transaction type for the error. 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30 -	This field is reserved. Reserved
31 VLD	Valid. Set as soon as valid information is captured in the error capture registers.

11.4.51 Memory error address capture (DDR_x_CAPTURE_ADDRESS)

The memory error address capture register holds the 32 lsbs of a transaction when a DDR ECC error is detected.

Address: Base address + E50h offset

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

DDR_x_CAPTURE_ADDRESS field descriptions

Field	Description
0–31 CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

11.4.52 Memory error extended address capture (DDRx_CAPTURE_EXT_ADDRESS)

The memory error extended address capture register holds the four most significant transaction bits when an error is detected.

Address: Base address + E54h offset

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

Reset 0

DDRx_CAPTURE_EXT_ADDRESS field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 CEADDR	Captured extended address. Captures the 4 msbs of the transaction address when an error is detected

11.4.53 Single-Bit ECC memory error management (DDRx_ERR_SBE)

The single-bit ECC memory error management register stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.

Address: Base address + E58h offset

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

Reset 0

DDRx_ERR_SBE field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.
16–23 -	This field is reserved. Reserved

Table continues on the next page...

DDR_x_ERR_SBE field descriptions (continued)

Field	Description
24–31 SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and an interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

11.5 DDR Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR2 and DDR3 SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

[Figure 11-227](#) is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40-bit wide memory. Bank sizes up to 8 Gbytes are supported, providing up to a maximum of 32 Gbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32- -bit data bus, detects all double-bit errors within the 64- or 32- bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

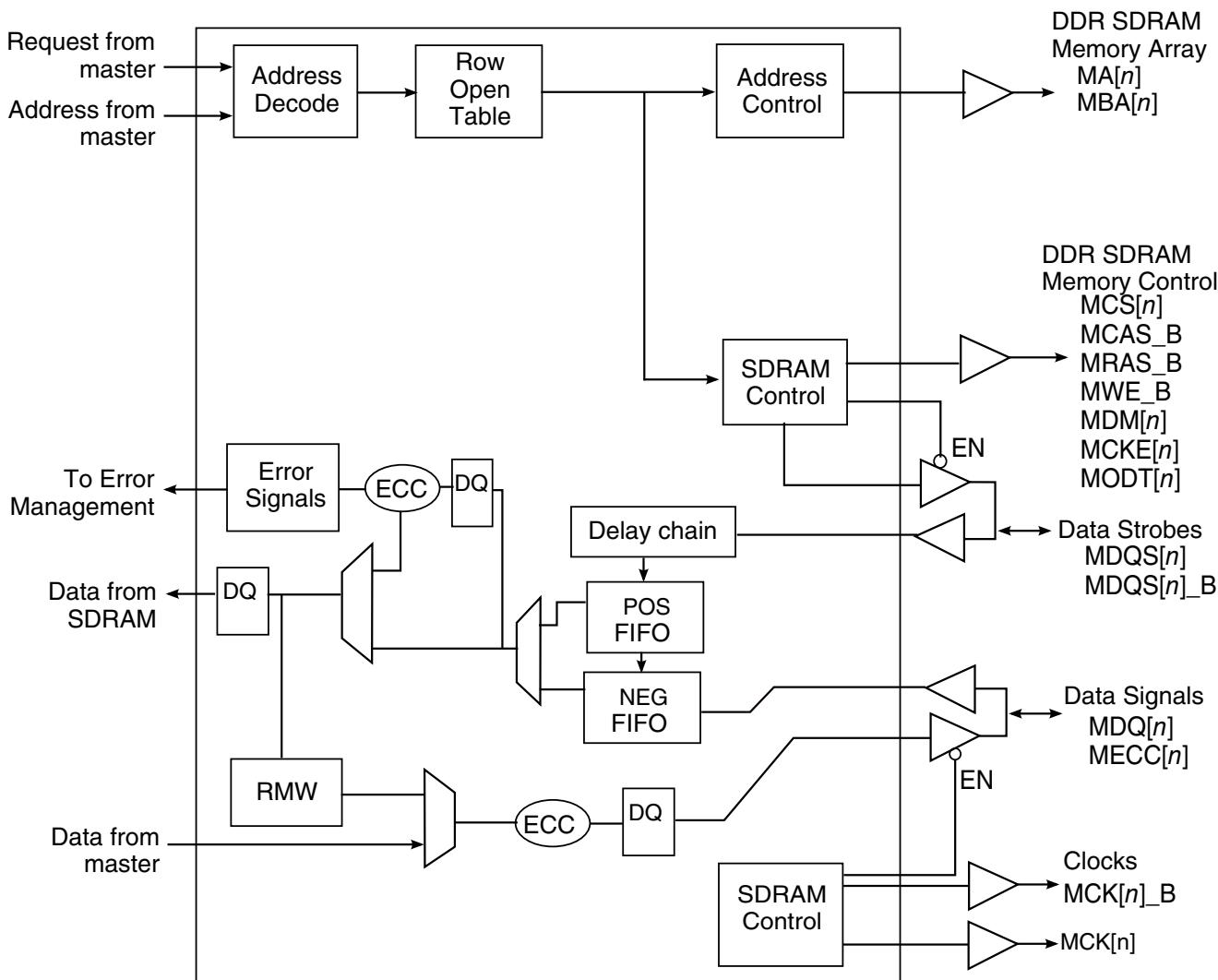


Figure 11-227. DDR Memory Controller Block Diagram

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (4 or 8) in a programmed sequence. Accesses to closed pages start with the registration of an ACTIVE command followed by a READ or WRITE. (Accessing open pages does not require an ACTIVE command.) The address bits registered coincident with the activate command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[n]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

[Figure 11-228](#) shows an example DDR SDRAM configuration with four logical banks.

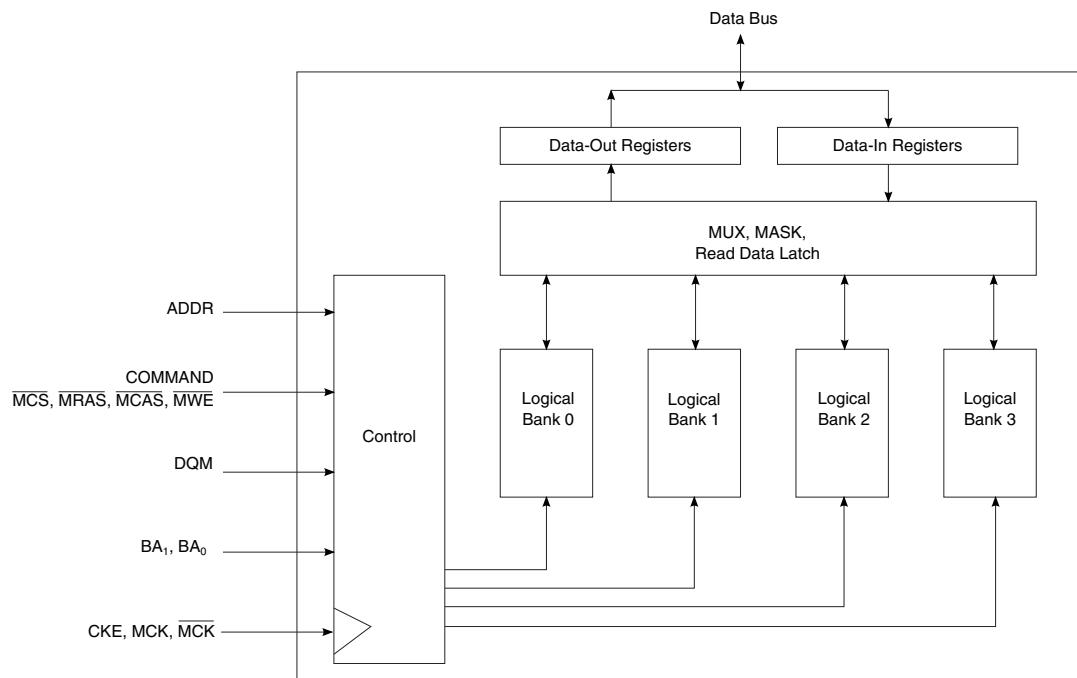


Figure 11-228. Typical Dual Data Rate SDRAM Internal Organization

[Figure 11-229](#) shows some typical signal connections.

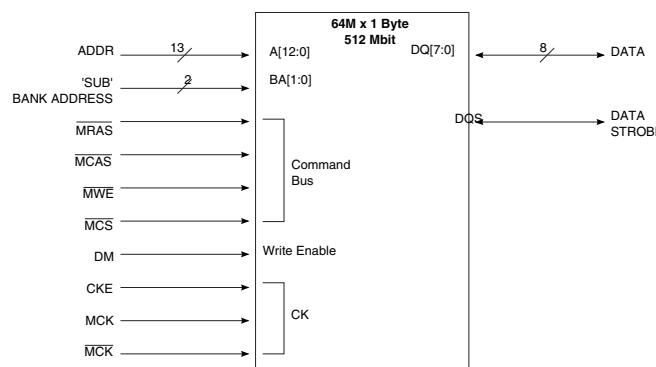


Figure 11-229. Typical DDR SDRAM Interface Signals

The following figure shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M x 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function.

Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controller drives 16 address pins, but in this example the DDR SDRAM devices use only 12 bits.

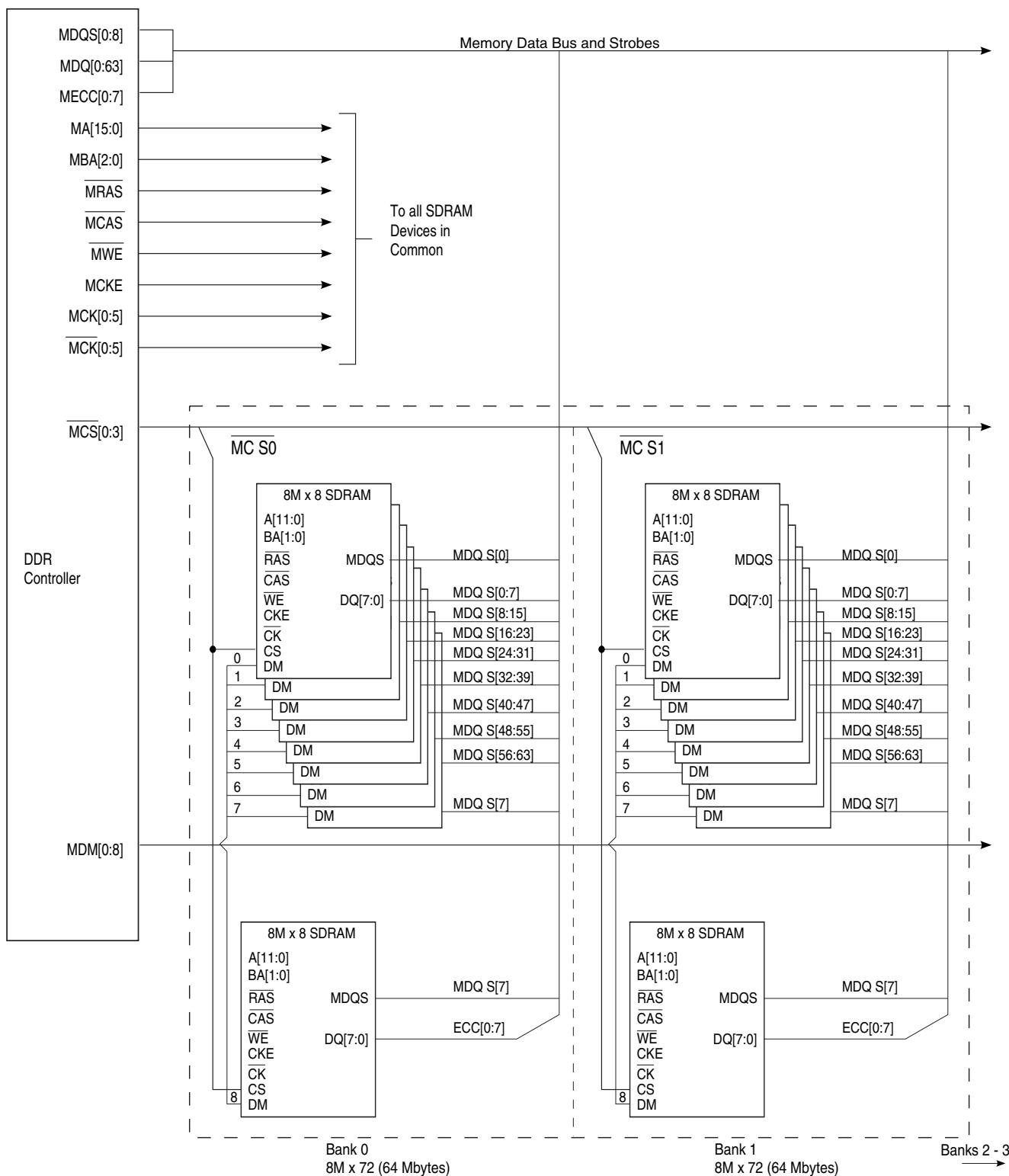


Figure 11-230. Example 256-Mbyte DDR SDRAM Configuration

[Error Management](#), explains how the DDR memory controller handles errors.

11.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Sixteen multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 4 Gbits. Four chip select (CS_B) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 Mbytes to 4 Gbytes. The physical banks can be constructed using x 8, x16, or x32 (with 1 DQS per data byte) memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit , 2 Gbits ,4 Gbits and 8Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

An 8-bit DDR SDRAM device has a DQM signal and eight data signals (DQ[0:7]). A 16-bit DDR SDRAM device has two DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection.

[Table 11-235](#) shows the DDR memory controller's relationships between data byte lane0-7 , MDM[0:7] , MDQS[0:7] , and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

Table 11-235. Byte Lane to Data Relationship

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 64-Bit Mode
0 (MSB)	MDM[0]	MDQS[0]	MDQ[0:7]
1	MDM[1]	MDQS[1]	MDQ[8:15]
2	MDM[2]	MDQS[2]	MDQ[16:23]
3	MDM[3]	MDQS[3]	MDQ[24:31]
4	MDM[4]	MDQS[4]	MDQ[32:39]
5	MDM[5]	MDQS[5]	MDQ[40:47]
6	MDM[6]	MDQS[6]	MDQ[48:55]
7 (LSB)	MDM[7]	MDQS[7]	MDQ[56:63]

NOTE

For a 32-bit bus mode data byte lanes 0-3 in the above table would be used.

11.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 16 16 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 16 row address bits, plus 2 or 3 logical bank-select bits and from 8-11 column address bits.

The following tables describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

Table 11-236. Supported DDR2 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
256 Mbits	32 Mbits x 8	13 x 10 x 2	256 Mbytes	1 Gbyte
256 Mbits	16 Mbits x 16	13 x 9 x 2	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits x 8	14 x 10 x 2	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	13 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbit	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes]
2 Gbits	128 Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes

Table 11-237. Supported DDR3 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	64-Bit Bank Size	Four Banks of Memory
512 Mbits	64 Mbits x 8	13 x 10 x 3	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	12 x 10 x 2	256 Mbytes	1 Gbyte
1 Gbits	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes
2 Gbits	128 Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes

Table continues on the next page...

Table 11-237. Supported DDR3 SDRAM Device Configurations (continued)

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	64-Bit Bank Size	Four Banks of Memory
4 Gbits	512 Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes
8 Gbits	1 Gbit x 8	16 x 11 x 3	8 Gbytes	32 Gbytes
8 Gbits	512 Mbits x 16	16 x 11 x 3	4 Gbytes	16 Gbytes

NOTE

32-bit data bus is also supported. The values in the 64-bit bank size column in the above table would be halved for a 32-bit mode

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in [Error Management](#).

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the DDR memory controller to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate MCSn_B signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

11.5.2 DDR SDRAM Address Multiplexing

The following tables show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[15 :0] use MA[15] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR2/DDR3 mode for reads and writes, so the column address can never use MA[10].

Table 11-238. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																														lsb
			4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	
16 x 10 x 3	MRAS_B	15	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0																
	MBA																	2	1	0													
	MCAS_B																				9	8	7	6	5	4	3	2	1	0			

Table continues on the next page...

Table 11-238. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled (continued)

Row x Col		msb 4	Address from Core Master																													lsb 33-35 32
			5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
15 x 10 x 3	MRAS_B		1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0											
	MCAS_B																															
14 x 10 x 3	MRAS_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																
	MBA																		2	1	0											
	MCAS_B																															
14 x 10 x 2	MRAS_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0												
	MCAS_B																															
13 x 10 x 3	MRAS_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																	
	MBA																		2	1	0											
	MCAS_B																															
13 x 10 x 2	MRAS_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																	
	MBA																		1	0												
	MCAS_B																															
13 x 9 x 2	MRAS_B						1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0												
	MCAS_B																															

Table 11-239. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb 4	Address from Core Master																															lsb 34-3 33 32
			5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32				
16 x 10 x 3	MRA_S_B		1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																
	MBA																		2	1	0													

Table continues on the next page...

Table 11-239. DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled (continued)

Row x Col	msb	Address from Core Master																														lsb 34-3 5		
		4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3				
	MCA S_B																																	
15 x 10 x 3	MRA S_B		1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																	
	MBA																																	
	MCA S_B																																	
14 x 10 x 3	MRA S_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																		
	MBA																																	
	MCA S_B																																	
14 x 10 x 2	MRA S_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																		
	MBA																																	
	MCA S_B																																	
13 x 10 x 3	MRA S_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																			
	MBA																																	
	MCA S_B																																	
13 x 10 x 2	MRA S_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																			
	MBA																																	
	MCA S_B																																	
13 x 9 x 2	MRA S_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0																			
	MBA																																	
	MCA S_B																																	

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 11-240 illustrates examples of address decode when interleaving between two chip selects, and **Table 11-241** shows examples of address decode when interleaving between four chip selects.

**Table 11-240. Example of Address Multiplexing for 32/64-Bit Data Bus
Interleaving Between Two Banks with Partial Array Self Refresh Disabled**

Row x Col		msb	Address from Core Master																											lsb		
			4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
14 x 10 x 3	MRA_S_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0												
	MCA_S_B																			9	8	7	6	5	4	3	2	1	0			
14 x 10 x 2	MRA_S_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	1	0													
	MCA_S_B																			9	8	7	6	5	4	3	2	1	0			
13 x 10 x 3	MRA_S_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																
	MBA																2	1	0													
	MCA_S_B																		9	8	7	6	5	4	3	2	1	0				
13 x 10 x 2	MRA_S_B		1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																
	MBA																1	0														
	MCA_S_B																		9	8	7	6	5	4	3	2	1	0				

**Table 11-241. Example of Address Multiplexing for 64-Bit Data Bus
Interleaving Between Four Banks with Partial Array Self Refresh Disabled**

Row x Col		msb	Address from Core Master																												lsb		
			4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
14 x 10 x 3	MRAS_B	13	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																	
	MBA																2	1	0														
	MCAS_B																		9	8	7	6	5	4	3	2	1	0					
14 x	MRAS_B		1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																

Table continues on the next page...

Table 11-241. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks with Partial Array Self Refresh Disabled (continued)

Row x Col	msb	Address from Core Master																												lsb	
		4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
10 x 2	MBA																			1	0										
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS_B	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																
	MBA															2	1	0													
	MCAS_B																		9	8	7	6	5	4	3	2	1	0			
13 x 10 x 2	MRAS_B	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	CS SEL																
	MBA															1	0														
	MCAS_B																		9	8	7	6	5	4	3	2	1	0			

Partial Array Self Refresh (PASR) can be enabled for any chip select using the CSn_CONFIG_2[PASR_CFG] fields. If PASR is enabled for a given chip select, then the sub-bank and row decode will be swapped, and the sub-bank will be decoded as the most significant portion of the DRAM address, as shown in [Table 11-242](#). If chip select interleaving and PASR are enabled for a chip select, then the interleaved chip select bit is placed immediately to the left of the column decode, as shown in [Table 11-243](#).

Table 11-242. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled

Table continues on the next page...

Table 11-242. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled (continued)

Table 11-243. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled

Table continues on the next page...

Table 11-243. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled (continued)

Row x Col	msb	Address from Core Master																														lsb 33-3 5			
		4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3	3	3						
		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	
	MCAS_B																																		
13 x 10 x 3	MRAS_B				1	1	1	9	8	7	6	5	4	3	2	1	0	C																	
	MBA		2	1	0													S																	
	MCAS_B																	E	9	8	7	6	5	4	3	2	1	0							
13 x 10 x 2	MRAS_B				1	1	1	9	8	7	6	5	4	3	2	1	0	C																	
	MBA			1	0												S																		
	MCAS_B																E	L	9	8	7	6	5	4	3	2	1	0							

11.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR2/DDR3 mode.

All read or write accesses to DDR SDRAM are performed by the DDR memory controller using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled using both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

The following DDR SDRAM interface commands (summarized in [Table 11-244](#)) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate-Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge-Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read-Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding

clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.

- Write-Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.
- Refresh (similar to MCAS_B before MRAS_B)-Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)-Allows setting of DDR SDRAM options. These options are: MCAS_B latency, additive latency (for DDR2/DDR3), write recovery (for DDR2/DDR3), burst type, and burst length. MCAS_B latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide MCAS_B latency {1,2,3} while some provide MCAS_B latency {1,2,3,4,5}). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR3 memory only. For DDR2 in 32-bit bus mode, all 32-byte burst accesses from the platform are split into two 16-byte (that is, 4 beat) accesses to the SDRAMs in the memory controller. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, MCAS_B latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software via use of the DDR_SDRAM_MD_CNTL register.
- Self refresh (for long periods of standby)-Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller will place all logical banks in a precharged state.

Table 11-244. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Curren- t	MCS_ B	MRAS_ B	MCAS_ B	MWE_ B	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto-precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column
Write with auto-precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

11.5.4 DDR SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in [Table 11-245](#) with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 11-245. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD} .
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS} .
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD} .
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the read latency is m clocks, the data is available nominally coincident with clock edge n + m.
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP} .
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as t_{RP} .
REFREC	The number of clock cycles from the refresh command until an activate command is allowed. This can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in nanoseconds. This field is also combined with TIMING_CFG_3[EXT_REFREC]
WR_DATA_DELAY	Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.
WRREC	The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WR} .
WRTORD	Last write pair to read command. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank as t_{WTR} .

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in [DDR SDRAM timing configuration 0 \(DDR_TIMING_CFG_0\)](#), [DDR SDRAM timing configuration 1 \(DDR_TIMING_CFG_1\)](#), [DDR SDRAM timing configuration 2 \(DDR_TIMING_CFG_2\)](#), and [DDR SDRAM timing configuration 3 \(DDR_TIMING_CFG_3\)](#),) and be kept in the DDR memory controller configuration register space.

The following figures show SDRAM timing for various types of accesses. System software is responsible (at reset) for optimally configuring SDRAM timing parameters. The programmable timing parameters apply to both read and write timing configuration. The configuration process must be completed and the DDR SDRAM initialized before any accesses to SDRAM are attempted.

The following figures show DDR SDRAM timing for various types of accesses. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are identical to burst-reads. These figures assume the CLK_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle.

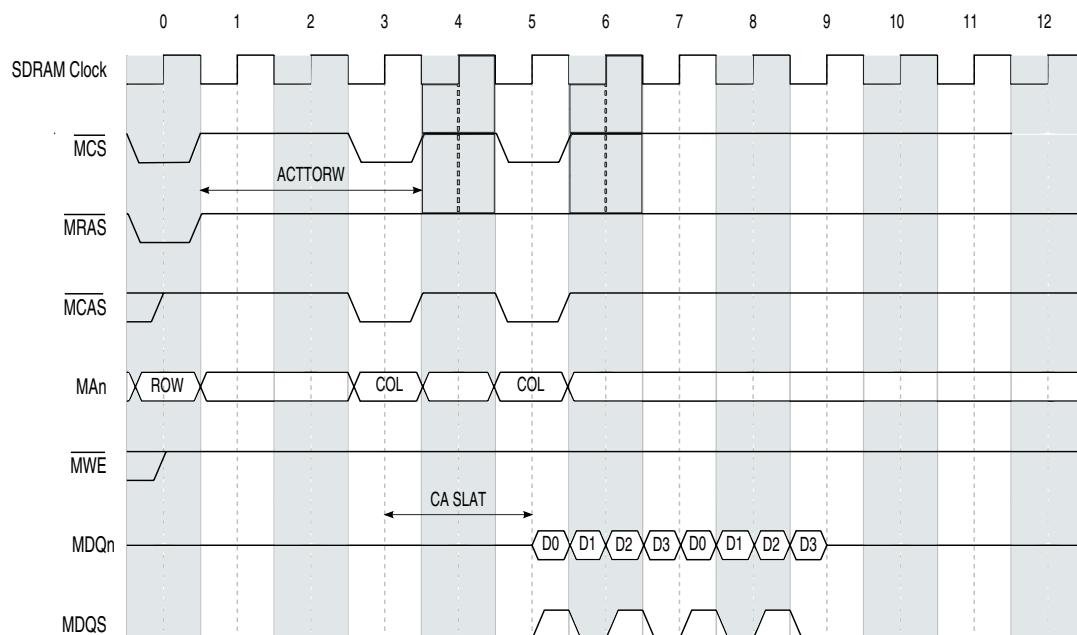


Figure 11-231. DDR SDRAM Burst Read Timing- $\text{ACTTORW} = 3$, $\text{MCAS_B Latency} = 2$

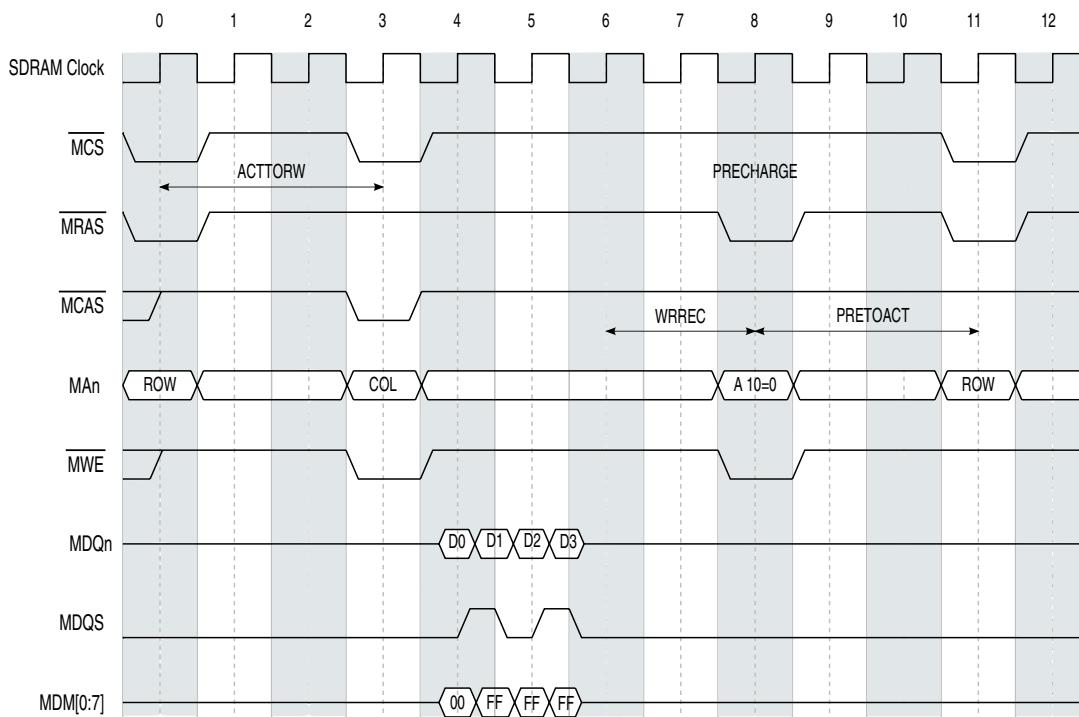


Figure 11-232. DDR SDRAM Single-Beat (Double Word) Write Timing-ACTTROW = 3

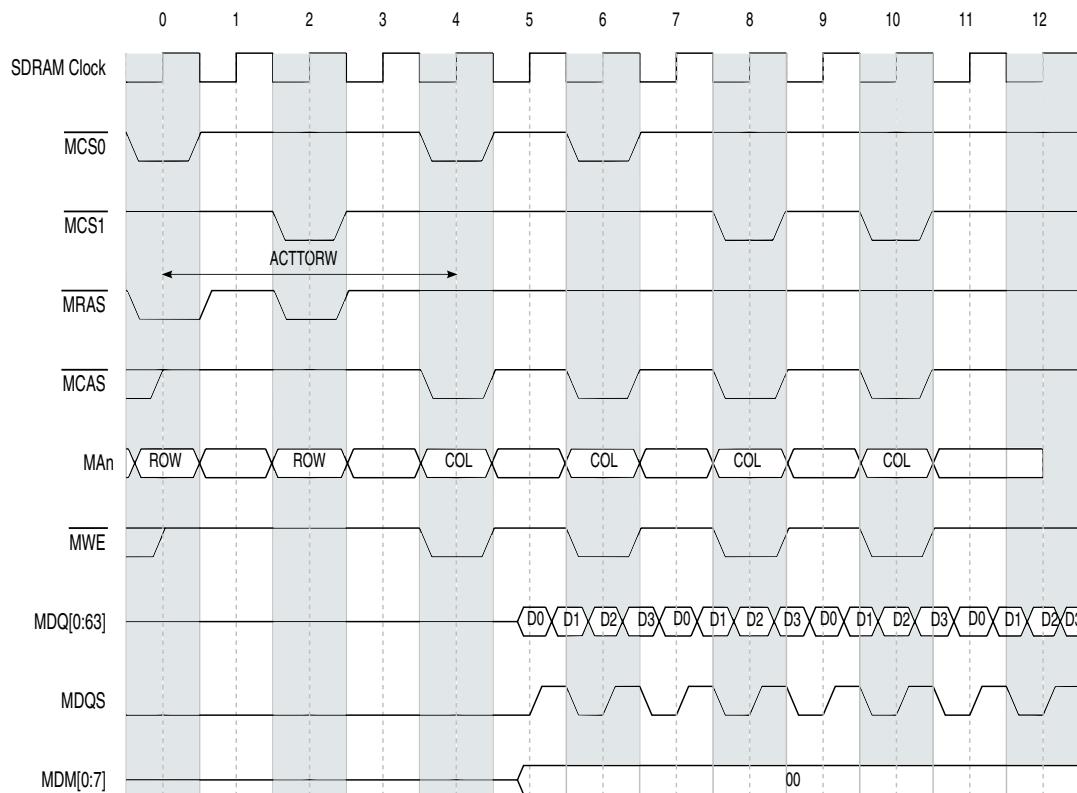


Figure 11-233. DDR SDRAM 4-Beat Burst Write Timing-ACTTROW = 4

11.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the mode register set commands to the SDRAM array, and it uses the setting of TIMING_CFG_0[MRS_CYC] for the Mode Register Set cycle time.

Figure 11-234 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

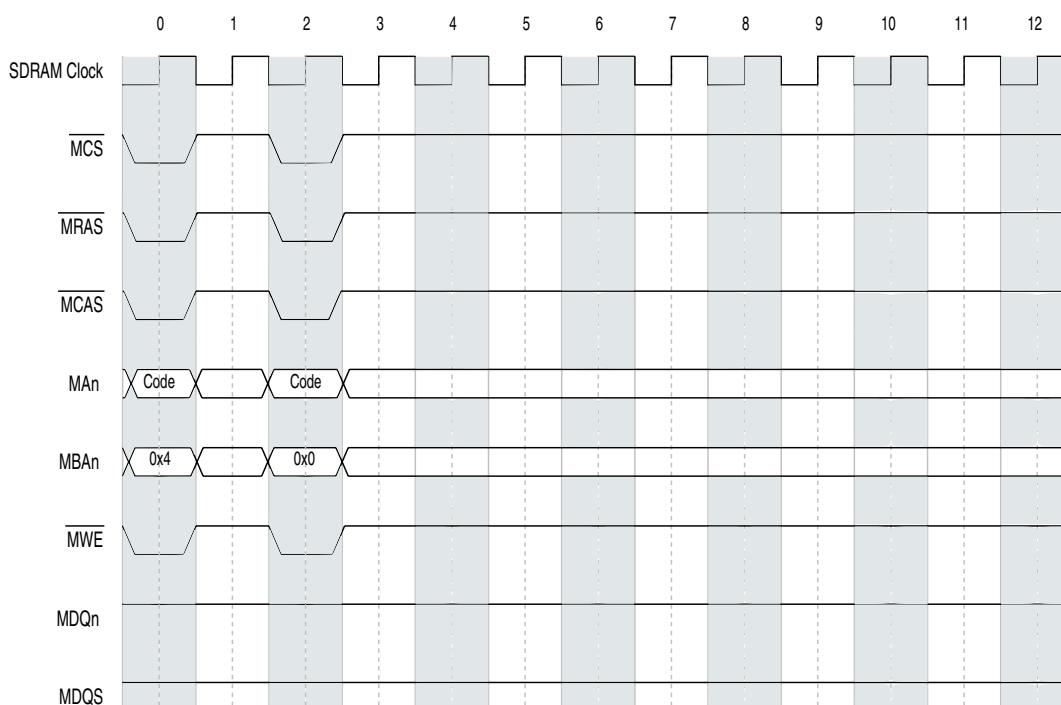


Figure 11-234. DDR SDRAM Mode-Set Command Timing

11.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting DDR_SDRAM_CFG[RD_EN] compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR

memory controller configuration registers, and must deassert the reset signal on DDR memory devices before DDR_SDRAM_CFG[MEM_EN] is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

The following figure shows the registered DDR SDRAM DIMM single-beat write timing.

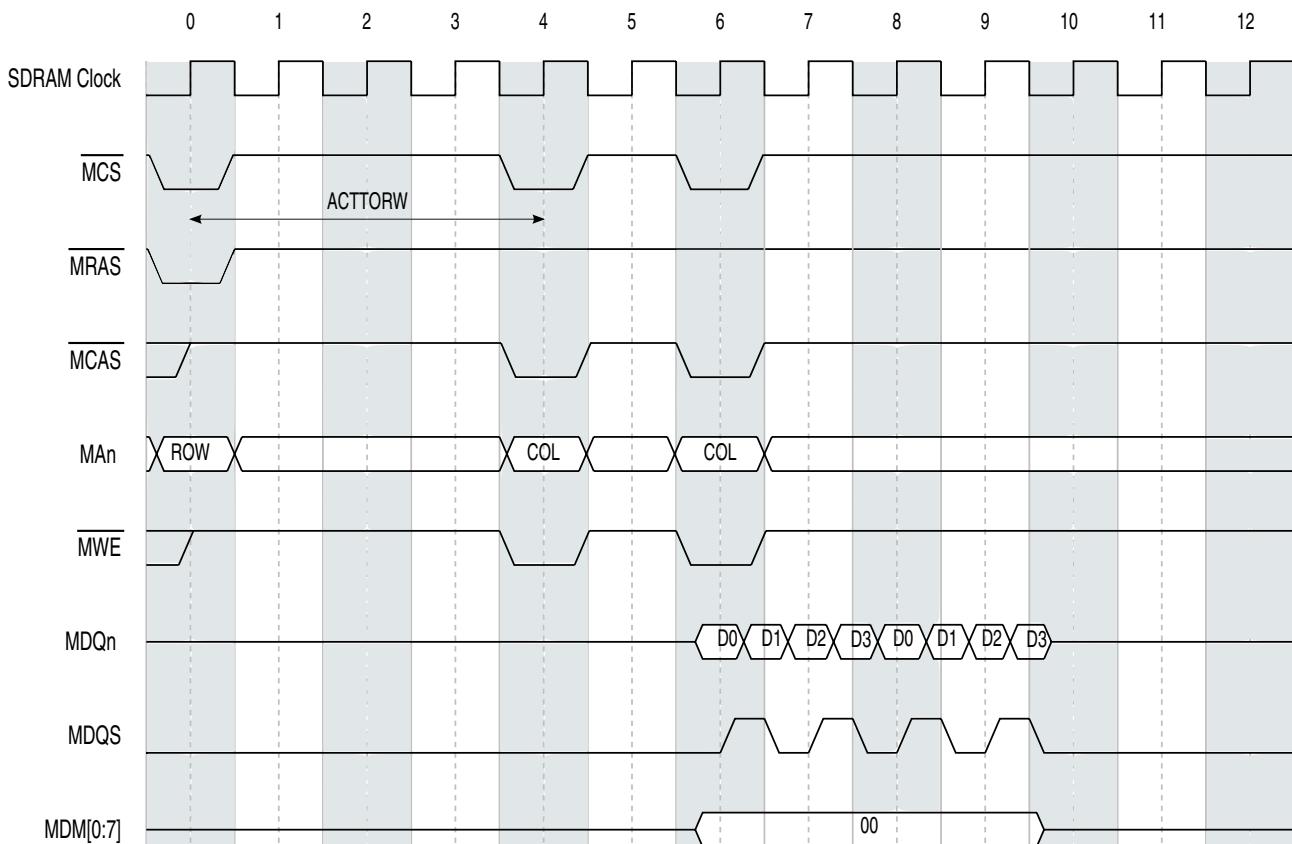


Figure 11-235. Registered DDR SDRAM DIMM Burst Write Timing

11.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period-and no later than 125% of a clock period-from the capturing clock edge of the command/

address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMMs.

TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

[Figure 11-236](#) shows the use of the WR_DATA_DELAY parameter.

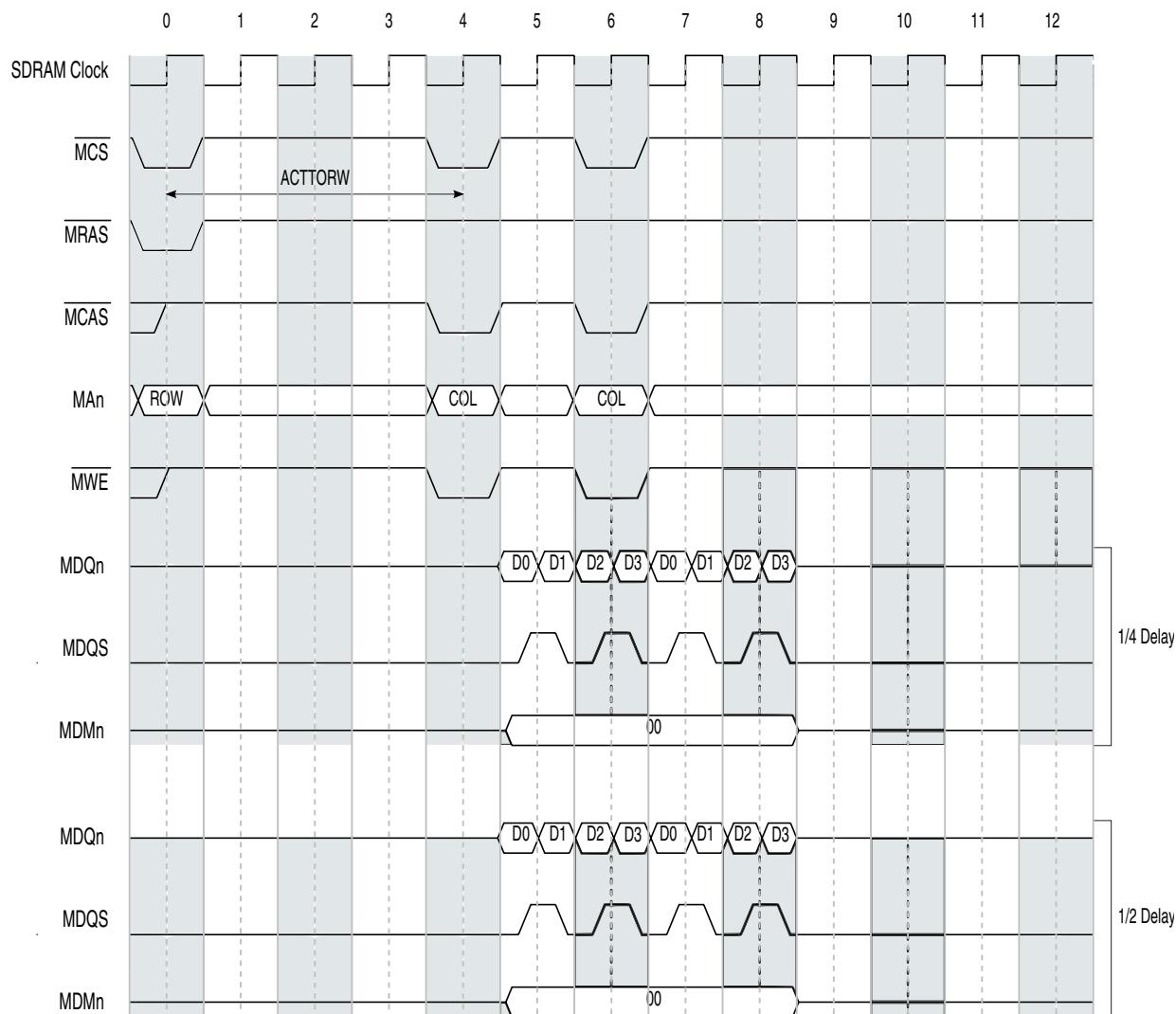


Figure 11-236. Write Timing Adjustments Example for Write Latency = 1

11.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.
2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto-refresh commands are staggered across the four possible banks to reduce the system's instantaneous power requirements. Two sets of auto refresh commands will be issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in two groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. For this entire refresh sequence, no cycle optimization occurs for the usual case where fewer than four banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

11.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in the following figure (TIMING_CFG_1 [REFREC] = 10 in this example).

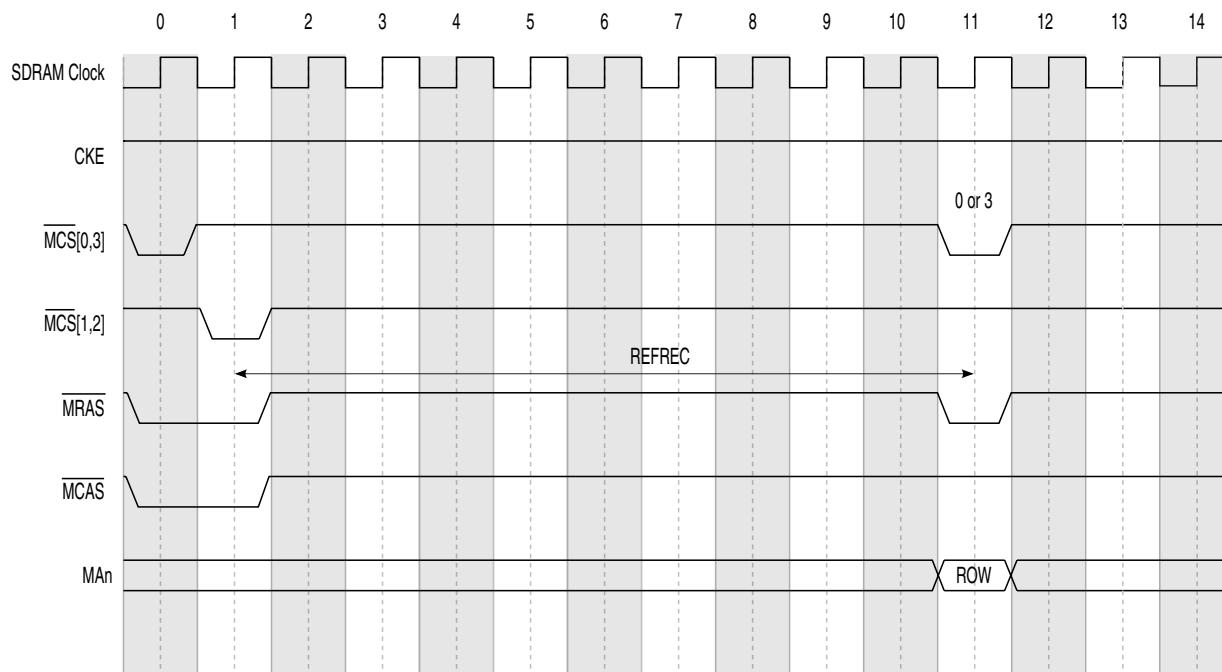


Figure 11-237. DDR SDRAM Bank Staggered Auto Refresh Timing

NOTE

Although the figure shows the controller activating bank 0 or 3 after the refresh recovery interval has expired in cycle 11, the controller could activate bank 1 or 2 instead of bank 0 or 3.

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

11.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

[Table 11-246](#) summarizes the refresh types available in each power-saving mode.

Table 11-246. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	-

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in [Figure 11-238](#).

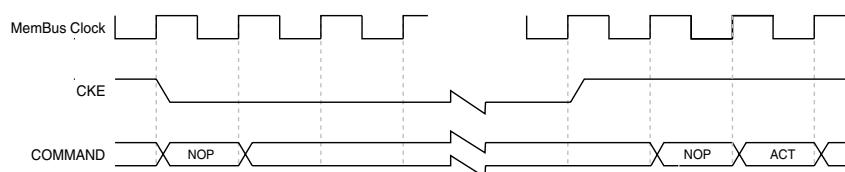


Figure 11-238. DDR SDRAM Power-Down Mode

11.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in [Figure 11-239](#) and [Figure 11-240](#).

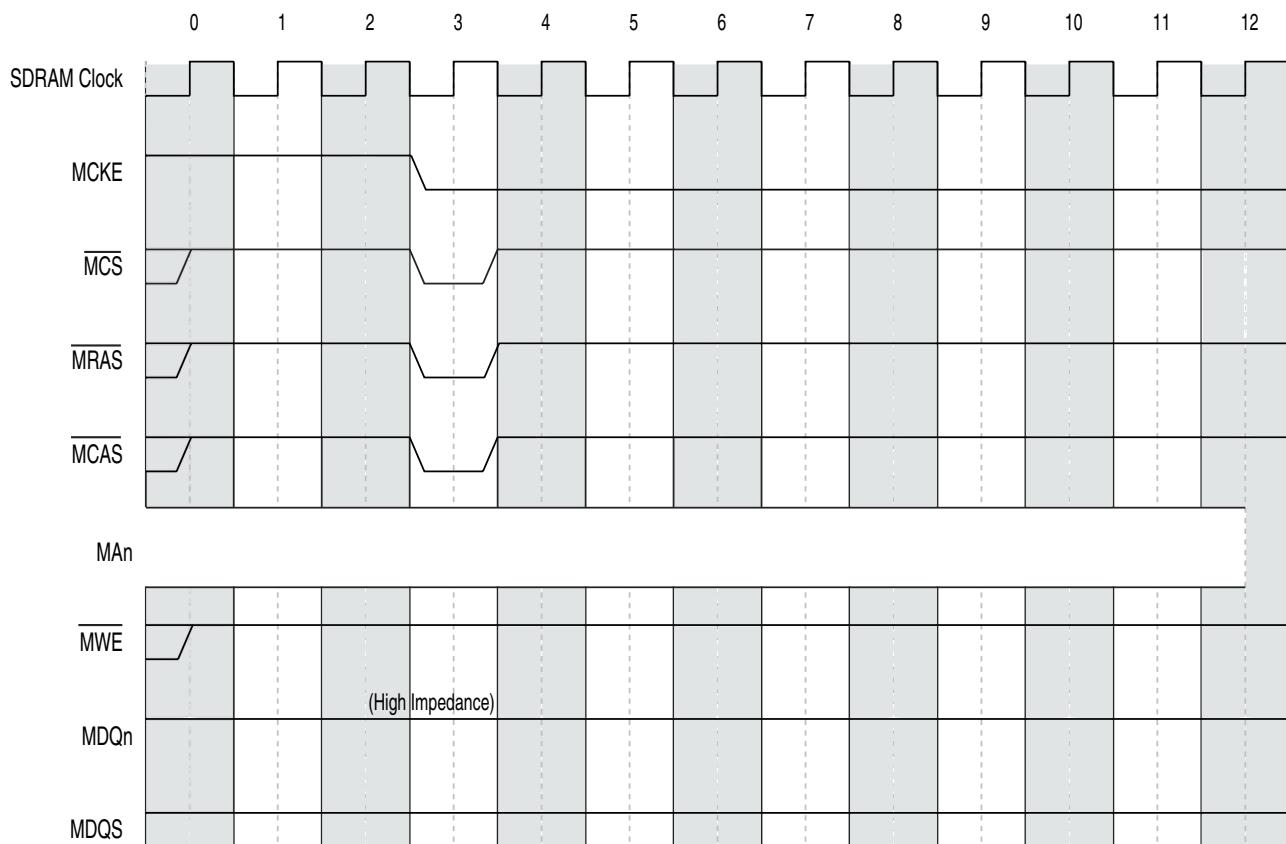


Figure 11-239. DDR SDRAM Self-Refresh Entry Timing

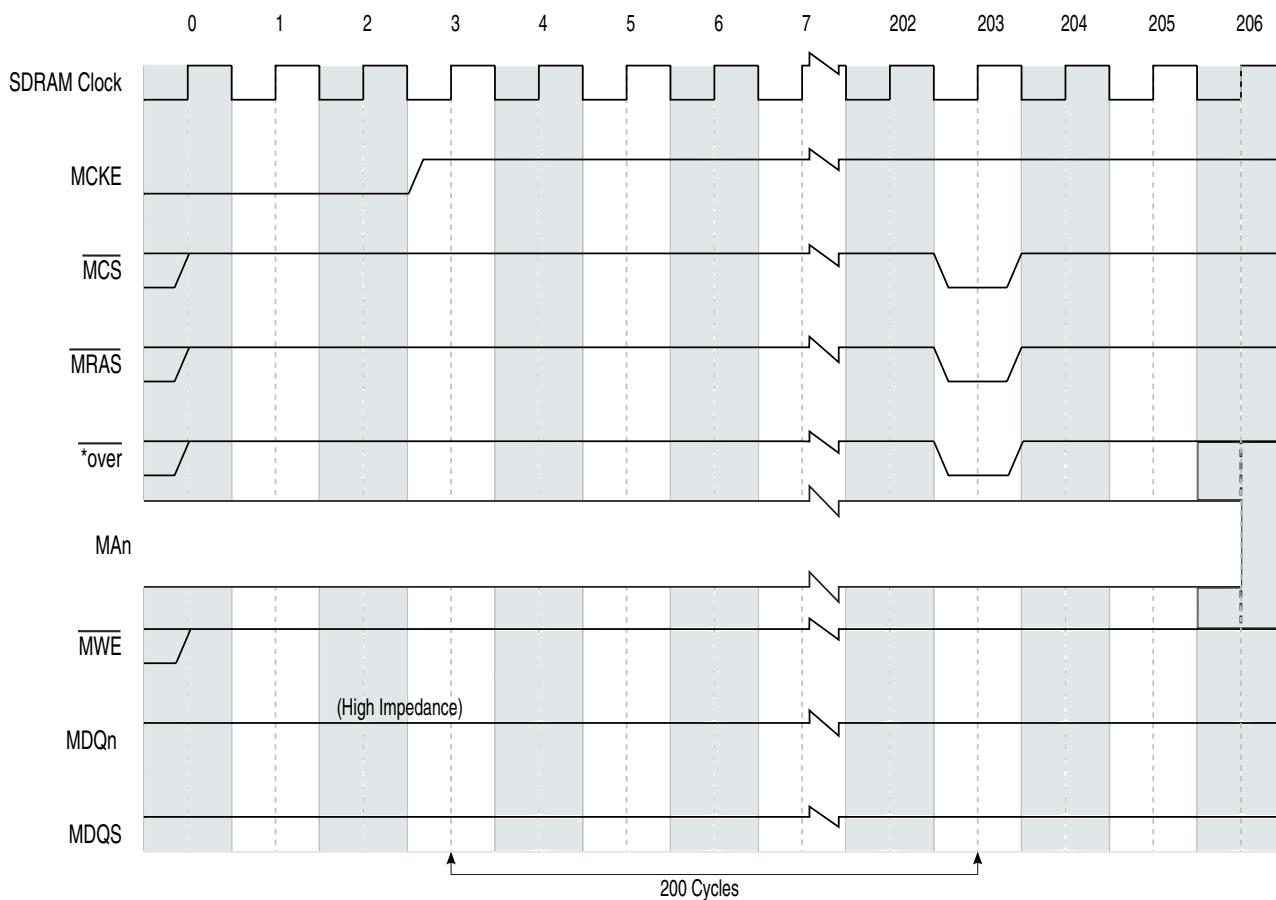


Figure 11-240. DDR SDRAM Self-Refresh Exit Timing

NOTE

Deep sleep mode is not supported for DDR3 registered DIMMS.

11.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not doubleword aligned or the size is not a multiple of a doubleword, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is doubleword aligned with a size that is a multiple of a doubleword, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the second, third, and fourth beats of data are not written to DRAM.

All writes for DDR3 mode are aligned to beat 0 of the DRAM.

11.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains open until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CS n_CONFIG[AP_nEN].

11.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Double-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in [Table 11-247](#) and [Table 11-248](#).

Table 11-247. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
0	•	•						•	32			•	•				•
1	•		•					•	33			•		•			•
2	•			•				•	34	•		•		•			
3	•				•			•	35		•	•		•			
4	•	•				•			36			•	•		•		
5	•		•			•			37			•		•	•		
6	•			•		•			38	•		•		•	•		•
7	•				•	•			39		•	•		•	•		•
8	•	•					•		40			•	•				•
9	•		•				•		41			•		•			•
10	•			•			•		42	•		•		•	•		•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•				•	•	
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	
20		•	•				•		52		•				•		•
21		•		•			•		53			•			•		•
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•					•		•
24		•	•					•	56		•				•	•	
25		•		•				•	57			•			•	•	
26		•			•			•	58				•		•	•	
27	•	•			•			•	59	•					•	•	
28		•	•	•			•	•	60				•	•		•	

Table continues on the next page...

**Table 11-247. DDR SDRAM ECC Syndrome Encoding
(continued)**

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 11-248. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

11.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in [Memory error interrupt enable \(DDR_ERR_INT_EN\)](#), [Memory error disable \(DDR_ERR_DISABLE\)](#), and [Memory error detect \(DDR_ERR_DETECT\)](#).

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates an error interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates an error interrupt (if enabled, as described in [Memory error disable \(DDR_ERR_DISABLE\)](#)). Another error the DDR memory controller detects is a

memory select error, which causes the DDR memory controller to log the error and generate an error interrupt (if enabled, as described in [Memory error detect \(DDR_ERR_DETECT\)](#)). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. [Table 11-249](#) shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 11-249. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.		
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.	The error is reported via an error interrupt if enabled.	The error control register only logs read versus write, not full type
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

11.5.13 DDR Rapid Clear of Memory

Upon Hard Fail condition, the security monitor drives an interrupt to the DDR controller indicating the memory needs to be cleared, the DDR controller will clear all of memory defined by the CSn_BNDS registers for all enabled chip selects. DDRDSR_2[RPD_ST] will be set once the DDR controller begins clearing memory. At this time, writes to the DDR CCSR space are not allowed to modify the register values. Once the rapid clear of memory sequence is complete, writes to the DDR registers may proceed. The DDRDSR_2[RPD_EN] bit is provided so software can determine when the rapid clear of memory sequence is complete and can therefore update registers again. Software can clear the RPD_ST and RPD_EN bits after the rapid memory clear is complete.

11.6 DDR Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate MCSn_B signal for memory accesses according to the provided bank depths. System software must also

configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in [Chip select n configuration \(DDR_CS_n_CONFIG\)](#). Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See the [DDR Memory Controller Memory Map](#) for more detailed descriptions of the configuration registers. These parameters are shown in [Table 11-250](#).

Table 11-250. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter		Section/Page
CS n_BNDS	Chip select memory bounds	SA n EA n		Chip select n memory bounds (DDR_CS_n_BNDS)
CS n_CONFIG	Chip select configuration	CS_n_EN INTLV_EN INTLV_CTL AP_n_EN ODT_RD_CFG ODT_WR_CFG	BA_BITS_CS_n ROW_BITS_CS_n COL_BITS_CS_n	Chip select n configuration (DDR_CS_n_CONFIG)
CS _n _CONFIG_2	Chip select configuration 2	PASR_CFG		Chip select n configuration 2 (DDR_CS_n_CONFIG_2)
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC EXT_ACTTOPRE EXT_CASLAT CNTL_ADJ		DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3)
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE	WR_DATA_DELAY CKE_PLS FOUR_ACT	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)

Table continues on the next page...

**Table 11-250. Memory Interface Configuration Register Initialization Parameters
(continued)**

Name	Description	Parameter		Section/Page
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW	NCAP 2T_EN 3T_EN BA_INTLV_CTL x32_EN HSE BI	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)
DDR_SDRAM_CFG_2	Control configuration	SR_IE DLL_RST_DIS DQS_CFG ODT_CFG	NUM_PR AP_EN D_INIT RCW_EN MD_EN	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE		DDR SDRAM mode configuration (DDR_DDR_SDRAM_MODE)
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3		DDR SDRAM mode configuration 2 (DDR_DDR_SDRAM_MODE_2)
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE		DDR SDRAM interval configuration (DDR_DDR_SDRAM_INTERVAL)
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE		DDR SDRAM data initialization (DDR_DDR_DATA_INIT)
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST		DDR SDRAM clock control (DDR_DDR_SDRAM_CLK_CNTL)
DDR_INIT_ADDR	Initialization address	INIT_ADDR		DDR training initialization address (DDR_DDR_INIT_ADDR)
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR		DDR training initialization extended address (DDR_DDR_INIT_EXT_ADDRESS)
TIMING_CFG_4	Timing configuration	RWT WRT		DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)

Table continues on the next page...

**Table 11-250. Memory Interface Configuration Register Initialization Parameters
(continued)**

Name	Description	Parameter	Section/Page
		RRT WWT DLL_LOCK	
TIMING_CFG_5	Timing configuration	RODT_ON RODT_OFF WODT_ON WODT_OFF	DDR SDRAM timing configuration 5 (DDR_TIMING_CFG_5)
DDR_ZQ_CNTL	ZQ calibration control	ZQ_EN ZQINIT ZQOPER ZQCS	DDR ZQ calibration control (DDR_DDR_ZQ_CNTL)
DDR_WRLVL_CNTL	Write leveling control	WRLVL_EN WRLVL_MRД WRLVL_ODTEN WRLVL_DQSEN WRLVL_SMPL WRLVL_WLR WRLVL_START	DDR write leveling control (DDR_DDR_WRLVL_CNTL)
DDR_WRLVL_CNTL_2	Write leveling control	WRLVL_START_1 WRLVL_START_2 WRLVL_START_3 WRLVL_START_4	DDR write leveling control 2 (DDR_DDR_WRLVL_CNTL_2)
DDR_WRLVL_CNTL_3	Write leveling control	WRLVL_START_5 WRLVL_START_6 WRLVL_START_7 WRLVL_START_8	DDR write leveling control 3 (DDR_DDR_WRLVL_CNTL_3)
DDR_SR_CNTR	Self refresh control	SR_IT	DDR Self Refresh Counter (DDR_DDR_SR_CNTR)
DDR_SDRAM_RCW_1	Register control words configuration	RCW0 RCW1 RCW2 RCW3 RCW4 RCW5 RCW6 RCW7	DDR Register Control Words 1 (DDR_DDR_SDRAM_RCW_1)

Table continues on the next page...

Table 11-250. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter		Section/Page
DDR_SDRAM_RCW_2	Register control words configuration	RCW8 RCW9 RCW10 RCW11 RCW12 RCW13 RCW14 RCW15		DDR Register Control Words 2 (DDR_DDR_SDRAM_RCW_2)
DDRCDR_1	Driver control	DHC_EN ODT DSO_C_EN DSO_D_EN	DSO_CPZ DSO_CNZ DSO_DPZ DSO_DNZ	DDR Control Driver Register 1 (DDR_DDRCDR_1)
DDRCDR_2	Driver control	DSO_CLK_EN DSO_CLKPZ DSO_CLKNZ		DDR Control Driver Register 2 (DDR_DDRCDR_2)

11.6.1 Programming Differences Between Memory Types

Depending on the memory type used, certain fields must be programmed differently.

[Table 11-251](#) illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 11-251. Programming Differences Between Memory Types

Parameter	Description	Differences		Section/page
APn_EN	Chip Select nAuto Precharge Enable	DDR2	Can be used to place chip select <i>n</i> in auto precharge mode	Chip select n configuration (DDR_CSn_CONFIG)
		DDR3	Can be used to place chip select <i>n</i> in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.	Chip select n configuration (DDR_CSn_CONFIG)
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology.	

Table continues on the next page...

Table 11-251. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
		However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.		
ODT_WR_CFG	Chip Select ODT Write Configuration	DDR2	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	Chip select n configuration (DDR_CSn_CONFIG)
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	DDR2	Should be set according to the DDR2 specifications for the memory used. The JEDEC parameter this applies to is t_{AXPD} .	DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)
		DDR3	Should be set to 0001 for DDR3. The powerdown times (t_{XP} and $t_{XP DLL}$) required for DDR3 are controlled via $TIMING_CFG_0[ACT_PD_EXIT]$ and $TIMING_CFG_0[PRE_PD_EXIT]$.	
PRETOACT	Precharge to Activate Timing	DDR2	Should be set according to the specifications for the memory used (t_{RP})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set according to the specifications for the memory used (t_{RP})	
ACTTOPRE	Activate to Precharge Timing	DDR2	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	DDR2	Should be set according to the specifications for the memory used (t_{RCD})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set according to the specifications for the memory used (t_{RCD})	

Table continues on the next page...

Table 11-251. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
CASLAT	CAS Latency	DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	DDR2	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	
WRREC	Write Recovery	DDR2	Should be set according to the specifications for the memory used (t_{WR})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set according to the specifications for the memory used (t_{WR}).	
ACTTOACT	Activate A to Activate B	DDR2	Should be set according to the specifications for the memory used (t_{RRD})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set according to the specifications for the memory used (t_{RRD})	
WRTORD	Write to Read Timing	DDR2	Should be set according to the specifications for the memory used (t_{WTR})	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
		DDR3	Should be set according to the specifications for the memory used (t_{WTR})	
ADD_LAT	Additive Latency	DDR2	Should be set to the desired additive latency. This must be set to a value less than $\text{TIMING_CFG_1}[ACTTORW]$	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)
		DDR3	Should be set to the desired additive latency. This must be set to a value less than $\text{TIMING_CFG_1}[ACTTORW]$	
WR_LAT	Write Latency	DDR2	Should be set to CAS latency - 1 cycle. For example, if the CAS latency is 5 cycles, then this field should be set to 100 (4 cycles). The minimum WR_LAT that can be used in 1T timing mode is 5 cycles if $\text{DDR_RATE}=0$	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)

Table continues on the next page...

Table 11-251. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
		DDR3	Should be set to the desired write latency. Note that DDR3 SDRAMs do not necessarily require the write latency to equal the CAS latency minus 1 cycle. The minimum WR_LAT that can be used in 1T timing mode is 5 cycles if DDR_RATE=0	
RD_TO_PRE	Read to Precharge Timing	DDR2	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)
		DDR3	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR2	Should be set according to the specifications for the memory used (t_{CKE})	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)
		DDR3	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	DDR2	Should be set according to the specifications for the memory used (t_{FAW}). Only applies to eight logical banks.	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)
		DDR3	Should be set according to the specifications for the memory used (t_{FAW}).	
RD_EN	Registered DIMM Enable	DDR2	If registered DIMMs are used, then this field should be set to 1	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)
		DDR3	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	DDR2	Should be set to 0 regardless of bus width. In 32-bit bus mode, 32-byte burst accesses from the platform will be split into two 16-byte (4-beat) bursts to the SDRAMs.	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)
		DDR3	If a 64-bit bus is used, this should be set to 0. Otherwise, this should be set to 1. If this is set to 0, then other requirements in TIMING_CFG_4 will be needed to ensure tCCD is met.	
2T_EN	2T Timing Enable	DDR2	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)

Table continues on the next page...

Table 11-251. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
		DDR3	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/command bandwidth.	
DLL_RST_DIS	DLL Reset Disable	DDR2	Should typically be set to 0, unless it is desired to bypass the DLL reset when exiting self refresh.	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)
		DDR3	Should be set to 1	
DQS_CFG	DQS Configuration	DDR2	Should be set to 01. Only differential data strobes are supported.	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)
		DDR3	Should be set to 01	
ODT_CFG	ODT Configuration	DDR2	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)
		DDR3	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
RWT	Read-to-write turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)
		DDR3	This can be used to force a longer read-to-write turnaround time when accessing the same chip select. This is useful for burst chop mode, as there are some timing requirements to the same chip select that still must be met.	
WRT	Write-to-read turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)
		DDR3	This could be used to force a certain turnaround time between a write and read to the same chip select. This is useful for burst chop mode. However, it is expected that TIMING_CFG_1[WRTORD] will be programmed appropriately such that TIMING_CFG_4[WRT] can be set to 0000.	
RRT	Read-to-read turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)
		DDR3	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	
WWT	Write-to-write turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)
		DDR3	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	

Table continues on the next page...

Table 11-251. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
ZQ_EN	ZQ Calibration Enable	DDR2	Should be set to 0	DDR ZQ calibration control (DDR_DDR_ZQ_CNTL)
		DDR3	Should be set to 1. The other fields in DDR_ZQ_CNTL should also be programmed appropriately based on the DRAM specifications.	
WRLVL_EN	Write Leveling Enable	DDR2	Should be set to 0	DDR write leveling control (DDR_DDR_WRLVL_CNTL)
		DDR3	Can be set to 1 if write leveling is desired. Otherwise the value used in TIMING_CFG_2[WR_DATA_DELAY] will be used to shift all bytes during writes to DRAM. If write leveling will be used, all other fields in DDR_WRLVL_CNTL should be programmed appropriately based on the DRAM specifications.	
BSTOPRE	Burst To Precharge Interval	DDR2	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	DDR SDRAM interval configuration (DDR_DDR_SDRAM_INTERVAL)
		DDR3	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

11.6.2 Supported DDR Interleaving Configurations

Considering the options for chip select, cache line, page, bank, and super-bank interleaving, the DDR controllers could potentially support hundreds of combinations, many of which are of limited utility. For this reason, only a subset of the possible combinations are supported. The following table describes the supported DDR interleaving configurations. There are additional restrictions for CPC SRAM mode and DDR interleaving options. See [Supported SRAM Mode Configurations](#), for more information.

NOTE

If only one memory controller is being used, any non-reserved bank (chip select) interleaving (DDR_SDRAM_CFG[BA_INTLV_CTL]) setting may be selected. However, if both memory controllers are being used, the bank (chip select) interleaving (DDR_SDRAM_CFG[BA_INTLV_CTL]) settings of both controllers must be the same.

Table 11-252. Supported DDR Interleaving Configurations

Memory Controller Interleaving CS0_CONFIG[INTLV_EN], CS0_CONFIG[INTLV_CTL]	DDR Controller 1 Bank (Chip Select) Interleaving DDR_SDRAM_CFG[BA_INTLV_CTL]	DDR Controller 2 Bank (Chip Select) Interleaving DDR_SDRAM_CFG[BA_INTLV_CTL]
Disabled	Disabled	DDR controller 2 not used
INTLV_EN = 0	BA_INTLV_CTL = 0x00 External memory banks 0/1 BA_INTLV_CTL = 0x40 External memory banks 2/3 BA_INTLV_CTL = 0x20 External memory banks 0/1 and 2/3 BA_INTLV_CTL = 0x60 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	
	DDR controller 1 not used BA_INTLV_CTL = 0x00 External memory banks 0/1 BA_INTLV_CTL = 0x40 External memory banks 2/3 BA_INTLV_CTL = 0x20 External memory banks 0/1 and 2/3 BA_INTLV_CTL = 0x60 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	Disabled BA_INTLV_CTL = 0x00 External memory banks 0/1 BA_INTLV_CTL = 0x40 External memory banks 2/3 BA_INTLV_CTL = 0x20 External memory banks 0/1 and 2/3 BA_INTLV_CTL = 0x60 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
	Disabled BA_INTLV_CTL = 0x00 External memory banks 0/1 BA_INTLV_CTL = 0x40 External memory banks 2/3 BA_INTLV_CTL = 0x20 External memory banks 0/1 and 2/3 BA_INTLV_CTL = 0x60 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	Disabled BA_INTLV_CTL = 0x00 External memory banks 0/1 BA_INTLV_CTL = 0x40 External memory banks 2/3 BA_INTLV_CTL = 0x20 External memory banks 0/1 and 2/3 BA_INTLV_CTL = 0x60 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
Memory controller cache line interleaving INTLV_EN = 1 INTLV_CTL = 0x0	Disabled ^{1, 2} BA_INTLV_CTL = 0x00 External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	Disabled ^{1, 2} BA_INTLV_CTL = 0x00 External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40 External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04

Table continues on the next page...

Table 11-252. Supported DDR Interleaving Configurations (continued)

Memory Controller Interleaving CS0_CONFIG[INTLV_EN], CS0_CONFIG[INTLV_CTL]	DDR Controller 1 Bank (Chip Select) Interleaving DDR_SDRAM_CFG[BA_INTLV_CTL]	DDR Controller 2 Bank (Chip Select) Interleaving DDR_SDRAM_CFG[BA_INTLV_CTL]
Memory controller page interleaving INTLV_EN = 1 INTLV_CTL = 0x1	Disabled ^{1, 2} BA_INTLV_CTL = 0x00	Disabled ^{1, 2} BA_INTLV_CTL = 0x00
	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40
	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
	Disabled ^{1, 2} BA_INTLV_CTL = 0x00	Disabled ^{1, 2} BA_INTLV_CTL = 0x00
	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40	External memory banks 0/1 BA_INTLV_CTL = 0x40 ^{1, 3}
Memory controller bank interleaving INTLV_EN = 1 INTLV_CTL = 0x2	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40
	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40
	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04
Memory controller super-bank interleaving INTLV_EN = 1 INTLV_CTL = 0x3	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40	External memory banks 0/1 ^{1, 3} BA_INTLV_CTL = 0x40
	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04	External memory banks 0/1/2/3 BA_INTLV_CTL = 0x04

1. If controller interleaving is enabled, then all chip-selects of a DDR controller must be interleaved or left unused. The chip-selects that are not interleaved by chip-select interleaving are inaccessible during controller interleaving.
2. Memory at chip selects 1, 2, and 3 are inaccessible.
3. Memory at chip selects 2 and 3 are inaccessible.

11.6.3 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs (500 µs for DDR3) must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller will conduct an automatic initialization sequence to the memory, which will follow the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

11.6.4 Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System

This section describes the options offered by this device to support battery-backed main memory.

11.6.4.1 Hardware Based Self-Refresh Scheme

An external voltage sense device can be connected to this device through one of the external interrupt lines IRQ_n . The external interrupt from the voltage sensor would then be steered through the programmable interrupt controller (MPIC) to the internal SoC interrupt event signal, $sie0$. Note that the $sie0$ signal must remain high until power is removed. Also note that $sie0$ is sent to both DDR controllers at the same time.

If $\text{DDR_SDRAM_CFG_2}[SR_IE]$ is set, the $sie0$ signal from the interrupt controller is then automatically detected by the DDR controller, which immediately causes main memory to enter self-refresh mode. See [Chip select n configuration 2 \(DDR_CS \$_n\$ _CONFIG_2\)](#), for further information on this bit.

These fields in the appropriate registers in the MPIC must be set for self refresh to function:

- $EIVPR_n[PRIORITY]$ should be set to 0xF (highest priority)
- $EILR_n[INTTGT]$ should be set to route the incoming signal to $sie0$

See [External interrupt n \(IRQ \$_n\$ \) vector/priority register \(MPIC_EIVPR\)](#), and [External interrupt n \(IRQ \$_n\$ \) level register \(MPIC_EILR\)](#), for descriptions of these registers.

11.6.4.2 Software Based Self-Refresh Scheme

The DDR controller also has a software-programmable bit, $\text{DDR_SDRAM_CFG_2}[FRC_SR]$, that immediately puts main memory into self-refresh mode. See [DDR SDRAM control configuration 2 \(DDR_DDR_SDRAM_CFG_2\)](#), for a description of this register.

It is expected that a critical interrupt routine triggered by an external voltage sensing device will have time to set this bit.

11.6.4.3 Bypassing Re-initialization During Battery-Backed Operation

The DDR controller offers an initialization bypass feature (DDR_SDRAM_CFG[BI]), which system designers may use to prevent re-initialization of main memory during system power-on following an abnormal shutdown. See [DDR SDRAM control configuration 2 \(DDR_DDR_SDRAM_CFG_2\)](#), for information on this bit and [DDR training initialization address \(DDR_DDR_INIT_ADDR\)](#), for a discussion of avoiding possible ECC errors in this mode.

Note that the DDR controller will automatically wait 200 DRAM cycles before issuing any command after the assertion of MCKE[0:3] when this mode is used.

Chapter 12

I²C Modules

12.1 I²C Overview

This chapter describes the dual inter-integrated circuit (I²C) bus modules implemented on this device and covers the following topics:

- [Introduction to I²C](#)
- [I²C Signal Descriptions](#)
- [I²C Controller Memory Map](#)
- [I²C Functional Description](#)
- [I²C Initialization/Application Information](#)

12.2 Introduction to I²C

This section presents the following topics:

- [Definition: I²C Module](#)
- [Advantages of the I²C Bus](#)
- [I²C Module Block Diagram](#)
- [I²C Features Summary](#)
- [I²C Modes of Operation](#)
- [Definition: I²C Conditions](#)

12.2.1 Definition: I²C Module

The I²C module is a functional unit that provides a two-wire-serial data (SDA) and serial clock (SCL)-bidirectional serial bus that provides a simple efficient method of data exchange between this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs.

12.2.2 Advantages of the I²C Bus

The two-wire I²C bus minimizes interconnections between devices. The synchronous, multiple-master I²C bus allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

12.2.3 I²C Module Block Diagram

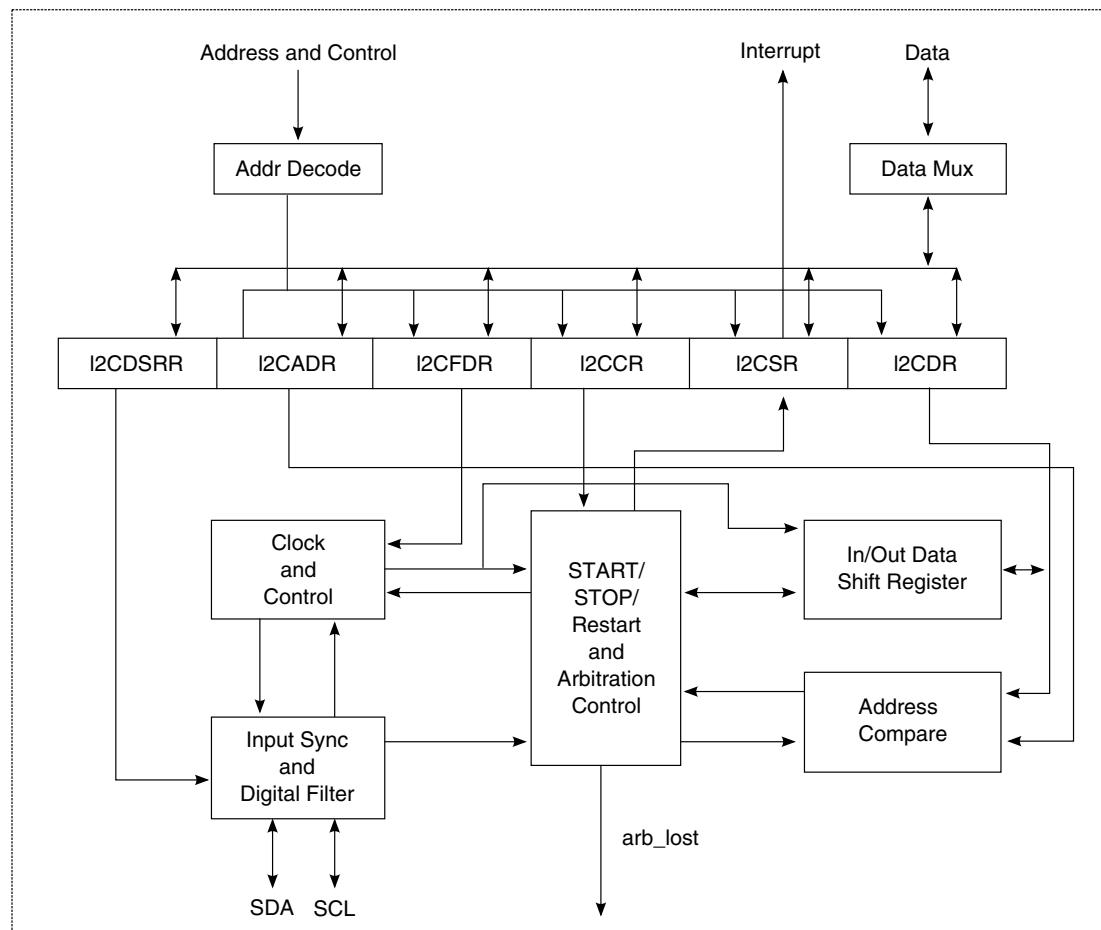


Figure 12-1. I²C Module Block Diagram

12.2.4 I²C Features Summary

The I²C module includes the following features:

- Acknowledge bit generation and detection
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Bus busy detection
- Calling-address identification interrupt
- Multiple-master operation
- On-chip filtering for spikes on the bus
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- START and STOP signal generation and detection
- Two-wire interface

12.2.5 I²C Modes of Operation

The I²C modules can operate in one of several modes, as shown in the following table.

Table 12-1. I²C Module Modes of Operation

Mode	Description	Important Notes
Master mode	The I ² C module is the driver of the SDA line.	<ul style="list-style-type: none"> • Do not use the I²C module's slave address as a calling address. • The I²C module cannot be a master and a slave simultaneously.
Slave mode	The I ² C module is not the driver of the SDA line.	<ul style="list-style-type: none"> • Enable the I²C module before a START condition from a non-I²C master is detected. • By default the I²C module performs as a slave receiver.
Interrupt-driven byte-to-byte data transfer mode	When successful slave addressing is achieved (and the I ² C module deasserts SCL), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling master.	Follow each byte of data by an acknowledge bit, which is signaled from the receiving device.

12.2.6 Definition: I²C Conditions

Table 12-2. I²C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.
STOP	A condition generated by the master to terminate a transfer and free the bus.

12.3 I²C Signal Descriptions

This section presents the following topics:

- [Signal Overview](#)
- [I²C Detailed Signal Descriptions](#)

12.3.1 Signal Overview

The I²C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the chip .

12.3.2 I²C Detailed Signal Descriptions

The SDA and SCL signals are described in the following table.

Table 12-3. I²C Module-Detailed Signal Descriptions

Signal	I/O	Description	
IICn_SCL	I/O	Serial Clock. Performs as an input when the device is programmed as an I ² C slave. SCL also performs as an output when the device is programmed as an I ² C master.	Idle State: High
	O	When the I ² C module is a master, it drives SCL along with SDA when transmitting. As a slave, the I ² C module drives SCL low for data pacing.	As output for the bidirectional serial clock, SCL operates as described below.
		State Meaning	Asserted/Negated - SCL is driven along with SDA as the clock for the data.
	I	When the I ² C module is idle or is acting as a slave, SCL is an input by default. The I ² C module uses SCL to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.	As input for the bidirectional serial clock, SCL operates as described below.
		State Meaning	Asserted/Negated - The I ² C module uses the SCL signal to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.

Table continues on the next page...

Table 12-3. I²C Module-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
IICn_SDA	I/O	Serial Data. Performs as an input when the device is in a receiving mode. SDA also performs as an output signal when the device is transmitting (either as an I ² C master or slave).	
	O	Idle State: High When the I ² C module is writing as a master or slave, it drives data on SDA synchronous to SCL. As output for the bidirectional serial data, SDA operates as described below.	
		State Meaning	Asserted/Negated - Data is driven.
	I	When the I ² C module is idle or is in a receiving mode, SDA is an input by default. The unit receives data from other I ² C devices on SDA. The bus is assumed to be busy when SDA is detected low. As input for the bidirectional serial data, SDA operates as described below.	
		State Meaning	Asserted/Negated - SDA is used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

12.4 I²C Controller Memory Map

The I²C module registers and their offsets are described in this section, as well as register access and register figure conventions. This section also contains important notes about the location, byte order, and read/write accesses of registers.

- Register Location: Locate all I²C registers in a cache-inhibited page.
- Byte Order: The I²C registers are shown in big-endian format. For a system that is in little-endian mode, the software must swap the bytes appropriately. Byte swapping is necessary because the I²C registers are byte registers.
- Read/Write Accesses: To guarantee in-order execution, execute a synchronizing assembly instruction after each I²C register read/write access.
- Writes to Reserved Fields: Reserved bits must be written with the value that they returned when read. Program the register by reading its value, modifying the appropriate fields, and writing back the value.

This section is organized as follows:

- [I²C Controller Memory Map](#)
- [I²C address register \(I2C_I2CADR\)](#)
- [I²C frequency divider register \(I2C_I2CFDR\)](#)
- [I²C control register \(I2C_I2CCR\)](#)
- [I²C status register \(I2C_I2CSR\)](#)

- I²C data register (I2C_I2CDR)
- I²C digital filter sampling rate register (I2C_I2CDFSRR)

The following table lists the I²C registers in offset order and provides links to the complete register descriptions.

I²C memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_8000	I ² C address register (I2C1_I2CADR)	8	R/W	00h	12.4.1/627
11_8004	I ² C frequency divider register (I2C1_I2CFDR)	8	R/W	00h	12.4.2/627
11_8008	I ² C control register (I2C1_I2CCR)	8	R/W	00h	12.4.3/629
11_800C	I ² C status register (I2C1_I2CSR)	8	R/W	81h	12.4.4/631
11_8010	I ² C data register (I2C1_I2CDR)	8	R/W	00h	12.4.5/632
11_8014	I ² C digital filter sampling rate register (I2C1_I2CDFSRR)	8	R/W	10h	12.4.6/633
11_8100	I ² C address register (I2C2_I2CADR)	8	R/W	00h	12.4.1/627
11_8104	I ² C frequency divider register (I2C2_I2CFDR)	8	R/W	00h	12.4.2/627
11_8108	I ² C control register (I2C2_I2CCR)	8	R/W	00h	12.4.3/629
11_810C	I ² C status register (I2C2_I2CSR)	8	R/W	81h	12.4.4/631
11_8110	I ² C data register (I2C2_I2CDR)	8	R/W	00h	12.4.5/632
11_8114	I ² C digital filter sampling rate register (I2C2_I2CDFSRR)	8	R/W	10h	12.4.6/633
11_9000	I ² C address register (I2C3_I2CADR)	8	R/W	00h	12.4.1/627
11_9004	I ² C frequency divider register (I2C3_I2CFDR)	8	R/W	00h	12.4.2/627
11_9008	I ² C control register (I2C3_I2CCR)	8	R/W	00h	12.4.3/629
11_900C	I ² C status register (I2C3_I2CSR)	8	R/W	81h	12.4.4/631
11_9010	I ² C data register (I2C3_I2CDR)	8	R/W	00h	12.4.5/632
11_9014	I ² C digital filter sampling rate register (I2C3_I2CDFSRR)	8	R/W	10h	12.4.6/633
11_9100	I ² C address register (I2C4_I2CADR)	8	R/W	00h	12.4.1/627
11_9104	I ² C frequency divider register (I2C4_I2CFDR)	8	R/W	00h	12.4.2/627
11_9108	I ² C control register (I2C4_I2CCR)	8	R/W	00h	12.4.3/629
11_910C	I ² C status register (I2C4_I2CSR)	8	R/W	81h	12.4.4/631
11_9110	I ² C data register (I2C4_I2CDR)	8	R/W	00h	12.4.5/632
11_9114	I ² C digital filter sampling rate register (I2C4_I2CDFSRR)	8	R/W	10h	12.4.6/633

12.4.1 I²C address register (I2Cx_I2CADR)

The I²C address register (I2CADR) specifies the address to which the I²C module responds if the I²C is addressed as a slave. This is not the address sent on the bus during the address-calling cycle when the I²C module is in master mode.

Address: Base address + 0h offset

Bit	0	1	2	3	4	5	6	7
Read				ADDR				
Write							Reserved	

Reset	0	0	0	0	0	0	0	0
-------	---	---	---	---	---	---	---	---

I2Cx_I2CADR field descriptions

Field	Description
0–6 ADDR	Slave address. The ADDR field specifies the slave address to which the I ² C module responds if it is addressed as a slave. If the I ² C is in slave mode and a transaction's calling address matches I2CADR[ADDR], the module will set I2CSR[MIF], signaling a pending interrupt. (For more information about I2CSR[MIF], see I2C status register (I2C_I2CSR) .)
7 -	This field is reserved. Reserved

12.4.2 I²C frequency divider register (I2Cx_I2CFDR)

The I²C frequency divider register (I2CFDR), shown in the following figure, specifies the ratio used to prescale the clock for selecting a specific bit rate.

For additional guidance about the proper use of I2CFDR and I2CDFSRR on Power Architecture™ integrated host/communications processors, refer to the application note AN2919, *Determining the I²C Frequency Divider Ratio for SCL*.

NOTE

Writing to the Reserved Field: While writing to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

The I2CFDR fields and FDR field settings for clock divider values are listed in the following table.

Address: Base address + 4h offset

Bit	0	1	2	3	4	5	6	7
Read								
Write	Reserved			FDR				

Reset	0	0	0	0	0	0	0	0
-------	---	---	---	---	---	---	---	---

I2Cx_I2CFDR field descriptions

Field	Description																																																																														
0–1 -	This field is reserved. Reserved																																																																														
2–7 FDR	Frequency divider ratio. Specifies the ratio used to prescale the clock for bit rate selection. The serial bit clock frequency of the SCL is equal to the platform clock divided by the designated divider. However, the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described in the following list: FDR Divider (Decimal) <table> <tbody> <tr><td>0x00</td><td>384</td></tr> <tr><td>0x01</td><td>416</td></tr> <tr><td>0x02</td><td>480</td></tr> <tr><td>0x03</td><td>576</td></tr> <tr><td>0x04</td><td>640</td></tr> <tr><td>0x05</td><td>704</td></tr> <tr><td>0x06</td><td>832</td></tr> <tr><td>0x07</td><td>1024</td></tr> <tr><td>0x08</td><td>1152</td></tr> <tr><td>0x09</td><td>1280</td></tr> <tr><td>0x0A</td><td>1536</td></tr> <tr><td>0x0B</td><td>1920</td></tr> <tr><td>0x0C</td><td>2304</td></tr> <tr><td>0x0D</td><td>2560</td></tr> <tr><td>0x0E</td><td>3072</td></tr> <tr><td>0x0F</td><td>3840</td></tr> <tr><td>0x10</td><td>4608</td></tr> <tr><td>0x11</td><td>5120</td></tr> <tr><td>0x12</td><td>6144</td></tr> <tr><td>0x13</td><td>7680</td></tr> <tr><td>0x14</td><td>9216</td></tr> <tr><td>0x15</td><td>10240</td></tr> <tr><td>0x16</td><td>12288</td></tr> <tr><td>0x17</td><td>15360</td></tr> <tr><td>0x18</td><td>18432</td></tr> <tr><td>0x19</td><td>20480</td></tr> <tr><td>0x1A</td><td>24576</td></tr> <tr><td>0x1B</td><td>30720</td></tr> <tr><td>0x1C</td><td>36864</td></tr> <tr><td>0x1D</td><td>40960</td></tr> <tr><td>0x1E</td><td>49152</td></tr> <tr><td>0x1F</td><td>61440</td></tr> <tr><td>0x20</td><td>256</td></tr> <tr><td>0x21</td><td>288</td></tr> <tr><td>0x22</td><td>320</td></tr> <tr><td>0x23</td><td>352</td></tr> <tr><td>0x24</td><td>384</td></tr> <tr><td>0x25</td><td>448</td></tr> <tr><td>0x26</td><td>512</td></tr> </tbody> </table>	0x00	384	0x01	416	0x02	480	0x03	576	0x04	640	0x05	704	0x06	832	0x07	1024	0x08	1152	0x09	1280	0x0A	1536	0x0B	1920	0x0C	2304	0x0D	2560	0x0E	3072	0x0F	3840	0x10	4608	0x11	5120	0x12	6144	0x13	7680	0x14	9216	0x15	10240	0x16	12288	0x17	15360	0x18	18432	0x19	20480	0x1A	24576	0x1B	30720	0x1C	36864	0x1D	40960	0x1E	49152	0x1F	61440	0x20	256	0x21	288	0x22	320	0x23	352	0x24	384	0x25	448	0x26	512
0x00	384																																																																														
0x01	416																																																																														
0x02	480																																																																														
0x03	576																																																																														
0x04	640																																																																														
0x05	704																																																																														
0x06	832																																																																														
0x07	1024																																																																														
0x08	1152																																																																														
0x09	1280																																																																														
0x0A	1536																																																																														
0x0B	1920																																																																														
0x0C	2304																																																																														
0x0D	2560																																																																														
0x0E	3072																																																																														
0x0F	3840																																																																														
0x10	4608																																																																														
0x11	5120																																																																														
0x12	6144																																																																														
0x13	7680																																																																														
0x14	9216																																																																														
0x15	10240																																																																														
0x16	12288																																																																														
0x17	15360																																																																														
0x18	18432																																																																														
0x19	20480																																																																														
0x1A	24576																																																																														
0x1B	30720																																																																														
0x1C	36864																																																																														
0x1D	40960																																																																														
0x1E	49152																																																																														
0x1F	61440																																																																														
0x20	256																																																																														
0x21	288																																																																														
0x22	320																																																																														
0x23	352																																																																														
0x24	384																																																																														
0x25	448																																																																														
0x26	512																																																																														

Table continues on the next page...

I2Cx_I2CFDR field descriptions (continued)

Field	Description	
0x27	576	
0x28	640	
0x29	768	
0x2A	896	
0x2B	1024	
0x2C	1280	
0x2D	1536	
0x2E	1792	
0x2F	2048	
0x30	2560	
0x31	3072	
0x32	3584	
0x33	4096	
0x34	5120	
0x35	6144	
0x36	7168	
0x37	8192	
0x38	10240	
0x39	12288	
0x3A	14336	
0x3B	16384	
0x3C	20480	
0x3D	24576	
0x3E	28672	
0x3F	32768	

12.4.3 I2C control register (I2Cx_I2CCR)

The I²C control register (I2CCR) contains fields for controlling several actions, modes, messages, conditions, and capabilities.

NOTE

Writing to the Reserved Field: While writing to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

Address: Base address + 8h offset

Bit	0	1	2	3	4	5	6	7
Read	MEN	MIEN	MSTA	MTX	TXAK	RSTA	Reserved	BCST
Write	0	0	0	0	0	0	0	0
Reset								

I2Cx_I2CCR field descriptions

Field	Description
0 MEN	<p>Module enable. Controls the software reset of the I²C module.</p> <p>0 The module is reset and disabled. The module is held in reset, but the registers can still be accessed. 1 The I²C module is enabled. MEN must be set before any other control register fields have any effect. All I²C registers for slave receive or master START can be initialized before setting this field, however.</p>
1 MIEN	<p>Module interrupt enable. Enables the interrupt to be reported to the interrupt controller and/or (ultimately) the CPU.</p> <p>0 Interrupt reporting from the I²C module is disabled. If any pending interrupt conditions exist, they are not cleared. 1 Interrupt reporting from the I²C module is enabled. If an interrupt condition is detected and I2CSR[MIF] is also set, the interrupt is reported.(For more information about I2CSR fields, see I2C status register (I2C_I2CSR).)</p>
2 MSTA	<p>Master/Slave Mode START.</p> <p>0 If MSTA is changed from 1 to 0, a STOP condition is generated and the I²C mode changes from master to slave. MSTA is cleared without generating a STOP condition when the master loses arbitration. 1 If MSTA is changed from 0 to 1, a START condition is generated on the bus and master mode is selected for the I²C.</p>
3 MTX	<p>Transmit/Receive Mode Select. Specifies the direction of master and slave transfers. If the I²C module is configured as a slave, the software should set MTX to match I2CSR[SRW]. If the I²C module is in master mode, MTX should be set according to the type of transfer required. For address cycles, therefore, this field is always high (has a value of 1). MTX is cleared if the master loses arbitration.</p> <p>0 Receive mode is selected. 1 Transmit mode is selected.</p>
4 TXAK	<p>Transfer Acknowledge. Specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The TXAK value applies only if the I²C module is configured as a receiver, not as a transmitter. The TXAK setting does not apply to address cycles. When the device is addressed as a slave, an acknowledge is always sent.</p> <p>0 An acknowledge signal (low value on the SDA) is sent out to the bus at the ninth clock after receiving 1 byte of data. 1 No acknowledge signal (high value on the SDA) is sent.</p>
5 RSTA	<p>Repeated START. Specifies whether to generate a repeated START condition. Setting RSTA always generates a repeated START condition on the bus and provides the device with the current bus master. The RSTA field is not readable; an attempt to read RSTA returns a 0.</p> <p>0 No START condition is generated. 1 A repeated START condition is generated.</p>
6 -	This field is reserved. Reserved
7 BCST	<p>Broadcast.</p> <p>0 The broadcast accept capability is disabled. 1 The I²C is enabled to accept broadcast messages at address 0.</p>

12.4.4 I²C status register (I2Cx_I2CSR)

The I²C status register (I2CSR) is read-only with the exception of the MIF and MAL fields, which can be cleared by software.

Address: Base address + Ch offset

Bit	0	1	2	3	4	5	6	7
Read	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
Write								
Reset	1	0	0	0	0	0	0	1

I2Cx_I2CSR field descriptions

Field	Description
0 MCF	Data Transfer. When one byte of data is transferred, MCF is cleared. MCF is set by the falling edge of the ninth clock of a byte transfer. 0 A byte transfer is in progress. MCF is cleared when I2CDR is read in receive mode or written in transmit mode. (For more information about I2CDR, see I2C data register (I2C_I2CDR) .) 1 The byte transfer is completed.
1 MAAS	Addressed as a slave. MAAS is set if the module is acting as a slave and has detected that the I2CADR address matches with the transaction's calling address. The processor is interrupted if I2CCR[MIEN] is set. Next, the processor must check the SRW value and set I2CCR[MTX] accordingly. Any write to I2CCR automatically clears MAAS. For more information, see I2C address register (I2C_I2CADR) (I2CADR fields) and I2C control register (I2C_I2CCR) (I2CCR fields). 0 Not addressed as a slave. 1 Addressed as a slave.
2 MBB	Bus Busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 The I ² C bus is idle. 1 The I ² C bus is busy.
3 MAL	Arbitration Lost. MAL is automatically set (value of 1) if arbitration is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software. 1 Arbitration is lost.
4 BCSTM	Broadcast Match. The broadcast address is always all zeros. BCSTM can be set only if I2CCR[BCST] is set to enable it. (For more information about I2CCR[BCST], see I2C control register (I2C_I2CCR) .) 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. BCSTM is also set if the I ² C drives an address of all zeros and broadcast mode is enabled.
5 SRW	Slave Read/Write. If MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master. By checking SRW, the processor can select slave transmit/receive mode according to the master's command.

Table continues on the next page...

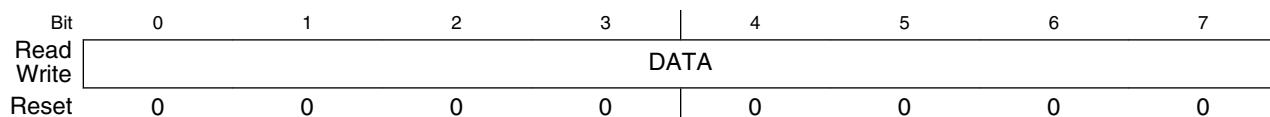
I2Cx_I2CSR field descriptions (continued)

Field	Description
	<p>0 Slave receive mode is selected, with the master writing to the slave.</p> <p>1 Slave transmit mode is selected, with the master reading from the slave. SRW is valid only if both of the following conditions are met:</p> <ul style="list-style-type: none"> • A complete transfer occurred and no other transfers have been initiated. • The I²C module is configured as a slave and has an address match.
6 MIF	<p>Module Interrupt. MIF is set if an interrupt is detected. An interrupt is reported if I2CCR[MIEN] is set. The interrupts for I²C1 and I²C2 are combined into one interrupt, which is sourced by the dual I²C module 1; similarly, the interrupts for I²C3 and I²C4 are combined into one interrupt, which is sourced by the dual I²C module 2.(For more information about I2CCR fields, see I2C control register (I2C_I2CCR)).</p> <p>0 No interrupt is detected. MIF can be cleared only by software.</p> <p>1 An interrupt is detected. MIF is set when one of the following events occurs:</p> <ul style="list-style-type: none"> • One byte of data is transferred (set at the falling edge of the ninth clock). • The I2CADR value matches with the calling address in Slave Receive mode. • Arbitration is lost.
7 RXAK	<p>Received Acknowledge. The value of SDA during the reception of a bus cycle's acknowledge bit. If RXAK = 0, it indicates that an acknowledge signal has been received after the 8 bits of data have been transmitted on the bus. If RXAK = 1, it means no acknowledge signal has been detected at the ninth clock.</p> <p>0 An acknowledge signal has been received.</p> <p>1 No acknowledge signal has been received.</p>

12.4.5 I2C data register (I2Cx_I2CDR)

I2CDR specifies the calling address and data to be transmitted (if the I²C is in master or slave transmit mode) or allows the I²C module to receive the next byte of data on the I²C module (if the I²C is in master or slave receive mode).

Address: Base address + 10h offset

**I2Cx_I2CDR field descriptions**

Field	Description
0-7 DATA	<p>Specifies data to be transmitted or allows receipt of the next data byte on the I²C module, depending on the I²C mode:</p> <ul style="list-style-type: none"> • Transmit mode: Data transmission is initiated when data is written to I2CDR. For master transmit mode, the first byte of data written to I2CDR is used for the address transfer and is follows the format described in Transactions. When bytes are written to I2CDR in transmit mode, they cannot be verified by reading them back. • Receive mode: Reading I2CDR allows the I²C module to receive the next byte of data on the I²C interface (in addition to reading the contents of I2CDR). <p>In all cases, the most significant bit is sent first.</p>

I2Cx_I2CDR field descriptions (continued)

Field	Description
	I2CCR[MTX] must be set appropriately for the desired behavior. For example, if I2CCR[MTX] is set for transmit mode (1) instead of receive mode (0), reading I2CDR does not initiate receipt of the next data byte. (For more information about I2CCR[MTX], see I2C control register (I2C_I2CCR) .) For both master receive and slave receive modes, the very first read is always a dummy read.

12.4.6 I²C digital filter sampling rate register (I2Cx_I2CDFSRR)

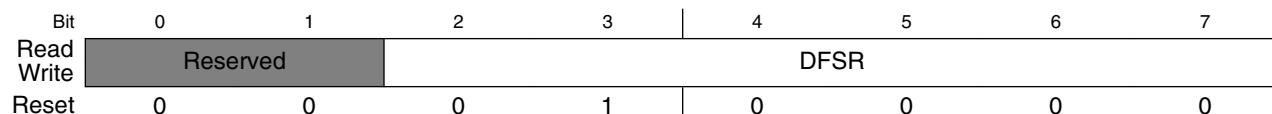
The digital filter sampling rate register (I2CDFSRR) specifies the sample rate for filtering out signal noise.

For additional guidance about the proper use of I2CFDR and I2CDFSRR on Power Architecture™ integrated host/communications processors, refer to the application note AN2919, *Determining the I²C Frequency Divider Ratio for SCL*.

NOTE

Writing to the Reserved Field: While writing to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

Address: Base address + 14h offset



I2Cx_I2CDFSRR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 DFSR	Digital Filter Sampling Rate. Specifies the sample rate for the program to use in filtering out signal noise. This rate is used to prescale the frequency the digital filter uses to take samples from the I ² C bus. The resulting sampling rate is calculated by dividing the platform frequency by the non-zero value of the DFSR. If I2CDFSRR = 0, the I ² C bus sample points to the reset divisor.

12.5 I²C Functional Description

This section presents the following topics:

- [Notes About Module Operation](#)
- [Transactions](#)
- [Protocol Implementation Details](#)

- Bus Arbitration
- Clock Behavior
- Filtering of SCL and SDA Lines

12.5.1 Notes About Module Operation

- The I²C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I²C module is acting as a master, it must not try to call its own slave address.

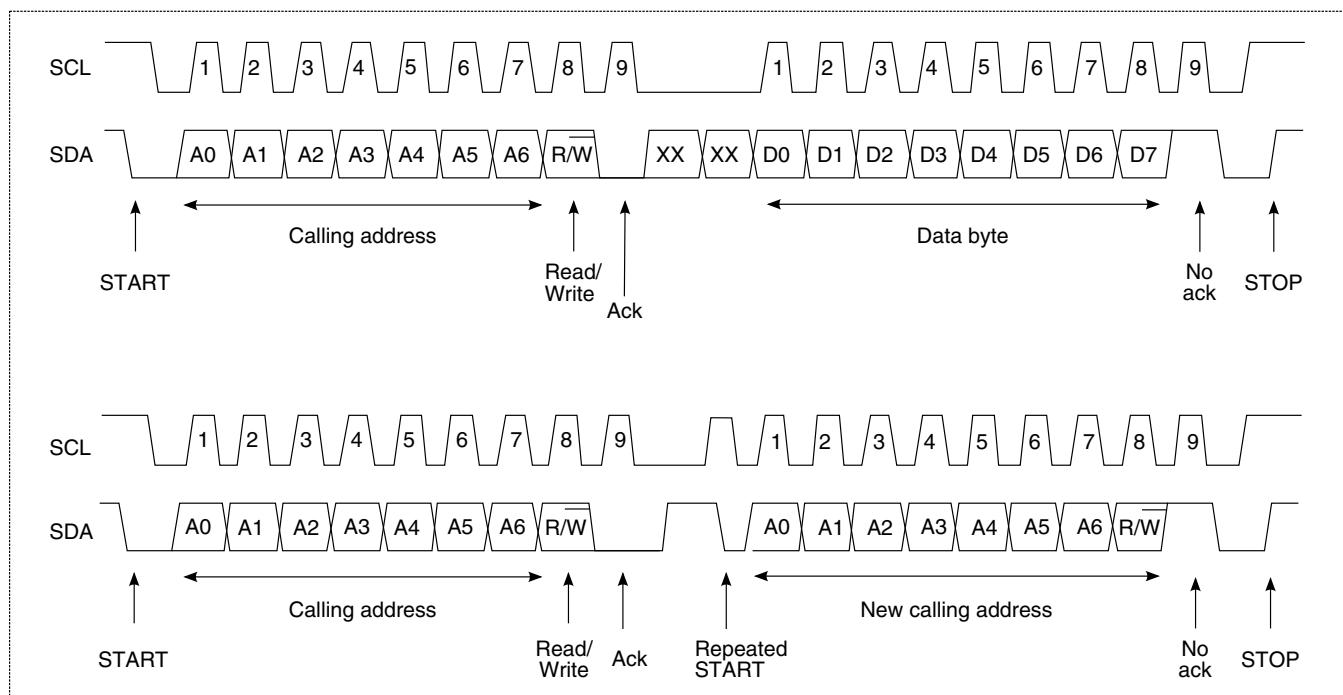
12.5.2 Transactions

This section covers the following topics:

- Protocol Overview
- Definitions
- I²C Calling Address Requirements
- High-Level Protocol Steps
- START Condition
- Slave Address Transmission
- General Call (Broadcast) Addressing
- Data Transmission
- STOP Condition
- Repeated START Condition

12.5.2.1 Protocol Overview

The following figure shows the behavior of SCL and SDA during a typical I²C transaction.

Figure 12-32. I²C Transaction Protocol

12.5.2.2 Definitions

This section defines several important terms presented in [Figure 12-32](#).

Table 12-34. I²C Definitions

Term	Definition
START	A START condition, as defined in Definition: I²C Conditions
STOP	A STOP condition, as defined in Definition: I²C Conditions
Calling (slave) address	A seven-bit address used to identify a slave on the I ² C bus. The requirements for specifying this address are presented in I²C Calling Address Requirements .
Read/write (R/W_B)	A bit that specifies the direction of the data transfer to the slave as follows: <ul style="list-style-type: none"> • 0 = The data is being transferred from the master to the slave ("write") • 1 = The data is being transferred from the slave to the master ("read")
Acknowledge	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

12.5.2.3 I²C Calling Address Requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

12.5.2.4 High-Level Protocol Steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in [Figure 12-32](#). The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 12-35. I²C High-Level Protocol Steps

Standard Transfer	Repeated START Transfer
1. START condition 2. Slave target or general call address transmission 3. Data transfer 4. STOP condition 5. (repeat Steps 1-4)	1. START condition 2. Slave target or general call address transmission 3. Data transfer 4. Repeated START condition 5. (repeat Steps 2-4 as needed) 6. STOP condition. 7. (repeat Steps 1-7)

12.5.2.5 START Condition

A master on the I²C bus initiates a data transfer by sending a START condition on the I²C bus when the bus is not engaged (both SDA and SCL are high).

On this device, the START condition is sent by setting I2CCR[MSTA]. (For more information about I2CCR[MSTA], see [I²C control register \(I²C_I2CCR\)](#).)

12.5.2.6 Slave Address Transmission

The master transmits the slave address immediately after the START condition (see [START Condition](#)). The process of slave address transmission is presented in [Table 12-36](#).

Table 12-36. Slave Address Transmission Process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W_B bit.

Table continues on the next page...

Table 12-36. Slave Address Transmission Process (continued)

Step	Action
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: <ul style="list-style-type: none"> • The acknowledge bit is set: The master must generate a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

12.5.2.7 General Call (Broadcast) Addressing

The master may also initiate a general call (broadcast) command by transmitting a slave address of 0x00. In this case, the second byte of the broadcast message is the master address. This device will not check the R/W_B bit in a broadcast address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

This device responds to a general call (broadcast) command if I2CCR[BCST] is set.

12.5.2.8 Data Transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data.
- Awakens all slaves.
- Proceeds on a byte-by-byte basis in the direction specified by the R/W_B bit sent by the calling master.

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.

- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

12.5.2.9 STOP Condition

A master on the I²C bus can terminate a data transfer by sending a STOP condition. It can do so even if the slave has sent an acknowledge bit. In this case, the slave must release the I²C bus.

On this device, the STOP condition is sent by clearing I2CCR[MSTA]. (For more information about I2CCR[MSTA], see [I2C control register \(I2C_I2CCR\)](#).)

A master is not required to send a STOP condition at the end of every transfer. For more information, see [Repeated START Condition](#).

12.5.2.10 Repeated START Condition

The I²C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of [Figure 12-32](#).

12.5.3 Protocol Implementation Details

This section provides details of how aspects of the I²C protocol are implemented in this device. The following sections are included:

- [Transaction Monitoring](#)
- [Control Transfer](#)

12.5.3.1 Transaction Monitoring

The different conditions of the I²C data transactions (see [Transactions](#)) are monitored as follows:

- START conditions are detected when SDA falls while SCL is high.
- STOP conditions are detected when SDA rises while SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

12.5.3.2 Control Transfer

The I²C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I²C. The SCL output is driven low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or repeated START condition. Otherwise, the SDA output is held constant.

[Table 12-37](#) presents the behavior of the SCL and SDA signals under the various protocol conditions.

Table 12-37. SDA and SCL Signal Behavior

Mode	Conditions When SDA Is Driven Low	Conditions When SCL Corresponds to the Internal SCL Signal
Master	Data bit (transmission) Acknowledge bit (receive) START condition STOP condition Repeated START condition	Bus owner Lost arbitration START condition STOP condition Repeated START condition begin Repeated START condition end
Slave	Acknowledging address match Data bit (transmit) Acknowledge bit (receive)	Address cycle Transmit cycle Acknowledge cycle

12.5.4 Bus Arbitration

This section covers the following topics related to bus arbitration:

- Bus Arbitration Overview
- Loss of Arbitration
- Module Startup During a Data Transfer

12.5.4.1 Bus Arbitration Overview

If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I²C module) determines the bus clock. The low part of the period is equal to the longest clock low period and the high part of the period is equal to the shortest one among the masters.

12.5.4.2 Loss of Arbitration

A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. In this case, the losing master performs the following tasks:

1. Switches to slave-receive mode.
2. Stops driving the SDA line without generating a STOP condition.
3. Sets I2CSR[MAL] to indicate the loss of arbitration.
4. Services the transaction if it is directed to itself.

12.5.4.3 Module Startup During a Data Transfer

[Table 12-38](#) presents the behavior of the I²C module if it is enabled in the middle of an ongoing byte transfer.

Table 12-38. Module Behavior at Startup During a Data Transfer

Mode	Behavior
Slave	The I ² C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
Master	The I ² C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately results in the current bus master of the I ² C bus losing arbitration, after which bus operations return to normal.

12.5.5 Clock Behavior

This section covers the following topics of SCL synchronization:

- SCL Synchronization

- Clock Stretching
- Handshaking

12.5.5.1 SCL Synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

12.5.5.2 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

12.5.5.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL low after completion of a 1-byte transfer (9 bits). In such cases, the mechanism halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

12.5.6 Filtering of SCL and SDA Lines

This section covers the following topics:

- Filtering of SCL and SDA Lines-Overview
- Sample Rate Control

12.5.6.1 Filtering of SCL and SDA Lines-Overview

The SCL and SDA inputs are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

12.5.6.2 Sample Rate Control

The sampling rate is controlled by the value stored in I2CDFSRR (as described in [I²C digital filter sampling rate register \(I2C_I2CDFSRR\)](#)). The duration of the sampling cycle is controlled by a down counter. This allows a software write to I2CDFSRR to control the filtered sampling rate.

12.6 I²C Initialization/Application Information

This section discusses the following topics related to I²C initialization and application:

- [Recommended Interrupt Service Flow](#)
- [General Programming Guidelines \(for Both Master and Slave Mode\)](#)
- [Programming Guidelines Specific to Master Mode](#)
- [Programming Guidelines Specific to Slave Mode](#)

12.6.1 Recommended Interrupt Service Flow

Figure 12-33 shows a flowchart for the recommended I²C interrupt service routine. Deviation from the flowchart may result in unpredictable I²C bus behavior. Although the flowchart does not explicitly show the synchronization after every I²C register access, it is recommended that a synchronizing instruction follow each I²C register read or write to guarantee in-order instruction execution.

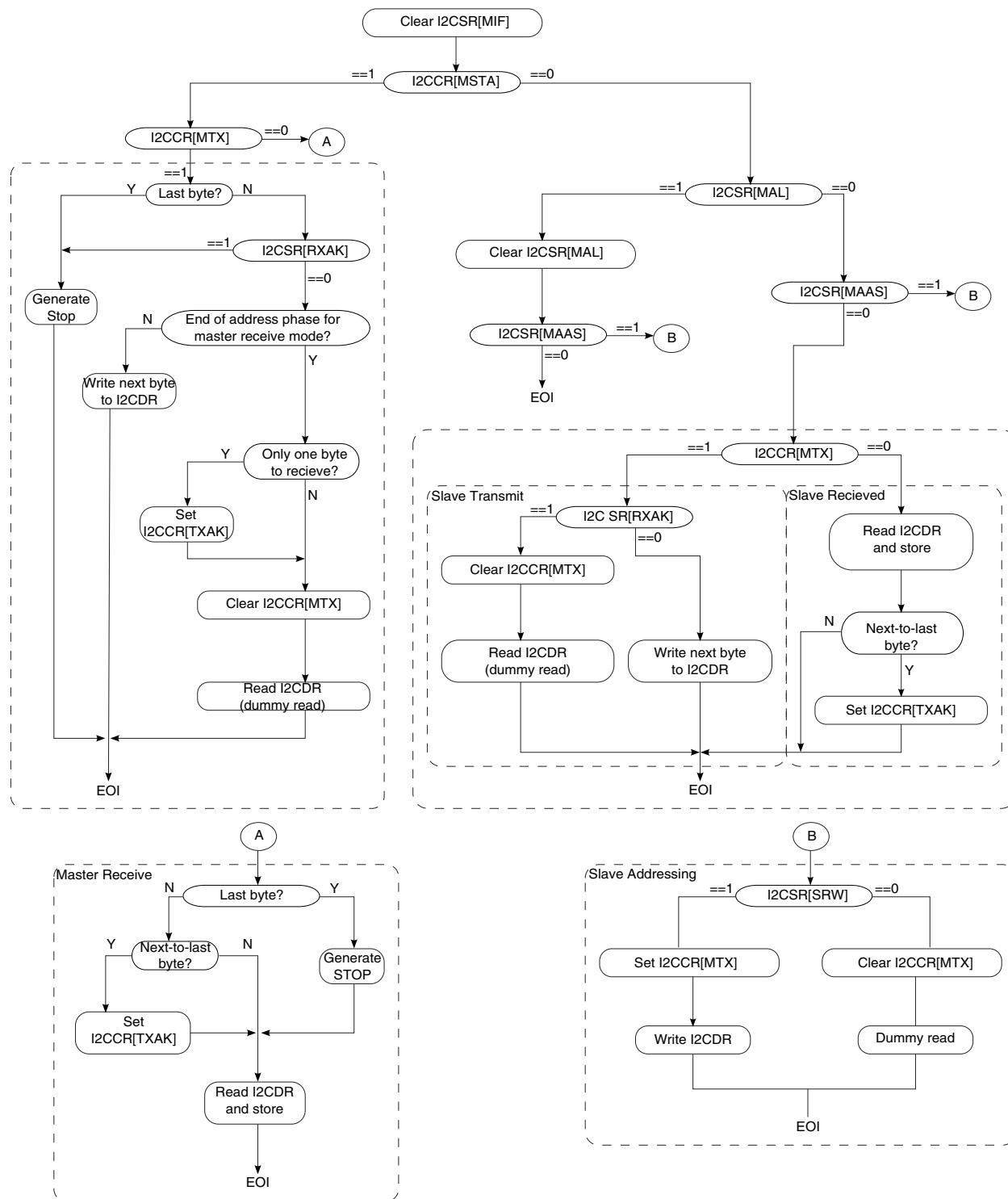


Figure 12-33. Recommended I²C Interrupt Service Routine Flowchart

12.6.2 General Programming Guidelines (for Both Master and Slave Mode)

This section provides programming guidelines recommended for the I²C module in both master and slave mode. It describes the following procedures and processes:

- [Initializing the Module](#)
- [Software Response After a Transfer](#)
- [Generating SCL when SDA is Low](#)

NOTE

Illegal or irregular bus activity: The I²C module does not guarantee its recovery from all illegal I²C bus activity. Additionally, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I²C bus hangs. The recovery routine should also handle the case when the status bits returned after an interrupt are not consistent with what was expected due to illegal I²C bus protocol behavior.

12.6.2.1 Initializing the Module

The following sequence initializes the I²C unit:

1. Program I2CFDR[FDR] with the appropriate ratio for the desired SCL frequency. (For more information about the I2CFDR[FDR], see [I2C frequency divider register \(I2C_I2CFDR\)](#).)
2. Update I2CADR to define the slave address for this device. (For more information about I2CADR fields, see [I2C address register \(I2C_I2CADR\)](#).)
3. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable; set I2CCR[MEN] to enable the I²C module. (For more information about I2CCR fields, see [I2C control register \(I2C_I2CCR\)](#).)

12.6.2.2 Software Response After a Transfer

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. If the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set), the I²C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the PIC. In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF].
2. Read the contents of the I²C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this access to I2CDR causes I2CSR[MCF] to be cleared automatically. See [Recommended Interrupt Service Flow](#).

Note the programming guidelines for the following conditions:

- An interrupt at the end of the address cycle-When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] must be toggled at this stage. See [Figure 12-33](#).
- Monitoring I2CSR[MIF] when the interrupt function is disabled-If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I²C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed in order to give the I²C signals sufficient time to settle.
- Slave-mode addressing-During slave-mode addressing, when I2CSR[MAAS] is set, I2CSR[SRW] should be read to determine the direction of the subsequent transfer, and I2CCR[MTX] should be programmed accordingly.
- Slave-mode data transfers-For slave-mode data transfers (MAAS is cleared), I2CSR[SRW] is not valid, and I2CCR[MTX] must be read to determine the direction of the current transfer. See [Figure 12-33](#) for more details.

12.6.2.3 Generating SCL when SDA is Low

When a system reset does not cause all I²C devices to be reset, it is sometimes necessary to force the I²C module to become the I²C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates the bus is busy). Thus, SDA can be driven low by another I²C device while this I²C module is coming out of reset and will stay low indefinitely.

The following procedure can be used to force this I²C module to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I²C module (I2CCR[MEN] = 0) and change to master mode (I2CCR[MSTA] = 1) by programming the value 0x20 into I2CCR.
2. Re-enable the I²C module (I2CCR[MEN] = 1) by programming the value 0xA0 into I2CCR.
3. Read I2CDR.
4. Return the I²C module to slave mode (I2CCR[MSTA] = 0) by programming the value 0x80 into I2CCR.

12.6.3 Programming Guidelines Specific to Master Mode

This section includes the following programming guidelines specific to master mode:

- Generating START
- Generating STOP
- Generating Repeated START
- Loss of Arbitration and Forcing Slave Mode

12.6.3.1 Generating START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I²C system, test the state of I2CSR[MBB] to check whether the serial bus is available (I2CSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
3. Write the slave address being called into I2CDR. The data written to I2CDR[0-6] comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

NOTE

The scenario above assumes that the I²C interrupt bit (I2CSR[MIF]) is cleared. If MIF is set at any time, an I²C interrupt is generated (if interrupt reporting is enabled with I2CCR[MIEN] =1).

NOTE

The interrupts for I²C1 and I²C2 are combined into a single interrupt to the PIC; similarly, interrupts for I²C3 and I²C4 are combined into a single interrupt to the PIC.

12.6.3.2 Generating STOP

A data transfer ends with a STOP condition generated by the master device.

A master transmitter generates a STOP condition after all the data has been transmitted. When the I²C module is acting as the master transmitter, and all the data has been transmitted (that is, after last byte to be transmitted has been written to I2CDR), software clears I2CCR[MSTA] to generate a STOP condition.

When the I²C module is acting as the master receiver, the master indicates the termination of the transfer by not acknowledging the final byte and by generating a STOP condition. For a multiple byte transfer before reading the next-to-last byte in I2CDR, software sets I2CCR[TXAK], then software reads the next-to-last byte in I2CDR, which causes the master receiver to not acknowledge the next transfer and to automatically generate a STOP at the conclusion of the next transfer. For single-byte transfers, at the conclusion of the address phase, software should set I2CCR[TXAK] and clear I2CCR[MTX], and then perform a dummy read to I2CDR. Prior to subsequent I²C transactions, software should clear I2CCR[TXAK]. This can be performed when setting up the I2CCR for the next transfer.

12.6.3.3 Generating Repeated START

At the end of a data transfer, if the master wants to continue communicating on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition. This is accomplished by setting I2CCR[RSTA].

12.6.3.4 Loss of Arbitration and Forcing Slave Mode

When a master loses arbitration, the following conditions all occur:

- I2CSR[MAL] is set.
- I2CCR[MSTA] is cleared (changing the master to slave mode).
- An interrupt occurs (if enabled) at the falling edge of the ninth clock of this transfer.

Thus, the slave interrupt service routine should first test I2CSR[MAL] and software should clear it if it is set.

12.6.4 Programming Guidelines Specific to Slave Mode

This section includes the following programming guidelines specific to slave mode:

- [Slave Mode Interrupt Service Routine](#)
- [Acknowledge Receipt During Transmission](#)

12.6.4.1 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the slave should be tested as follows to determine whether a calling of its own address has been received:

If I2CSR[MAAS] is set, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/W command bit (I2CSR[SRW]). Writing to I2CCR clears MAAS automatically. MAAS is read as set only in the interrupt handler at the end of that address cycle in which an address match occurred; interrupts resulting from subsequent data transfers clear MAAS.

A data transfer can then be initiated by writing to I2CDR for slave transmits or dummy reading from I2CDR in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode. See [Figure 12-33](#).

12.6.4.2 Acknowledge Receipt During Transmission

In the slave transmitter routine, the received acknowledge bit (I2CSR[RXAK]) must be checked before the next byte of data is sent:

- If I2CSR[RXAK] is cleared, the master has acknowledged the previous data transfer and the next byte of data may be transmitted.
- If I2CSR[RXAK] is set, the master is signaling an end-of-data by not acknowledging the previous data transfer. Software running on the slave transmitter must do the following:
 - a. Clear I2CCR[MTX] to switch the slave from transmitter to receiver mode.
 - b. Perform a dummy read of I2CDR, which releases SCL so the master can generate a STOP condition.

In the slave receiver routine, the receiver transmit acknowledge bit (I2CCR[TXAK]) must be set after the next-to-last byte of data from I2CDR is read.

See [Recommended Interrupt Service Flow](#).

Chapter 13

Enhanced Local Bus Controller

13.1 eLBC introduction

This chapter describes the enhanced local bus controller (eLBC) block.

It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), NAND Flash control machine (FCM), and user-programmable machines (UPMs) of the eLBC. Finally, it includes an initialization and applications information section with many specific examples of its use.

The figure below is a functional block diagram of the eLBC, which supports three interfaces: GPCM, FCM, and UPM controllers.

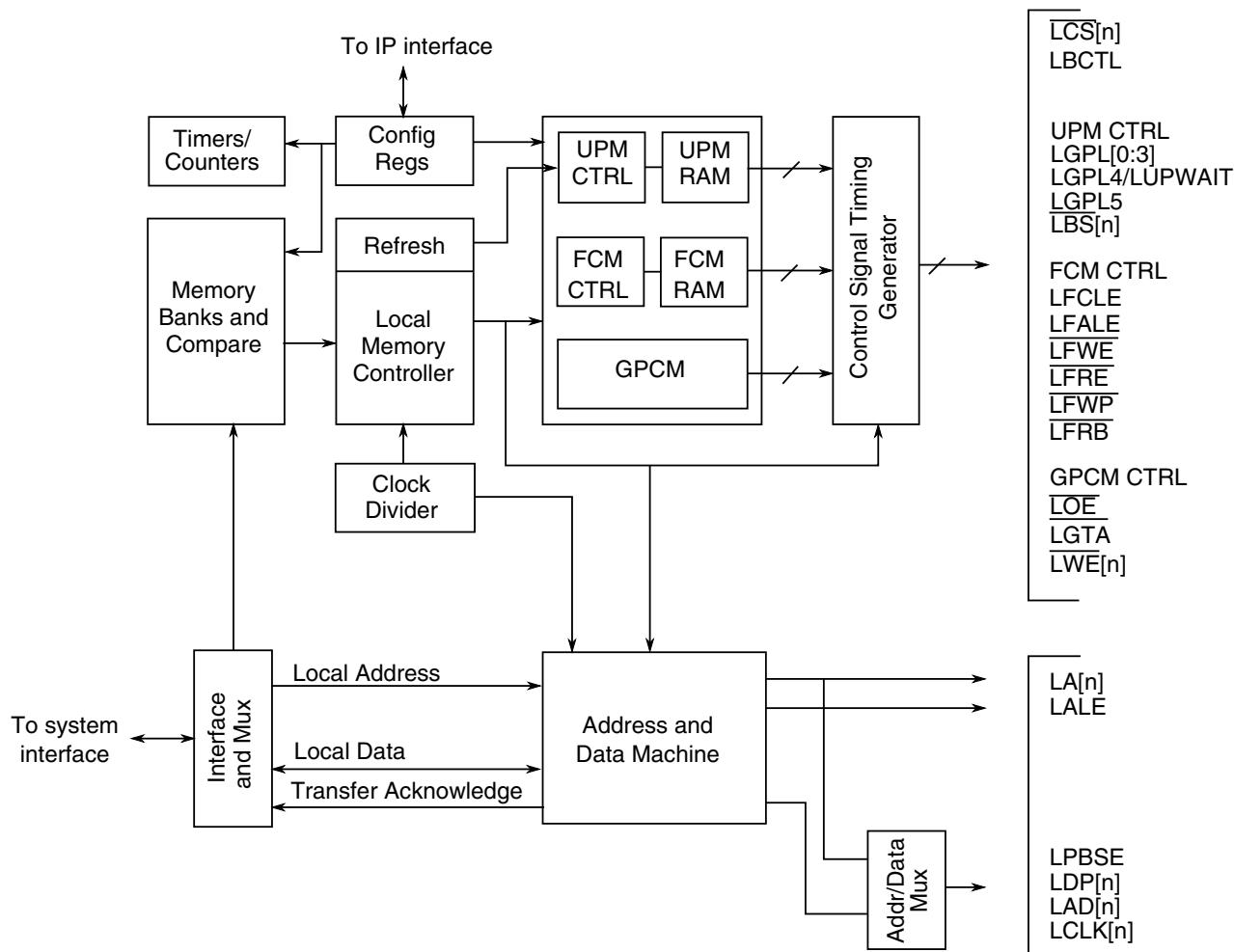


Figure 13-1. Enhanced local bus controller block diagram

13.1.1 Overview

The main component of the eLBC is its memory controller, which provides a seamless 16-bit interface to many types of memory devices and peripherals.

The memory controller is responsible for controlling eight memory banks shared by a GPCM, an FCM, and up to three UPMs. As such, it supports a minimal glue logic interface to SRAM, EPROM, NOR Flash EEPROM, NAND Flash EEPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LALE) allows multiplexing of addresses with data signals to reduce the device pin count.

The eLBC also includes a number of data checking and protection features such as data parity generation and checking, write protection, and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

13.1.2 Features

The eLBC main features are as follows:

- Memory controller with eight memory banks
 - 32-bit address decoding with mask
 - Variable memory block sizes (32 Kbytes to 4 Gbytes)
 - Selection of control signal generation on a per-bank basis
 - Data buffer controls activated on a per-bank basis
 - Automatic segmentation of large transactions into memory accesses optimized for bus width and addressing capability
 - Odd/even parity checking for single access
 - Write-protection capability
 - Parity byte-select
- General-purpose chip-select machine (GPCM)
 - Compatible with SRAM, EPROM, NOR Flash EEPROM, and peripherals
 - Global (boot) chip-select available at system reset
 - Boot chip-select support for 8- and 16-bit devices
 - Minimum three-clock access to external devices
 - Two byte-write-enable signals (LWE_B[0:1])
 - Output enable signal (LOE_B)
 - External access termination signal (LGTA_B)
- NAND Flash control machine (FCM)
 - Compatible with small (512 + 16 bytes) and large (2048 + 64 bytes) page parallel NAND Flash EEPROM
 - Global (boot) chip-select available at system reset, with 4-Kbyte boot block buffer for execute-in-place boot loading
 - ECC checking enable/disable feature supported during boot
 - Read-only ECC registers to verify after write operation
 - Boot chip-select support for 8-bit devices
 - Dual 2-Kbyte/eight 512-byte buffers allow simultaneous data transfer during Flash reads and programming
 - Event Interrupt-driven block transfer for reads and writes
 - Programmable command and data transfer sequences of up to eight steps supported
 - Generic command and address registers support proprietary Flash interfaces
 - Block write locking to ensure system security and integrity

- Three user-programmable machines (UPMs)
 - Programmable-array-based machine controls external signal timing with a granularity of up to one quarter of an external bus clock period
 - User-specified control-signal patterns run when an internal master requests a single-beat or burst read or write access.
 - UPM refresh timer runs a user-specified control signal pattern to support refresh
 - User-specified control-signal patterns can be initiated by software
 - Each UPM can be defined to support DRAM devices with depths of 64, 128, 256, and 512 Kbytes, and 1, 2, 4, 8, 16, 32, 64, 128, and 256 Mbytes
 - Support for 8- and 16-bit devices
 - Page mode support for successive transfers within a burst
 - Internal address multiplexing supporting 64-, 128-, 256-, and 512-Kbyte, and 1-, 2-, 4-, 8-, 16-, 32-, 64-, 128-, and 256-Mbyte page banks
- Optional monitoring of transfers between local bus internal masters and local bus slaves (local bus error reporting on error interrupt and status registers)
- Different machines (FCM/GPCM/UPM) share the address, data, and control signals. While the eLBC is servicing a transaction, subsequent transactions are queued until the current transaction has completed.

13.1.3 Modes of operation

The eLBC provides one GPCM, one FCM, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine.

The internal transaction address is limited to 32 bits, so all chip selects must fall within the 4-Gbyte window addressed by the internal transaction address. When a memory transaction is dispatched to the eLBC, the internal transaction address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, FCM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus, with the eLBC in GPCM or FCM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction except in the case of UPM refresh where all UPM machines that are enabled for refresh have concurrent chip select assertion.

13.1.3.1 eLBC bus clock and clock ratios

The eLBC supports ratios of 8, 16, and 32 between the faster internal (platform) clock and slower external bus clock (LCLK[0:1]).

This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]).

This ratio affects the resolution of signal timing shifts in GPCM and FCM modes and the interpretation of UPM array words in UPM mode. The bus clock is driven identically onto pins, LCLK[0:1], to allow the clock load to be shared equally across a set of signal nets, thereby enhancing the edge rates of the bus clock.

NOTE

The eLBC is clocked by platform clock/2. The ratios given in this chapter include this divider.

During power-on reset when the chip is fetching the RCW, the system logic is clocked directly by SYSCLK. After the RCW has been fetched and the internal PLLs have locked, the eLBC is clocked by the platform clock/2.

Table 13-1. LCLK at power-on reset

LCLK	cfg_rcw_src	cfg_rcw_src
LCLK	FCM (NAND Flash)	GPCM (NOR Flash)
During RCW fetch	SYSCLK/8	SYSCLK/16
After PLLs have locked, but before pre-boot initialization	Platform clock/16	Platform clock/32

13.2 eLBC external signal descriptions

The table contains a list of external signals related to the eLBC and summarizes their function.

Table 13-2. Signal properties-summary

Name	Alternate Function(s)	Mode	Descriptions	No. of Signals	I/O
LALE	-	-	External address latch enable	1	O
LCS_B[0:7]	-	-	Chip select 0	8	O
LWE0_B/ LFWE_B	LWE0_B	GPCM	Write enable 0	1	O
	LFWE_B	FCM	Flash Write enable	1	
LBS0	LBS0_B	UPM	Byte (lane) select 0	1	
LWE 1_B/ LBS1_B	LWE_B	GPCM	Write enable 1	1	O
	LBS_B	UPM	Byte (lane) select 1	1	
LGPO/ LFCLE	LGPO	UPM	General purpose line 0	1	O
	LFCLE	FCM	Flash command latch enable	1	
LGPO1/ LFALE	LGPO1	UPM	General purpose line 1	1	O
	LFALE	FCM	Flash address latch enable	1	

Table continues on the next page...

Table 13-2. Signal properties-summary (continued)

Name	Alternate Function(s)	Mode	Descriptions	No. of Signals	I/O
LOE_B/ LGPL2/ LFRE_B	LOE_B	GPCM	Output enable	1	O
	LFRE_B	FCM	Flash read enable	1	
	LGPL2	UPM	General purpose line 2	1	
LGPL3/ LFWP_B	LGPL3	UPM	General purpose line 3	1	O
	LFWP_B	FCM	Flash write protect	1	
LGTA_B/	LGTA_B	GPCM	Transaction termination	1	I
LFRB_B/	LFRB_B	FCM	Flash ready/busy_B , open-drain shared pin_B	1	I
LGPL4/	LGPL4	UPM	General purpose line 4	1	O
LUPWAIT/ LPBSE	LUPWAIT	UPM	External device wait	1	I
	LPBSE	-	Local bus parity byte select	1	O
LGPL5	-	UPM	General purpose line 5	1	O
LBCTL	-	-	Data buffer control	1	O
LA[16:31]	-	-	Non-multiplexed address bus	16	O
LAD[0:15]	-	-	Multiplexed address/data bus	16	I/O
LDP[0: 1]	-	-	Local bus data parity	2	I/O
LCLK[0:1]	-	-	Local bus clocks	2	O

Table 13-3. Enhanced local bus controller detailed signal descriptions

Signal	I/O	Description
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device pins.
		State Meaning Asserted/Negated-LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. Note that no other control signals are asserted during the assertion of LALE.
LCS_B[0:7]	O	Chip selects. Eight chip selects are provided that are mutually exclusive.
		State Meaning Asserted/Negated-Used to enable specific memory devices or peripherals connected to the eLBC. LCS_B[0:7] are provided on a per-bank basis with LCS0_B corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.
LWE0_B/ LFWE_B/ LBS0_B, LWE1 _B/ LBS1 _B	O	GPCM write enable 0/FCM write enable/UPM byte select 0. These signals select or validate each byte lane of the data bus. For an 8-bit port size, bit 0 is the only defined signal. The least-significant address bits of each access also determine which byte lanes are considered valid for a given data transfer.
		State Meaning Asserted/Negated-For GPCM operation, LWE_B[0:1] assert for each byte lane enabled for writing. LFWE_B enables command, address, and data writes to NAND Flash EEPROMS controlled by FCM. LBS_B[0:1] are programmable byte-select signals in UPM mode. See General-purpose chip-select machine (GPCM) , for programming details about LBS_B[0:1].
		Timing Assertion/Negation-See RAM array , for details regarding the timing of LWE_B[0:1].
LGPL0/ LFCLE	O	General purpose line 0/FCM command latch enable.
		State Meaning Asserted/Negated-In UPM mode, LGPL0 is one of six general purpose signals; it is driven with a value programmed into the UPM array. In FCM mode, LFCLE enables command cycles to NAND Flash EEPROMS.

Table continues on the next page...

Table 13-3. Enhanced local bus controller detailed signal descriptions (continued)

Signal	I/O	Description	
LGPL1/ LFALE	O	General-purpose line 1/FCM address latch enable. State Meaning Asserted/Negated-In UPM mode, LGPL1 is one of six general purpose signals; it is driven with a value programmed into the UPM array. In FCM mode, LFALE enables address cycles to NAND Flash EEPROMs.	
LOE_B/ LGPL2/ LFRE_B	O	GPCM output enable/General-purpose line 2/FCM read enable. State Meaning Asserted/Negated-Controls the output buffer of memory when accessing memory/devices in GPCM mode. In UPM mode, LGPL2 is one of six general purpose signals; it is driven with a value programmed into the UPM array. LFRE_B enables data read cycles from NAND Flash EEPROMs controlled by FCM.	
LGPL3/ LFWP_B	O	General-purpose line 3/FCM write protect. State Meaning Asserted/Negated-In UPM mode, LGPL3 is one of six general purpose signals; it is driven with a value programmed into the UPM array. In FCM mode LFWP_B protects NAND Flash EEPROMs from accidental erasure and programming when LFWP_B is asserted low-see Flash mode register (eLBC_FMR) , for programming of FCM operations to control LFWP_B .	
LGTA_B/ LGPL4/ LFRB_B LUPWAIT/ LPBSE	I/O	GPCM transfer acknowledge/General-purpose line 4/FCM Flash ready-busy/UPM wait/parity byte select. State Meaning Asserted/Negated-Input in GPCM or FCM modes used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. FCM uses LFRB_B to stall during long-latency read and programming operations, continuing once LFRB_B returns high. When configured as LPBSE, it disables any use in GPCM, FCM, or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing LBSn_B through external logic to achieve the logical function of this byte-select can affect memory access timing. The LBC provides this optional byte-select signal connection to RMW-parity devices.	
LGPL5	O	General-purpose line 5 State Meaning Asserted/Negated-One of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.	
LBCTL	O	Data buffer control. The memory controller activates LBCTL for the local bus when a GPCM-, UPM-, or FCM-controlled bank is accessed. Buffer control is disabled by setting ORn[BCTL0]. State Meaning Asserted/Negated-The LBCTL pin normally functions as a write/read_B control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the eLBC when LBCTL is high, because LBCTL remains high after reset and during address phases.	
LA[16:31]	O	Nonmultiplexed address bus. All bits driven are defined for 8-bit port sizes. For 16-bit port sizes LA[31] is a don't care. State Meaning Asserted/Negated-LA is the address bus used to transmit addresses to external RAM devices. Refer to eLBC initialization/application information , for address signal multiplexing.	
LAD[0:15]	I/O	Multiplexed address/data bus. For a port size of 16 bits, LAD[0:7] connect to the most-significant byte lane (at address offset 0), while LAD[8:15] connect to the least-significant byte lane (at address offset 1). For a port size of 8 bits, only LAD[0:7] are connected to the external RAM. State Meaning Asserted/Negated-LAD is the shared 16-bit address/data bus through which external RAM devices transfer data and receive addresses. Timing Assertion/Negation-During assertion of LALE, LAD are driven with the RAM address for the access to follow. External logic should propagate the address on LAD while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD are either driven by write data or are made high-impedance by the eLBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD are again taken into a high-impedance state.	

Table continues on the next page...

Table 13-3. Enhanced local bus controller detailed signal descriptions (continued)

Signal	I/O	Description	
LDP[0: 1]	I/O	Local bus data parity. Drives and receives the data parity corresponding with the data phases on LAD for GPCM and UPM controlled banks.	
		State Meaning	Asserted/Negated-During write accesses, a parity bit is generated for each 8 bits of LAD[0:15], such that LDP0 is even/odd parity for LAD[0:7], while LDP[1] is even/odd parity for LAD[8:15]. Unused byte lanes for port sizes less than 16 bits have undefined parity.
		Timing	Assertion/Negation-Drive and receive the data parity corresponding with the data phases on LAD. For read accesses, the parity bits for each byte lane are sampled on LDP[0: 1] with the same timing that read data is sampled on LAD.
LCLK[0:1]	O	Local bus clocks	
		State Meaning	Asserted/Negated-LCLK[0:1] drive an identical bus clock signal for distributed loads.

13.3 Enhanced Local Bus Controller (eLBC) Memory Map

The table below shows the memory mapped registers of the eLBC. Undefined 4-byte address spaces within offset 0x0000-0xFFFF are reserved and should not be accessed for reading or writing. Similarly, only zero should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

Bits designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

eLBC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_4000	Base register 0 (eLBC_BR0)	32	R/W	See section	13.3.1/659
12_4004	Options register 0 layout for GPCM Mode (eLBC_ORg0)	32	R/W	0000_0FF7h	13.3.2/661
12_4004	Options register 0 layout for FCM Mode (eLBC_ORf0)	32	R/W	See section	13.3.3/664
12_4004	Options register 0 layout for UPM Mode (eLBC_ORu0)	32	R/W	See section	13.3.4/668
12_4008	Base register n (eLBC_BR1)	32	R/W	0000_0000h	13.3.5/672
12_400C	Options register n layout for GPCM Mode (eLBC_ORg1)	32	R/W	0000_0000h	13.3.6/673
12_400C	Options register n layout for FCM Mode (eLBC_ORf1)	32	R/W	0000_0000h	13.3.7/677
12_400C	Options register n layout for UPM Mode (eLBC_ORu1)	32	R/W	0000_0000h	13.3.8/681
12_4010	Base register n (eLBC_BR2)	32	R/W	0000_0000h	13.3.5/672
12_4014	Options register n layout for GPCM Mode (eLBC_ORg2)	32	R/W	0000_0000h	13.3.6/673
12_4014	Options register n layout for FCM Mode (eLBC_ORf2)	32	R/W	0000_0000h	13.3.7/677
12_4014	Options register n layout for UPM Mode (eLBC_ORu2)	32	R/W	0000_0000h	13.3.8/681

Table continues on the next page...

eLBC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_4018	Base register n (eLBC_BR3)	32	R/W	0000_0000h	13.3.5/672
12_401C	Options register n layout for GPCM Mode (eLBC_ORg3)	32	R/W	0000_0000h	13.3.6/673
12_401C	Options register n layout for FCM Mode (eLBC_ORf3)	32	R/W	0000_0000h	13.3.7/677
12_401C	Options register n layout for UPM Mode (eLBC_ORu3)	32	R/W	0000_0000h	13.3.8/681
12_4020	Base register n (eLBC_BR4)	32	R/W	0000_0000h	13.3.5/672
12_4024	Options register n layout for GPCM Mode (eLBC_ORg4)	32	R/W	0000_0000h	13.3.6/673
12_4024	Options register n layout for FCM Mode (eLBC_ORf4)	32	R/W	0000_0000h	13.3.7/677
12_4024	Options register n layout for UPM Mode (eLBC_ORu4)	32	R/W	0000_0000h	13.3.8/681
12_4028	Base register n (eLBC_BR5)	32	R/W	0000_0000h	13.3.5/672
12_402C	Options register n layout for GPCM Mode (eLBC_ORg5)	32	R/W	0000_0000h	13.3.6/673
12_402C	Options register n layout for FCM Mode (eLBC_ORf5)	32	R/W	0000_0000h	13.3.7/677
12_402C	Options register n layout for UPM Mode (eLBC_ORu5)	32	R/W	0000_0000h	13.3.8/681
12_4030	Base register n (eLBC_BR6)	32	R/W	0000_0000h	13.3.5/672
12_4034	Options register n layout for GPCM Mode (eLBC_ORg6)	32	R/W	0000_0000h	13.3.6/673
12_4034	Options register n layout for FCM Mode (eLBC_ORf6)	32	R/W	0000_0000h	13.3.7/677
12_4034	Options register n layout for UPM Mode (eLBC_ORu6)	32	R/W	0000_0000h	13.3.8/681
12_4038	Base register n (eLBC_BR7)	32	R/W	0000_0000h	13.3.5/672
12_403C	Options register n layout for GPCM Mode (eLBC_ORg7)	32	R/W	0000_0000h	13.3.6/673
12_403C	Options register n layout for FCM Mode (eLBC_ORf7)	32	R/W	0000_0000h	13.3.7/677
12_403C	Options register n layout for UPM Mode (eLBC_ORu7)	32	R/W	0000_0000h	13.3.8/681
12_4068	UPM address register (eLBC_MAR)	32	R/W	0000_0000h	13.3.9/683
12_4070	UPMn mode register (eLBC_MAMR)	32	R/W	0000_0000h	13.3.10/684
12_4074	UPMn mode register (eLBC_MBMR)	32	R/W	0000_0000h	13.3.10/684
12_4078	UPMn mode register (eLBC_MCMR)	32	R/W	0000_0000h	13.3.10/684
12_4084	Memory refresh timer prescaler register (eLBC_MRTPR)	32	R/W	0000_0000h	13.3.11/687
12_4088	UPM data register (eLBC_MDRu)	32	R/W	0000_0000h	13.3.12/687
12_4088	FCM data register (eLBC_MDRf)	32	R/W	0000_0000h	13.3.13/688
12_4090	Special operation initiation register (eLBC_LSOR)	32	R/W	0000_0000h	13.3.14/688
12_40A0	UPM refresh timer (eLBC_LURT)	32	R/W	0000_0000h	13.3.15/689
12_40B0	Transfer error status register (eLBC_LTESR)	32	w1c	0000_0000h	13.3.16/690
12_40B4	Transfer error disable register (eLBC_LTEDR)	32	R/W	0000_0000h	13.3.17/692

Table continues on the next page...

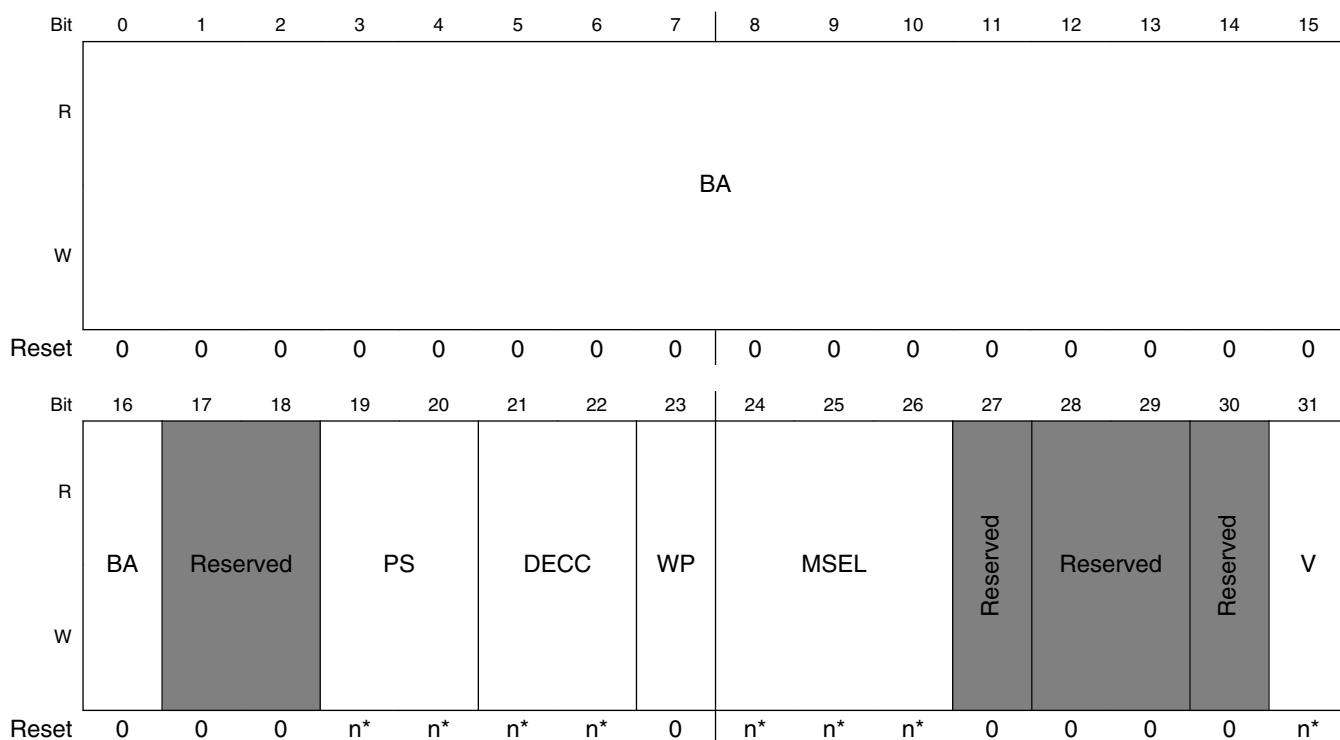
eLBC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_40B8	Transfer error interrupt register (eLBC_LTEIR)	32	R/W	0000_0000h	13.3.18/ 693
12_40BC	Transfer error attributes register (eLBC_LTEATR)	32	R/W	0000_0000h	13.3.19/ 694
12_40C0	Transfer error address register (eLBC_LTEAR)	32	R/W	0000_0000h	13.3.20/ 695
12_40C4	Transfer error ECC register (eLBC_LTECCR)	32	w1c	0000_0000h	13.3.21/ 696
12_40D0	Configuration register (eLBC_LBCR)	32	R/W	4000_0000h	13.3.22/ 697
12_40D4	Clock ratio register (eLBC_LCRR)	32	R/W	8000_0000h	13.3.23/ 698
12_40E0	Flash mode register (eLBC_FMR)	32	R/W	See section	13.3.24/ 700
12_40E4	Flash instruction register (eLBC_FIR)	32	R/W	0000_0000h	13.3.25/ 702
12_40E8	Flash command register (eLBC_FCR)	32	R/W	0000_0000h	13.3.26/ 704
12_40EC	Flash block address register (eLBC_FBAR)	32	R/W	0000_0000h	13.3.27/ 704
12_40F0	Flash page address register [Large Page Device (ORx[PGS] = 1)] (eLBC_FPARI)	32	R/W	0000_0000h	13.3.28/ 705
12_40F0	Flash page address register [Small Page Device (ORx[PGS] = 0)] (eLBC_FPARs)	32	R/W	0000_0000h	13.3.29/ 706
12_40F4	Flash byte count register (eLBC_FBCR)	32	R/W	0000_0000h	13.3.30/ 707
12_4100	Flash ECC block n registers (eLBC_FECC0)	32	R	0000_0000h	13.3.31/ 708
12_4104	Flash ECC block n registers (eLBC_FECC1)	32	R	0000_0000h	13.3.31/ 708
12_4108	Flash ECC block n registers (eLBC_FECC2)	32	R	0000_0000h	13.3.31/ 708
12_410C	Flash ECC block n registers (eLBC_FECC3)	32	R	0000_0000h	13.3.31/ 708

13.3.1 Base register 0 (eLBC_BR0)

The base registers (BR_n) contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, $BR0[V]$ is set, $BR1[V]$ - $BR7[V]$ are cleared, and the value of $BR0[PS]$ reflects the initial port size configured by the boot ROM location field of the reset configuration word.

Address: 12_4000h base + 0h offset = 12_4000h



* Notes:

- V field: BR0 has its valid bit (V) set for RCW[BOOT_LOC] = eLBC .
- MSEL field: M = 0 for MSEL of GPCM, 1 for MSEL of FCM at boot.
- DECC field: The reset value for DECC is determined by cfg_elbc_ecc .
- PS field: Thus bank 0 is valid with the port size (PS) configured from RCW[BOOT_LOC]as loaded during reset .

eLBC_BR0 field descriptions

Field	Description
0–16 BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits OR n [AM].

Table continues on the next page...

eLBC_BR0 field descriptions (continued)

Field	Description
17–18 -	This field is reserved. Reserved
19–20 PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the boot ROM location field in the reset configuration word as loaded during reset . For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8-bit (supported for GPCM, UPM, FCM) 10 16-bit (supported for GPCM, UPM) 11 Reserved
21–22 DECC	Specifies the method for data error checking. 00 Data error checking disabled , but normal parity generation for GPCM and UPM . No ECC generation for FCM. 01 ECC checking is enabled, but ECC generation is disabled, for FCM on full-page transfers. Normal parity generation and checking for GPCM and UPM. 10 ECC checking and generation are enabled for FCM on full-page transfers. 11 Reserved
23 WP	Write protect. 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert LCS[n]_B on write cycles to this memory bank. LTESR[WP] is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26 MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (possible reset value) 001 FCM (possible reset value) 010 Reserved 011 Reserved 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27 -	This field is reserved. Reserved
28–29 -	This field is reserved. Reserved
30 -	This field is reserved. Reserved
31 V	Valid bit. Indicates that the contents of the BR <i>n</i> and OR <i>n</i> pair are valid. LCS[n]_B does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

13.3.2 Options register 0 layout for GPCM Mode (eLBC_ORg0)

The OR n registers define the sizes of memory banks and access attributes. The OR n attribute bits support the following three modes of operation as defined by BR n [MSEL]:

- GPCM mode
- FCM mode
- UPM mode

The OR n registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the GPCM controls bank n .

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-6. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_0F07

The address mask field of the option registers (OR n [AM]) masks up to 17 corresponding BR n [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-7. Memory Bank Sizes in Relation to Address Mask

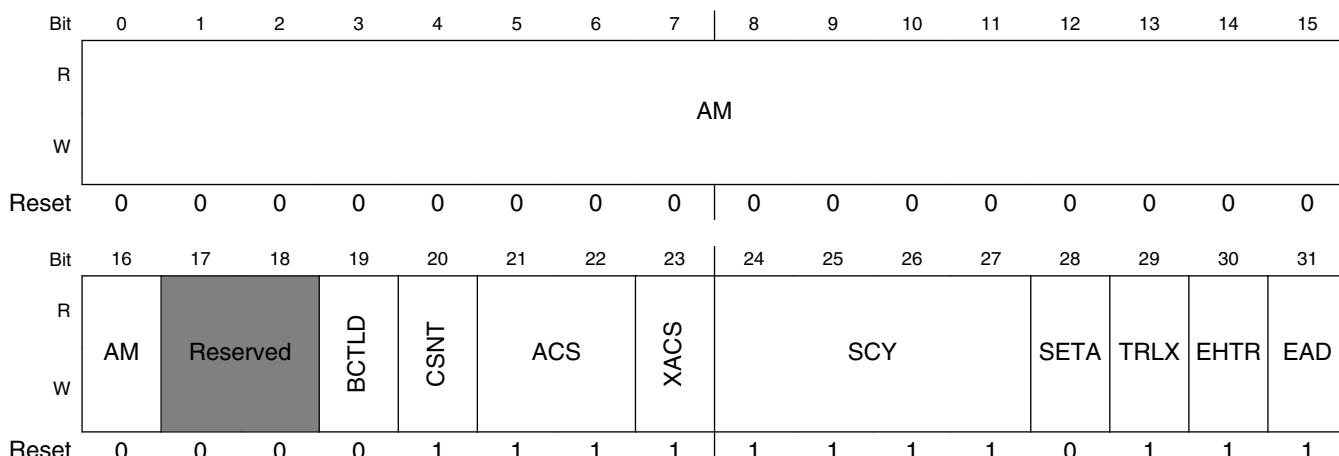
AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes

Table continues on the next page...

Table 13-7. Memory Bank Sizes in Relation to Address Mask (continued)

AM	Memory Bank Size
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Address: 12_4000h base + 4h offset = 12_4004h

**eLBC_ORg0 field descriptions**

Field	Description
0–16 AM	GPCM address mask. Masks corresponding BRn bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked and therefore don't care for address checking. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18 -	This field is reserved. Reserved
19 BCTL	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20 CSNT	Chip select negation time. Determines when LCS[n]_B and LWE_B are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only LWE_B is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals.

Table continues on the next page...

eLBC_ORg0 field descriptions (continued)

Field	Description
	0 LCSn_B and LWE_B are negated normally. 1 LCSn_B and LWE_B are negated one quarter of a bus clock cycle earlier.
21–22 ACS	Address to chip-select setup. Determines the delay of the LCS[n]_B assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11. 00 LCSn_B is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT = 0. 01 Reserved. 10 LCSn_B is output one quarter bus clock cycle after the address lines. 11 LCSn_B is output one half bus clock cycle after the address lines.
23 XACS	Extra address to chip-select setup. Setting this bit increases the delay of the LCS[n]_B assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1. 0 Address to chip-select setup is determined by ORx[ACS]. 1 Address to chip-select setup is extended (see Table 13-191 and Table 13-192).
24–27 SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, OR0[SCY] = 1111. 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states
28 SETA	External address termination. 0 Access is terminated internally by the memory controller unless the external device asserts LGTA_B earlier to terminate the access. 1 Access is terminated externally by asserting the LGTA_B external pin. (Only LGTA_B can terminate the access).
29 TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: <ul style="list-style-type: none"> • Adds an additional cycle between the address and control signals (only if ACS is not equal to 00). • Doubles the number of wait states specified by SCY, providing up to 30 wait states. • Works in conjunction with EHTR to extend hold time on read accesses. • LCSn_B (only if ACS is not equal to 00) and LWE_B signals are negated one cycle earlier during writes.
30 EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.

Table 13-8. TRLX and EHTR

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.

Table continues on the next page...

eLBC_ORg0 field descriptions (continued)

Field	Description		
Table 13-8. TRLX and EHTR (continued)			
	TRLX	EHTR	Meaning
	1	1	8 idle clock cycles are inserted.
31 EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).		

13.3.3 Options register 0 layout for FCM Mode (eLBC_ORf0)

The OR *n* registers define the sizes of memory banks and access attributes. The OR *n* attribute bits support the following three modes of operation as defined by BR *n* [MSEL]:

- GPCM mode
- FCM mode
- UPM mode

The OR *n* registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the FCM controls bank *n*.

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-10. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_0F07

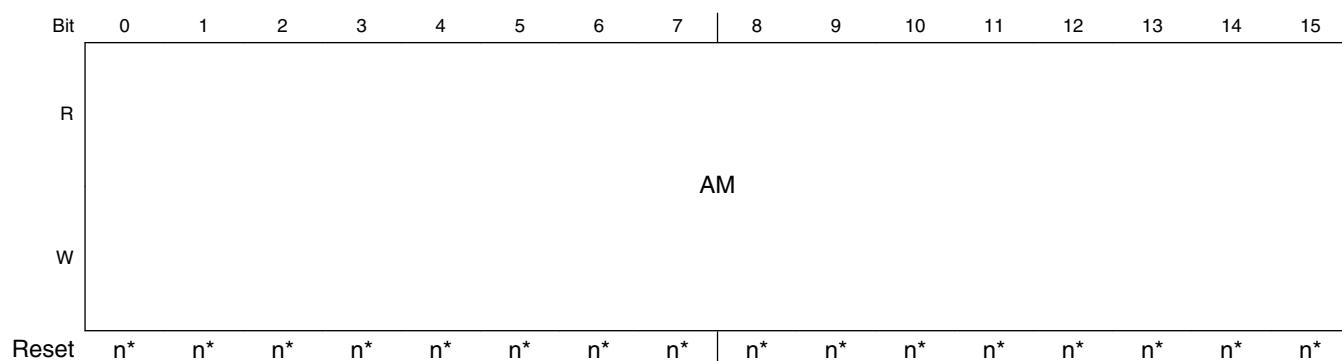
The address mask field of the option registers (OR *n* [AM]) masks up to 17 corresponding BR *n* [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be

used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-11. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Address: 12_4000h base + 4h offset = 12_4004h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
AM	Reserved		BCTLD	Reserved	PGS	CSCT	CST	CHT		SCY		RST	TRLX	EHTR		
W																Reserved

Reset n* n n n n n n n n n n n n n n n

* Notes:

- AM field: See Table "Reset value of OR0 Register" above for the OR0 reset value. All other option registers have all bits cleared.

eLBC_ORf0 field descriptions

Field	Description	
0–16 AM	FCM address mask. Masks corresponding BRn bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map.	
	0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.	
17–18 -	This field is reserved. Reserved	
19 BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank.	
	0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.	
20 -	This field is reserved. Reserved	
21 PGS	NAND Flash EEPROM page size, buffer size, and block size.	
	0 Page size of 512 main area bytes plus 16 spare area bytes (small page devices); FCM RAM buffers are 1 Kbyte each; Flash block size of 16 Kbytes. 1 Page size of 2048 main area bytes plus 64 spare area bytes (large page devices); FCM RAM buffers are 4 Kbytes each; Flash block size of 128 Kbytes.	
22 CSCT	Chip select to command time. Determines how far in advance LCS[n]_B is asserted prior to any bus activity during a NAND Flash access handled by the FCM. This helps meet chip-select setup times for slow memories.	

Table 13-16. TRLX and CSCT

TRLX	CSCT	Meaning
0	0	The chip-select is asserted 1 clock cycle before any command.
0	1	The chip-select is asserted 4 clock cycles before any command.
1	0	The chip-select is asserted 2 clock cycles before any command.
1	1	The chip-select is asserted 8 clock cycles before any command.

Table continues on the next page...

eLBC_ORf0 field descriptions (continued)

Field	Description																	
23 CST	Command setup time. Determines the delay of LFWE_B assertion relative to the command, address, or data change when the external memory access is handled by the FCM.																	
	Table 13-15. TRLX and CST																	
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>TRLX</th> <th>CST</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The write-enable is asserted coincident with any command.</td> </tr> <tr> <td>0</td> <td>1</td> <td>The write-enable is asserted 0.25 clock cycles after any command, address, or data.</td> </tr> <tr> <td>1</td> <td>0</td> <td>The write-enable is asserted 0.5 clock cycles after any command, address, or data.</td> </tr> <tr> <td>1</td> <td>1</td> <td>The write-enable is asserted 1 clock cycle after any command, address, or data.</td> </tr> </tbody> </table>			TRLX	CST	Meaning	0	0	The write-enable is asserted coincident with any command.	0	1	The write-enable is asserted 0.25 clock cycles after any command, address, or data.	1	0	The write-enable is asserted 0.5 clock cycles after any command, address, or data.	1	1	The write-enable is asserted 1 clock cycle after any command, address, or data.
TRLX	CST	Meaning																
0	0	The write-enable is asserted coincident with any command.																
0	1	The write-enable is asserted 0.25 clock cycles after any command, address, or data.																
1	0	The write-enable is asserted 0.5 clock cycles after any command, address, or data.																
1	1	The write-enable is asserted 1 clock cycle after any command, address, or data.																
24 CHT	Command hold time. Determines the LFWE_B negation prior to the command, address, or data change when the external memory access is handled by the FCM.																	
	Table 13-14. TRLX and CHT																	
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>TRLX</th> <th>CHT</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The write-enable is negated 0.5 clock cycles before any command, address, or data change.</td> </tr> <tr> <td>0</td> <td>1</td> <td>The write-enable is negated 1 clock cycle before any command, address, or data change.</td> </tr> <tr> <td>1</td> <td>0</td> <td>The write-enable is negated 1.5 clock cycles before any command, address, or data change.</td> </tr> <tr> <td>1</td> <td>1</td> <td>The write-enable is negated 2 clock cycles before any command, address, or data change.</td> </tr> </tbody> </table>			TRLX	CHT	Meaning	0	0	The write-enable is negated 0.5 clock cycles before any command, address, or data change.	0	1	The write-enable is negated 1 clock cycle before any command, address, or data change.	1	0	The write-enable is negated 1.5 clock cycles before any command, address, or data change.	1	1	The write-enable is negated 2 clock cycles before any command, address, or data change.
TRLX	CHT	Meaning																
0	0	The write-enable is negated 0.5 clock cycles before any command, address, or data change.																
0	1	The write-enable is negated 1 clock cycle before any command, address, or data change.																
1	0	The write-enable is negated 1.5 clock cycles before any command, address, or data change.																
1	1	The write-enable is negated 2 clock cycles before any command, address, or data change.																
25–27 SCY	Cycle length in bus clocks. Determines the following: <ul style="list-style-type: none"> the number of wait states inserted in command, address, or data transfer bus cycles, when the FCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. the delay between command/address writes and data write cycles, or the delay between write cycles and read cycles from NAND Flash EEPROM. A delay of 4x(2+SCY) clock cycles (TRLX = 0) or 8x(2+SCY) clock cycles (TRLX = 1) is inserted between the last write and the first data transfer to/from NAND Flash devices. the delay between a command write and the first sample point of the RDY/BSY_B pin (connected to LFRB_B). LFRB_B is not sampled until 8x(2+SCY) clock cycles (TRLX = 0) or 16x(2+SCY) clock cycles (TRLX = 1) have elapsed following the command. 																	
	000 No extra wait states 001 1 bus clock cycle wait state ... 111 7 bus clock cycle wait states																	
28 RST	Read setup time. Determines the delay of LFRE_B assertion relative to sampling of read data when the external memory access is handled by the FCM.																	
	Table 13-13. TRLX and RST																	
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>TRLX</th> <th>RST</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>The read-enable is asserted 0.75 clock cycles prior to any wait states.</td> </tr> <tr> <td>0</td> <td>1</td> <td>The read-enable is asserted 1 clock cycle prior to any wait states.</td> </tr> </tbody> </table>			TRLX	RST	Meaning	0	0	The read-enable is asserted 0.75 clock cycles prior to any wait states.	0	1	The read-enable is asserted 1 clock cycle prior to any wait states.						
TRLX	RST	Meaning																
0	0	The read-enable is asserted 0.75 clock cycles prior to any wait states.																
0	1	The read-enable is asserted 1 clock cycle prior to any wait states.																

Table continues on the next page...

eLBC_ORf0 field descriptions (continued)

Field	Description		
Table 13-13. TRLX and RST (continued)			
	1	0	The read-enable is asserted 0.5 clock cycles prior to any wait states.
	1	1	The read-enable is asserted 1 clock cycle prior to any wait states.
29 TRLX	<p>Timing relaxed. Modifies the settings of timing parameters for slow memories.</p> <p>0 Normal timing is generated by the FCM.</p> <p>1 Relaxed timing on the following parameters:</p> <ul style="list-style-type: none"> Doubles the number of clock cycles between LCS[n]_B assertion and commands. Doubles the number of wait states specified by SCY, providing up to 14 wait states. Works in conjunction with CST and RST to extend command/address/data setup times. Adds one clock cycle to the command/address/data hold times. Works in conjunction with CBT to extend the wait time for read/busy status sampling by 16 clock cycles. Works in conjunction with EHTR to double hold time on read accesses. 		
30 EHTR	<p>Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.</p>		
Table 13-12. TRLX and EHTR			
	TRLX	EHTR	Meaning
	0	0	1 idle clock cycle is inserted.
	0	1	2 idle clock cycles are inserted.
	1	0	4 idle clock cycles are inserted.
	1	1	8 idle clock cycles are inserted.
31 -	<p>This field is reserved.</p> <p>Reserved</p>		

13.3.4 Options register 0 layout for UPM Mode (eLBC_ORu0)

The OR *n* registers define the sizes of memory banks and access attributes. The OR *n* attribute bits support the following three modes of operation as defined by BR *n* [MSEL]:

- GPCM mode
- FCM mode
- UPM mode

The OR *n* registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the UPM controls bank *n*.

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-18. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_0F07

The address mask field of the option registers (OR n [AM]) masks up to 17 corresponding BR n [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-19. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Enhanced Local Bus Controller (eLBC) Memory Map

Address: 12_4000h base + 4h offset = 12_4004h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									AM							
W																
Reset	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AM	Reserved	BCTL	Reserved	BI	Reserved			TRLX	EHTR	EAD					
W																
Reset	n*	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

* Notes:

- AM field: See Table "Reset value of OR0 Register" above for the OR0 reset value. All other option registers have all bits cleared.

eLBC_ORu0 field descriptions

Field	Description
0–16 AM	UPM address mask. Masks corresponding BR n bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address pins.
17–18 -	This field is reserved. Reserved
19 BCTL	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20–22 -	This field is reserved. Reserved
23 BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.
24–28 -	This field is reserved. Reserved
29 TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.
30 EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.

Table continues on the next page...

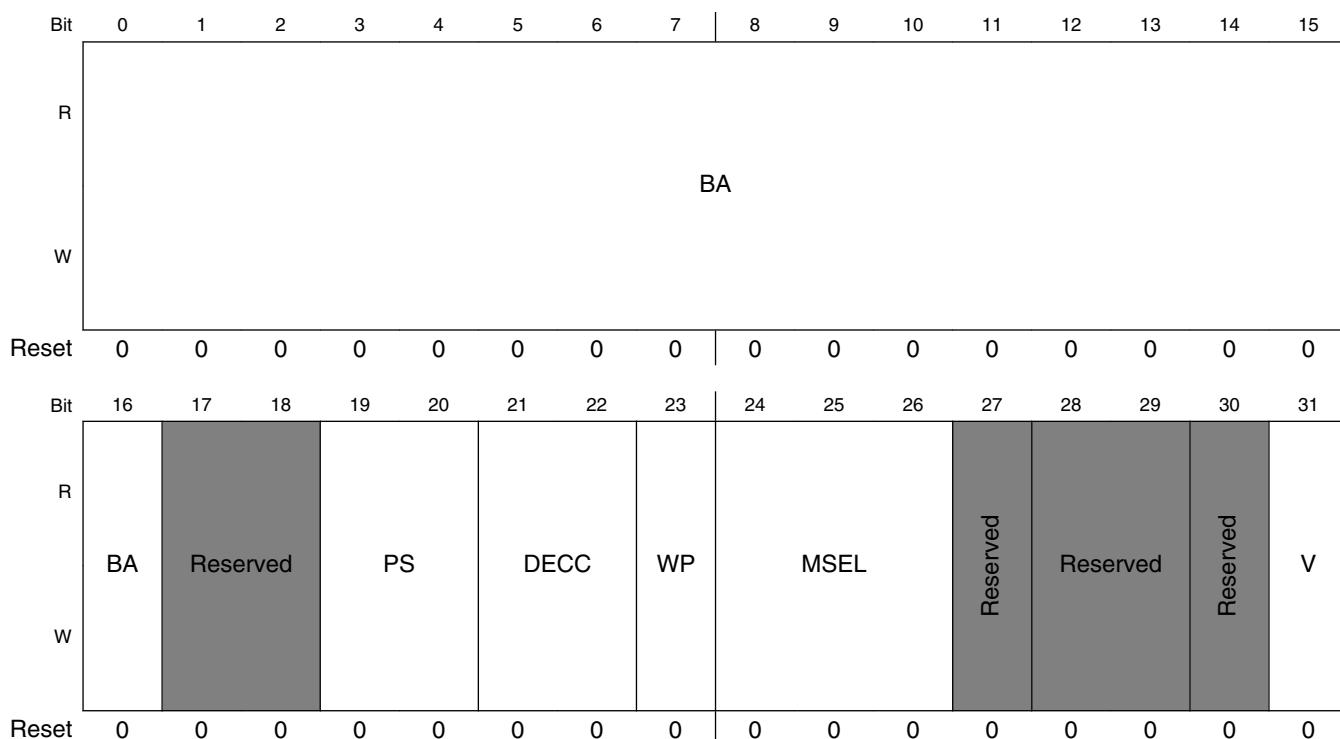
eLBC_ORu0 field descriptions (continued)

Field	Description		
Table 13-20. TRLX and EHTR			
	TRLX	EHTR	Meaning
	0	0	The memory controller generates normal timing. No additional cycles are inserted.
	0	1	1 idle clock cycle is inserted.
	1	0	4 idle clock cycles are inserted.
	1	1	8 idle clock cycles are inserted.
31 EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE).		
	0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).		

13.3.5 Base register n (eLBC_BRn)

The base registers (BR_n) contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, BR0[V] is set, BR1[V]-BR7[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location field of the reset configuration word .

Address: 12_4000h base + 8h offset + (8d × i), where i=0d to 6d



eLBC_BRn field descriptions

Field	Description
0–16 BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits OR _n [AM].
17–18 -	This field is reserved. Reserved
19–20 PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the boot ROM location field in the reset configuration word as loaded during reset . For all other banks the value is reset to 00 (port size not defined). 00 Reserved

Table continues on the next page...

eLBC_BRn field descriptions (continued)

Field	Description
	01 8-bit (supported for GPCM, UPM, FCM) 10 16-bit (supported for GPCM, UPM) 11 Reserved
21–22 DECC	Specifies the method for data error checking. 00 Data error checking disabled , but normal parity generation for GPCM and UPM . No ECC generation for FCM. 01 ECC checking is enabled, but ECC generation is disabled, for FCM on full-page transfers. Normal parity generation and checking for GPCM and UPM. 10 ECC checking and generation are enabled for FCM on full-page transfers. 11 Reserved
23 WP	Write protect. 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert LCS[n]_B on write cycles to this memory bank. LTESR[WP] is set (if WP is set) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.
24–26 MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (possible reset value) 001 FCM (possible reset value) 010 Reserved 011 Reserved 100 UPMA 101 UPMB 110 UPMC 111 Reserved
27 -	This field is reserved. Reserved
28–29 -	This field is reserved. Reserved
30 -	This field is reserved. Reserved
31 V	Valid bit. Indicates that the contents of the BR n and OR n pair are valid. LCS[n]_B does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

13.3.6 Options register n layout for GPCM Mode (eLBC_ORgn)

The OR n registers define the sizes of memory banks and access attributes. The OR n attribute bits support the following three modes of operation as defined by BR n [MSEL]:

- GPCM mode

- FCM mode
- UPM mode

The OR n registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the GPCM controls bank n .

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-28. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_0F07

The address mask field of the option registers (OR n [AM]) masks up to 17 corresponding BR n [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-29. Memory Bank Sizes in Relation to Address Mask

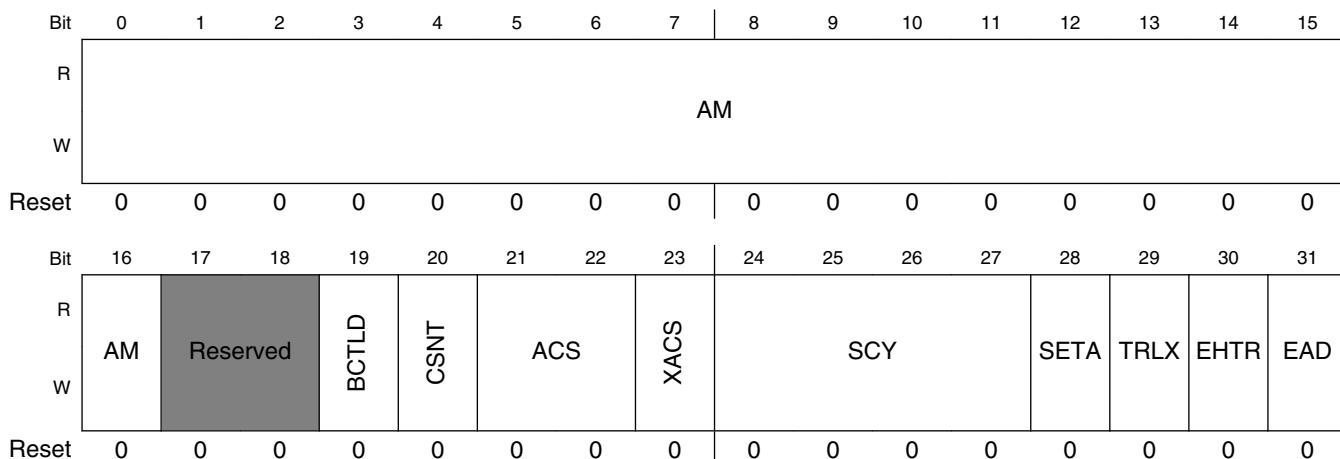
AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes

Table continues on the next page...

Table 13-29. Memory Bank Sizes in Relation to Address Mask (continued)

AM	Memory Bank Size
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Address: 12_4000h base + Ch offset + (8d × i), where i=0d to 6d



eLBC_ORgn field descriptions

Field	Description
0–16 AM	GPCM address mask. Masks corresponding BRn bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked and therefore don't care for address checking. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18 -	This field is reserved. Reserved
19 BCTL	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20 CSNT	Chip select negation time. Determines when LCS[n]_B and LWE_B are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only LWE_B is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 LCSn_B and LWE_B are negated normally. 1 LCSn_B and LWE_B are negated one quarter of a bus clock cycle earlier.
21–22 ACS	Address to chip-select setup. Determines the delay of the LCS[n]_B assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11. 00 LCSn_B is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT = 0. 01 Reserved.

Table continues on the next page...

eLBC_ORgn field descriptions (continued)

Field	Description
	10 LCSn_B is output one quarter bus clock cycle after the address lines. 11 LCSn_B is output one half bus clock cycle after the address lines.
23 XACS	Extra address to chip-select setup. Setting this bit increases the delay of the LCS[n]_B assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1. 0 Address to chip-select setup is determined by ORx[ACS]. 1 Address to chip-select setup is extended (see Table 13-191 and Table 13-192).
24–27 SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, OR0[SCY] = 1111. 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states
28 SETA	External address termination. 0 Access is terminated internally by the memory controller unless the external device asserts LGTA_B earlier to terminate the access. 1 Access is terminated externally by asserting the LGTA_B external pin. (Only LGTA_B can terminate the access).
29 TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: <ul style="list-style-type: none"> • Adds an additional cycle between the address and control signals (only if ACS is not equal to 00). • Doubles the number of wait states specified by SCY, providing up to 30 wait states. • Works in conjunction with EHTR to extend hold time on read accesses. • LCSn_B (only if ACS is not equal to 00) and LWE_B signals are negated one cycle earlier during writes.
30 EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
31 EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

Table 13-30. TRLX and EHTR

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.

13.3.7 Options register n layout for FCM Mode (eLBC_ORfn)

The OR n registers define the sizes of memory banks and access attributes. The OR n attribute bits support the following three modes of operation as defined by BR n [MSEL]:

- GPCM mode
- FCM mode
- UPM mode

The OR n registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the FCM controls bank n .

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-40. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_0F07

The address mask field of the option registers (OR n [AM]) masks up to 17 corresponding BR n [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-41. Memory Bank Sizes in Relation to Address Mask

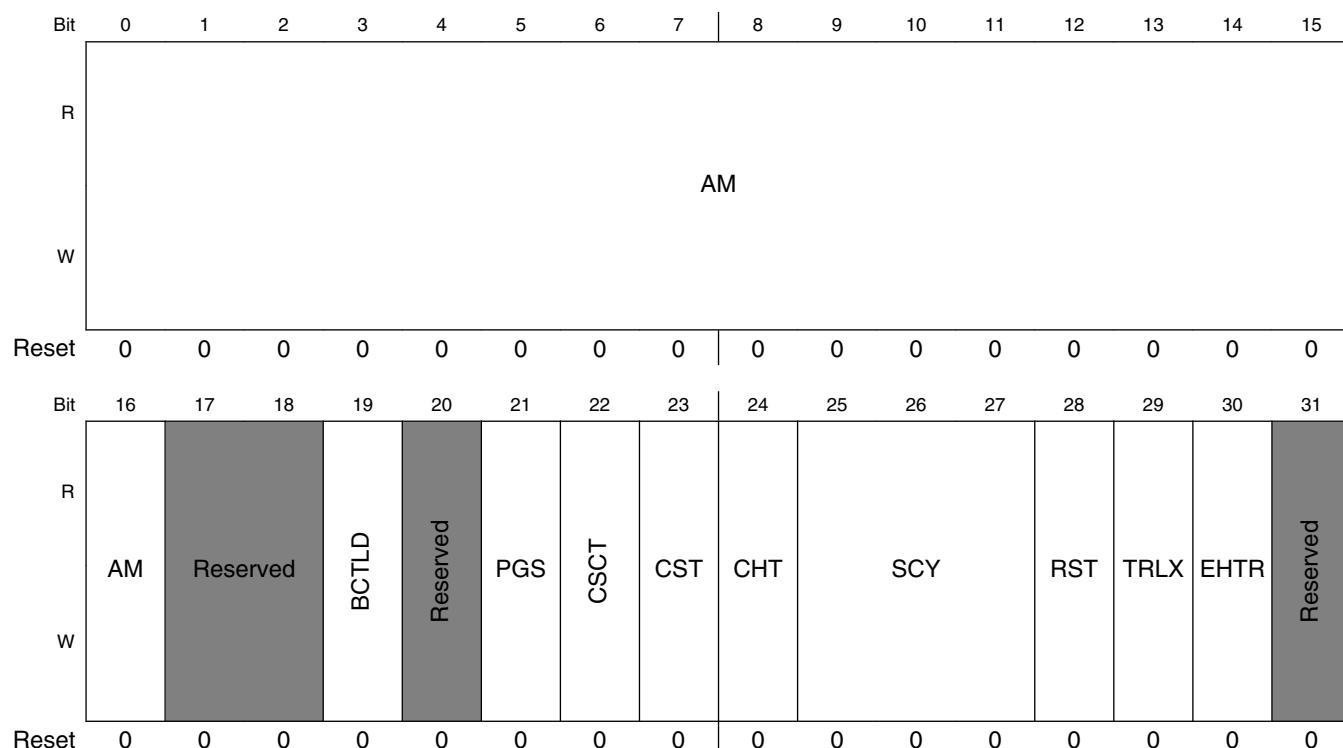
AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes

Table continues on the next page...

Table 13-41. Memory Bank Sizes in Relation to Address Mask (continued)

AM	Memory Bank Size
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Address: 12_4000h base + Ch offset + (8d × i), where i=0d to 6d



eLBC_ORfn field descriptions

Field	Description
0-16 AM	FCM address mask. Masks corresponding BRn bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.

Table continues on the next page...

eLBC_ORfn field descriptions (continued)

Field	Description															
17–18 -	This field is reserved. Reserved															
19 BCTL _D	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.															
20 -	This field is reserved. Reserved															
21 PGS	NAND Flash EEPROM page size, buffer size, and block size. 0 Page size of 512 main area bytes plus 16 spare area bytes (small page devices); FCM RAM buffers are 1 Kbyte each; Flash block size of 16 Kbytes. 1 Page size of 2048 main area bytes plus 64 spare area bytes (large page devices); FCM RAM buffers are 4 Kbytes each; Flash block size of 128 Kbytes.															
22 CSCT	Chip select to command time. Determines how far in advance LCS[n]_B is asserted prior to any bus activity during a NAND Flash access handled by the FCM. This helps meet chip-select setup times for slow memories.															
Table 13-46. TRLX and CSCT																
<table border="1"> <thead> <tr> <th>TRLX</th><th>CSCT</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>The chip-select is asserted 1 clock cycle before any command.</td></tr> <tr> <td>0</td><td>1</td><td>The chip-select is asserted 4 clock cycles before any command.</td></tr> <tr> <td>1</td><td>0</td><td>The chip-select is asserted 2 clock cycles before any command.</td></tr> <tr> <td>1</td><td>1</td><td>The chip-select is asserted 8 clock cycles before any command.</td></tr> </tbody> </table>		TRLX	CSCT	Meaning	0	0	The chip-select is asserted 1 clock cycle before any command.	0	1	The chip-select is asserted 4 clock cycles before any command.	1	0	The chip-select is asserted 2 clock cycles before any command.	1	1	The chip-select is asserted 8 clock cycles before any command.
TRLX	CSCT	Meaning														
0	0	The chip-select is asserted 1 clock cycle before any command.														
0	1	The chip-select is asserted 4 clock cycles before any command.														
1	0	The chip-select is asserted 2 clock cycles before any command.														
1	1	The chip-select is asserted 8 clock cycles before any command.														
23 CST	Command setup time. Determines the delay of LFWE_B assertion relative to the command, address, or data change when the external memory access is handled by the FCM.															
Table 13-45. TRLX and CST																
<table border="1"> <thead> <tr> <th>TRLX</th><th>CST</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>The write-enable is asserted coincident with any command.</td></tr> <tr> <td>0</td><td>1</td><td>The write-enable is asserted 0.25 clock cycles after any command, address, or data.</td></tr> <tr> <td>1</td><td>0</td><td>The write-enable is asserted 0.5 clock cycles after any command, address, or data.</td></tr> <tr> <td>1</td><td>1</td><td>The write-enable is asserted 1 clock cycle after any command, address, or data.</td></tr> </tbody> </table>		TRLX	CST	Meaning	0	0	The write-enable is asserted coincident with any command.	0	1	The write-enable is asserted 0.25 clock cycles after any command, address, or data.	1	0	The write-enable is asserted 0.5 clock cycles after any command, address, or data.	1	1	The write-enable is asserted 1 clock cycle after any command, address, or data.
TRLX	CST	Meaning														
0	0	The write-enable is asserted coincident with any command.														
0	1	The write-enable is asserted 0.25 clock cycles after any command, address, or data.														
1	0	The write-enable is asserted 0.5 clock cycles after any command, address, or data.														
1	1	The write-enable is asserted 1 clock cycle after any command, address, or data.														
24 CHT	Command hold time. Determines the LFWE_B negation prior to the command, address, or data change when the external memory access is handled by the FCM.															
Table 13-44. TRLX and CHT																
<table border="1"> <thead> <tr> <th>TRLX</th><th>CHT</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>The write-enable is negated 0.5 clock cycles before any command, address, or data change.</td></tr> <tr> <td>0</td><td>1</td><td>The write-enable is negated 1 clock cycle before any command, address, or data change.</td></tr> <tr> <td>1</td><td>0</td><td>The write-enable is negated 1.5 clock cycles before any command, address, or data change.</td></tr> </tbody> </table>		TRLX	CHT	Meaning	0	0	The write-enable is negated 0.5 clock cycles before any command, address, or data change.	0	1	The write-enable is negated 1 clock cycle before any command, address, or data change.	1	0	The write-enable is negated 1.5 clock cycles before any command, address, or data change.			
TRLX	CHT	Meaning														
0	0	The write-enable is negated 0.5 clock cycles before any command, address, or data change.														
0	1	The write-enable is negated 1 clock cycle before any command, address, or data change.														
1	0	The write-enable is negated 1.5 clock cycles before any command, address, or data change.														

Table continues on the next page...

eLBC_ORfn field descriptions (continued)

Field	Description				
Table 13-44. TRLX and CHT (continued)					
	1	1	The write-enable is negated 2 clock cycles before any command, address, or data change.		
25–27 SCY	<p>Cycle length in bus clocks. Determines the following:</p> <ul style="list-style-type: none"> the number of wait states inserted in command, address, or data transfer bus cycles, when the FCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. the delay between command/address writes and data write cycles, or the delay between write cycles and read cycles from NAND Flash EEPROM. A delay of 4x(2+SCY) clock cycles (TRLX = 0) or 8x(2+SCY) clock cycles (TRLX = 1) is inserted between the last write and the first data transfer to/from NAND Flash devices. the delay between a command write and the first sample point of the RDY/BSY_B pin (connected to LFRB_B). LFRB_B is not sampled until 8x(2+SCY) clock cycles (TRLX = 0) or 16x(2+SCY) clock cycles (TRLX = 1) have elapsed following the command. 				
	000	No extra wait states			
	001	1 bus clock cycle wait state			
	...				
	111	7 bus clock cycle wait states			
28 RST	<p>Read setup time. Determines the delay of LFRE_B assertion relative to sampling of read data when the external memory access is handled by the FCM.</p>				
	Table 13-43. TRLX and RST				
	TRLX	RST	Meaning		
	0	0	The read-enable is asserted 0.75 clock cycles prior to any wait states.		
	0	1	The read-enable is asserted 1 clock cycle prior to any wait states.		
	1	0	The read-enable is asserted 0.5 clock cycles prior to any wait states.		
	1	1	The read-enable is asserted 1 clock cycle prior to any wait states.		
29 TRLX	<p>Timing relaxed. Modifies the settings of timing parameters for slow memories.</p> <p>0 Normal timing is generated by the FCM.</p> <p>1 Relaxed timing on the following parameters:</p> <ul style="list-style-type: none"> Doubles the number of clock cycles between LCS[n]_B assertion and commands. Doubles the number of wait states specified by SCY, providing up to 14 wait states. Works in conjunction with CST and RST to extend command/address/data setup times. Adds one clock cycle to the command/address/data hold times. Works in conjunction with CBT to extend the wait time for read/busy status sampling by 16 clock cycles. Works in conjunction with EHTR to double hold time on read accesses. 				
30 EHTR	<p>Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.</p>				
	Table 13-42. TRLX and EHTR				
	TRLX	EHTR	Meaning		
	0	0	1 idle clock cycle is inserted.		

Table continues on the next page...

eLBC_ORfn field descriptions (continued)

Field	Description		
Table 13-42. TRLX and EHTR (continued)			
	TRLX	EHTR	Meaning
	0	1	2 idle clock cycles are inserted.
	1	0	4 idle clock cycles are inserted.
	1	1	8 idle clock cycles are inserted.
31	This field is reserved. Reserved		

13.3.8 Options register n layout for UPM Mode (eLBC_ORn)

The OR *n* registers define the sizes of memory banks and access attributes. The OR *n* attribute bits support the following three modes of operation as defined by BR *n* [MSEL]:

- GPCM mode
- FCM mode
- UPM mode

The OR *n* registers are interpreted differently depending on which of the three machine types is selected for that bank; this section describes the interpretation when the UPM controls bank *n*.

Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options .

Table 13-52. Reset value of OR0 Register

Boot Source	OR0 Reset Value
FCM (small page NAND Flash)	0000_03AE
FCM (large page NAND Flash)	0000_07AE
GPCM	0000_0FF7
eLBC not used as a boot source	0000_OF07

The address mask field of the option registers (OR *n* [AM]) masks up to 17 corresponding BR *n* [BA] fields. The 15 LSBs of the 32-bit internal transaction address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. The table below shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 13-53. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000_0	4 Gbytes
1000_0000_0000_0000_0	2 Gbytes
1100_0000_0000_0000_0	1 Gbyte
1110_0000_0000_0000_0	512 Mbytes
1111_0000_0000_0000_0	256 Mbytes
1111_1000_0000_0000_0	128 Mbytes
1111_1100_0000_0000_0	64 Mbytes
1111_1110_0000_0000_0	32 Mbytes
1111_1111_0000_0000_0	16 Mbytes
1111_1111_1000_0000_0	8 Mbytes
1111_1111_1100_0000_0	4 Mbytes
1111_1111_1110_0000_0	2 Mbytes
1111_1111_1111_0000_0	1 Mbyte
1111_1111_1111_1000_0	512 Kbytes
1111_1111_1111_1100_0	256 Kbytes
1111_1111_1111_1110_0	128 Kbytes
1111_1111_1111_1111_0	64 Kbytes
1111_1111_1111_1111_1	32 Kbytes

Address: 12_4000h base + Ch offset + (8d × i), where i=0d to 6d

**eLBC_ORun field descriptions**

Field	Description
0–16 AM	UPM address mask. Masks corresponding BR n bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address pins.

Table continues on the next page...

eLBC_ORun field descriptions (continued)

Field	Description	
17–18 -	This field is reserved. Reserved	
19 BCTL _D	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.	
20–22 -	This field is reserved. Reserved	
23 BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.	
24–28 -	This field is reserved. Reserved	
29 TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.	
30 EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.	
Table 13-54. TRLX and EHTR		
TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.
31 EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).	

Table 13-54. TRLX and EHTR

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.

13.3.9 UPM address register (eLBC MAR)

Address: 12 4000h base + 68h offset = 12 4068h

eLBC MAR field descriptions

Field	Description
0-31 A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

13.3.10 UPM_n mode register (eLBC_MnMR)

The UPM machine mode registers (MAMR, MBMR, and MCMR) contain the configuration for the three UPMS.

Address: 12_4000h base + 70h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W		RFEN		OP	UWPL		AM		DS		G0CL		GPL4		RLF	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W		RLF			WLF			TLF					MAD			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_MnMR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1 RFEN	Refresh enable. Indicates that the UPM needs refresh services. This bit must be set for UPMA (refresh executor) if refresh services are required on any UPM assigned chip selects. If MAMR[RFEN] = 0, no refresh services can be provided, even if UPMB and/or UPMC have their RFEN bit set. 0 Refresh services are not required 1 Refresh services are required
2-3 OP	Command opcode. Determines the command executed by the UPM <i>n</i> when a memory access hits a UPM assigned bank. 00 Normal operation 01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented.

Table continues on the next page...

eLBC_MnMR field descriptions (continued)

Field	Description
	<p>10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented.</p> <p>11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.</p>
4 UWPL	<p>LUPWAIT polarity active low. Sets the polarity of the LUPWAIT pin when in UPM mode.</p> <p>0 LUPWAIT is active high.</p> <p>1 LUPWAIT is active low.</p>
5–7 AM	<p>Address multiplex size. Determines how the address of the current memory cycle can be output on the address pins. This field is needed when interfacing with devices requiring row and column addresses multiplexed on the same pins. See Address multiplexing (AMX) for more information.</p> <p>000 Internal transaction address a[8:23] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>001 Internal transaction address a[7:22] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>010 Internal transaction address a[6:21] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>011 Internal transaction address a[5:20] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>100 Internal transaction address a[4:19] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>101 Internal transaction address a[3:18] driven on LA [16:31]; LAD[0:15] driven low.</p> <p>110 Reserved</p> <p>111 Reserved</p>
8–9 DS	<p>Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by UPM n. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPM n allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPM n is also allowed. To avoid conflicts between successive accesses to different banks, the minimum pattern in the RAM array for a request serviced, should not be shorter than the period established by DS.</p> <p>00 1-bus clock cycle disable period</p> <p>01 2-bus clock cycle disable period</p> <p>10 3-bus clock cycle disable period</p> <p>11 4-bus clock cycle disable period</p>
10–12 G0CL	<p>General line 0 control. Determines which logical address line can be output to the LGPL0 pin when the UPM n is selected to control the memory access.</p> <p>000 A12</p> <p>001 A11</p> <p>010 A10</p> <p>011 A9</p> <p>100 A8</p> <p>101 A7</p> <p>110 A6</p> <p>111 A5</p>
13 GPL4	LGPL4 output line disable. Determines how the LGPL4/LUPWAIT pin is controlled by the corresponding bits in the UPM n array. See Table 13-202 .

Table continues on the next page...

eLBC_MnMR field descriptions (continued)

Field	Description																	
	Table 13-161. Effect of GPL4 Settings																	
	Value	LGPL4/LUPWAIT Pin Function	Interpretation of UPM Word Bits															
			G4T1/DLT3	G4T3/WAEN														
	0	LGPL4 (output)	G4T1	G4T3														
	1	LUPWAIT (input)	DLT3	WAEN														
14–17 RLF	<p>Read loop field. Determines the number of times a loop defined in the UPM n will be executed for a burst- or single-beat read pattern or when $M \times MR[OP] = 11$ (run command)</p> <table> <tr><td>0000</td><td>16</td></tr> <tr><td>0001</td><td>1</td></tr> <tr><td>0010</td><td>2</td></tr> <tr><td>0011</td><td>3</td></tr> <tr><td colspan="2">...</td></tr> <tr><td>1110</td><td>14</td></tr> <tr><td>1111</td><td>15</td></tr> </table>				0000	16	0001	1	0010	2	0011	3	...		1110	14	1111	15
0000	16																	
0001	1																	
0010	2																	
0011	3																	
...																		
1110	14																	
1111	15																	
18–21 WLF	<p>Write loop field. Determines the number of times a loop defined in the UPM n will be executed for a burst- or single-beat write pattern.</p> <table> <tr><td>0000</td><td>16</td></tr> <tr><td>0001</td><td>1</td></tr> <tr><td>0010</td><td>2</td></tr> <tr><td>0011</td><td>3</td></tr> <tr><td colspan="2">...</td></tr> <tr><td>1110</td><td>14</td></tr> <tr><td>1111</td><td>15</td></tr> </table>				0000	16	0001	1	0010	2	0011	3	...		1110	14	1111	15
0000	16																	
0001	1																	
0010	2																	
0011	3																	
...																		
1110	14																	
1111	15																	
22–25 TLF	<p>Refresh loop field. Determines the number of times a loop defined in the UPM n will be executed for a refresh service pattern.</p> <table> <tr><td>0000</td><td>16</td></tr> <tr><td>0001</td><td>1</td></tr> <tr><td>0010</td><td>2</td></tr> <tr><td>0011</td><td>3</td></tr> <tr><td colspan="2">...</td></tr> <tr><td>1110</td><td>14</td></tr> <tr><td>1111</td><td>15</td></tr> </table>				0000	16	0001	1	0010	2	0011	3	...		1110	14	1111	15
0000	16																	
0001	1																	
0010	2																	
0011	3																	
...																		
1110	14																	
1111	15																	
26–31 MAD	<p>Machine address. RAM address pointer for the command executed. This field is incremented by 1, each time the UPM is accessed and the OP field is set to WRITE or READ. Address range is 64 words per UPM n.</p>																	

13.3.11 Memory refresh timer prescaler register (eLBC_MRTPR)

The refresh timer prescaler register (MRTPR) is used to divide the eLBC input clock (platform clock/2) to provide the UPM refresh timers clock.

Address: 12_4000h base + 84h offset = 12_4084h

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

eLBC_MRTPR field descriptions

Field	Description
0–7 PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The input clock (platform clock/2) is divided by PTP except when the value is 00000_0000, which represents the maximum divider of 256.
8–31 -	This field is reserved. Reserved

13.3.12 UPM data register (eLBC_MDRu)

The memory data register (MDR) contains data written to or read from the RAM array for UPM read or write commands. MDR also contains data written to or read from an external NAND Flash EEPROM for FCM write address, write data, and read status commands. MDR must be set up before issuing a write command to the UPM, or before issuing a FCM operation sequence that uses MDR to source address or data bytes.

Address: 12_4000h base + 88h offset = 12_4088h

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

eLBC_MDRu field descriptions

Field	Description
0–31 D	D is the data to be read or written into the RAM array when a write or read command is supplied to the UPM ($M \times MR[OP] = 01$ or $M \times MR[OP] = 10$).

13.3.13 FCM data register (eLBC_MDRf)

The memory data register (MDR) contains data written to or read from the RAM array for UPM read or write commands. MDR also contains data written to or read from an external NAND Flash EEPROM for FCM write address, write data, and read status commands. MDR must be set up before issuing a write command to the UPM, or before issuing a FCM operation sequence that uses MDR to source address or data bytes.

Address: 12_4000h base + 88h offset = 12_4088h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AS3							AS2							AS1							AS0										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

eLBC_MDRf field descriptions

Field	Description
0–7 AS3	AS3 is the fourth byte of address sent by a custom address write operation, or the fourth byte of data read from a read status operation.
8–15 AS2	AS2 is the third byte of address sent by a custom address write operation, or the third byte of data read from a read status operation.
16–23 AS1	AS1 is the second byte of address sent by a custom address write operation, or the second byte of data read from a read status operation.
24–31 AS0	AS0 is the first byte of address sent by a custom address write operation, or the first byte of data read from a read status operation.

13.3.14 Special operation initiation register (eLBC_LSOR)

The special operation initiation register (LSOR) is used by software to trigger a special operation on the indicated bank. Writing to LSOR activates a special operation on bank LSOR[BANK] provided that the bank is valid and controlled by a memory controller whose mode OP field is set to a value other than "normal operation." If eLBC is currently busy with a memory transaction, writing LSOR completes immediately, but the special operation request is queued until eLBC can service it. To avoid race conditions between software and a busy eLBC, registers that affect currently running special operation and LSOR must not be rewritten before a pending special operation has been completed. The UPM and FCM have different indications of when such special operations are completed. The behavior of eLBC is unpredictable if special operation modes are altered between LSOR being written and the relevant memory controller completing that access.

UPM special operation modes are set in registers $M \times MR[OP]$, see [UPMn mode register \(eLBC_MnMR\)](#). FCM special operation modes are set in $FMR[OP]$, see [Flash mode register \(eLBC_FMR\)](#). Writing LSOR has the same effect as setting a special controller mode and performing a dummy access to a bank associated with the controller in question, but use of LSOR avoids changing settings for the address space occupied by the bank. More details of special operation sequences appear in [UPM programming example \(two sequential writes to the RAM array\)](#).

Address: 12_4000h base + 90h offset = 12_4090h

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

eLBC_LSOR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 BANK	Bank on which a special operation is initiated. If the bank identified by BANK is marked valid ($BRn[V]$ set) and the bank is controlled by a memory controller whose current mode OP is non-zero-or a special operation-eLBC will request the special operation to be activated on the selected bank when this field is written. Otherwise, writing this field has no effect. 000 Bank 0 is triggered for special operation ... 111 Bank 7 is triggered for special operation

13.3.15 UPM refresh timer (eLBC_LURT)

The UPM refresh timer (LURT) generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled ($M \times MR[RFEN] = 1$). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

Address: 12_4000h base + A0h offset = 12_40A0h

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

eLBC_LURT field descriptions

Field	Description
0–7 LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation:

Table continues on the next page...

eLBC_LURT field descriptions (continued)

Field	Description
	<p>TimerPeriod = $\frac{\text{LURT}}{(\text{InputClock})}$ M RTPR[PTP]</p> <p>Where InputClock is platform clock/2.</p> <p>Example: For a 600-MHz platform clock, the input clock is 300 MHz. For a required service rate of 15.6 μs, given M RTPR[PTP] = 32, the LURT value should be 145 decimal. $145/(300 \text{ MHz}/32) = 15.5 \mu\text{s}$, which is less than the required service period of 15.6 μs.</p> <p>Note that the reset value (0x00) sets the maximum period to 256 x M RTPR[PTP] input clock cycles.</p>
8–31 -	This field is reserved. Reserved

13.3.16 Transfer error status register (eLBC_LTESR)

The transfer error status register (LTESR) indicates the cause of an error or event.

LTESR is a write-1-to-clear register. Reading LTESR occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear only the write protect error bit (LTESR[WP]) without affecting other LTESR bits, 0x0400_0000 should be written to the register. After any error/event reported by LTESR, LTEATR[V] must be cleared for LTESR to update again. Note that error statuses are only reflected in LTESR if they have been enabled in LTEDR.

Address: 12_4000h base + B0h offset = 12_40B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BM	FCT	PAR	Reserved	WP	Reserved						CS	Reserved			
W	w1c	w1c	w1c		w1c							w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												UCC	CC		
W													w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_LTESR field descriptions

Field	Description
0 BM	Bus monitor time-out 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[BMT] x LBCR[BMTPS] bus clock cycles from the start of a transaction.
1 FCT	FCM command time-out 0 No FCM command time-out occurred. 1 A CW0, CW1, CW2, or CW3 command issued to FCM timed-out with respect to the timer configured by FMR[CWTO].
2 PAR	Parity or ECC error 0 No local bus parity error 1 Local bus parity error (GPCM or UPM), or non-correctable ECC error (FCM). LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BNK] indicates which memory controller bank was accessed.
3–4 -	This field is reserved. Reserved
5 WP	Write protect error 0 No write protect error occurred. 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur (as the cycle is not automatically terminated).
6–11 -	This field is reserved. Reserved
12 CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the eLBC that did not hit any memory bank.
13–29 -	This field is reserved. Reserved
30 UCC	UPM Run pattern (MxMR[OP]=11) command completion event 0 No UPM Run pattern operation in progress, or operation pending. 1 UPM Run pattern operation has completed, allowing software to continue processing of results.
31 CC	FCM command completion event 0 No FCM operation in progress, or operation pending. 1 FCM operation has completed, allowing software to continue processing of results.

13.3.17 Transfer error disable register (eLBC_LTEDR)

The transfer error check disable register (LTEDR) is used to disable error/event checking, which are reported in LTESR. Note that control of error/event checking is independent of control of reporting of errors/events (LTEIR) through the interrupt mechanism.

Address: 12_4000h base + B4h offset = 12_40B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BMD	FCTD	PARD	Reserved	WPD				Reserved				CSD		Reserved	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									Reserved					UCCD		CCD
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_LTEDR field descriptions

Field	Description
0 BMD	Bus monitor disable 0 Bus monitor is enabled. 1 Bus monitor is disabled, but internal bus time-outs can still occur.
1 FCTD	FCM command time-out disable 0 FCM command timer is enabled. 1 FCM command time-out is disabled, but internal FCM command timer can terminate command waits.
2 PARD	Parity and ECC error checking disabled. 0 Parity and ECC error checking is enabled. 1 Parity and ECC error checking is disabled.
3–4 -	This field is reserved. Reserved
5 WPD	Write protect error checking disable. 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–11 -	This field is reserved. Reserved

Table continues on the next page...

eLBC_LTEDR field descriptions (continued)

Field	Description
12 CSD	Chip select error checking disable. 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–29 -	This field is reserved. Reserved
30 UCCD	UPM Run pattern command completion checking disable. 0 UPM Run pattern command completion checking is enabled. 1 UPM Run pattern command completion checking is disabled.
31 CCD	FCM command completion checking disable. 0 Command completion checking is enabled. 1 Command completion checking is disabled.

13.3.18 Transfer error interrupt register (eLBC_LTEIR)

The transfer error interrupt enable register (LTEIR) is used to send or block error/event reporting through the eLBC internal interrupt mechanism. Software should clear pending errors/events in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error/event bits negates the interrupt.

Address: 12_4000h base + B8h offset = 12_40B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BMI	FCTI	PARI	Reserved	WPI				Reserved				CSI		Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W									Reserved					UCCI	CCI	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_LTEIR field descriptions

Field	Description
0 BMI	Bus monitor error interrupt enable. 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1 FCTI	FCM command time-out interrupt enable. 0 FCM command time-out error reporting is disabled. 1 FCM command time-out error reporting is enabled.

Table continues on the next page...

eLBC_LTEIR field descriptions (continued)

Field	Description
2 PARI	Parity and ECC error interrupt enable. 0 Parity and ECC error reporting is disabled. 1 Parity and ECC error reporting is enabled.
3–4 -	This field is reserved. Reserved
5 WPI	Write protect error interrupt enable. 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–11 -	This field is reserved. Reserved
12 CSI	Chip select error interrupt enable. 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–29 -	This field is reserved. Reserved
30 UCCI	UPM Run pattern command completion Event interrupt enable. 0 UPM Run pattern command completion reporting is disabled. 1 UPM Run pattern command completion reporting is enabled.
31 CCI	FCM command completion Event interrupt enable. 0 Command completion reporting is disabled. 1 Command completion reporting is enabled.

13.3.19 Transfer error attributes register (eLBC_LTEATR)

The transfer error attributes register (LTEATR) captures source attributes of an error/event. After LTEATR[V] has been set, software must clear this bit to allow LTESR, LTEATR, and LTEAR to update following any subsequent events/errors.

NOTE

LTEATR may not capture accurate information for errors that occur when an FCM special operation is in progress.

Address: 12_4000h base + BCh offset = 12_40BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		RWB		Reserved								SRCID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PB								BNK				Reserved		V	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_LTEATR field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 RWB	Transaction type for the error: 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–7 -	This field is reserved. Reserved
8–15 SRCID	Captures the source of the transaction when this information is provided on the internal interface to the eLBC. See Global Source and Target IDs See Global Source and Target IDs , for the defined encodings.
16–19 PB	Parity error on byte or block . For GPCM and UPM, there are four parity error status bits, one per byte lane. A bit is set for the byte that had a parity error (bit 16 represents byte 0, the most significant byte lane). For FCM, there are at most four 512-byte page blocks (for a large page device) checked by ECC. A bit is set for the 512-byte block that had an non-correctable ECC error on read (bit 16 represents block 0, the first 512 bytes of a page; if ORx[PGS] = 0, bits 17-19 are always 0).
20–27 BNK	Memory controller bank. There is one error status bit per memory controller bank (bit 20 represents bank 0). A bit is set for the local bus memory controller bank that had an error.
28–30 -	This field is reserved. Reserved
31 V	Error attribute capture is valid. Indicates that the captured error information is valid. 0 Captured error attributes and address are not valid. 1 Captured error attributes and address are valid.

13.3.20 Transfer error address register (eLBC_LTEAR)

The transfer error address register (LTEAR) captures the address of a transaction that caused an error/event.

NOTE

LTEAR may not capture accurate information for errors that occur when an FCM special operation is in progress.

Address: 12_4000h base + C0h offset = 12_40C0h

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

eLBC_LTEAR field descriptions

Field	Description
0–31 A	Transaction address for the error. For GPCM and UPM, holds the 32-bit address of the transaction resulting in an error. For FCM, this register is undefined.

13.3.21 Transfer error ECC register (eLBC_LTECCR)

The transfer error ECC register (LTECCR) captures single bit and multibit errors per 512-byte sector in FCM mode. LTECCR is a write-1-to-clear register. Write operations can clear but not set bits. It captures the errors during full page read transfers on FCM command completion event, provided ECC check is enabled in BRx[DECC].

Address: 12_4000h base + C4h offset = 12_40C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												SBCE		Reserved												MBUE					
W													w1c														w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

eLBC_LTECCR field descriptions

Field	Description
0–11 -	This field is reserved. Reserved
12–15 SBCE	Single bit correctable error. There are at most four 512-byte page blocks (for a large page device) checked by ECC. A bit is set for the 512-byte block that had a single bit correctable ECC error on read (bit 12 represents block 0, the first 512 bytes of a page; if ORx[PGS] = 0, bits 13–15 are always 0).
16–27 -	This field is reserved. Reserved
28–31 MBUE	Multi bit uncorrectable error There are at most four 512-byte page blocks (for a large page device) checked by ECC. A bit is set for the 512-byte block that had an uncorrectable ECC error on read (bit 28 represents block 0, the first 512 bytes of a page; if ORx[PGS] = 0, bits 29–31 are always 0).

13.3.22 Configuration register (eLBC_LBCR)

Address: 12_4000h base + D0h offset = 12_40D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LDIS	LRDY	Reserved					BCTLC	AHD	Reserved			LPBSE	EPAR		
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BMT								Reserved			BMTPS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eLBC_LBCR field descriptions

Field	Description
0 LDIS	Local bus disable 0 Local bus is enabled. 1 Local bus is disabled. No internal transactions will be acknowledged.
1 LRDY	eLBC ready LCLK generated by eLBC / PLL is stable. This bit is cleared by the hardware in case any of the fields in LCRR register is changed. This bit is set by the hardware when the external LCLK is stable. Note that after changing any field of the LCRR register, the user should wait for at least 32 cycles before polling this bit.
2–7 -	This field is reserved. Reserved
8–9 BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/ R_B control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as LOE_B for GPCM accesses only. 10 LBCTL is used as LWE_B for GPCM accesses only. 11 Reserved
10 AHD	Address hold disable. Removes part of the hold time for LAD with respect to LALE in order to lengthen the LALE pulse. 0 During address phases on the local bus, the LALE signal negates four platform clock periods prior to the address being invalidated. 1 During address phases on the local bus, the LALE signal negates two platform clock period prior to the address being invalidated. This halves the address hold time, but extends the latch enable duration. This may be necessary for very high frequency designs.
11–13 -	This field is reserved. Reserved

Table continues on the next page...

eLBC_LBCR field descriptions (continued)

Field	Description
14 LPBSE	Enables parity byte select on LGTA_B/LFRB_B /LGPL4/LUPWAIT/LPBSE signal. 0 Parity byte select is disabled. LGTA_B/ LGPL4/LPBSE signal is available for memory control as LGPL4 (output) or LGTA_B/LFRB_B /LUPWAIT (input). 1 Parity byte select is enabled. LPBSE signal is dedicated as the parity byte select output, and LGTA_B/ LFRB_B /LUPWAIT is disabled.
15 EPAR	Determines odd or even parity. Writing GPCM or UPM controlled memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity; normal, odd-parity ECC 1 Even parity; inverted, even-parity ECC
16–23 BMT	Bus monitor timing. Defines the bus monitor time-out period. The number of LCLK clock cycles to count down before a time-out error is generated is given by: if BMT = 0, then bus cycles = 256 x PS if BMT ≠ 0, then bus cycles = BMT x PS where PS is set according to LBCR[BMTPS]. The value of bus cycles must not be less than 40 bus cycles for reliable operation. When BMT = 0, bus cycles = 256 x PS.
24–27 -	This field is reserved. Reserved
28–31 BMTPS	Bus monitor timer prescale. Defines the multiplier, PS, to scale LBCR[BMT] for determining bus time-outs. 0000 PS = 8 0001 PS = 16 0010 PS = 32 0011 PS = 64 0100 PS = 128 0101 PS = 256 0110 PS = 512 0111 PS = 1024 1000 PS = 2048 1001 PS = 4096 1010 PS = 8192 1011 PS = 16,384 1100 PS = 32,768 1101 PS = 65,536 1110 PS = 131,072 1111 PS = 262,144

13.3.23 Clock ratio register (eLBC_LCRR)

The clock ratio register sets the platform clock to eLBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

NOTE

For proper operation of the system, it is required that this register setting will not be altered while local bus memories or devices are being accessed. Special care needs to be taken when running instructions from an eLBC memory.

Address: 12_4000h base + D4h offset = 12_40D4h

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

eLBC_LCRR field descriptions

Field	Description
0–13 -	This field is reserved. Reserved. Note that bit 0 must remain set for proper operation.
14–15 EADC	External address delay cycles of LCLK. Defines the number of cycles for the assertion of LALE. 00 4 01 1 10 2 11 3
16–26 -	This field is reserved. Reserved
27–31 CLKDIV	Clock divider. Sets the frequency ratio between the platform clock and the local bus clock. Only the values shown below are allowed. Note that the reset value of CLKDIV depends on the RCW source configuration(cfg_rcw_src[0:4]). If the FCM is selected as the RCW source, the default value for CLKDIV is 0100; otherwise, the default is 1000. However, during reset, the clock behavior is different than the default would indicate. See eLBC bus clock and clock ratios , for more information. Also note that the eLBC is clocked by platform clock/2. The ratios given in this chapter include this divider. NOTE: It is critical that no transactions are being executed via the local bus while CLKDIV is being modified. As such, prior to modification, the user must ensure that code is not executing out of the local bus. Once LCRR[CLKDIV] is written, the register should be read, and then an isync should be executed. 00000-00001 Reserved 00010 8 00011 Reserved 00100 16 00101-00111 Reserved 01000 32 01001-11111 Reserved

13.3.24 Flash mode register (eLBC_FMR)

The local bus Flash mode register (FMR) controls global operation of the FCM.

Address: 12_4000h base + E0h offset = 12_40E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reset	0	0	0	0	n*	0	0	0	0	0	0	0	0	0	0	0
R	CWTO				BOOT	Reserved		ECCM	Reserved		AL	Reserved		OP		
W																

* Notes:

- BOOT field: Bit is set if power-on-reset configuration selects FCM as the boot ROM target.

eLBC_FMR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–19 CWTO	Command wait time-out. For FCM commands that wait on LFRB_B being sampled high (CW0, CW1, RBW, and RSW), FCM pauses execution of the instruction sequence until either LFRB_B is sampled high, or a timer controlled by CTO expires, whichever occurs first. The time-out in the latter case is as follows: 0000 256 cycles of LCLK 0001 512 cycles of LCLK 0010 1024 cycles of LCLK 0011 2048 cycles of LCLK 0100 4096 cycles of LCLK 0101 8192 cycles of LCLK 0110 16,384 cycles of LCLK 0111 32,768 cycles of LCLK 1000 65,536 cycles of LCLK 1001 131,072 cycles of LCLK 1010 262,144 cycles of LCLK 1011 524,288 cycles of LCLK 1100 1,048,576 cycles of LCLK 1101 2,097,152 cycles of LCLK

Table continues on the next page...

eLBC_FMR field descriptions (continued)

Field	Description
	1110 4,194,304 cycles of LCLK 1111 8,388,608 cycles of LCLK
20 BOOT	Flash auto-boot load mode. During system boot from NAND Flash EEPROM, this bit remains set to alter the use of the FCM buffer RAM. Software should clear BOOT once FCM is to be restored to normal operation. Setting BOOT without auto-boot in progress only alters the mapping of the buffer RAM. 0 FCM is operating in normal functional mode, with an 8 Kbyte FCM buffer RAM. 1 eLBC has been configured-either from reset or by a special operation OP = 01-to autoload a 4-Kbyte boot block into the FCM buffer RAM, which maps only the 4 Kbytes of NAND Flash main data region comprising the boot block. Any access to the buffer RAM is delayed until the entire boot block has been loaded.
21–22 -	This field is reserved. Reserved
23 ECCM	ECC mode. When hardware checking and/or generation of error correcting codes (ECC) is enabled (that is, when BRn[DECC] is 01 or 10, and full page transfers are specified with FBCR[BC] = 0), ECCM sets the ECC block size and position of the ECC code word(s) in the NAND Flash spare region for both checking and generation functions. The format of the ECC code word conforms with the Samsung/Toshiba spare region assignment specifications. 0 ECC is checked/calculated over 512-Byte blocks. A 24-bit ECC is assigned to spare region bytes at offsets (N x 16) + 6 through (N x 16) + 8 for spare region N, N = 0-3. 1 ECC is checked/calculated over 512-Byte blocks. A 24-bit ECC is assigned to spare region bytes at offsets (N x 16) + 8 through (N x 16) + 10 for spare region N, N = 0-3.
24–25 -	This field is reserved. Reserved
26–27 AL	Address length. AL sets the number of address bytes issued during page address (PA) operations. However, the number of address bytes issued for column address (CA) operations is determined by the device page size (for ORn[PGS] = 0, 1 CA byte is issued; for ORn[PGS] = 1, 2 CA bytes are issued). 00 2 bytes are issued for page addresses, thus a total of 3 (ORn[PGS] = 0) or 4 (ORn[PGS] = 1) address bytes are issued for a {CA,PA} sequence 01 3 bytes are issued for page addresses, thus a total of 4 (ORn[PGS] = 0) or 5 (ORn[PGS] = 1) address bytes are issued for a {CA,PA} sequence 10 4 bytes are issued for page addresses, thus a total of 5 (ORn[PGS] = 0) or 6 (ORn[PGS] = 1) address bytes are issued for a {CA,PA} sequence 11 Reserved
28–29 -	This field is reserved. Reserved
30–31 OP	Flash operation. For OP not equal to 00, a special operation is triggered on the next write to LSOR or dummy access to a bank controlled by FCM. Once a special operation has commenced, OP is automatically reset to 00 by FCM. Individual blocks may be temporarily unlocked for erase and reprogramming operations. 00 Normal operation. All read and write accesses to banks controlled by FCM access the shared FCM buffer RAM. No bus activity is caused by this operation. 01 Simulate auto-boot block loading, and set FMR[BOOT]. Boot block loading occurs from the bank triggered on the special operation, therefore the appropriate bank configuration must be initialized prior to issuing this operation.

Table continues on the next page...

eLBC_FMR field descriptions (continued)

Field	Description
10	Execute the command sequence contained in FIR, but with write protection enabled (pin LFWP_B asserted low) so that all Flash blocks are protected from accidental erasure and reprogramming.
11	Execute the command sequence contained in FIR, but permit the single block identified by FBAR[BLK] to be erased or reprogrammed, with pin LFWP_B remaining high during the access.

13.3.25 Flash instruction register (eLBC_FIR)

The local bus Flash instruction register (FIR) holds a sequence of up to eight instructions for issue by the FCM. Setting FMR[OP] non-zero and writing LSOR or accessing a bank controlled by FCM causes FCM to read FIR 4 bits at a time, starting at bit 0 and continuing with adjacent 4-bit opcodes, until only NOP opcodes remain. The programmed instruction sequence of OP0, OP1,..., OP7 is performed on the activated bank, using the data buffer addressed by FPAR. If LTEIR[CCI] = 1 and LTEDR[CCD] = 0, eLBC will generate an interrupt once the entire sequence has completed, and software should examine LTEATR and clear its V bit.

Software must not alter the contents of the addressed FCM buffer, FIR, MDR, FCR, FBAR, FPAR, or FBCR while an operation is in progress-or eLBC will behave unpredictably-but software can freely modify the contents of any currently unused FCM RAM buffer in preparation for the next operation.

Address: 12_4000h base + E4h offset = 12_40E4h

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

Reset 0

eLBC_FIR field descriptions

Field	Description
0–3 OP0	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. 0000 NOP-No-operation and end of operation sequence 0001 CA-Issue current column address as set in FPAR, with length set by ORx[PGS] 0010 PA-Issue current block+page address as set in FBAR and FPAR, with length set by FMR[AL] 0011 UA-Issue user-defined address byte from next AS field in MDR 0100 CM0-Issue command from FCR[CMD0] 0101 CM1-Issue command from FCR[CMD1] 0110 CM2-Issue command from FCR[CMD2] 0111 CM3-Issue command from FCR[CMD3] 1000 WB-Write FBCR bytes of data from current FCM buffer to Flash device 1001 WS-Write one byte (8b port) of data from next AS field of MDR to Flash device 1010 RB-Read FBCR bytes of data from Flash device into current FCM RAM buffer 1011 RS-Read one byte (8b port) of data from Flash device into next AS field of MDR 1100 CW0-Wait for LFRB_B to return high or time-out, then issue command from FCR[CMD0] 1101 CW1-Wait for LFRB_B to return high or time-out, then issue command from FCR[CMD1] 1110 RBW-Wait for LFRB_B to return high or time-out, then read FBCR bytes of data from Flash device into current FCM RAM buffer 1111 RSW-Wait for LFRB_B to return high or time-out, then read one byte (8b port) of data from Flash device into next AS field of MDR
4–7 OP1	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
8–11 OP2	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
12–15 OP3	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
16–19 OP4	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
20–23 OP5	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
24–27 OP6	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.
28–31 OP7	FCM operation codes. OP0 is executed first, followed by OP1, through to OP7. See description for OP0.

13.3.26 Flash command register (eLBC_FCR)

The local bus Flash command register (FCR) holds up to four NAND Flash EEPROM command bytes that may be referenced by opcodes in FIR during FCM operation. The values of the commands should follow the manufacturer's datasheet for the relevant NAND Flash device.

Address: 12_4000h base + E8h offset = 12_40E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMD0							CMD1							CMD2							CMD3										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

eLBC_FCR field descriptions

Field	Description
0–7 CMD0	General purpose FCM Flash command byte 0. Opcodes in FIR that issue command index 0 write CMD0 to the NAND Flash command/data bus.
8–15 CMD1	General purpose FCM Flash command byte 1. Opcodes in FIR that issue command index 1 write CMD1 to the NAND Flash command/data bus.
16–23 CMD2	General purpose FCM Flash command byte 2. Opcodes in FIR that issue command index 2 write CMD2 to the NAND Flash command/data bus.
24–31 CMD3	General purpose FCM Flash command byte 3. Opcodes in FIR that issue command index 3 write CMD3 to the NAND Flash command/data bus.

13.3.27 Flash block address register (eLBC_FBAR)

The local bus Flash block address register (FBAR) locates the NAND Flash block index for the page currently accessed.

Address: 12_4000h base + EC offset = 12_40EC

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

eLBC_FBAR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved

Table continues on the next page...

eLBC_FBAR field descriptions (continued)

Field	Description
8–31 BLK	Flash block address. The size of the NAND Flash, as configured in ORn[PGS] and FMR[AL], determines the number of bits of BLK that are issued to the EEPROM during block address phases.

13.3.28 Flash page address register [Large Page Device (ORx[PGS] = 1)] (eLBC_FPARI)

The local bus Flash page address register (FPAR) locates the current NAND Flash page in both the external NAND Flash device and FCM buffer RAM.

Address: 12_4000h base + F0h offset = 12_40F0h

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

eLBC_FPARI field descriptions

Field	Description
0–13 -	This field is reserved. Reserved
14–19 PI	Page index. PA indexes the page in NAND Flash EEPROM at the current block defined by FBAR, and locates the corresponding transfer buffer in the FCM buffer RAM. The LSB of PI indexes one of the two 4 Kbyte buffers in the FCM buffer RAM as follows: 0 The page is transferred to/from FCM buffer 0, address offsets 0x0000-0xFFFF 1 The page is transferred to/from FCM buffer 1, address offsets 0x1000-0x1FFF
20 MS	Main/spare region locator. In the case that FBCR[BC] = 0, MS is treated as 0. 0 Data is transferred to/from the main region of the FCM buffer; that is, the first 2048 bytes of the buffer are used as the starting address. 1 Data is transferred to/from the spare region of the FCM buffer; that is, the second 2048 bytes of the buffer are used as the starting address, but only an initial 64 bytes of spare region are defined.
21–31 CI	Column index. CI indexes the first byte to transfer to/from the main or spare region of the NAND Flash EEPROM and corresponding transfer buffer. In the case that FBCR[BC] = 0, CI is treated as 0. For MS = 0, CI can range 0x000-0x7FF; for MS = 1, CI can range 0x000-0x03F.

13.3.29 Flash page address register [Small Page Device (ORx[PGS] = 0)] (eLBC_FPARs)

The local bus Flash page address register (FPAR) locates the current NAND Flash page in both the external NAND Flash device and FCM buffer RAM.

Address: 12_4000h base + F0h offset = 12_40F0h

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

eLBC_FPARs field descriptions

Field	Description
0–16 -	This field is reserved. Reserved
17–21 PI	Page index. PI indexes the page in NAND Flash EEPROM at the current block defined by FBAR, and locates the corresponding transfer buffer in the FCM buffer RAM. The 3 LSBs of PI index one of the eight 1 Kbyte buffers in the FCM buffer RAM as follows: 000 The page is transferred to/from FCM buffer 0, address offsets 0x0000-0x03FF 001 The page is transferred to/from FCM buffer 1, address offsets 0x0400-0x07FF 010 The page is transferred to/from FCM buffer 2, address offsets 0x0800-0x0BFF 011 The page is transferred to/from FCM buffer 3, address offsets 0x0C00-0x0FFF 100 The page is transferred to/from FCM buffer 4, address offsets 0x1000-0x13FF 101 The page is transferred to/from FCM buffer 5, address offsets 0x1400-0x17FF 110 The page is transferred to/from FCM buffer 6, address offsets 0x1800-0x1BFF 111 The page is transferred to/from FCM buffer 7, address offsets 0x1C00-0x1FFF
22 MS	Main/spare region locator. In the case that FBCR[BC] = 0, MS is treated as 0. 0 Data is transferred to/from the main region of the FCM buffer; that is, the first 512 bytes of the buffer are used as the starting address. 1 Data is transferred to/from the spare region of the FCM buffer; that is, the second 512 bytes of the buffer are used as the starting address, but only an initial 16 bytes of spare region are defined.
23–31 CI	Column index. CI indexes the first byte to transfer to/from the main or spare region of the NAND Flash EEPROM and corresponding transfer buffer. In the case that FBCR[BC] = 0, CI is treated as 0. For MS = 0, CI can range 0x000-0x1FF; for MS = 1, CI can range 0x000-0x00F.

13.3.30 Flash byte count register (eLBC_FBCR)

The local bus Flash byte count register (FBCR) defines the size of FCM block transfers for reads and writes to the NAND Flash EEPROM.

Address: 12_4000h base + F4h offset = 12_40F4h

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

Reset 0

eLBC_FBCR field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–31 BC	Byte count determines how many bytes are transferred by the FCM during data read (RB) or data write (WB) opcodes. The first byte accessed in the NAND Flash EEPROM is located by the FPAR register, and successive bytes are transferred until either BC bytes have been counted, or the end of the spare region of the currently addressed Flash page has been reached. If BC = 0, an entire Flash page and its spare region will be transferred by FCM, in which case FPAR[MS] and FPAR[CI] are treated as zero regardless of their values. BC = 0 is the only setting that permits FCM to generate and check ECC.

13.3.31 Flash ECC block n registers (eLBC_FECCn)

The local bus Flash ECC block n register (FECC n) specifies the ECC value calculated during writes or reads by eLBC. It can be used for verify after write feature in software. Note that the valid bit sets before the command completion event and hence the correct ECC could be read before actual completion of writes/reads.

Address: 12_4000h base + 100h offset + (4d \times i), where i=0d to 3d

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

eLBC_FECCn field descriptions

Field	Description
0 V	Valid bit. This bit denotes that the ECC stored in this register is valid. It is set for full page write/read transfers if ECC generation/checking is enabled in BRn[DECC].
1–7 -	This field is reserved. Reserved
8–31 ECC	24 bit ECC; For n^{th} 512 bytes of a page in case of large page or for $(4k + n)^{\text{th}}$ 512 byte page for small page where $k = 0, 1, 2, \dots$. It stores calculated ECC value during writes/reads.

13.4 eLBC functional description

The eLBC allows the implementation of memory systems with very specific timing requirements.

- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. It has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading from NVRAM or NOR Flash, and access to low-performance memory-mapped peripherals.
- The FCM interfaces the eLBC to NAND Flash EEPROMs with 8-bit data bus. The FCM has an automatic boot-loading feature that allows the CPU to boot from high

density EEPROM, loading the boot block into 4 Kbytes of RAM for execution of the first level boot code. Following boot, FCM provides a flexible instruction sequencer that allows a user-defined command, address, and data transfer sequence of up to 8 steps to be executed against a memory-mapped buffer RAM. Programmable set-up time, hold time, and wait states permit the FCM to maximize the performance of NAND Flash block transfers, which can proceed in parallel with software processing of the multiple RAM buffers. A single-pass ECC engine in the FCM permits zero-overhead error checking, reporting, and correction in both boot blocks and page data transfers if enabled.

- The UPM supports refresh timers, address multiplexing of the external bus, and generation of programmable control signals for row address and column address strobes, to allow for a minimal glue logic interface to DRAMs, burstable SRAMs, and almost any other kind of peripheral with asynchronous timing or single data rate clocking. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

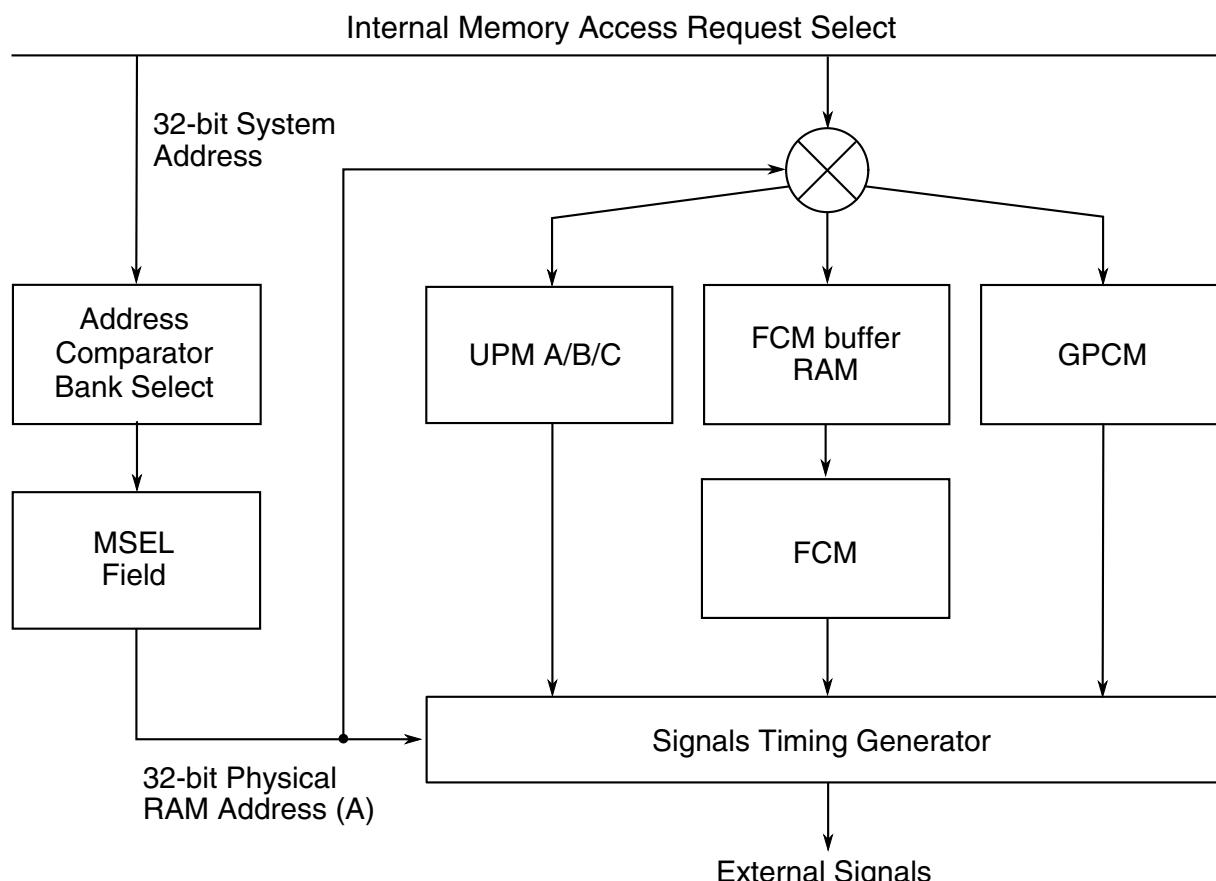


Figure 13-68. Basic operation of memory controllers in the eLBC

Each memory bank (chip select) can be assigned to any of these three types of machines through the machine select bits of the base register for that bank ($BR_n[MSEL]$), as illustrated in the above figure. If a bank match occurs, the corresponding machine (GPCM, FCM, or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

NOTE

Different machines (FCM/GPCM/UPM) share the address, data, and control signals. While the eLBC is servicing a transaction, subsequent transactions are queued until the current transaction has completed.

13.4.1 Basic architecture

The following subsections describe the basic architecture of the eLBC.

13.4.1.1 Address and address space checking

The defined base addresses are written to the BR_n registers, while the corresponding address masks are written to the OR_n registers.

Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 17 MSBs of the address, masked by $OR_n[AM]$, with the base address for each bank ($BR_n[BA]$). If a match is found on a memory controller bank, the attributes defined in the BR_n and OR_n for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

13.4.1.2 External address latch enable signal (LALE)

The local bus uses a multiplexed address/data bus.

Therefore the eLBC must distinguish between address and data phases, which take place on the same bus (LAD pins). The LALE signal, when asserted, signifies an address phase during which the eLBC drives the memory address on the LAD pins. An external address latch uses this signal to capture the address and provide it to the address pins of the memory or peripheral device. When LALE is negated, LAD then serves as the (bi-directional) data bus for the access. Any address phase initiates the assertion of LALE, which has a programmable duration of between 1 and 4 bus clock cycles.

To ensure adequate hold time on the external address latch, LALE negates earlier than the address changes on LAD during address phases. By default, LALE negates earlier by 1 platform clock period. For example, if the platform clock is operating at 533 MHz, then 1.8 ns of address hold time is introduced. However, at higher frequencies, the duration of the shortened LALE pulse may not meet the minimum latch enable pulse width specifications of some latches. In such cases, setting LBCR[AHD] = 1 increases the LALE pulse width by $\frac{1}{2}$ platform clock cycle, but decreases the address hold time by the same amount. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the ORn[EAD] and LCRR[EADC] fields, and the LBCR[AHD] bit can be left at 0. However, this will add latency to all address tenures.

The frequency of LALE assertion varies across the three memory controllers:

- For GPCM, every assertion of $LCSn_B$ is considered an independent access, and accordingly, LALE asserts prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and $LCSn_B$ 32 times in order to satisfy a 32-byte cache line transfer.
- For FCM, LALE asserts prior to each multi-command operation sequence, but LALE can be ignored on NAND Flash EEPROM accesses as the signal does *not* enable address latching in such devices. The value on the LAD and LA pins during LALE assertion is driven low-impedance, but otherwise not defined for FCM banks.
- In the case of UPM, the frequency of LALE assertion depends on how the UPM RAM is programmed. UPM single accesses typically assert LALE once, upon commencement, but it is possible to program UPM to assert LALE several times, and to change the values of LA_n with and without LALE being involved.

In general, when using the GPCM controller it is typically required to use both LA and the latched version of LAD to capture the entire address during LALE phases. The UPMs may require LA if the eLBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the eLBC, the following figure shows eLBC signals for the GPCM performing a 32-byte write starting at address 0x1234_5420. Note that during each of the 32 assertions of LALE, LA[16:31] and LAD[0:15] together drive the address, but during data phases, only LAD[0:7] and LDP[0] are driven with valid data and parity, respectively.

If the RCW is not loaded by the eLBC (for example, I2C or hard-coded options are used), then LALE is at an unknown value until the PLL is locked and should be ignored until the negation of HRESET.

13.4.1.3 Data transfer acknowledge (TA)

The three memory controllers in the eLBC generate an internal transfer acknowledge signal, TA, to allow data on LAD to be either sampled (for reads) or changed (on writes).

The data sampling/data change always occurs at the end of the bus cycle in which the eLBC asserts TA internally. The GPCM controller automatically generates TA according to the timing parameters programmed for them in the option and mode registers; FCM generates TA whenever data read and write instructions are executed out of register FIR; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set. [Figure 13-69](#) shows LALE, TA (internal), and LCSn_B . Note that TA and LALE are never asserted together, and that for the duration of LALE, LCSn_B (or any other control signal) remains negated or frozen.

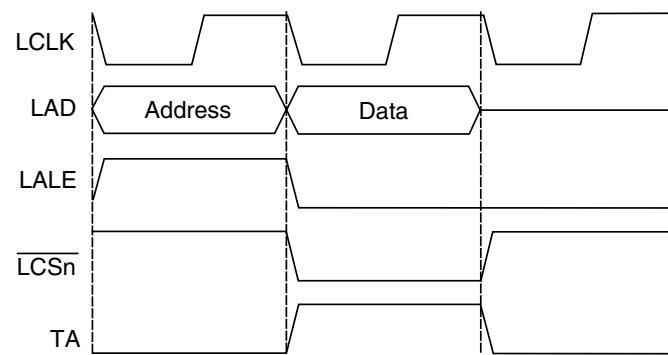


Figure 13-69. Basic eLBC bus cycle with LALE, TA, and LCSn_B (PLL bypass)

13.4.1.4 Data buffer control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL).

This signal is activated when a GPCM-, FCM-, or UPM-controlled bank is accessed. LBCTL can be disabled by setting ORn[BCTLD]. LBCTL can be further configured by LBCR[BCTLC] to act as an extra LWE_B or an extra LOE_B signal when in GPCM mode.

If LBCTL is configured as a data buffer control (LBCR[BCTLC] = 00), the signal is asserted (high) on the rising edge of the bus clock on the first cycle of the memory controller operation, coincident with LALE. If the access is a write, LBCTL remains high for the whole duration. However, if the access is a read, LBCTL is negated (low) with the negation of LALE so that the memory device is able to drive the bus. If back-to-back read accesses are pending, LBCTL is asserted (high) one bus clock cycle before the next transaction starts (that is, one bus clock cycle before LALE) to allow a whole bus cycle for the bus to turn around before the next address is driven.

13.4.1.5 Parity generation and checking (LDP)

Parity can be configured for any GPCM or UPM bank by programming BR_n[DECC]. Parity is generated and checked on a per-byte basis using LDP[0:1] for the bank if BR_n[DECC] = 01 (normal parity). Byte lane parity on LDP[0:1] is generated regardless of the BR_n[DECC] setting.

FCM calculates an ECC over 512-byte blocks, and hence does not use the LDP[0:1] pins. The setting of BR_n[DECC] = 01 enables ECC checking only, while BR_n[DECC] = 10 enables ECC generation and checking; in either case, LBCR[EPAR] determines the global type of block parity for ECC (odd or even).

13.4.1.6 Bus monitor

A bus monitor is provided to ensure that each transaction is terminated within a reasonable (user defined) period.

When a transaction starts, the bus monitor starts counting down from the time-out value (LBCR[BMT] x LBCR[BMTPS]) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the countdown until the data tenure completes and then idles if there is no pending transaction. Setting LTEDR[BMD] disables bus monitor error checking (that is, the LTESR[BM] bit is not set by a bus monitor time-out); however, the bus monitor is still active and can generate a UPM exception (as noted in [Exception requests](#),) or terminate a GPCM access.

It is very important to ensure that the value of LBCR[BMT] is not set too low; otherwise spurious bus time-outs may occur during normal operation-resulting in incomplete data transfers. Accordingly, the time-out value represented by the LBCR[BMT], LBCR[BMTPS] pair must not be set below 40 bus cycles for time-out under any circumstances.

NOTE

When the FCM is in the middle of a long transaction (such as NAND erase, write, and so on), another transaction on the GPCM or UPM will trigger the bus monitor to start, even though the GPCM or UPM is waiting for the FCM to finish. If the bus monitor times out, it could corrupt the current NAND Flash operation as well as terminate the GPCM or UPM operation. To avoid such cases, it is recommended that while using FCM the bus monitor timeout be programmed to its

maximum setting of LBCR[BMT] = 0 and LBCR[BMTPS] = 0xF.

13.4.1.7 PLL Bypass mode

In PLL bypass mode, eLBC drives new address, data, and control signals effectively on falling edges of LCLK, but continues to sample synchronous read data on rising edges of LCLK to maximize the set-up margin for reads.

NOTE

Because LCLK is not used for NAND Flash EEPROMs controlled by FCM, the eLBC drives and samples data on the falling edge on FCM controlled banks.

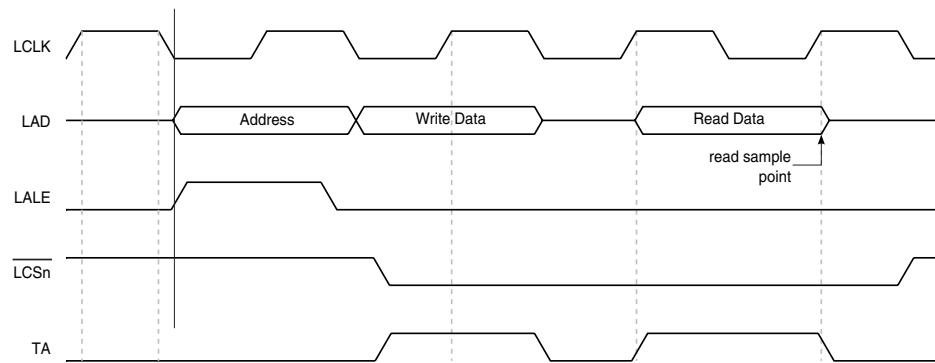


Figure 13-70. eLBC bus cycles in PLL-bypassed mode (GPCM and UPM only)

13.4.2 General-purpose chip-select machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, EEPROM, ROM devices, and external peripherals.

The GPCM contains two basic configuration register groups-BR n and OR n .

Byte-write enable signals (LWE_B) are available for each byte written to memory. Also, the output enable signal (LOE_B) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select (LCS0_B) prior to the system being fully configured.

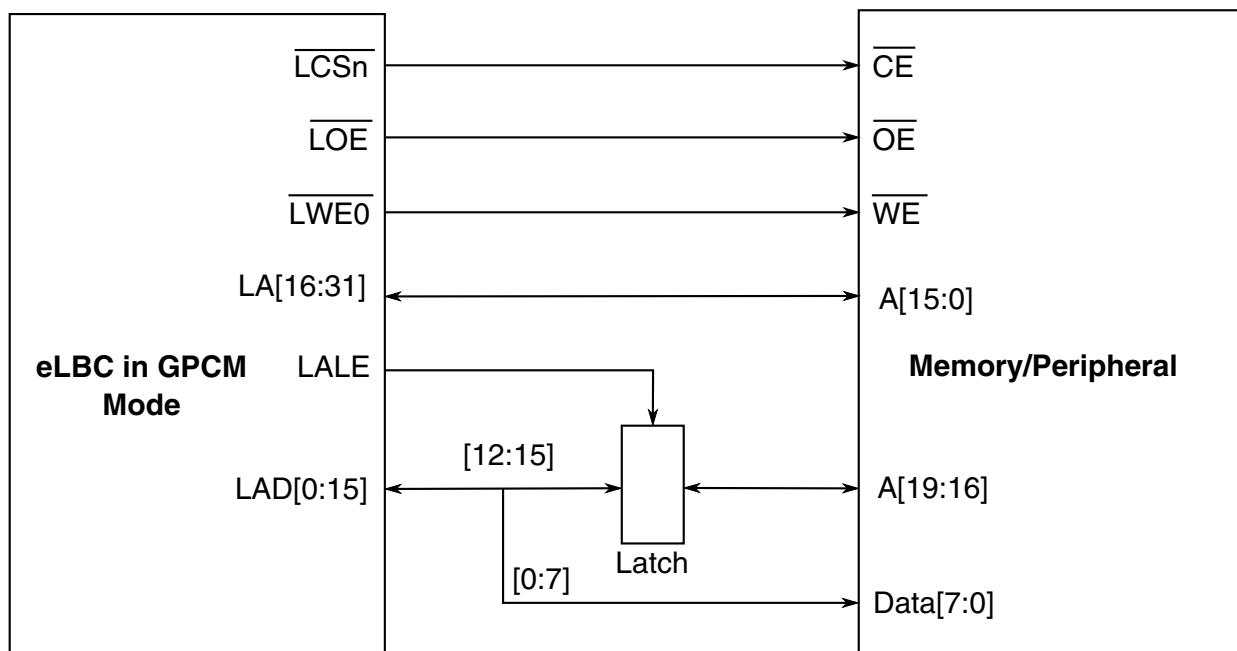


Figure 13-71. Enhanced local bus to GPCM device interface

The figure below shows **LCS_B** as defined by the setup time required between the address lines and **CE_B**. The user can configure **ORn[ACS]** to specify **LCS_B** to meet this requirement. Generally, the attributes for the memory cycle are taken from **ORn**. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR, and SETA fields.

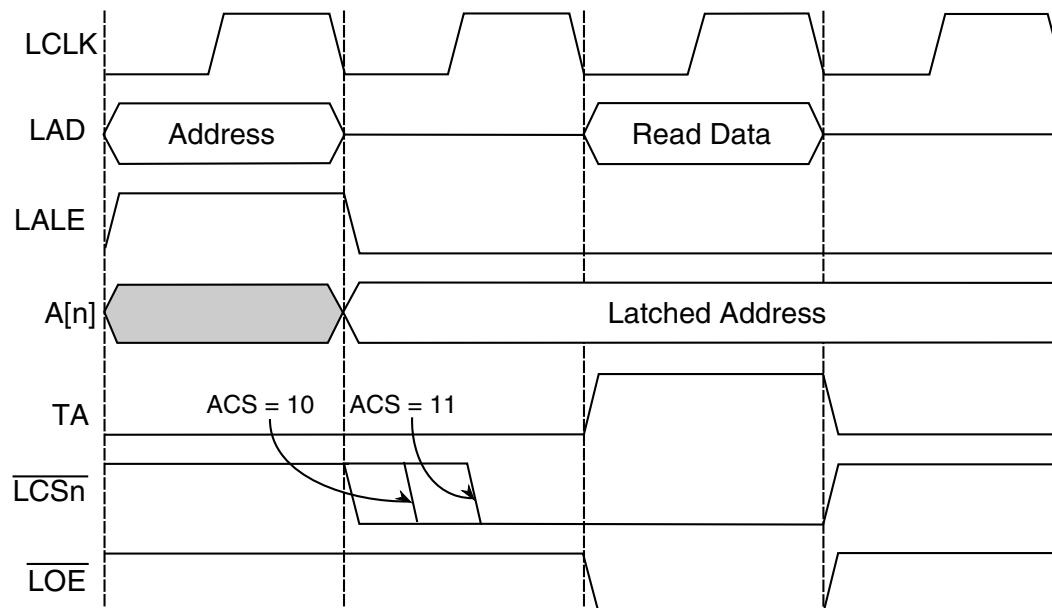
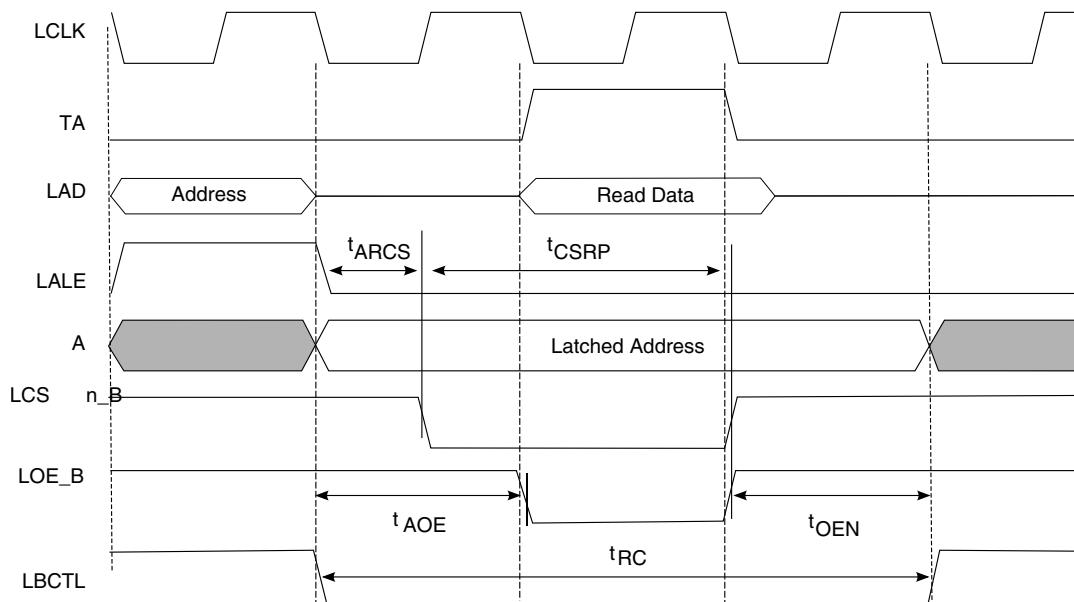


Figure 13-72. GPCM basic read timing (XACS = 0, ACS = 1x, TRLX = 0)

13.4.2.1 GPCM read signal timing

The basic GPCM read timing parameters that may be set by the ORn attributes are shown below.

The read access cycle commences upon latching of the memory address (LALE negated), and concludes when LBCTL returns high to turn the local bus around for a subsequent address phase. Read data is captured on the rising edge of LCLK when TA is asserted. LOE_B and LCSn_B negate high simultaneously, in some cases before the end of the read access to provide additional hold time for the external memory.



Notes:

t_{RC} = Read cycle time

t_{CSRP} = Read chip-select assertion period.

t_{ARCS} = Address valid to read chip-select time.

t_{OEN} = Output enable negated time.

t_{AOE} = Address valid to output enable time.

Figure 13-73. GPCM general read timing parameters with PLL bypass

The table below lists the signal timing parameters for a GPCM read access as the option register attributes are varied.

Table 13-191. GPCM read control signal timing

Option Register Attributes				Signal Timing (LCLK clock cycles)				
TRLX	EHTR	XACS	ACS	t_{ARCS}	t_{CSRP}	t_{AOE}	t_{OEN}	t_{RC}
0	0	0	0X	0	2 + SCY	1	0	2 + SCY
0	0	0	10	$\frac{1}{4}$	$1\frac{3}{4} + SCY$	1	0	2 + SCY
0	0	0	11	$\frac{1}{2}$	$1\frac{1}{2} + SCY$	1	0	2 + SCY
0	0	1	0X	0	2 + SCY	1	0	2 + SCY
0	0	1	10	1	$1 + SCY$	1	0	2 + SCY
0	0	1	11	2	$1 + SCY$	2	0	$3 + SCY$

Table continues on the next page...

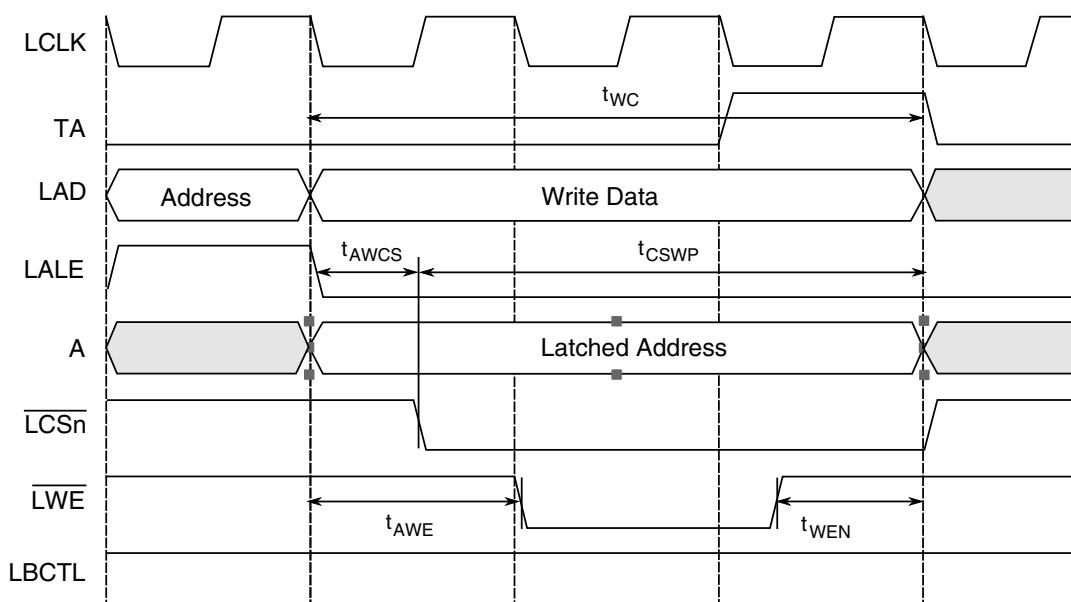
Table 13-191. GPCM read control signal timing (continued)

Option Register Attributes				Signal Timing (LCLK clock cycles)				
TRLX	EHTR	XACS	ACS	t _{ARCS}	t _{CSR} P	t _{AOE}	t _{OEN}	t _{RC}
0	1	0	0X	0	2 + SCY	1	1	3 + SCY
0	1	0	10	½	1¾ + SCY	1	1	3 + SCY
0	1	0	11	½	1½ + SCY	1	1	3 + SCY
0	1	1	0X	0	2 + SCY	1	1	3 + SCY
0	1	1	10	1	1 + SCY	1	1	3 + SCY
0	1	1	11	2	1 + SCY	2	1	4 + SCY
1	0	0	0X	0	2 + 2 x SCY	1	4	6 + 2 x SCY
1	0	0	10	1¼	1¾ + 2 x SCY	2	4	7 + 2 x SCY
1	0	0	11	1½	1½ + 2 x SCY	2	4	7 + 2 x SCY
1	0	1	0X	0	2 + 2 x SCY	1	4	6 + 2 x SCY
1	0	1	10	2	1 + 2 x SCY	2	4	7 + 2 x SCY
1	0	1	11	3	1 + 2 x SCY	3	4	8 + 2 x SCY
1	1	0	0X	0	2 + 2 x SCY	1	8	10 + 2 x SCY
1	1	0	10	1¼	1¾ + 2 x SCY	2	8	11 + 2 x SCY
1	1	0	11	1½	1½ + 2 x SCY	2	8	11 + 2 x SCY
1	1	1	0X	0	2 + 2 x SCY	1	8	10 + 2 x SCY
1	1	1	10	2	1 + 2 x SCY	2	8	11 + 2 x SCY
1	1	1	11	3	1 + 2 x SCY	3	8	12 + 2 x SCY

13.4.2.2 GPCM write signal timing

The basic GPCM write timing parameters that may be set by the OR_n attributes are shown in the figure below.

The write access cycle commences upon latching of the memory address (LALE negated), and concludes when LCS_n_B returns high. LBCTL remains stable for the entire cycle to drive data onto any secondary data bus. Write data becomes invalid following the falling edge of TA. LWE_B may, in some cases, negate high before the end of the write access to provide additional hold time for the external memory.

**Notes:** t_{WC} = Write cycle time. t_{CSWP} = Write chip-select assertion period t_{AWCS} = Address valid to write chip-select time. t_{WEN} = Write enable negated time wrt t_{AWE} = Address valid to write enable time. chip-select negation time**Figure 13-74. GPCM general write timing parameters****Table 13-192. GPCM write control signal timing**

Option register attributes				Signal Timing (LCLK clock cycles)				
TRLX	XACS	ACS	CSNT	t_{AWCS}	t_{CSWP}	t_{AWE}	t_{WEN}	t_{WC}
0	0	00	0	0	2 + SCY	1	0	2 + SCY
0	0	10	0	1/4	1 3/4 + SCY	1	0	2 + SCY
0	0	11	0	1/2	1 1/2 + SCY	1	0	2 + SCY
0	1	00	0	0	2 + SCY	1	0	2 + SCY
0	1	10	0	1	1 + SCY	1	0	2 + SCY
0	1	11	0	2	1 + SCY	2	0	3 + SCY
0	0	00	1	0	2 + SCY	1	1/4	2 + SCY
0	0	10	1	1/4	1 1/2 + SCY	1	0	1 3/4 + SCY
0	0	11	1	1/2	1 1/4 + SCY	1	0	1 3/4 + SCY
0	1	00	1	0	2 + SCY	1	1/4	2 + SCY
0	1	10	1	1	3/4 + SCY	1	0	1 3/4 + SCY
0	1	11	1	2	3/4 + SCY	2	0	2 3/4 + SCY
1	0	00	0	0	2 + 2 x SCY	1	0	2 + 2 x SCY
1	0	10	0	1 1/4	1 3/4 + 2 x SCY	2	0	3 + 2 x SCY
1	0	11	0	1 1/2	1 1/2 + 2 x SCY	2	0	3 + 2 x SCY
1	1	00	0	0	2 + 2 x SCY	1	0	2 + 2 x SCY
1	1	10	0	2	1 + 2 x SCY	2	0	3 + 2 x SCY
1	1	11	0	3	1 + 2 x SCY	3	0	4 + 2 x SCY

Table continues on the next page...

Table 13-192. GPCM write control signal timing (continued)

Option register attributes				Signal Timing (LCLK clock cycles)				
TRLX	XACS	ACS	CSNT	t _{AWCS}	t _{CSPW}	t _{AWE}	t _{WEN}	t _{WC}
1	0	00	1	0	3 + 2 x SCY	1	1¼	3 + 2 x SCY
1	0	10	1	1¼	1½ + 2 x SCY	2	0	2¾ + 2 x SCY
1	0	11	1	1½	1¼ + 2 x SCY	2	0	2¾ + 2 x SCY
1	1	00	1	0	3 + 2 x SCY	1	1¼	3 + 2 x SCY
1	1	10	1	2	¾ + 2 x SCY	2	0	2¾ + 2 x SCY
1	1	11	1	3	¾ + 2 x SCY	3	0	3¾ + 2 x SCY

13.4.2.3 Chip-select assertion timing

The banks selected to work with the GPCM support an option to drive the LCS_B_n signal with different timings (with respect to the external address/data bus).

LCS_B_n can be driven in any of the following ways:

- Simultaneous with the latched memory address. (This refers to the externally latched address and not the address timing on LAD. That is, the chip select does not assert during LALE.)
- One quarter of a clock cycle later.
- One half of a clock cycle later.
- One clock cycle later , when ORn[XACS] = 1.
- Two clock cycles later, when ORn[XACS] = 1.
- Three clock cycles later, when ORn[XACS] = 1 and ORn[TRLX] = 1.

The timing diagram in [Figure 13-72](#) shows two chip-select assertion timings.

13.4.2.3.1 Programmable wait state configuration

The GPCM supports internal generation of transfer acknowledge.

It allows between zero and 30 wait states to be added to an access by programming ORn[SCY] and ORn[TRLX]. Internal generation of transfer acknowledge is enabled if ORn[SETA] = 0. If LGTA_B is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by LGTA_B; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of ORn[SETA], wait states prolong the assertion duration of both LOE_B and LWE_B_n in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from ORn[SCY] cycles to 2 x ORn[SCY] cycles, allowing a maximum of 30 wait states.

13.4.2.3.2 Chip-select and write enable negation timing

The figure below shows a basic connection between the local bus and a static memory device.

In this case, LCS_B n is connected directly to CE_B of the memory device. The LWE_B[0:1] signals are connected to the respective WE_B[1:0] signals on the memory device where each LWE_B[0:1] signal corresponds to a different data byte.

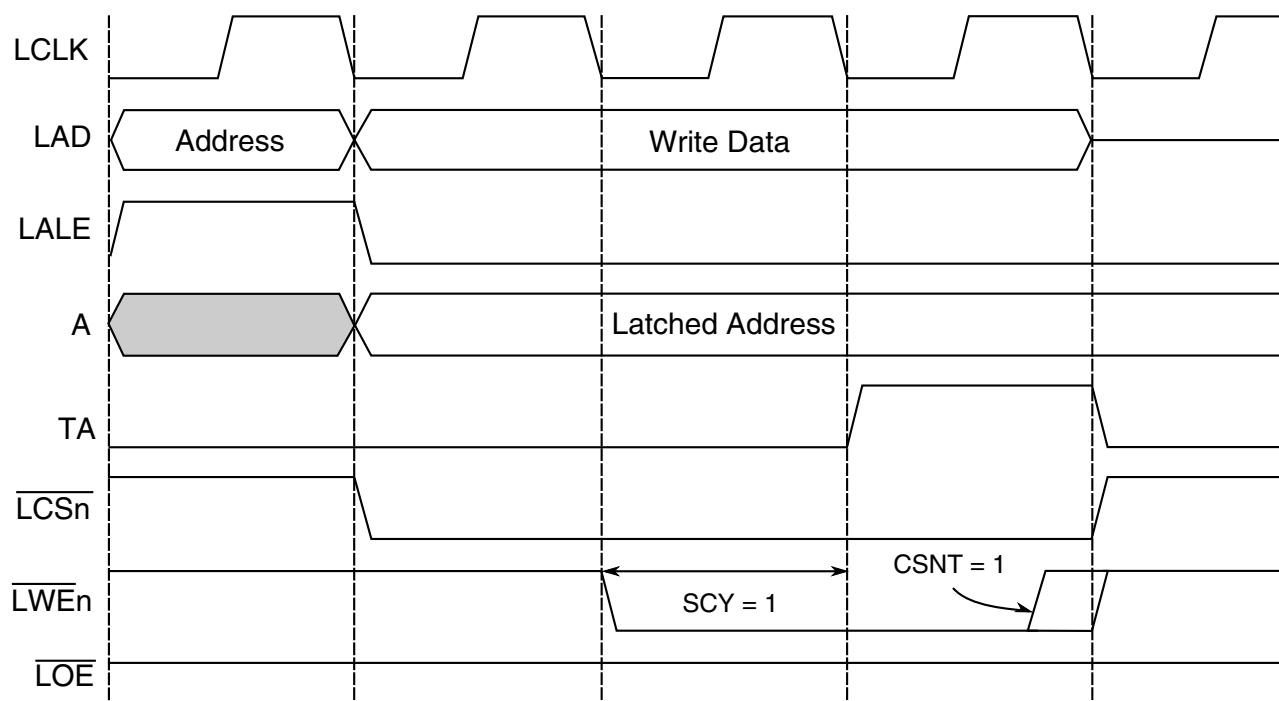


Figure 13-75. GPCM basic write timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0)

The timing for LCS_B n is the same as for the latched address. The strobes for the transaction are supplied by LOE_B or LWE_B n, depending on the transaction direction-read or write (write case shown in the figure). ORn[CSNT], along with ORn[TRLX], control the timing for the appropriate strobe negation in write cycles. When this attribute is asserted, the strobe is negated one quarter of a clock before the normal case. For example, when ACS = 00 and CSNT = 1, LWE_B n is negated one quarter of a clock earlier, as shown in [Figure 13-75](#).

1. LCS_B n is affected by CSNT and TRLX only if ACS[0] is non zero. However, LWE_B n is affected independent of ACS.
2. When CSNT attribute is asserted, the strobe is negated one quarter of a clock before the normal case.
3. TRLX = 1 in conjunction with CSNT = 1, negates the LCS_B n and LWE_B n 1 + 1/4 cycle earlier.

For example, when ACS = 00, CSNT = 1 and TRLX = 0, LWE_B_n is negated one quarter of a clock earlier and LCS_B_n is negated normally as shown in [Figure 13-75](#).

13.4.2.3.3 Relaxed timing

ORx[TRLX] is provided for memory systems that require more relaxed timing between signals.

Setting TRLX = 1 has the following effect on timing:

- An additional bus cycle is added between the address and control signals (but only if ACS is not equal to 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses (EHTR) is extended further.
- LCSn_B signals are negated one cycle earlier during writes (but only if ACS is not equal to 00).
- LWE_B[0:1] signals are negated one cycle earlier during writes.

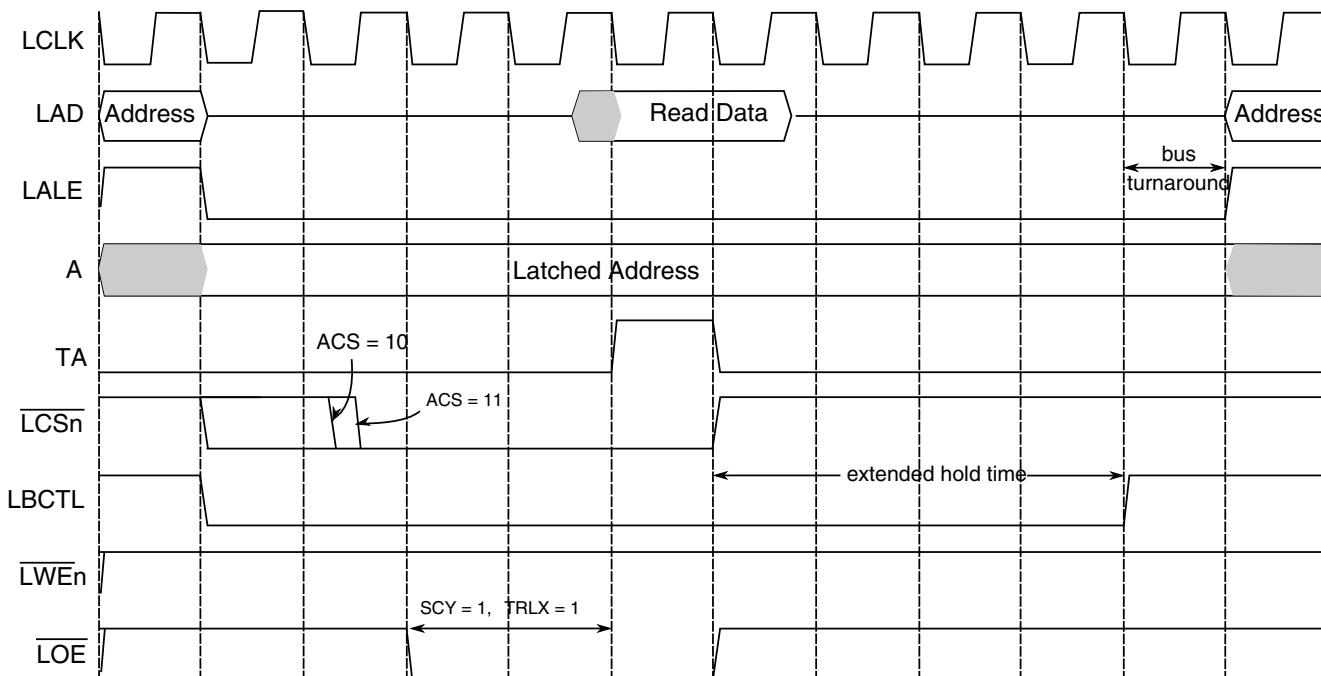


Figure 13-76. GPCM relaxed timing back-to-back reads with PLL bypass (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, EHTR = 0)

The example in [Figure 13-77](#) shows address and data multiplexing on LAD for a pair of writes issued consecutively.

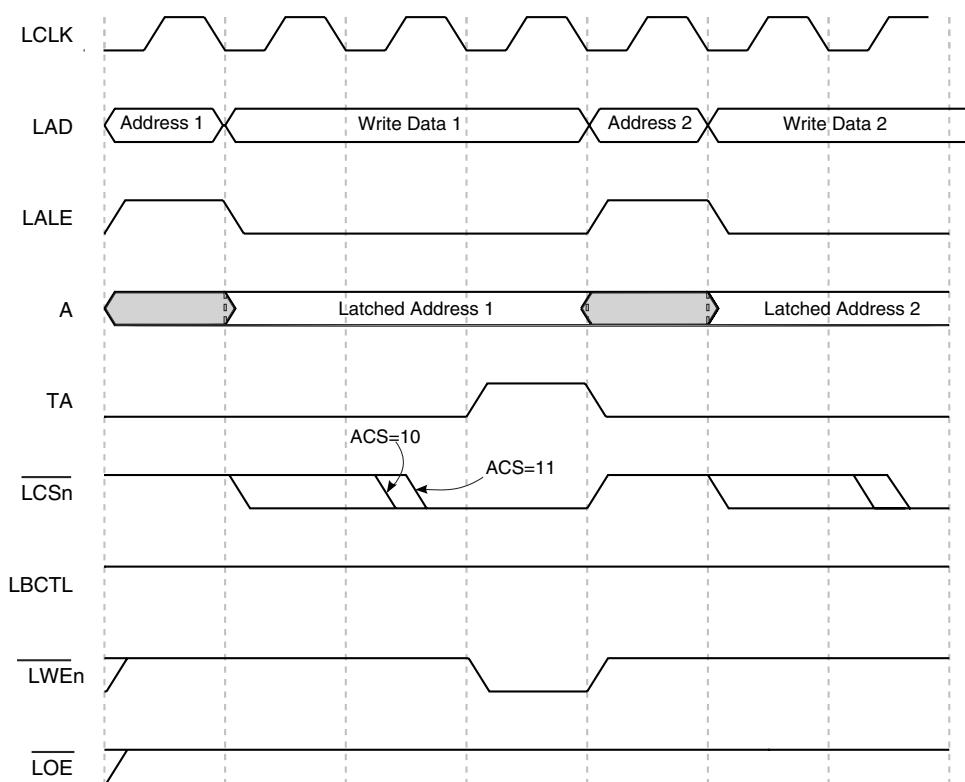


Figure 13-77. GPCM relaxed timing back-to-back writes with PLL bypass (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1)

When TRLX and CSNT are set in a write access, the LWE_B[0:1] strobe signals are negated one clock earlier than in the normal case. If ACS ≠ 00, LCSn_B is also negated one clock earlier.

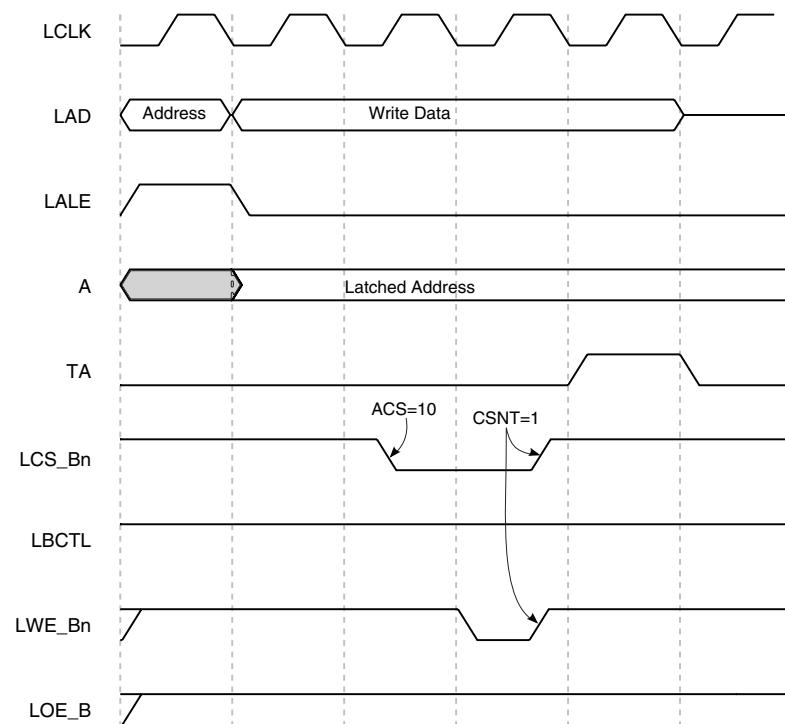


Figure 13-78. GPCM relaxed timing write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)

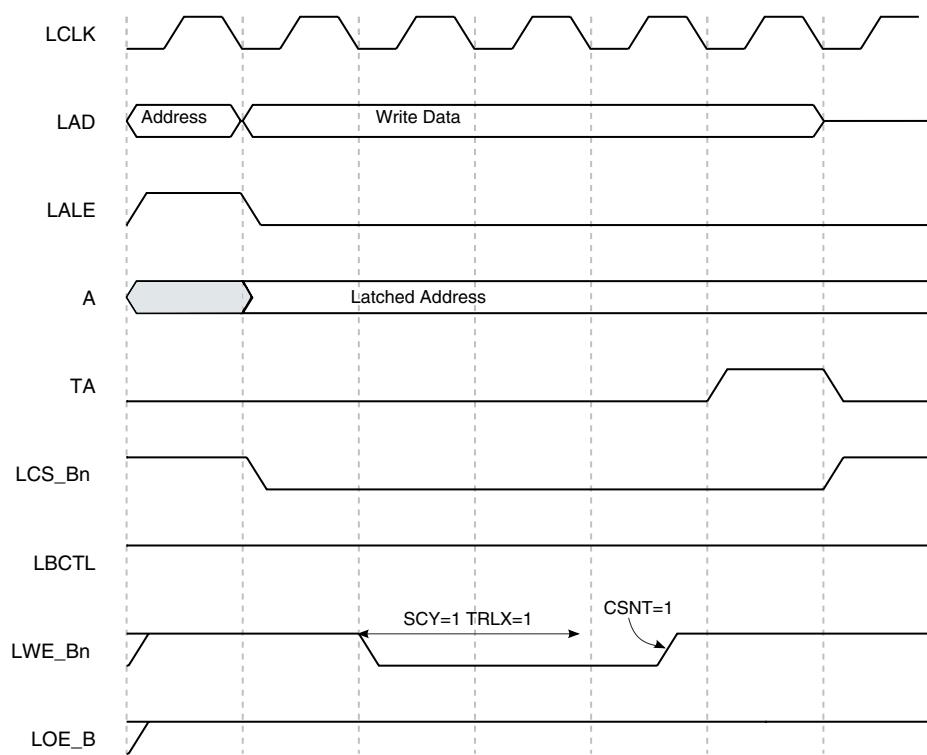


Figure 13-79. GPCM relaxed timing write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1)

13.4.2.3.4 Output enable (LOE_B) timing

The timing of the LOE_B is affected only by TRLX.

It always asserts and negates on the rising edge of the bus clock. LOE_B asserts either on the rising edge of the bus clock after LCSn_B is asserted or coinciding with LCSn_B (if XACS = 1 and ACS = 10 or ACS = 11). Accordingly, assertion of LOE_B can be delayed (along with the assertion of LCSn_B by programming TRLX = 1. LOE_B negates on the rising clock edge coinciding with LCSn_B negation

13.4.2.3.5 Extended hold time on read accesses-GPCM

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of ORn[TRLX,EHTR].

Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in [Options register 0 layout for GPCM Mode \(eLBC_ORg0\)](#) in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the eLBC for reads, regardless of the setting of ORn[EHTR].

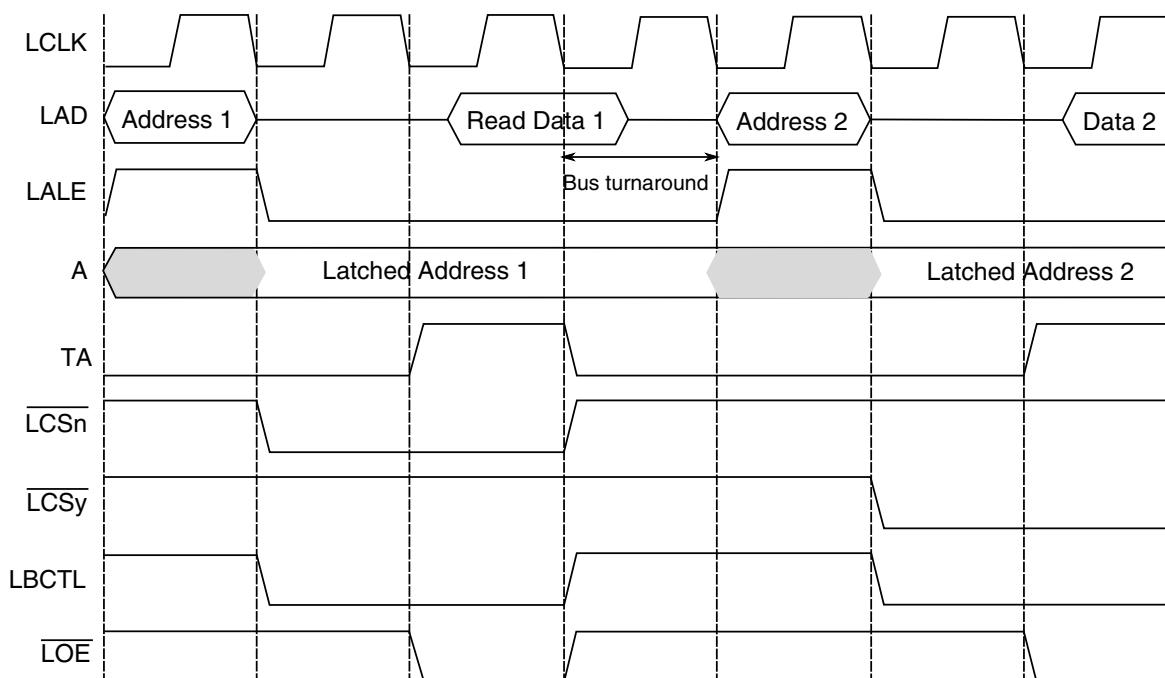


Figure 13-80. GPCM read followed by read (TRLX = 0, EHTR = 0, fastest timing)

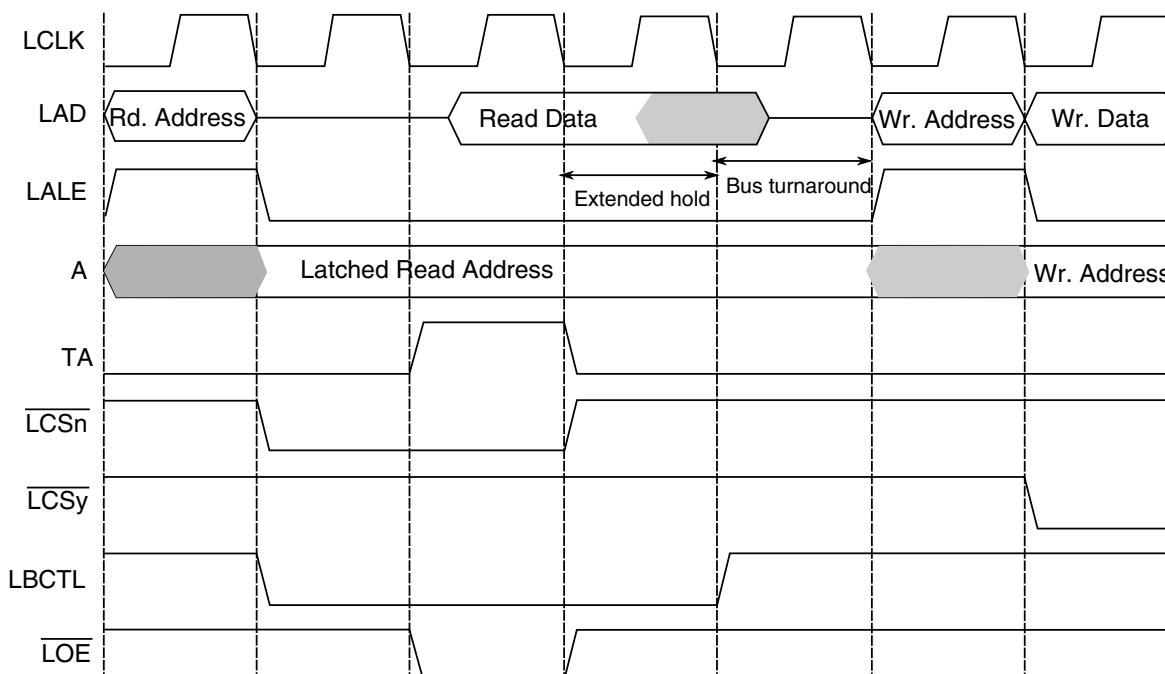


Figure 13-81. GPCM read followed by write (TRLX = 0, EHTR = 1, one-cycle extended hold time on reads)

13.4.2.4 External access termination (LGTA_B)

External access termination is supported by the GPCM using the asynchronous LGTA_B input signal, which is synchronized and sampled internally by the local bus.

If, during assertion of LCSn_B, the sampled LGTA_B signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of ORn[SETA]). LGTA_B must be asserted for at least two bus cycles to be effective. Note that because LGTA_B is synchronized, bus termination occurs two cycles after LGTA_B assertion, so in case of read cycle, the device still must drive data as long as LOE_B is asserted.

The user selects whether transfer acknowledge is generated internally or externally (LGTA_B) by programming ORn[SETA]. Asserting LGTA_B always terminates an access, even if ORn[SETA] = 0 (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if ORn[SETA] = 1.

In PLL bypass mode, the timing of LGTA_B is illustrated by the example in the figure below.

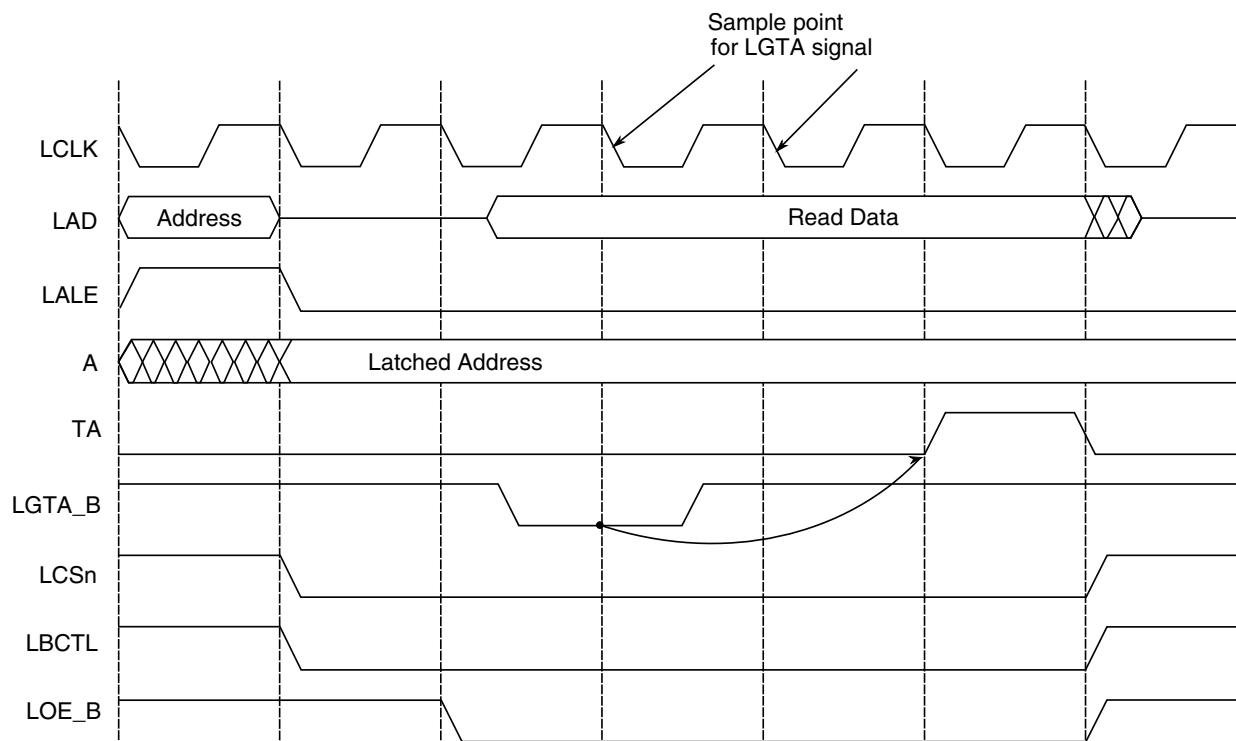


Figure 13-82. External termination of GPCM access (PLL bypass mode)

13.4.2.5 GPCM boot chip-select operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization.

LCS0_B is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, LCS0_B is asserted for every local bus access until BR0 or OR0 is reconfigured.

The boot chip-select also provides a programmable port size, which is configured during reset. The boot chip-select does not provide write protection. LCS0_B operates this way until the first write to OR0 and it can be used as any other chip-select register after the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only with a hardware reset.

Table 13-193. Boot bank field values after reset for GPCM as boot controller

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	PS	From RCW[BOOT_LOC]
	DECC	00
	WP	0
	MSEL	000
	-	-
	V	1
OR0	AM	0000_0000_0000_0000_0
	BCTL0	0
	CSNT	1
	ACS	11
	XACS	1
	SCY	1111
	SETA	0
	TRLX	1
	EHTR	1
	EAD	1

13.4.3 Flash control machine (FCM)

The FCM provides a glueless interface to parallel-bus NAND Flash EEPROM devices.

The FCM contains three basic configuration register groups-BRn, ORn, and FMR.

The figure below shows a simple connection between an 8-bit port size NAND Flash EEPROM and the eLBC in FCM mode. Commands, address bytes, and data are all transferred on LAD[0:7]⁸, with LFWE_B asserted for transfers written to the device, or LFRE_B asserted for transfers read from the device. eLBC signals LFCLE and LFALE determine whether writes are of type command (only LFCLE asserted), address (only LFALE asserted), or write data (neither LFCLE nor LFALE asserted). The NAND Flash RDY/BSY_B pin is normally open-drain, and should be pulled high by a 4.7-KΩ resistor. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select (LCS0_B) prior to the system being fully configured.

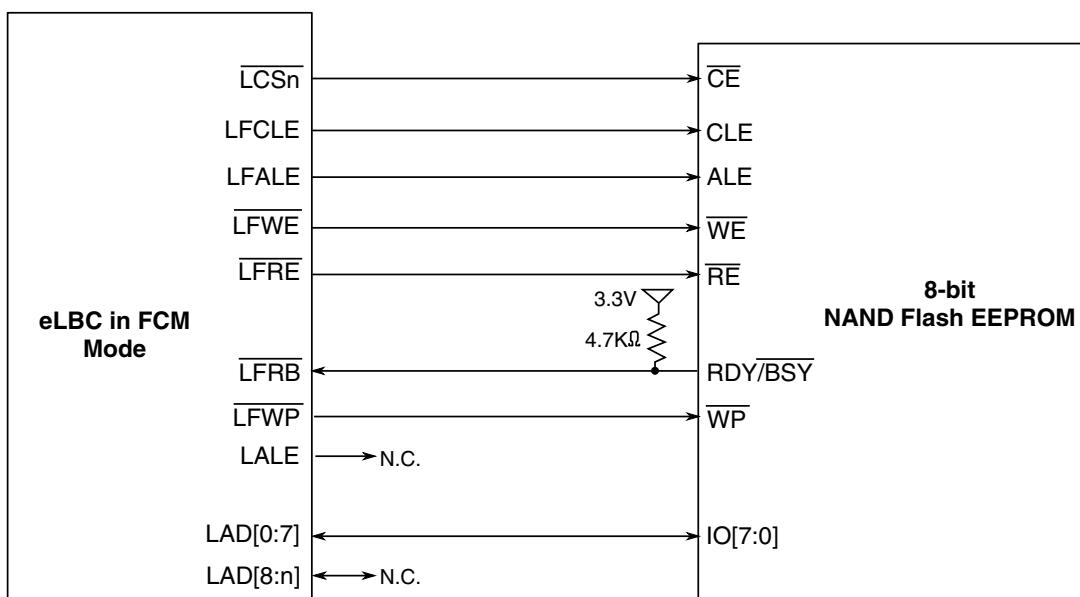


Figure 13-83. Local bus to 8-bit FCM device interface

Although LCLK is shown for reference, NAND Flash EEPROMs do not make use of the clock.

8. Note bit numbering reversal: LAD[0] (msb) connects to Flash IO[7], while LAD[7] (lsb) connects to IO[0].

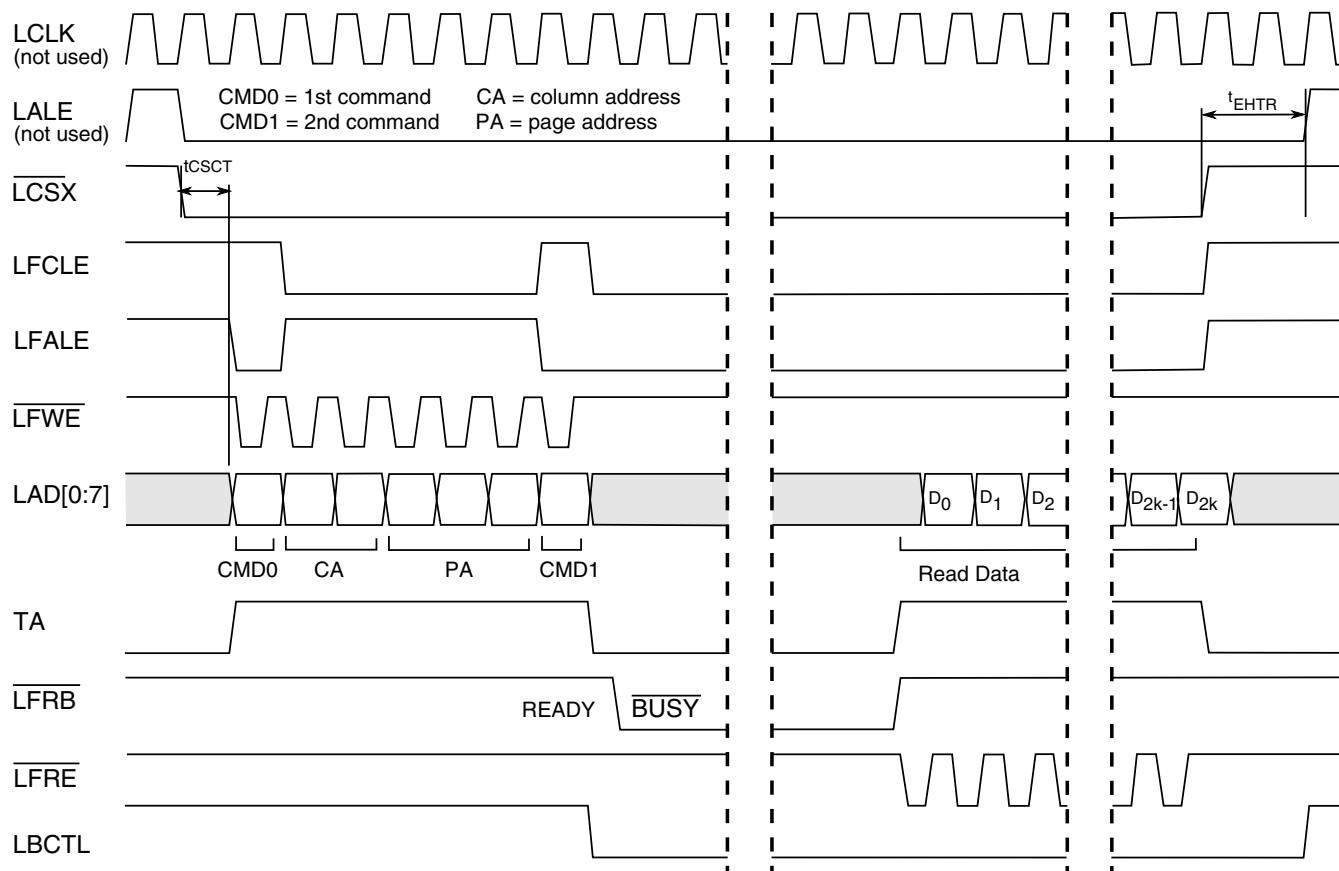


Figure 13-84. FCM basic page read timing (PGS = 1, CSCT = 0, CST = 0, CHT = 1, RST = 1, SCY = 0, TRLX = 0, EHTR = 1)

Following the assertion of LALE, FCM asserts LCSn_B to commence a command sequence to the Flash device. After a delay of t_{CSCT}, the first command can be written to the device on assertion of LFWE_B, followed by any parameters (typically address bytes and data), and concluded with a secondary command. In many cases, the second command initiates a long-running operation inside the Flash device, which pulls the wired-OR pin LFRB_B low to indicate that the device is busy. Since in Figure 13-84 FCM is now expecting a read response, it takes LBCTL low to turnaround any bus transceivers that are present. Upon LFRB_B indicating ready status, FCM asserts LFRE_B repeatedly to recover bytes of read data, and the bytes are stored in eLBC's FCM buffer RAM while an ECC is optionally computed on the bytes transferred. Finally, FCM negates LCSn_B and delays eLBC by t_{EHTR} before any subsequent memory access occurs.

13.4.3.1 FCM buffer RAM

Read and write accesses to eLBC banks controlled by FCM do not access attached NAND Flash EEPROMs directly.

Rather, these accesses read and write the FCM buffer RAM-a single, shared 8-Kbyte space internal to eLBC and mapped by the base address of every FCM bank. Even though each FCM-controlled bank will have a different base address to differentiate it, all accesses to such banks will access the same buffer space. External eLBC signals, such as LALE and LCSn_B, will not assert upon accesses to the buffer RAM. The FCM buffer RAM is logically divided into two or more buffers, depending on the setting of ORn[PGS], with different buffers being accessible concurrently by software and FCM.

To perform a page read operation from a NAND Flash device, software initializes the FCM command, mode, and address registers, before issuing a special operation (FMR[OP] set non-zero) to a particular FCM-controlled bank. FCM will execute the sequence of op-codes held in FIR, reading data from the Flash device into the shared buffer RAM. While this read is taking place, software is free to access any data stored in other, currently inactive buffers of the FCM buffer RAM through reads or writes to any bank controlled by FCM. If command completion interrupts are enabled, an event interrupt will be generated once FCM has completed the read. When FCM has completed its last command, software can switch to the newly read buffer and issue further commands.

To perform a page write operation, software first prepares data to be written in a fresh buffer. Then, the FCM command, mode, and address registers are initialized, and a special operation (FMR[OP] set non-zero) is issued to a particular FCM-controlled bank. FCM will execute the sequence of op-codes held in FIR, writing data from shared buffer RAM to the Flash device. To ensure that the device is enabled for programming, software must initialize FMR[OP] = 11, which prevents assertion of LFWP_B during the write. While this write is taking place, software is free to access any data stored in other, currently inactive buffers of the FCM buffer RAM through reads or writes to any bank controlled by FCM. When FCM has completed its last command, software can re-use the previously written buffer and issue further commands.

See [Boot block loading into the FCM buffer RAM](#) for a description of the shared buffer RAM layout during boot.

13.4.3.1.1 Buffer layout and page mapping for small-page NAND flash devices

The FCM buffer space is divided into eight 1-Kbyte buffers for small-page devices (ORn[PGS] = 0), mapped as shown in [Figure 13-85](#).

Each page in a small-page NAND Flash comprises 528 bytes, where 512 bytes appear as main region data, and 16 bytes appear as spare region data. The EEPROM's page numbered P is associated with buffer number $(P \bmod 8)$, where $P = \text{FPAR}[PI]$. Since the bank size set by ORn[AM] will be greater than 8 Kbytes, an identical image of the FCM

buffer RAM appears replicated every 8 Kbytes throughout the bank address space. It is recommended that the bank size be set to 32 Kbytes, which covers a single NAND Flash block for small-page devices.

For FCM commands, register FPAR sets the page address and, therefore, also the buffer number. In the case that FBCR[BC] = 0, FCM transfers an entire page, comprising the 512-byte main region followed by the 16-byte spare region; the 496-byte reserved region is not accessible, and remains undefined for software. However, for commands given a specific byte count in FBCR[BC], FPAR[MS] locates the starting address in either the main region (MS = 0) or the spare region (MS = 1). Where different eLBC banks control both small and large-page devices, a large-page 4-Kbyte buffer must be assigned to either the first 4 or last 4 small-page buffers.

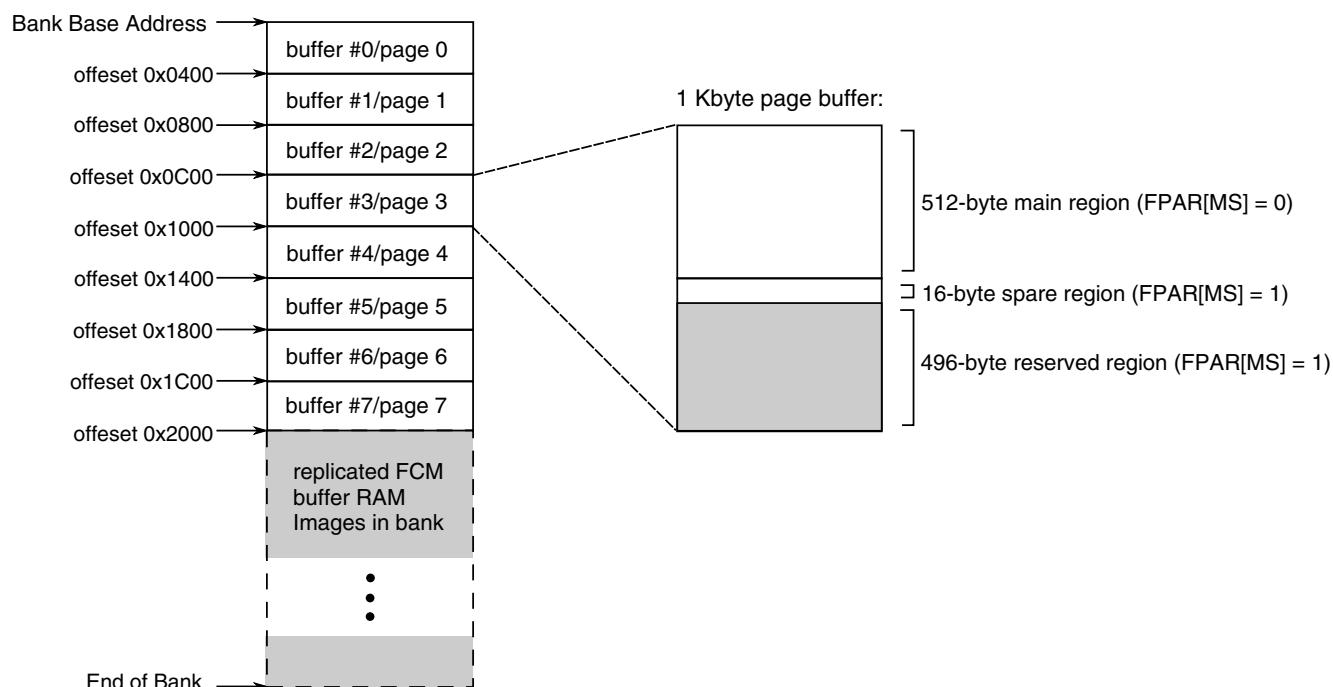


Figure 13-85. FCM buffer RAM memory map for small-page (512-byte page) NAND flash devices

13.4.3.1.2 Buffer layout and page mapping for large-page NAND flash devices

The FCM buffer space is divided into two 4 Kbyte buffers for large-page devices ($ORn[PGS] = 1$), mapped as shown in [Figure 13-86](#).

Each page in a large-page NAND Flash comprises 2112 bytes, where 2048 bytes appear as main region data, and 64 bytes appear as spare region data. The EEPROM's page numbered P is associated with buffer number $(P \bmod 2)$, where $P = FPAR[PI]$. Since the bank size set by $ORn[AM]$ will be greater than 8 Kbytes, an identical image of the FCM

buffer RAM appears replicated every 8 Kbytes throughout the bank address space. It is recommended that the bank size be set to 256 Kbytes, which covers a single NAND Flash block for large-page devices.

For FCM commands, register FPAR sets the page address and, therefore, also the buffer number. In the case that FBCR[BC] = 0, FCM transfers an entire page, comprising the 2048-byte main region followed by the 64-byte spare region; the 1984-byte reserved region is not accessed, and remains undefined for software. However, for commands given a specific byte count in FBCR[BC], FPAR[MS] locates the starting address in either the main region (MS = 0) or the spare region (MS = 1). Where different eLBC banks control both small and large-page devices, a large-page 4 Kbyte buffer must be assigned to either the first 4 or last 4 small-page buffers.

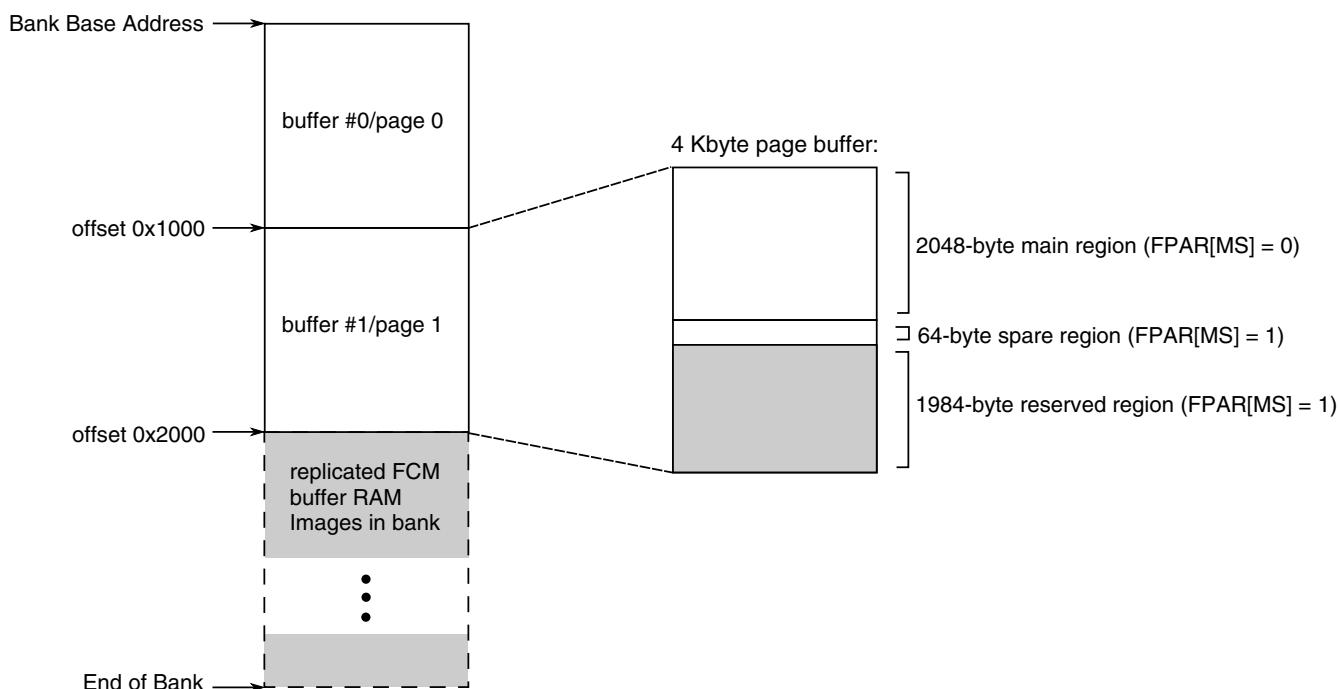


Figure 13-86. FCM buffer RAM memory map for large-page (2-Kbyte page) NAND flash devices

13.4.3.1.3 Error correcting codes and the spare region

The FCM's ECC engine makes use of data in the NAND Flash spare region to store pre-computed ECC code words.

ECC is calculated in a single pass over blocks of 512 bytes of data in the main region. The setting of FMR[ECCM] determines the location of the 24-bit ECC in the spare region.

The basic ECC algorithm is depicted in [Figure 13-87](#). The stream of data bytes is considered to form a matrix having 8 columns (corresponding with the device bus IO[7:0] or IO[15:8]) and 512 rows (corresponding with each byte in the ECC block). Six bits of parity, $\{P_4, P_4', P_2, P_2', P_1, P_1'\}$, are calculated across the columns, and at most 18 bits of parity $\{P_{2048}, P_{2048}', \dots, P_{16}, P_{16}', P_8, P_8'\}$ are calculated across the rows to create a 24-bit Hamming code for the data block. In this calculation, parity bit P_N' is the exclusive-OR of every alternate N -bit group of bits positioned at even intervals (starting at N -bit group 0, then continuing to group 2, 4, and so on), while parity bit P_N is the exclusive-OR of every alternate N -bit group of bits positioned at odd intervals (starting at N -bit group 1, then continuing to group 3, 5, and so on).

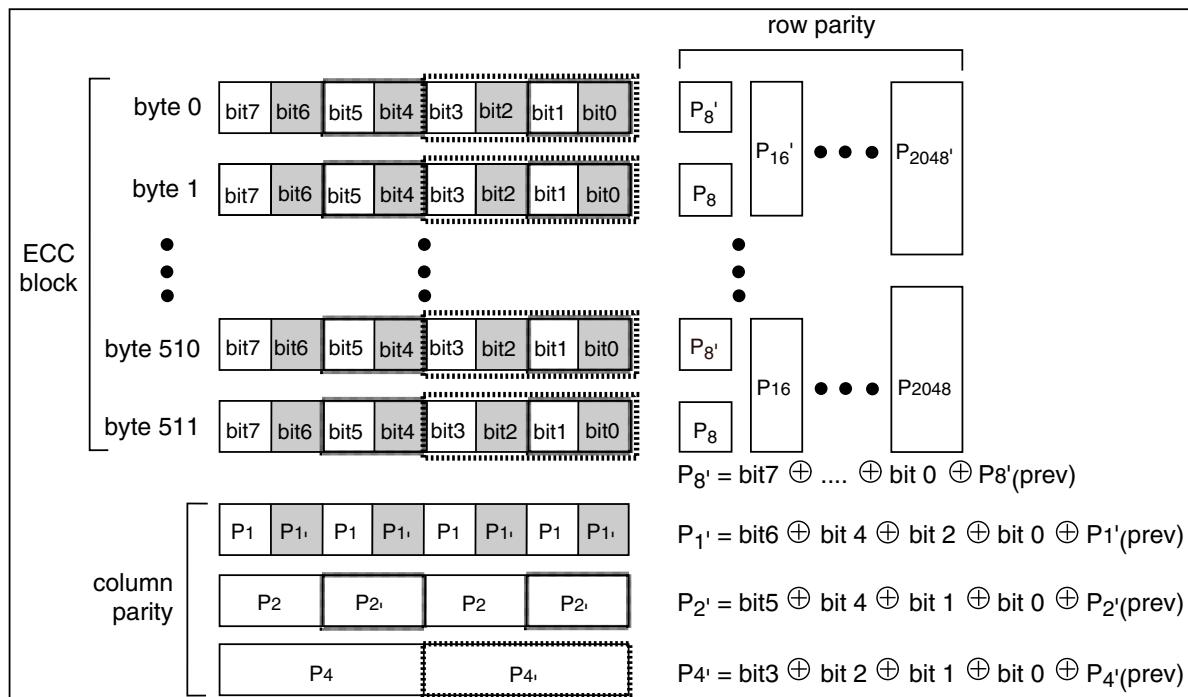


Figure 13-87. FCM ECC calculation

The 24-bit ECC code word format is shown in [Table 13-194](#) for normal ECC polarity. Setting LBCR[EPAR] = 1 changes ECC polarity, and thus omits negation of each P_N and P_N' bit.

Table 13-194. ECC layout for LBCR[EPAR] = 0 (~ represents logical negation)

	0 (MSB)	1	2	3	4	5	6	7 (LSB)
EC0	$\sim P_{64}$	$\sim P_{64}'$	$\sim P_{32}$	$\sim P_{32}'$	$\sim P_{16}$	$\sim P_{16}'$	$\sim P_8$	$\sim P_8'$
EC1	$\sim P_{1024}$	$\sim P_{1024}'$	$\sim P_{512}$	$\sim P_{512}'$	$\sim P_{256}$	$\sim P_{256}'$	$\sim P_{128}$	$\sim P_{128}'$
EC2	$\sim P_4$	$\sim P_4'$	$\sim P_2$	$\sim P_2'$	$\sim P_1$	$\sim P_1'$	$\sim P_{2048}$	$\sim P_{2048}'$

The placement of ECC code words in relation to FMR[ECCM] is shown in [Table 13-195](#). For small-page devices, only a single 512-byte main region is ECC-protected. For large-page devices, there are four adjacent main regions, and each has a 16-byte spare region-of which only one is shown in the figure. If eLBC is configured to generate ECC ($BRn[DECC] = 10$), FCM will substitute on full-page write transfers the three code word bytes in place of the spare region data originally provided at the locations shown in [Table 13-195](#). Transfers shorter than a full page, however, require software to prepare the appropriate ECC in the spare region. Similarly, FCM can check and correct bit errors on full-page reads if $BRn[DECC] = 01$ or 10 . A correctable error is a single bit error in any 512-byte block of main region data, as judged by comparison of a regenerated ECC with the ECC retrieved from the spare region, or a single bit error in the retrieved ECC only. Bit errors in the main region are corrected before FCM completes its final read transfer and signals an event in LTESR[CC]. Errors that appear more complex (two or more bits in error per 512-byte block) are not corrected, but are flagged as parity errors by FCM. The bit vector in LTEATR[PB] can be checked to determine which 512-byte blocks in a large-page NAND Flash main region were found to be non-correctable.

Table 13-195. ECC placement in NAND flash spare regions in relation to FMR[ECCM]

ECCM	Byte 0	Byte 511	Other Mains	Spare 0	5	6	7	8	9	10	11	12	13	14	15
0	Main Region			-				EC0	EC1	EC2	-				
1	Main Region			-						EC0	EC1	EC2	-		

13.4.3.2 Programming FCM

FCM has a fully general command and data transfer sequencer that caters for both common and specific/proprietary NAND Flash command sequences.

The command sequencer reads a program out of the FIR register, which can hold up to 8 instructions, each represented by a 4-bit op-code, as illustrated in [Figure 13-88](#). The first instruction executed is read from FIR[OP0], the next is read from FIR[OP1], and likewise to subsequent instructions, ending at FIR[OP7] or until the only instructions remaining are NOPs. If FIR contains nothing but NOP instructions, FCM will not assert $LCSn_B$, otherwise, $LCSn_B$ is asserted prior to the first instruction and remains asserted until the last instruction has completed. If LTESR[CC] is enabled, completion of the last instruction will trigger a command completion event interrupt from eLBC.

Prior to executing a sequence, necessary operands for the instructions will need to be set in the FMR, FCR, MDR, FBCR, FBAR, and FPAR registers. The AS0-AS3 address and data pointers associated with FCM's use of MDR all reset to select AS0 at the start of the instruction sequence. A complete list of op-codes can be found in [Flash instruction register \(eLBC_FIR\)](#).

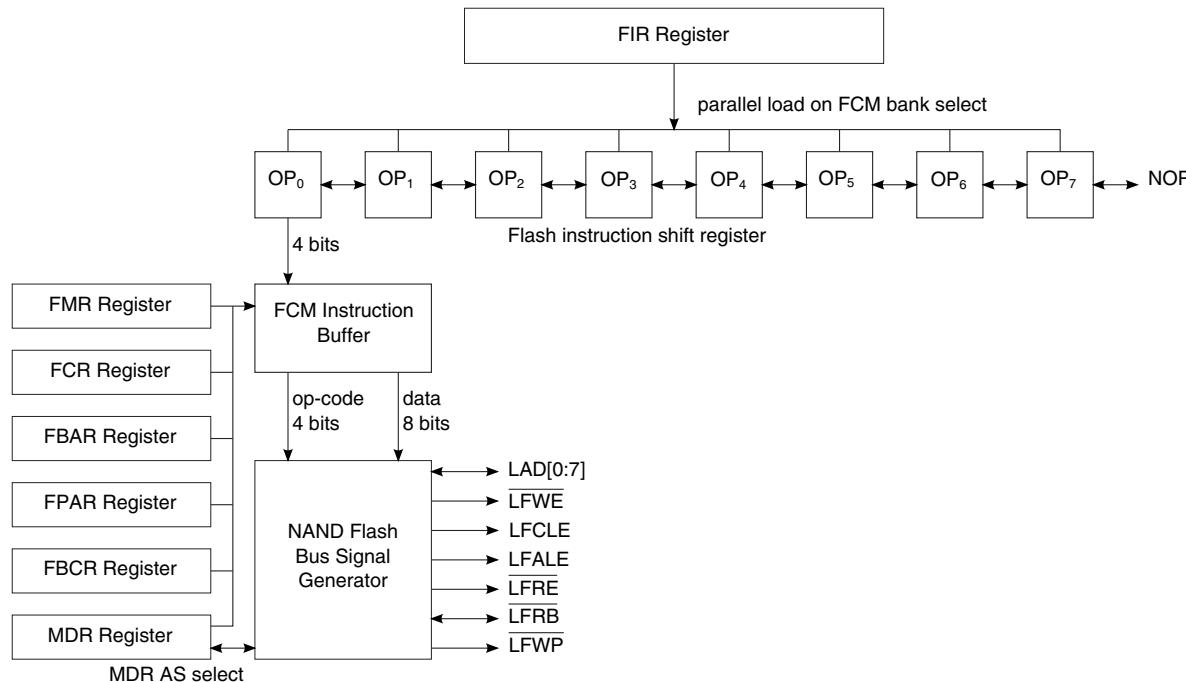


Figure 13-88. FCM instruction sequencer mechanism

13.4.3.2.1 FCM command instructions

There are two kinds of command instruction:

- Commands that issue immediately-CM0, CM1, CM2, and CM3. These commands write a single command byte by asserting LFCLE and LFWE_B while driving an 8-bit command onto LAD[0:7]. Op-code CM n sources its command byte from field FCR[CMD n], therefore up to four different commands can be issued in any FCM instruction sequence.
- Commands that wait for LFRB_B to be sampled high (EEPROM in ready state) before issuing-CW0, and CW1. These commands first poll the LFRB_B pin, waiting for it to go high, before writing a single command byte onto LAD[0:7], sourced from FCR[CMD n] for op-code CW n . It is necessary to use CW n op-codes whenever the EEPROM is expected to be in a busy state (such as following a page read, block erase, or program operation) and therefore initially unresponsive to commands. To avoid deadlock in cases where the device is already available, FCM does not expect a transition on LFRB_B. Rather, FCM waits for 8 x (2 + OR n [SCY]) clock cycles

(when $ORn[TRLX] = 0$) or $16 \times (2 + ORn[SCY])$ clock cycles (when $ORn[TRLX] = 1$) before sampling the level of LFRB_B. If the level of LFRB_B does not return high before a time-out set by FMR[CWTO] occurs, FCM proceeds to issue the command normally, and a FCT event is issued to LTESR.

The manufacturer's datasheet should be consulted to determine values for programming into the FCR register, and whether a given command in the sequence is expected to initiate busy device behavior.

13.4.3.2.2 FCM no-operation instruction

A NOP instruction that appears in FIR ahead of the last instruction is executed with the timing of a regular command instruction, but neither LFCLC nor LFWE_B are asserted.

Thus a NOP instruction may be used to insert a pause matching the time taken for a regular command write.

13.4.3.2.3 FCM address instructions

Address instructions are used to issue addresses to the NAND Flash EEPROM.

A complete device address is formed from a sequence of one or more bytes, each written onto LAD[0:7] with LFALE and LFWE_B asserted together. There are three kinds of address generation provided:

- Column address-CA. A column address comprises one byte ($ORn[PGS] = 0$) or two bytes ($ORn[PGS] = 1$) locating the starting byte or word to be transferred in the next page read or write sequence. FPAR[CI] sets the value of the column index provided that FBCR[BC] is non-zero. In the case that FBCR[BC] = 0, a column index of zero is issued to the device, regardless of the value in FPAR[CI].
- Page address-PA. A page address comprises 2, 3, or 4 bytes, depending on the setting of FMR[AL], and locates the data page in the NAND Flash address space. The complete page address is the concatenation of the block index, read from FBAR[BLK], with the page-in-block index, read from FPAR[PI]. The page address length set in FMR[AL] should correspond with the size of EEPROM being accessed. Similarly, the block index in FBAR[BLK] must not exceed the maximum block index for the device, as most devices require reserved address bits to be written as zero.
- User-defined address-UA. This instruction allows the FCM to write a user-defined address byte, which is read from the next AS field in MDR, starting at MDR[AS0]. Each subsequent UA instruction reads an adjacent AS field in MDR, until all four AS bytes (MDR[AS0], MDR[AS1], MDR[AS2], MDR[AS3]) have been sent; a fifth and any following UA instructions send zero as the address byte. Note that each UA

instruction advances the MDR pointer for writes by one byte, and therefore a mix of UA and WS instructions can consume adjacent bytes from MDR.

13.4.3.2.4 FCM data read instructions

Data read instructions assert LFRE_B repeatedly to transfer one or more bytes of read data from the NAND Flash EEPROM.

Data read instructions are distinguished by their data destination:

- Read data to buffer RAM immediately- RB. This instruction reads FBCR[BC] bytes of data into the current FCM RAM buffer addressed by FPAR. If FBCR[BC] = 0, an entire page (including spare region) is transferred in a burst, starting at the page boundary, and the ECC calculation is checked against the ECC stored in the spare region. Correctable ECC errors are corrected; other errors may cause an interrupt if enabled. If the value of FBCR[BC] takes the read pointer beyond the end of the spare region in the buffer, FCM discards any excess bytes read.
- Read data/status to MDR immediately- RS. This instruction asserts LFRE_B exactly once to read one byte (8-bit port size) of data into the next AS field of MDR. Reads beyond the fourth byte of MDR are discarded. The MDR read pointer is independent of the MDR write pointer used by UA and WS instructions.
- Read data to buffer RAM once waited on ready- RBW. This instruction first polls the LFRB_B pin, waiting for it to go high, before proceeding with a read to buffer as described for the RB instruction. Sampling and time-outs for polling the LFRB_B pin follow the behavior of CW_n instructions.
- Read data/status to MDR once waited on ready- RSW. This instruction first polls the LFRB_B pin, waiting for it to go high, before proceeding with a status read to MDR as described for the RS instruction. Sampling and time-outs for polling the LFRB_B pin follow the behavior of CW_n instructions.

13.4.3.2.5 FCM data write instructions

Data write instructions assert LFWE_B repeatedly (with LFCLE and LFALE both negated) to transfer one or more bytes of write data to the NAND Flash EEPROM.

Data write instructions are distinguished by their data source:

- Write data from FCM buffer RAM-WB. This instruction writes FBCR[BC] bytes of data from the current FCM RAM buffer addressed by FPAR. If FBCR[BC] = 0, an entire page (including spare region) is transferred in a burst, starting at the page boundary, and the ECC calculation is stored in the and spare region in accordance with the setting of FMR[ECCM]. If the value of FBCR[BC] takes the write pointer

beyond the end of the spare region in the buffer, the value of data written by FCM is undefined.

- Write data/status from MDR-WS. This instruction asserts LFWE_B exactly once to write one byte (8-bit port size) of data taken from the next AS field of MDR. Attempts to write beyond four bytes of MDR has the effect of writing zeros. The MDR write pointer is independent of the MDR read pointer used by RS and RSW instructions.

13.4.3.3 FCM signal timing

If BR_n[MSEL] selects the FCM, the attributes for the memory cycle are taken from OR_n. These attributes include the CSCT, CST, CHT, RST, SCY, TRLX, and EHTR fields.

13.4.3.3.1 FCM chip-select timing

The timing of LCS_n_B assertion in FCM mode is illustrated by the timing diagram in [Figure 13-84](#).

LCS_n_B is asserted immediately following LALE negation, and remains asserted until the last instruction in FIR has completed.

The delay, t_{CSCT}, between LCS_n_B assertion and commencement of the first NAND Flash instruction is controlled by OR_n[CSCT] and OR_n[TRLX], as shown in [Table 13-196](#). OR_n[CSCT] should be set in accordance with the NAND Flash EEPROM chip-select to WE_B set-up time specification.

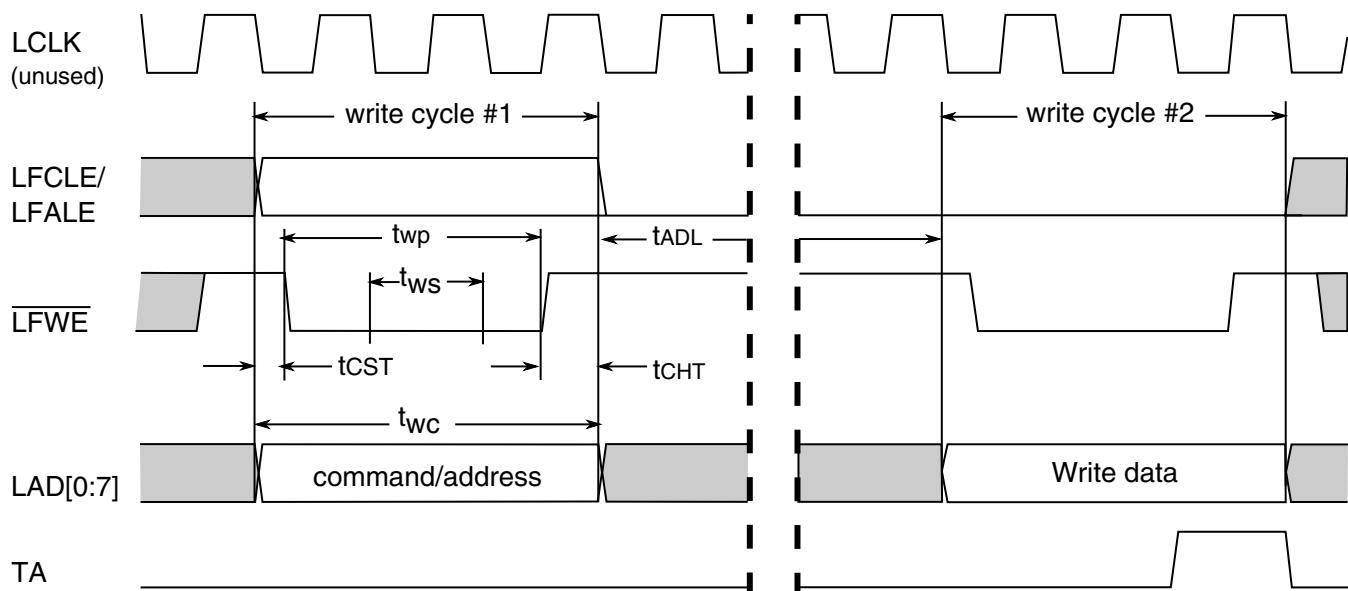
Table 13-196. FCM chip-select to first command timing

OR _n [TRLX]	OR _n [CSCT]	LCS _n _B to First Command Delay
0	0	1 LCLK clock cycle
0	1	4 LCLK clock cycles
1	0	2 LCLK clock cycles
1	1	8 LCLK clock cycles

13.4.3.3.2 FCM command, address, and write data timing

The FCM command (CM0-CM3, CW0, CW1), address (CA, PA, UA), and data write (WB, WS) instructions all share the same basic timing attributes.

Assertion of LFWE_B initiates transfer via LAD[0:7], and the options in OR_n for FCM mode establish the set-up, hold, and wait state timings with respect to LFWE_B, as shown in [Figure 13-89](#).



Notes:

- t_{CST} = Command to \overline{LFWE} set-up time.
- t_{CHT} = Command to \overline{LFWE} hold time.
- t_{ADL} = Command/address to write data delay.
- t_{WP} = \overline{LFWE} pulse time, driven low.
- t_{WS} = Command wait state time.
- t_{WC} = Command cycle time.

Figure 13-89. Timing of FCM command/address and write data cycles (for $TRLX = 0$, $CHT = 0$, $CST = 1$, $SCY = 1$)

Table 13-197. FCM command, address, and write data timing parameters

Option Register Attributes			Timing Parameter (LCLK Clock Cycles) ¹					
TRLX	CHT	CST	t_{CST}	t_{CHT}	t_{WS}	t_{WP}	t_{WC}	t_{ADL}
0	0	0	0	$\frac{1}{2}$	SCY	$1\frac{1}{2} + SCY$	$2 + SCY$	$4 \times (2 + SCY)$
0	0	1	$\frac{1}{4}$	$\frac{1}{2}$	SCY	$1\frac{1}{4} + SCY$	$2 + SCY$	$4 \times (2 + SCY)$
0	1	0	0	1	SCY	$1 + SCY$	$2 + SCY$	$4 \times (2 + SCY)$
0	1	1	$\frac{1}{4}$	1	SCY	$\frac{3}{4} + SCY$	$2 + SCY$	$4 \times (2 + SCY)$
1	0	0	$\frac{1}{2}$	$1\frac{1}{2}$	$2xSCY$	$1 + 2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$
1	0	1	1	$1\frac{1}{2}$	$2xSCY$	$\frac{1}{2} + 2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$
1	1	0	$\frac{1}{2}$	2	$2xSCY$	$\frac{1}{2} + 2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$
1	1	1	1	2	$2xSCY$	$2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$

1. In the parameters, SCY refers to a delay of ORn[SCY] clock cycles.

An example of minimum delay command timing appears in the figure below. Note that the set-up, wait-state, and hold timing of command, address, and write data cycles with respect to LFWE_B assertion are all identical, and that the minimum cycle extends for two LCLK clock cycles.

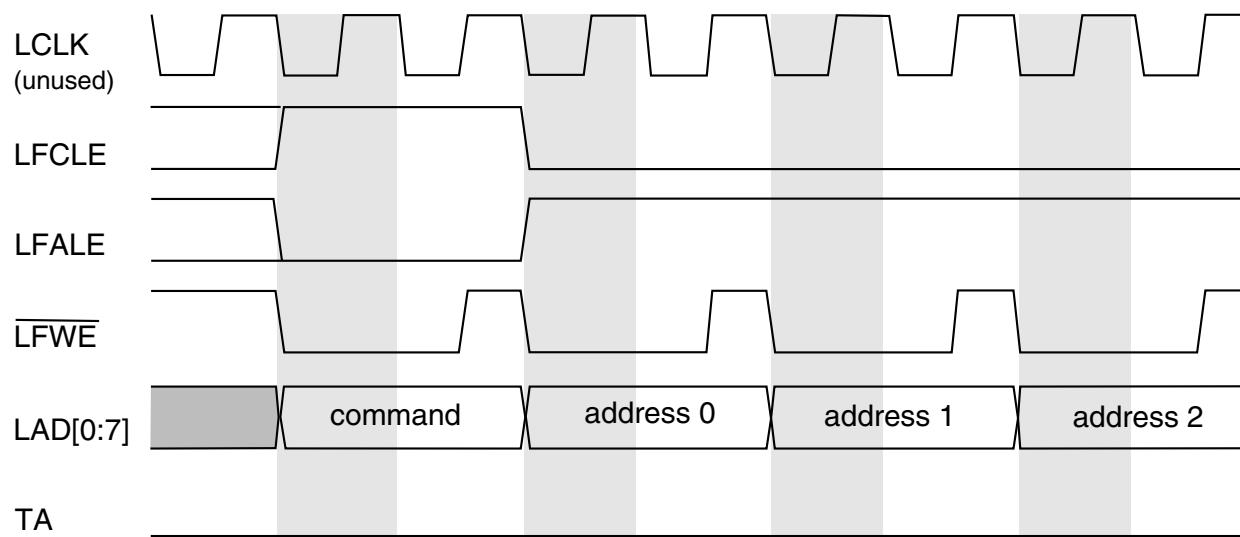


Figure 13-90. Example of FCM command and address timing with minimum delay parameters (for TRLX = 0, CHT = 0, CST = 0, SCY = 0)

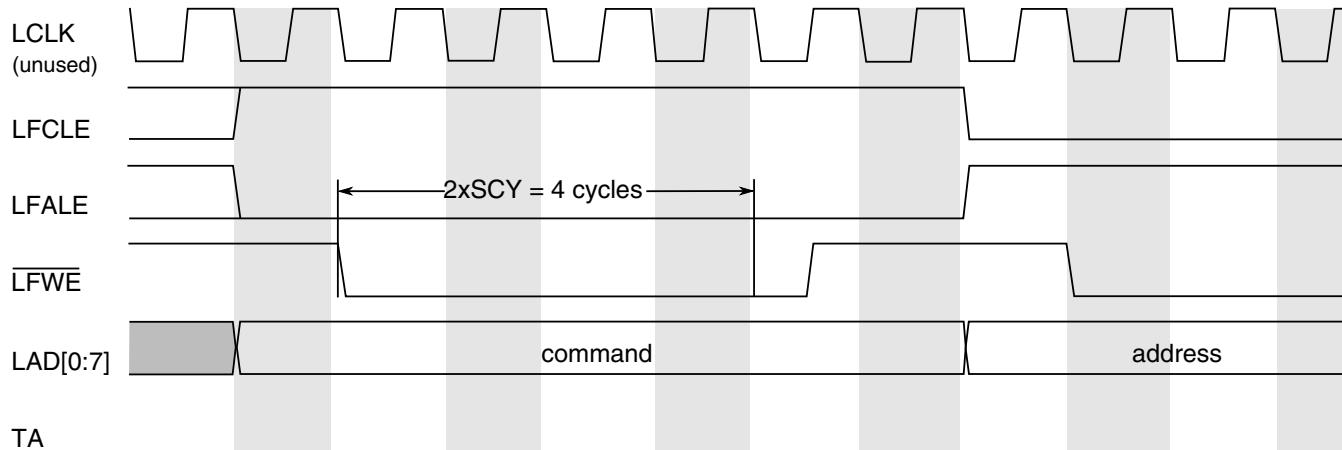


Figure 13-91. Example of FCM command and address timing with relaxed parameters (for TRLX = 1, CHT = 0, CST = 1, SCY = 2)

13.4.3.3.3 FCM ready/busy timing

Instructions CW0, CW1, RBW, and RSW force FCM to observe the state of the LFRB_B pin, which may be driven low by a long-latency NAND Flash operation, such as a page read.

Following the issue of such commands, FCM waits as shown in [Figure 13-92](#) before sampling the state of LFRB_B. This guards against observing LFRB_B before it has been properly driven low by the device, but does not preclude LFRB_B from remaining high after a command. In addition, FCM samples and compares the state of LFRB_B on two consecutive cycles of LCLK to filter out noise on this signal as it rises to the ready state (LFRB_B = 1).

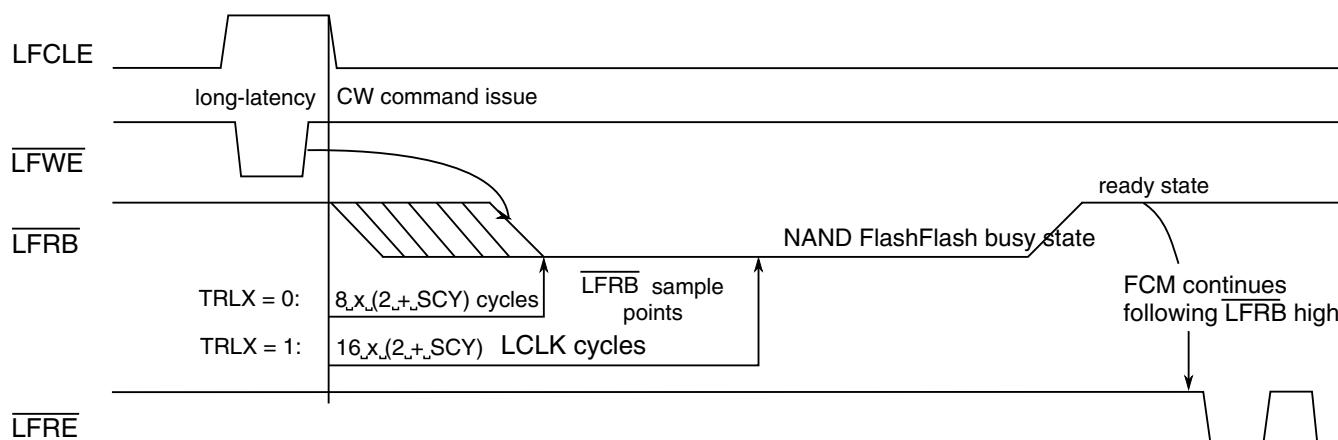


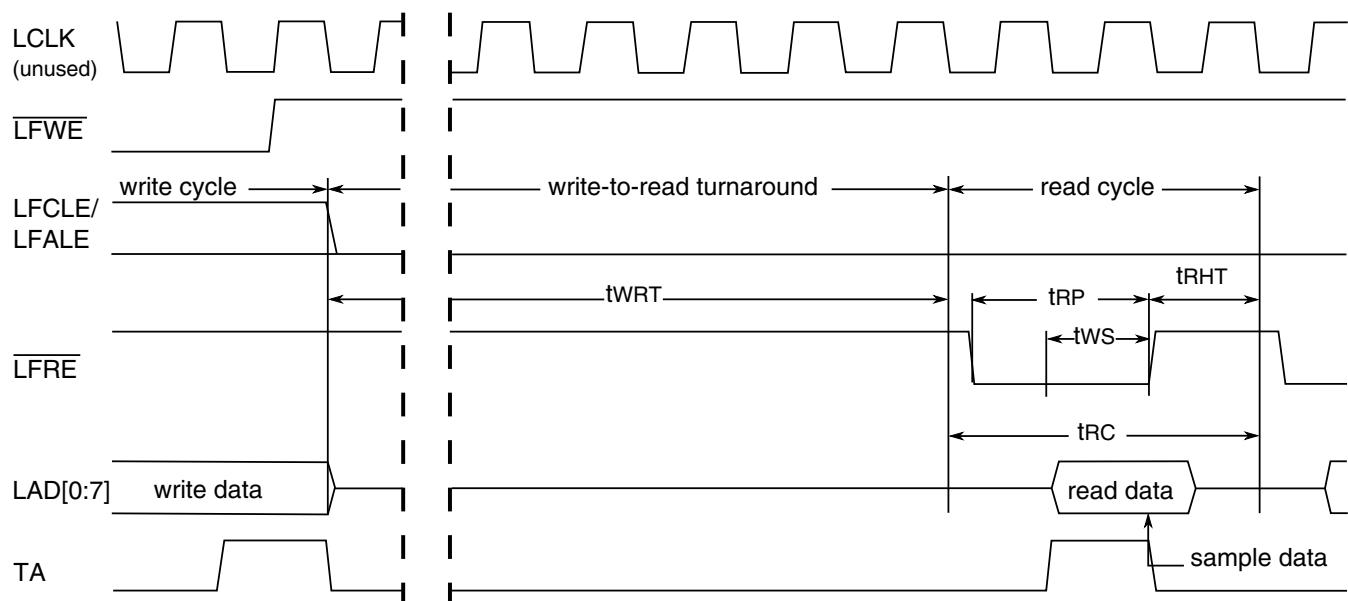
Figure 13-92. FCM delay prior to sampling LFRB_B state

13.4.3.3.4 FCM read data timing

The timing for read data transfers is shown in the figure below.

Upon assertion of LFRE_B, the Flash device will enable its output drivers and drive valid read data while LFRE_B is held low.

FCM samples read data on the rising edge of LFRE_B, which follows an optional number of wait states. Note that FCM will delay the first read if a RBW or RSW instruction is issued, in which case LFRB_B sample timing takes effect (see [FCM ready/busy timing](#)).



Notes:

- $t_{RP} = \overline{LFRE}$ pulse time, read period.
- $t_{RHT} = \overline{LFRE}$ hold time.
- $t_{WS} =$ Read wait state time.
- $t_{RC} =$ Read data cycle time.
- $t_{WRT} =$ Write to read turnaround time.

Figure 13-93. FCM read data timing (for TRLX = 0, RST = 0, SCY = 1)

Table 13-198. FCM read data timing parameters

Option Register Attributes		Timing Parameter (LCLK Clock Cycles) ¹				
TRLX	RST	t_{RP}	t_{RHT}	t_{WS}	t_{RC}	t_{WRT}
0	0	$\frac{3}{4} + SCY$	1	SCY	$2 + SCY$	$4 \times (2 + SCY)$
0	1	$1 + SCY$	1	SCY	$2 + SCY$	$4 \times (2 + SCY)$
1	0	$\frac{1}{2} + 2 \times SCY$	2	$2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$
1	1	$1 + 2 \times SCY$	2	$2 \times SCY$	$3 + 2 \times SCY$	$8 \times (2 + SCY)$

- In the parameters, SCY refers to a delay of ORn[SCY] clock cycles.

13.4.3.3.5 FCM extended read hold timing

Allowance for slow output driver turn-off when reading NAND Flash EEPROMs is made via setting of ORn[EHTR] and ORn[TRLX].

The extended read data hold time, shown at t_{EHTR} in [Figure 13-84](#) and [Figure 13-94](#), is a delay inserted by FCM between the last data read and another eLBC memory access (requiring LALE assertion). LCSn_B is negated during t_{EHTR} to allow external devices and bus transceivers time to disable their drivers.

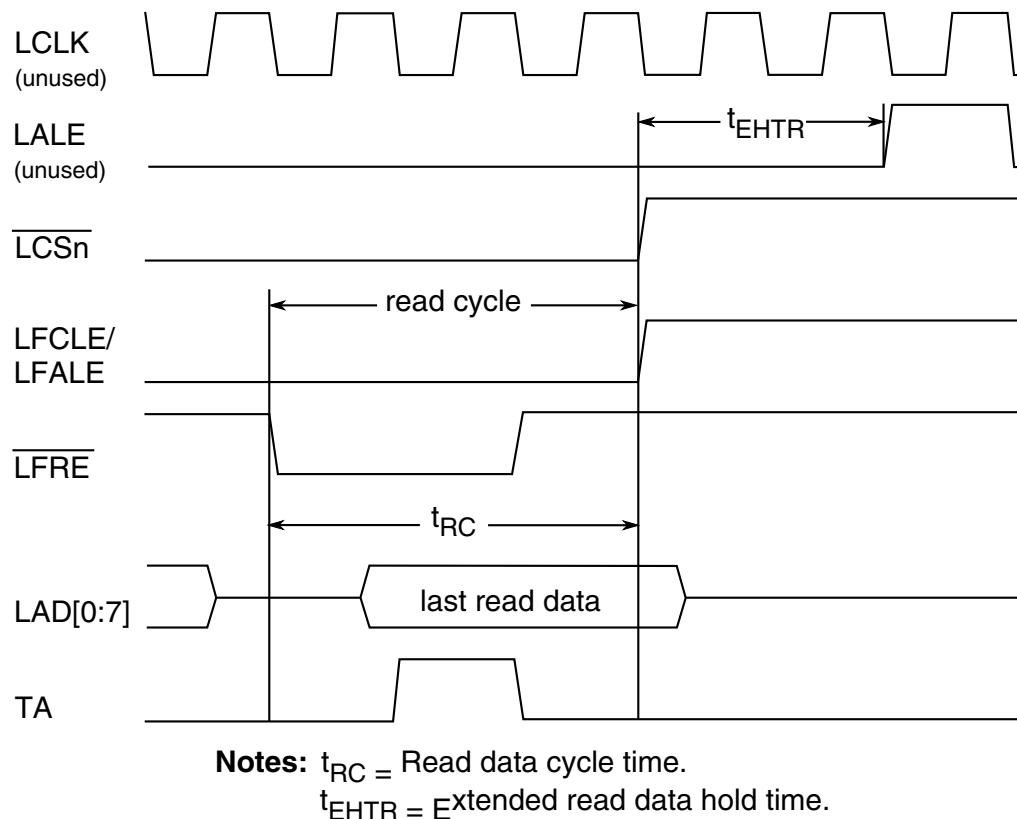


Figure 13-94. FCM read data timing with extended hold time (for TRLX = 0, EHTR = 1, RST = 1, SCY = 1)

13.4.3.4 FCM boot chip-select operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization.

LCS0_B is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, LCS0_B is asserted initially to load a 4-Kbyte boot block into the FCM buffer RAM, but core instruction fetches occur from the buffer RAM.

13.4.3.4.1 FCM bank 0 reset initialization

The boot chip-select also provides a programmable port size, which is configured during reset.

The boot chip-select does not provide write protection. LCS0_B operates this way until the first write to OR0 and it can be used as any other chip-select register after the preferred address range is loaded into BR0. After the first write to OR0, the boot chip-select can be restarted only with a hardware reset.

Table 13-199. Boot bank field values after reset for FCM as boot controller

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	PS	From RCW[BOOT_LOC]
	DECC	From cfg_elbc_ecc
	WP	0
	MSEL	001
	V	1
OR0	AM	0000_0000_0000_0000_0
	BCTL0	0
	PGS	From RCW[BOOT_LOC]
	CSCT	1
	CST	1
	CHT	1
	RST	1
	SCY	010
	TRLX	1
	EHTR	1

13.4.3.4.2 Boot block loading into the FCM buffer RAM

If FCM is selected as the boot ROM controller from power-on-reset configuration, eLBC will automatically load from bank 0 a single 4 Kbyte page of boot code into the FCM buffer RAM during HRESET_B.

The CPU can execute boot code directly from the FCM buffer RAM, but must ensure that any further data read from the NAND Flash EEPROM is transferred under software control in order to continue the bootstrap process.

Since OR0[AM] is initially cleared during reset, all CPU fetches to eLBC will access the FCM buffer RAM, which appears in the memory map as a 4-Kbyte RAM. No NAND Flash spare regions are mapped during boot, therefore only 4 Kbytes of contiguous, main region data, loaded from the first pages of the boot block, are accessible in eLBC bank 0.

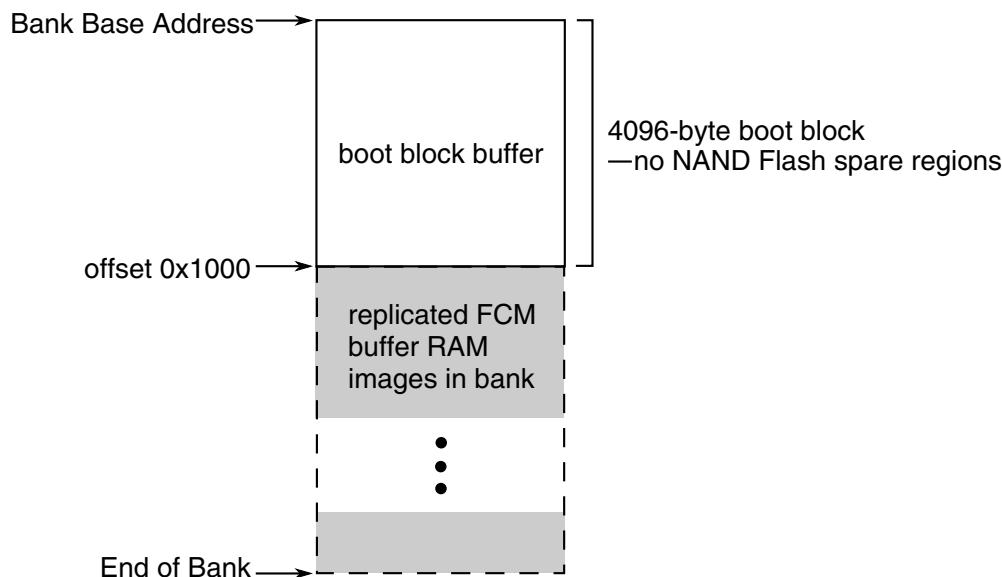


Figure 13-95. FCM buffer RAM memory map during boot loading

The process for booting is as follows:

1. Following negation of HRESET_B, eLBC is released from reset and commences automatic boot block loading if FCM is selected as the boot ROM location. Small-page or large-page, 8-bit NAND Flash devices can be used for boot loading when enabled with LCS0_B. eLBC drives LFWP_B low during boot accesses to prevent accidental erasure of the NAND Flash boot ROM.
2. FCM starts searching for a valid boot block at block index 0.
3. FCM reads the spare regions of the first two pages of the current block, checking the bad block indication (BI) bytes to validate the block for reading. BI bytes must all hold the value 0xFF for the page to be considered readable.
 - For small-page devices, BI is a single byte read from spare region byte offset 5.
 - For large-page devices, BI is a single byte read from spare region byte offset 0.

If either of the first two pages of the current block are marked invalid, then the boot block index is incremented by 1, and FCM repeats step 3. eLBC will continue searching for a bootable block indefinitely, therefore at least one block must be marked valid for boot loading to proceed. At the conclusion of the boot block search, the value of FBAR[BLK] points to the boot block.

4. The FCM optionally performs ECC checking at boot time depending on the configuration selected during reset. If ECC checking is enabled, the FCM recovers from the spare region the stored ECC for each 512-byte block of boot data. The boot block must be prepared with ECC protection. During ECC generation, software should use FMR[ECCM] = 0 for small-page devices, and FMR[ECCM] = 1 for large-page devices.
5. FCM performs a sequence of random-access page reads, reading entire pages from the boot block until 4 Kbytes have been saved to the FCM buffer RAM. If ECC checking is enabled, the ECC of each 512-byte region is verified and single-bit errors are corrected if possible. If FCM is unable to correct ECC errors, eLBC halts the boot process and signals an unrecoverable error by asserting the *hreset_req_B* signal.
6. The CPU now commences fetching instructions, in random order, from the FCM buffer RAM. This first-level boot loader typically copies a secondary boot loader into system memory, and continues booting from there. Boot software must clear FMR[BOOT] to enable normal operation of FCM.

13.4.4 User-programmable machines (UPMs)

UPMs are flexible interfaces that connect to a wide range of memory devices.

At the heart of each UPM is an internal RAM array that specifies the logical value driven on the external memory control signals (LCSn_B, LBS_B[0:1] and LGPL[0:5]) for a given clock cycle. Each word in the RAM array provides bits that allow a memory access to be controlled with a resolution of up to one quarter of the external bus clock period on the byte-select and chip-select lines. A gap of 2 dead LCLK cycles is present on the UPM interface between UPM transactions.

NOTE

If the LGPL4/ LGTA_B/ LFRB_B/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pull-up is required. See the device hardware specifications for details regarding termination options.

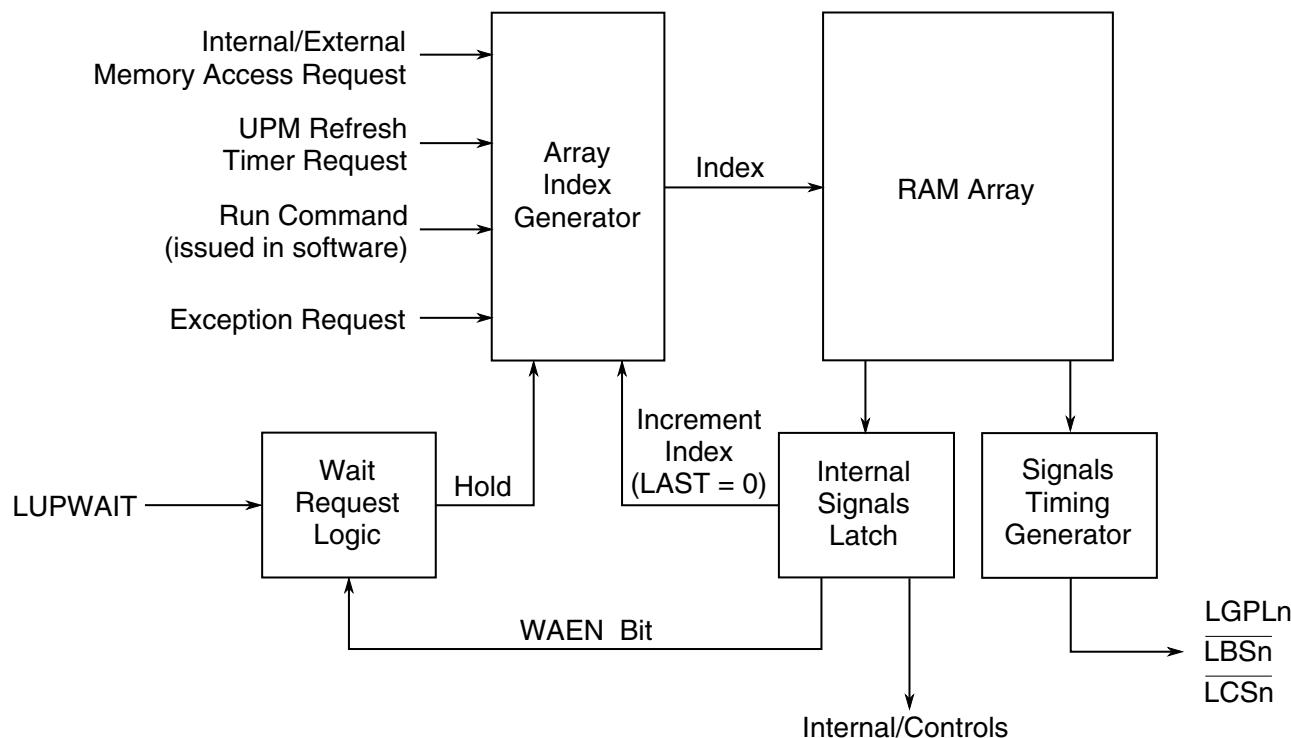


Figure 13-96. User-programmable machine functional block diagram

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

13.4.4.1 UPM requests

A special pattern location in the RAM array is associated with each of the possible UPM requests.

An internal device's request for a memory access initiates one of the following patterns ($MxMR[OP] = 00$):

- Read single-beat pattern (RSS)

- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

A UPM refresh timer request pattern initiates a refresh timer pattern (RTS).

An exception (caused by a bus monitor time-out error) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

The figure and table below show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands ($MxMR[OP] = 11$), however, can initiate patterns starting at any of the 64 UPM RAM words.

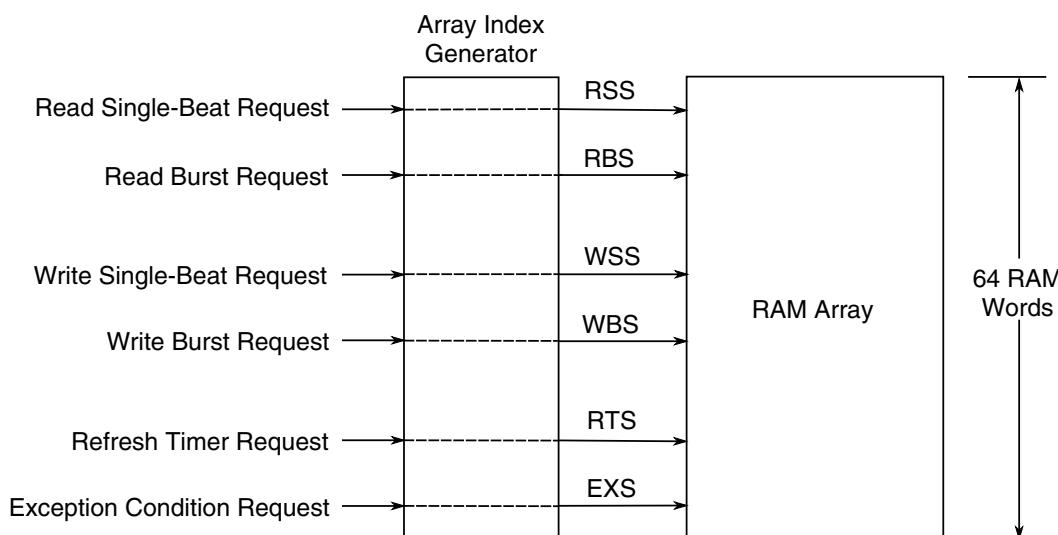


Figure 13-97. RAM array indexing

Table 13-200. UPM routines start addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

13.4.4.1.1 Memory access requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly 4 double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting ORn[BI]. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

13.4.4.1.2 UPM refresh timer requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array.

The figure below shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

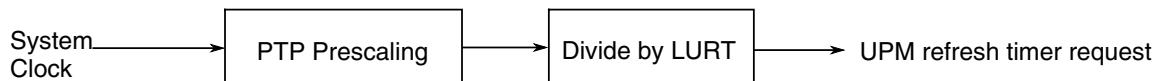


Figure 13-98. Memory refresh timer request block diagram

By default, all local bus refreshes are performed using the refresh pattern of UPMA. This means that if refresh is required, MAMR[RFEN] must be set. It also means that only one refresh routine should be programmed and be placed in UPMA, which serves as the refresh executor. Any banks assigned to a UPM are provided with the common UPMA refresh pattern if the RFEN bit of the corresponding UPM is set, concurrently. UPMA assigned banks, therefore, always receive refresh services when MAMR[RFEN] is set, while UPMB and UPMC assigned banks also receive (the same) refresh services if the corresponding MxMR[RFEN] bits are set. In this scenario, more than one chip select may assert at the same time, as refresh pattern runs for all banks assigned to UPM with RFEN bit set.

13.4.4.1.3 Software requests-RUN command

Software can start a request to the UPM by issuing a RUN command to the UPM.

Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The RUN command is issued by setting MxMR[OP] = 11 and accessing UPM n memory region with any write transaction that hits the corresponding UPM machine. MxMR[MAD] determines the starting address in the RAM array for the pattern.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

13.4.4.1.4 Exception requests

When the eLBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data.

The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

13.4.4.2 Programming the UPMS

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles.

Follow these steps to program the UPMS:

1. Set up BR n and OR n registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT and MAMR[RFEN] if refresh is required.
4. Program MxMR.

Patterns are written to the RAM array by setting MxMR[OP] = 01 and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each

time followed by a read from MDR and then followed by a (dummy) write transaction to the relevant UPM assigned bank. A read from MDR is required to ensure that the MDR update has occurred prior to the (dummy) write transaction.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when $MxMR[OP] = 10$).

NOTE

$MxMR/MDR$ registers should not be updated while dummy read/write access is still in progress. If the $MxMR[MAD]$ is incremented then the previous dummy transaction is already completed.

In order to enforce proper ordering between updates to the $MxMR/MDR$ register and the dummy accesses to the UPM memory region, two rules must be followed:

- Since the result of any update to the $MxMR/MDR$ register must be in effect before the dummy read or write to the UPM region, a write to $MxMR/MDR$ should be followed immediately by a read of $MxMR/MDR$.
- The UPM memory region should have the same MMU settings as the memory region containing the $MxMR$ configuration register; both should be mapped by the MMU as cache-inhibited and guarded. This prevents the CPU from re-ordering a read of the UPM memory around the read of $MxMR$. Once the programming of the UPM array is complete the MMU setting for the associated address range can be set to the proper mode for normal operation, such as cacheable and copyback.

For proper signalling, the following guidelines must be followed while programming UPM RAM words:

- For UPM reads, program UTA and LAST in the same or consecutive RAM words.
- For UPM burst reads, program last UTA and LAST in the same or consecutive RAM words.
- For UPM writes, program UTA and LAST in the same RAM word.
- For UPM burst writes, program last UTA and LAST in the same RAM word.

13.4.4.2.1 UPM programming example (two sequential writes to the RAM array)

The following example further illustrates the steps required to perform two writes to the RAM array at non-sequential addresses assuming that the relevant BR_n and OR_n registers have been previously set up:

1. Program MxMR for the first write (with the desired RAM array address).
2. Write pattern/data to MDR to ensure that the MxMR has already been updated with the desired configuration.
3. Read MDR to ensure that the MDR has already been updated with the desired pattern. (Or, read MxMR register if step 2 is not performed.)
4. Perform a dummy write transaction.
5. Read/check MxMR[MAD]. If incremented, the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program MxMR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the MxMR has already been updated with the desired configuration.
8. Read MDR to ensure that the MDR has already been updated with the desired pattern.
9. Perform a dummy write transaction.
10. Read/check MxMR[MAD]. If incremented, the previous dummy write transaction is completed.

Note that if steps 1 and 2 or steps 6 and 7 are reversed, step 3 or 8 (as appropriate) is replaced by the following:

- Read MxMR to ensure that the MxMR has already been updated with the desired configuration.

13.4.4.2.2 UPM programming example (two sequential reads from the RAM array)

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR ($MxMR[OP] = 0b10$).

The following example further illustrates the steps required to perform two reads from the RAM array at non-sequential addresses assuming that the relevant BR_n and OR_n registers have been previously set up:

1. Program MxMR for the first read with the desired RAM array address.
2. Read MxMR to ensure that the MxMR has already been updated with the desired configuration, such as RAM array address.
3. Perform a dummy read transaction.

4. Read/check MxMR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program MxMR for the second read with the desired RAM array address.
7. Read MxMR to ensure that the MxMR has already been updated with the desired configuration, such as RAM array address.
8. Perform a dummy read transaction.
9. Read/check MxMR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

13.4.4.3 UPM signal timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle.

The signal timing generator causes external signals to behave according to timing specified in the current RAM word. Each bit in the RAM word relating to $LCSn_B$ and LBS_B timing specifies the value of the corresponding external signal at each quarter phase of the bus clock.

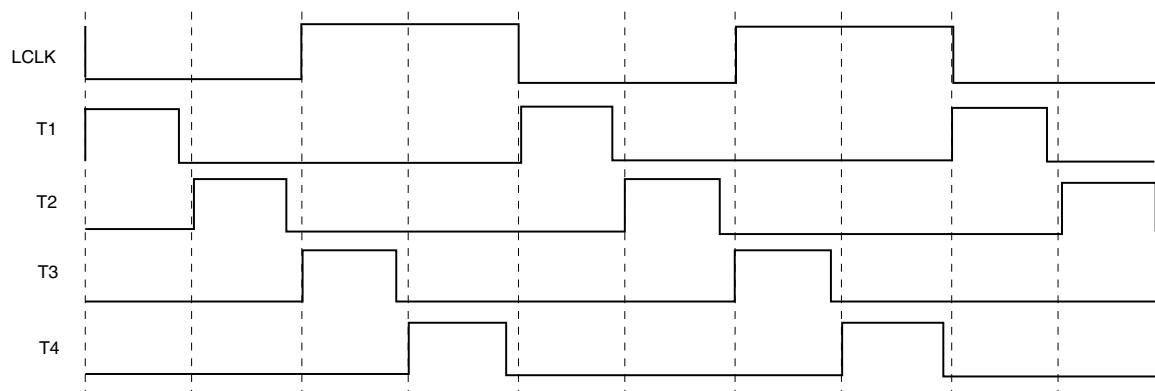


Figure 13-99. UPM clock scheme for $LCRR[CLKDIV] = 4$ and higher order ratios (PLL bypass)

13.4.4.4 RAM array

The RAM array for each UPM is 64 locations deep and 32 bits wide.

The signals at the bottom of the figure are UPM outputs. The selectedLCSn_B is for the bank that matches the current address. The selected LBS_B is for the byte lanes read or written by the access.

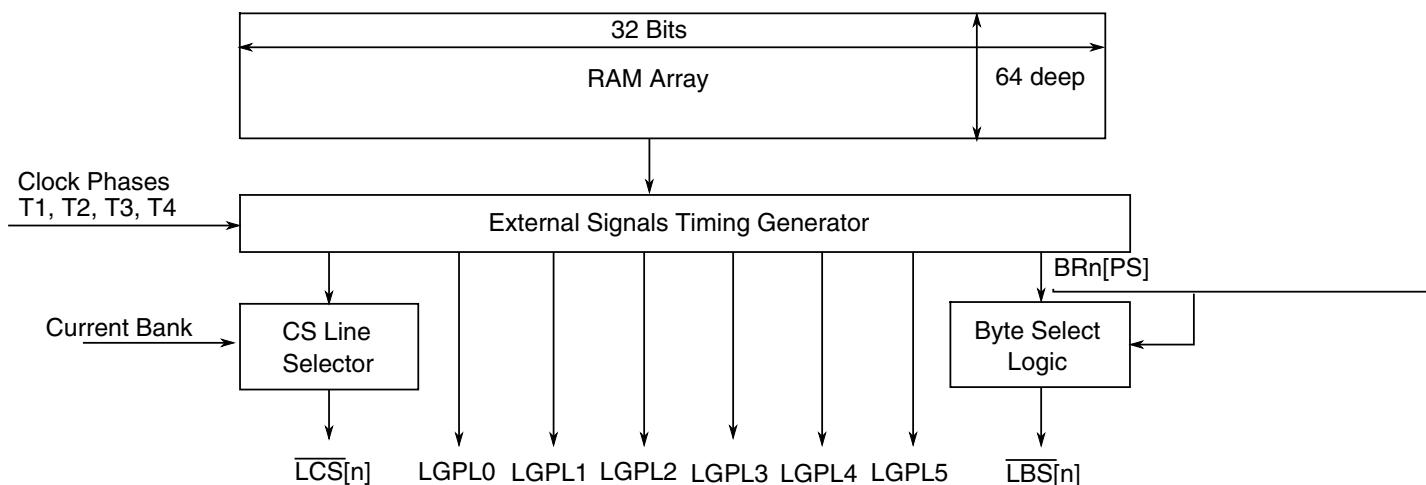


Figure 13-100. RAM array and signal generation

13.4.4.4.1 RAM words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array.

It specifies timing for external signals controlled by the UPM. The CSTn and BSTn bits determine the state of UPM signals LCSn_B and LBS_B[0:1] at each quarter phase of the bus clock.

Table 13-201. RAM word fields

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CST	CST	CST	CST4	BST	BST	BST	BST	G0L	G0H	G1T1	G1T3	G2T1	G2T		
W	1	2	3		1	2	3	4								3
Reset	All zeros															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	G3T	G3T	G4T	G4T3/ WAEN	G5T	G5T	REDO		LOOP	EXE N	AMX		NA	UTA	TODT	LAS T
W	1	3	1/ DLT3		1	3										
Reset	All zeros															

Table 13-202. RAM word field descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of LCSn_B during bus clock quarter phase 1.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of LCSn_B during bus clock quarter phase 2.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of LCSn_B during bus clock quarter phase 3.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of LCSn_B during bus clock quarter phase 4.
4	BST1	Byte select timing 1. Defines the state (0 or 1) of LBS_B during bus clock quarter phase 1 .
5	BST2	Byte select timing 2. Defines the state (0 or 1) of LBS_B during bus clock quarter phase 2 .
6	BST3	Byte select timing 3. Defines the state (0 or 1) of LBS_B during bus clock quarter phase 3 .
7	BST4	Byte select timing 4. Defines the state (0 or 1) of LBS_B during bus clock quarter phase 4 .
8-9	G0L	General purpose line 0 lower. Defines the state of LGPL0 during the bus clock quarter phases 1 and 2 (first half phase). 00 Value defined by MxMR[G0CL] 01 Reserved 10 0 11 1
10-11	G0H	General purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by MxMR[G0CL] 01 Reserved 10 0 11 1
12	G1T1	General purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).
13	G1T3	General purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DTL3	General purpose line 4 timing 1/delay time 3. The function of this bit is determined by MxMR[GPL4].

Table continues on the next page...

Table 13-202. RAM word field descriptions (continued)

Bits	Name	Description
		<p>If MxMR[GPL4] = 0 and LGPL4/LUPWAIT pin functions as an output (LGPL4), G4T1/DLT3 defines the state (0 or 1) of LGPL4 during bus clock quarter phases 1 and 2 (first half phase).</p> <p>If MxMR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), if a read burst or single read is executed, G4T1/DLT3 defines the sampling of the data bus as follows:</p> <ul style="list-style-type: none"> 0 In the current word, the data bus should be sampled at the start of bus clock quarter phase 1 of the next bus clock cycle. 1 In the current word, the data bus should be sampled at the start of bus clock quarter phase 3 of the current bus clock cycle.
19	G4T3/WAEN	<p>General purpose line 4 timing 3/wait enable. Bit function is determined by MxMR[GPL4].</p> <p>If MxMR[GPL4] = 0 and LGPL4/LUPWAIT pin functions as an output (LGPL4), G4T3/WAEN defines the state (0 or 1) of LGPL4 during bus clock quarter phases 3 and 4 (second half phase).</p> <p>If MxMR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), G4T3/WAEN is used to enable the wait mechanism:</p> <ul style="list-style-type: none"> 0 LUPWAIT detection is disabled. 1 LUPWAIT is enabled. If LUPWAIT is detected as being asserted, a freeze in the external signals logical values occurs until LUPWAIT is detected as being negated.
20	G5T1	General purpose line 5 timing 1. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General purpose line 5 timing 3. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 3 and 4 (second half phase).
22-23	REDO	<p>Redo current RAM word. Defines the number of times to execute the current RAM word.</p> <ul style="list-style-type: none"> 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	<p>Loop start/end. The first RAM word in the RAM array where LOOP is 1 is recognized as the loop start word. The next RAM word where LOOP is 1 is the loop end word. RAM words between, and including the start and end words, are defined as part of the loop. The number of times the UPM executes this loop is defined in the corresponding loop fields of the MxMR.</p> <ul style="list-style-type: none"> 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop. <p>NOTE: AMX must not change values in any RAM word which begins a loop</p>
25	EXEN	Exception enable. Allows branching to an exception pattern at the exception start address (EXS). When an internal bus monitor time-out exception is recognized and EXEN in the RAM word is set, the UPM branches to the special exception start address (EXS) and begins operating as the pattern defined there specifies.

Table continues on the next page...

Table 13-202. RAM word field descriptions (continued)

Bits	Name	Description
		<p>The user should provide an exception pattern to negate signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate RAS and CAS to prevent data corruption. If EXEN = 0, exceptions are ignored by UPM (but not by local bus) and execution continues. After the UPM branches to the exception start address, it continues reading until the LAST bit is set in the RAM word.</p> <p>0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out.</p> <p>1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.</p>
26-27	AMX	<p>Address multiplexing. Determines the source of LAD during an LALE phase. Any change in the AMX field initiates a new LALE (address) phase.</p> <p>00 LAD (and/or in conjunction with LA) is the non-multiplexed address. For example, column address.</p> <p>01 Reserved</p> <p>10 LAD (and/or in conjunction with LA) is driven with the multiplexed address according to MxMR[AM]. For example, row address. See Address multiplexing (AMX) for more information.</p> <p>11 LAD (and/or in conjunction with LA) is driven with the contents of MAR. Used, for example, to initialize a mode.</p> <p>NOTE: AMX must not change values in any RAM word which begins a loop.</p>
28	NA	<p>Next burst address. Determines when the address is incremented during a burst access.</p> <p>0 The address increment function is disabled.</p> <p>1 The address is incremented in the next cycle. In conjunction with the BRn[PS], the increment value of LAn is 1 or 2 for port sizes of 8 and 16 bits , respectively.</p>
29	UTA	<p>UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle.</p> <p>0 Transfer acknowledge is not asserted in the current cycle.</p> <p>1 Transfer acknowledge is asserted in the current cycle.</p> <p>In case of UPM writes, program UTA and LAST in same RAM word.</p> <p>In case of UPM reads, program UTA and LAST in consecutive or same RAM words.</p>
30	TODT	<p>Turn-on disable timer. The disable timer associated with each UPM allows a minimum time to be guaranteed between two successive accesses to the same memory bank. This feature is critical when DRAM requires a RAS precharge time. TODT turns the timer on to prevent another UPM access to the same bank until the timer expires. The disable timer period is determined in MxMR[DSn]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored.</p> <p>0 The disable timer is turned off.</p> <p>1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.</p>

Table continues on the next page...

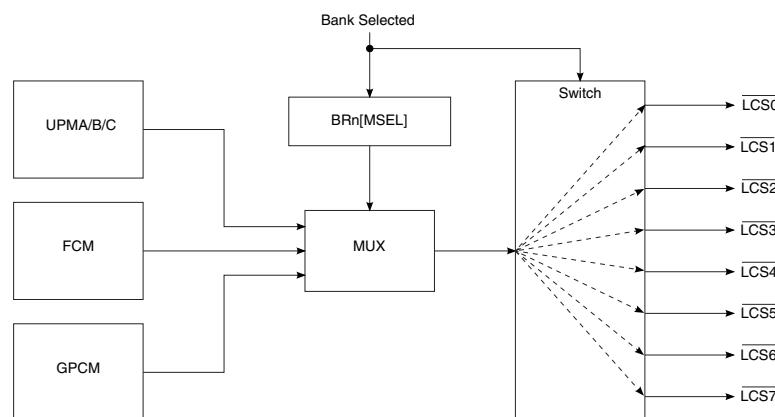
Table 13-202. RAM word field descriptions (continued)

Bits	Name	Description
31	LAST	<p>Last word. When LAST is read in a RAM word, the current UPM pattern terminates and control signal timing set in the RAM word is applied to the current (and last) cycle. However, if the disable timer is activated and the next access is to the same bank, execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in MxMR[DSn].</p> <p>Note that UPM continue to execute RAM words until it finds LAST, irrespective of the assigned region for various patterns like RSS.</p> <p>0 The UPM continues executing RAM words.</p> <p>1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.</p> <p>In case of UPM writes, program UTA and LAST in same RAM word.</p> <p>In case of UPM reads, program UTA and LAST in consecutive or same RAM words.</p>

13.4.4.4.2 Chip-select signal timing (CSTn)

If BRn[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the LCSn_B for that bank with timing as specified in the UPM RAM word CSTn fields.

The selected UPM affects only the assertion and negation of the appropriate LCSn_B signal. The state of the selected LCSn_B signal of the corresponding bank depends on the value of each CSTn bit.

**Figure 13-101. LCSn_B Signal selection**

13.4.4.4.3 Byte select signal timing (BSTn)

If $BRn[MSEL]$ of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate $LBS_B[0:1]$ signal.

The timing of both byte-select signals is specified in the RAM word. However, $LBS_B[0:1]$ are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed.

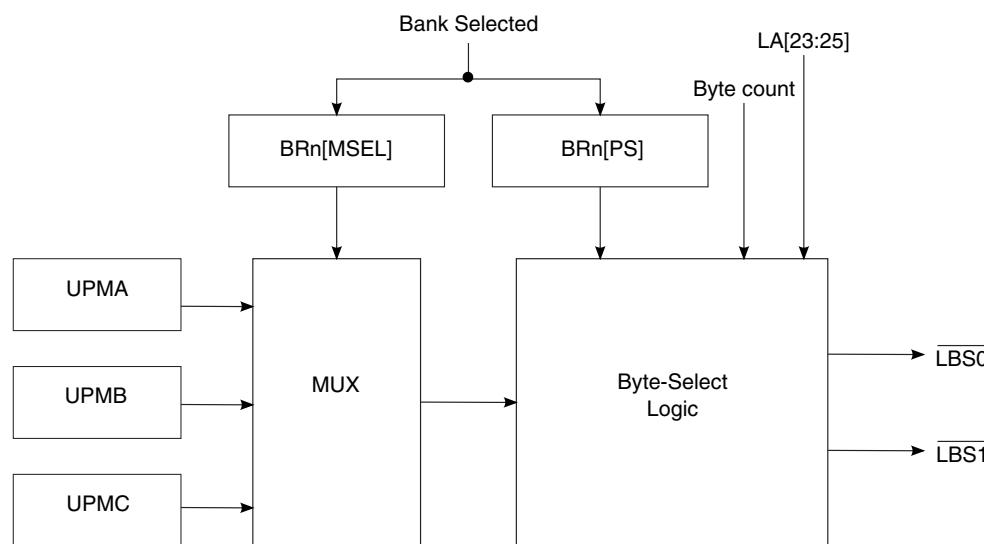


Figure 13-102. LBS_B Signal selection

The uppermost byte select (LBS0_B), when asserted, indicates that $LAD[0:7]$ contains valid data during a cycle. Likewise, LBS1_B indicates that $LAD[8:15]$ contain valid data. For a UPM refresh timer request, all $LBS_B[0:1]$ signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, the $LBS_B[0:1]$ signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

13.4.4.4 General-purpose signals (GnTn, GOn)

The general-purpose signals ($LGPL[0:5]$) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of the bus clock and/or at the falling edge of the bus clock.

$LGPL_0$ offers enhancements beyond the other $LGPL_n$ lines.

LGPL0 can be controlled by an address line specified in MxMR[G0CL]. To use this feature, G0H and G0L should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

13.4.4.4.5 Loop control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated.

The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in the table below. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken:

- LAST and LOOP must not be set together.
- Loop start word should not have an AMX change with regard to the previous word.

Table 13-203. MxMR loop field use

Request Serviced	Loop Field
Read single-beat cycle	RLF
Read burst cycle	RLF
Write single-beat cycle	WLF
Write burst cycle	WLF
Refresh timer expired	TLF
RUN command	RLF

13.4.4.4.6 Repeat execution of current RAM word (REDO)

The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words.

Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
 - When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
 - When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
 - LAST and REDO must not be set together.
 - REDO should not be used within the exception routine.

13.4.4.4.7 Address multiplexing (AMX)

Address lines can be controlled by the user-provided pattern in the UPM. The address multiplex (AMX) bits in the RAM word can choose between driving the transaction address ($AMX = 00$), driving it according to the multiplexing specified by the $MxMR[AM]$ field ($AMX = 10$), or driving the contents of MAR ($AMX = 11$) on the address signals. The next address (NA) bit of the RAM word does not affect LA signals, unless $AMX = 00$ and chooses the column address for $NA = 1$.

In all cases, LA[27:31] of the eLBC are driven by the five lsbs of the address selected by AMX, regardless of whether the next address (NA) bit of the RAM word is used to increment the current address. The effect of NA = 1 is visible only when AMX = 00 chooses the column address.

[Table 13-204](#) shows how the RAM word AMX bits and MxMR[AM] settings can be used to affect row x column address multiplexing on the LA [16:31] signals.

Table 13-204. UPM address multiplexing

Table continues on the next page...

Table 13-204. UPM address multiplexing (continued)

	msb	Internal Transaction Address																													I s b				
		0	1	2	3	4	5	6	7	8	9	1	0	1	1	1	1	1	1	1	1	1	2	0	2	1	2	2	2	2	2	2			
AMX = 10 MxMR[AM] = 010 (Row)								1 6	1 7	1 8	1 9	0 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1												
AMX = 00 (Col)																									2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1	
AMX = 10 MxMR[AM] = 011 (Row)						1 6	1 7	1 8	1 9	0 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1														
AMX = 00 (Col)																								2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1	
AMX = 10 MxMR[AM] = 100 (Row)					1 6	1 7	1 8	1 9	0 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1															
AMX = 00 (Col)																							2 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1	
AMX = 10 MxMR[AM] = 101 (Row)				1 6	1 7	1 8	1 9	0 0	1 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1																
AMX = 00 (Col)																							1 9	2 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7	2 8	2 9	3 0	3 1
AMX = 10 MxMR[AM] = 110		Reserved																																	
AMX = 10 MxMR[AM] = 111		Reserved																																	

Note that any change to the AMX field from one RAM word to the next RAM word executed results in an address phase on the {LAD_n, LAn} bus with the assertion of LALE for the number of cycles set for LALE in the OR_n and LCRR registers. The LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

NOTE

AMX must not change values in any RAM word which begins a loop.

13.4.4.4.8 Data valid and data sample control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the eLBC), the value of the DLT3 bit in the same RAM word, in conjunction with MxMR[GPL4], determines when the data input is sampled by the eLBC as follows:

- If MxMR[GPL4] = 1 (G4T4/DLT3 functions as DLT3) and DLT3 = 1 in the RAM word, data is latched on the falling edge of the bus clock instead of the rising edge. The eLBC samples the data on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.
- If MxMR[GPL4] = 0 (G4T4/DLT3 functions as G4T4), or if MxMR[GPL4] = 1 but DLT3 = 0 in the RAM word, data is latched on the rising edge of the bus clock, which occurs at the end of the current bus clock cycle (normal operation).

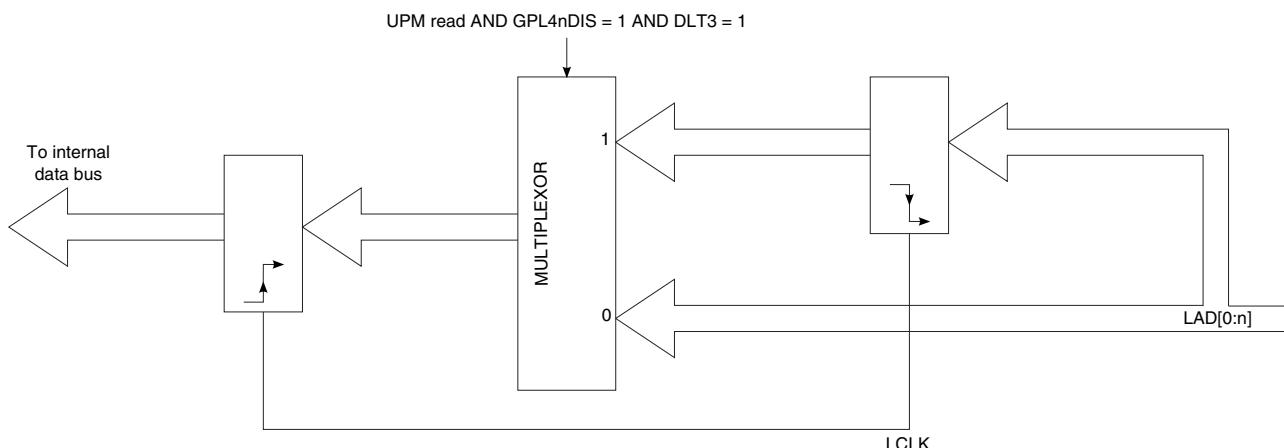


Figure 13-103. UPM read access data sampling

13.4.4.4.9 LGPL[0:5] signal negation (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern is terminated at the end of the current cycle.

On the next cycle (following LAST) all the UPM signals are negated unconditionally (driven to logic 1), unless there is a back-to-back UPM request pending. In this case, the signal values for the cycle following the one in which the LAST bit was set are taken from the first RAM word of the pending UPM routine.

In case of UPM writes, program UTA and LAST in same RAM word. In case of UPM reads, program UTA and LAST in consecutive or same RAM words.

13.4.4.4.10 Wait mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words.

If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if LAST = 1 in the same RAM word.

Synchronization of LUPWAIT starts at the falling edge of the LCLK and takes at least 1 LCLK cycle to get affected. If LUPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

The figure below shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the LCSn_B and LGPL1 states and the WAEN value are frozen until LUPWAIT is recognized as negated. WAEN is typically set before the line that contains UTA = 1. Note that if WAEN and NA are both set in the same RAM word, NA causes the burst address to increment once as normal regardless of whether the UPM freezes.

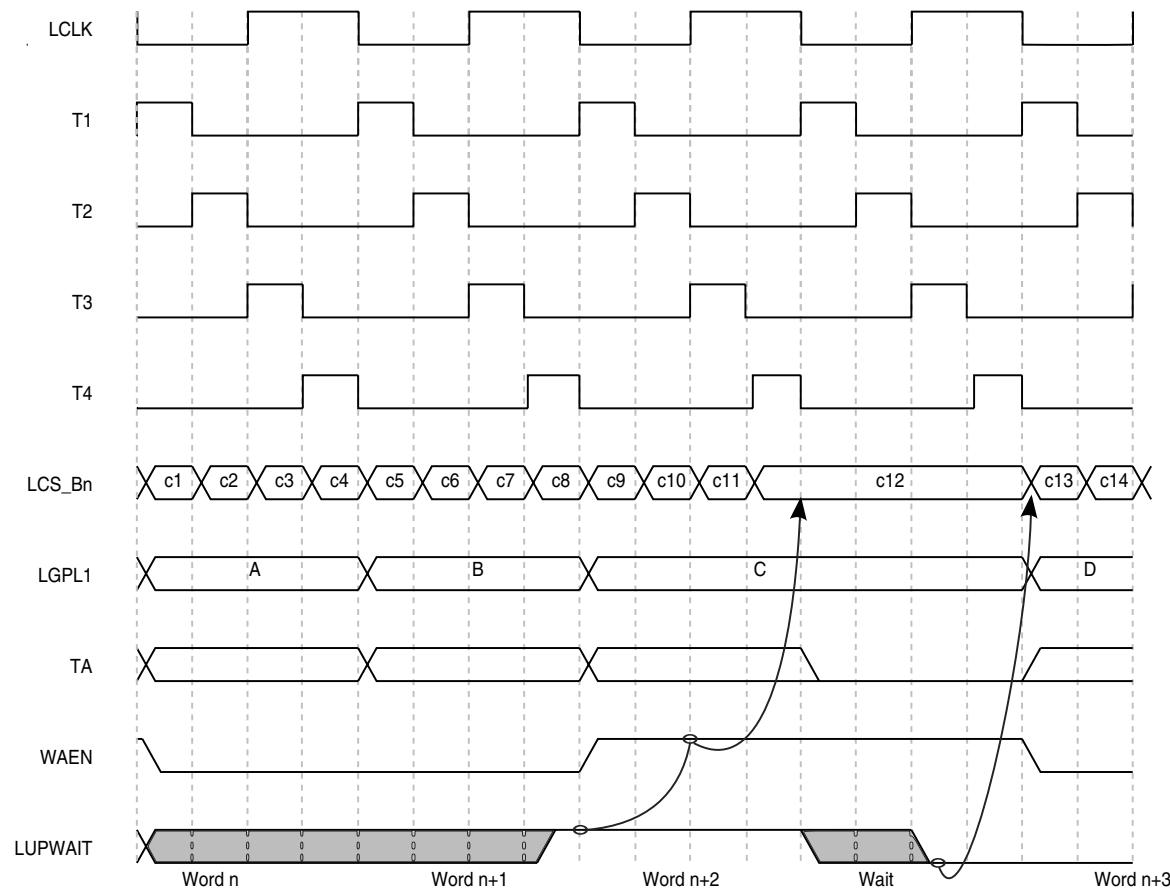


Figure 13-104. Effect of LUPWAIT signal with PLL bypass

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

13.4.4.5 Extended hold time on read accesses-UPM

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of OR_n[TRLX] and OR_n[EHTR].

The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the OR_n register in addition to any existing bus turnaround cycle.

13.5 eLBC initialization/application information

This section provides information about the following:

- Interfacing to peripherals in different address modes
- Bus turnaround
- Interface to different port-size devices
- Command sequence examples for NAND flash EEPROM
- Interfacing to fast-page mode DRAM using UPM
- Interfacing to ZBT SRAM using UPM

13.5.1 Interfacing to peripherals in different address modes

This section provides guidelines for interfacing to peripherals.

13.5.1.1 Multiplexed address/data bus for 32-bit addressing

In order to reduce pins on the local bus, address and data signals are multiplexed. To build the address, an external latch is used to demultiplex and reconstruct the original address.

Because the LALE signal provides the correct timing to control a standard logic latch, no external intelligence is needed. To pass data, the LAD signals can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus begin with an address phase. The eLBC drives the transaction address on the LAD signals and asserts the LALE signal to latch the address. This assertion causes address bits A[0:15] to appear on LAD[0:15]. The eLBC can then continue on into the data phase.

The eLBC supports port sizes of 8 and 16 bits. When there is an access larger than the port size, the eLBC breaks up the access into smaller transactions using the non-multiplexed address signals LA_n . For 16-bit devices, LA31 is irrelevant since this address bit is implicit in the byte lanes which carry data. However, note for 8-bit devices, LA31 is necessary.

In addition, the eLBC supports burst transfers in the UPM machine. To minimize the amount of address phases needed on the local bus and to optimize the throughput, LA_n are driven separately. All other address bits, A[0:15], must be reconstructed through the latch.

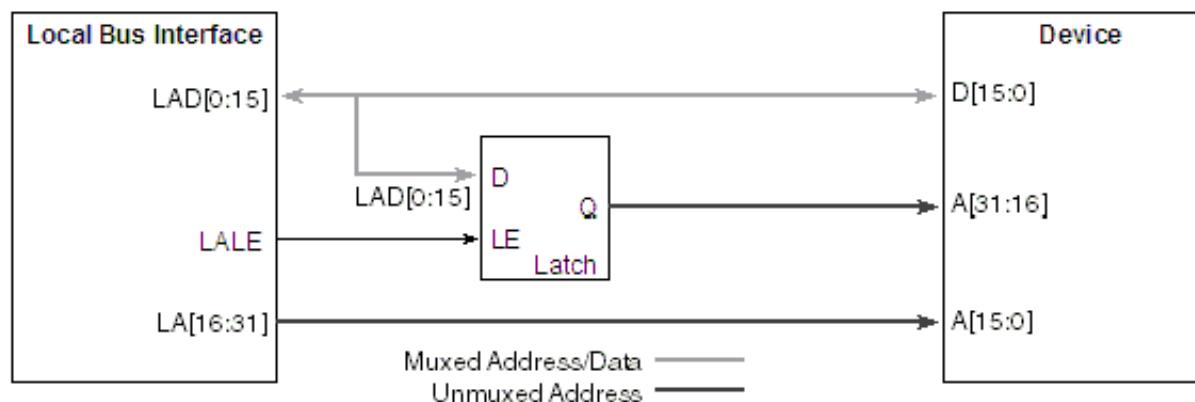


Figure 13-105. Multiplexed address/data bus for 32-bit addressing and 16-Bit Data Bus

13.5.1.2 Non-multiplexed address and data buses

For small address space applications the address latch may be eliminated entirely if the local bus address is taken entirely from LA[16:31], in which case addresses driven onto LAD during address phases are simply ignored. The connection is illustrated in the figure below. In non-multiplexed mode, the waveforms remain the same except for a few things, LAD bus turnaround time, LALE timings, that can be ignored.



Figure 13-106. Non-multiplexed address and data buses

13.5.1.3 Peripheral hierarchy on the local bus for high bus speeds

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the bus.

For best results, only one bank of synchronous SRAMs should have a direct connection, and a bus demultiplexor should be used to replace separate latch and separate bus transceiver combinations. The figure below shows an example of such a hierarchy. This section is only a guideline, and the board designer must simulate the electric characteristics of the scenario to determine the maximum operating frequency.

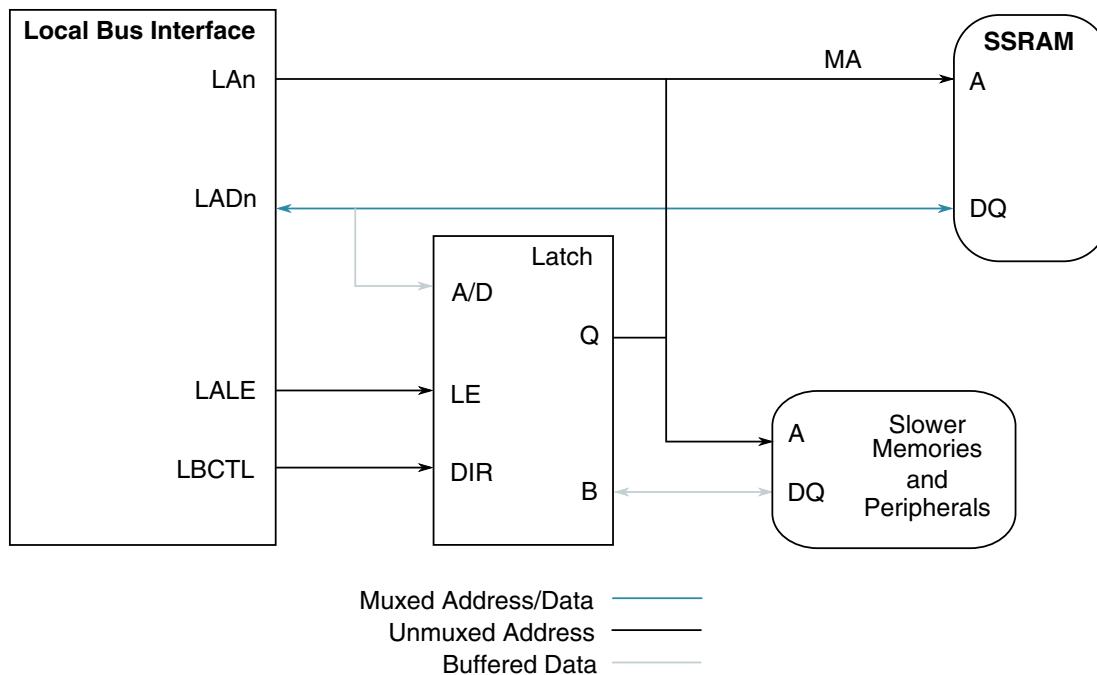


Figure 13-107. Local bus peripheral hierarchy for high bus speeds

13.5.1.4 GPCM timings

In case a system contains a memory hierarchy with high speed synchronous memories (synchronous SRAM) and lower speed asynchronous memories (for example, FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus.

Those buffers have to be taken into account for the timing calculations.

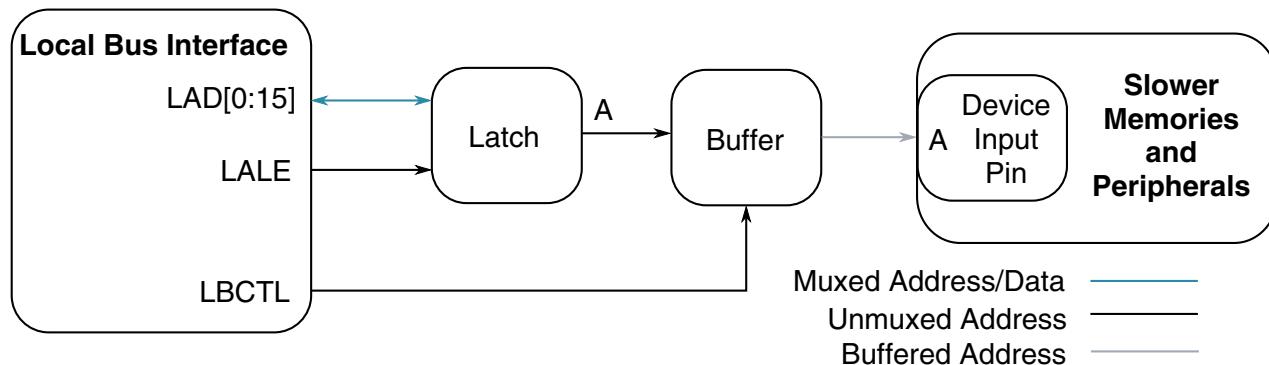


Figure 13-108. GPCM address timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the two propagation delays are in the order of 3 to 6 ns, so for a 100-MHz bus frequency, LCS_B should arrive on the order of 2 to 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered.

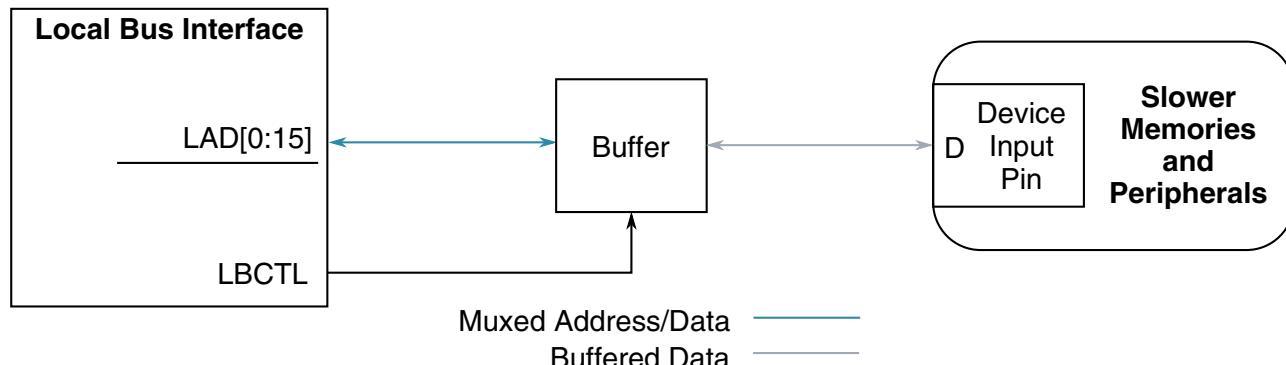


Figure 13-109. GPCM data timings

13.5.2 Bus turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround.

The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

The bus does not change direction for the following cases so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

13.5.2.1 Address phase after previous read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD.

After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the eLBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The eLBC then inserts a bus turnaround cycle to avoid contention.

The external device has now already placed its data signals in high impedance and no bus contention will occur.

13.5.2.2 Read data phase after address phase

During the address phase, LAD actively drives the address and LBCTL is high, driving the bus transceivers in the same direction as during a write.

After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The eLBC places the LAD signals in high impedance after its $t_{dis}(LB)$. The LBCTL will have its new state after $t_{en}(LB)$ and, because this is an asynchronous input, the transceiver starts to drive those signals after its $t_{en}(\text{transceiver})$ time. The system designer has to ensure, that $[t_{en}(LB) + t_{en}(\text{transceiver})]$ is larger than $t_{dis}(LB)$ to avoid bus contention.

13.5.2.3 Read-modify-write cycle for parity protected memory banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle.

Because the write cycle will have a new address phase in any case, this basically is the same case as an address phase after a previous read.

13.5.2.4 UPM cycles with additional address phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern.

The eLBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

13.5.3 Interface to different port-size devices

The eLBC supports 8- and 16-bit data port sizes.

However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on LAD[0-15], and an 8-bit port must reside on LAD[0-7]. The local bus always tries to transfer the maximum amount of data on all bus cycles

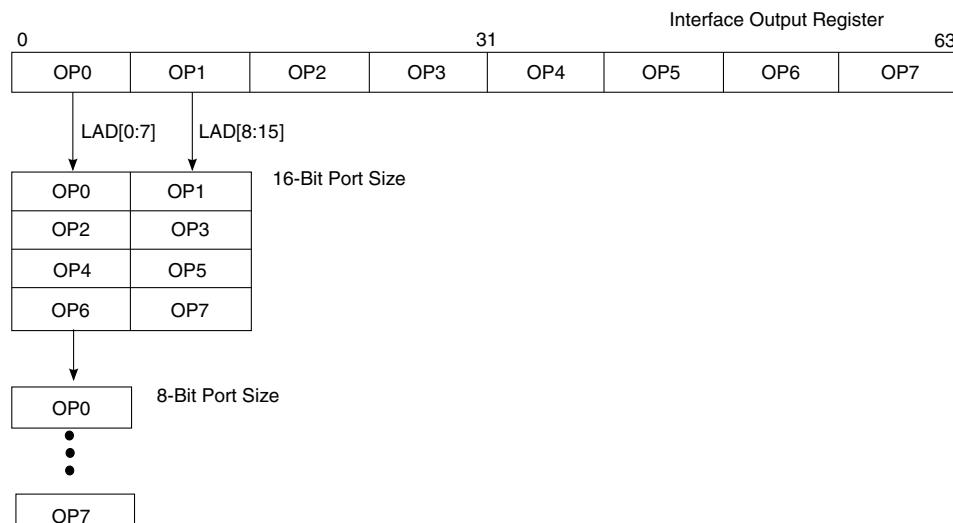


Table 13-205. Data bus drive requirements for read cycles

Transfer Size	Address State ¹ 3 lsbs	Port Size/LAD Data Bus Assignments							
		16-Bit				8-Bit			
		0-7	8-15	16-23	24-31	0-7	8-15	16-23	24-31
Byte	000	OP0	-			OP0			
	001	-	OP1			OP1			
	010	OP2	-			OP2			
	011	-	OP3			OP3			
	100	OP4	-			OP4			
	101	-	OP5			OP5			
	110	OP6	-			OP6			
	111	-	OP7			OP7			
Half Word	000	OP0	OP1			OP0			
	001	-	OP1			OP1			
	010	OP2	OP3			OP2			
	100	OP4	OP5			OP4			
	101	-	OP5			OP5			
	110	OP6	OP7			OP6			
Word	000	OP0	OP1			OP0			
	100	OP4	OP5			OP4			

1. Address state is the calculated address for port size.

13.5.4 Command sequence examples for NAND flash EEPROM

To program the eLBC and FCM for executing NAND Flash command sequences, command codes and pause states should be obtained from the relevant NAND Flash device data sheet and programmed into FCM configuration registers.

This section illustrates some common sequences for large-page, multi-gigabit NAND Flash EEPROMs; however, details should be verified against manufacturers' specific programming data.

Throughout these examples it is assumed that one or more banks of eLBC has been configured under FCM control ($BRn[MSEL] = 001$), with base address, port size, ECC mode, and timing parameters configured in accordance with the device hardware specifications.

13.5.4.1 NAND flash soft reset command sequence example

An example of configuring FCM to execute a soft reset command to large-page NAND Flash is shown in the table below.

This sequence does not require use of the shared FCM buffer RAM.

The sequence is initiated by writing FMR[OP] = 10, and issuing a special operation to the bank. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled.

Table 13-206. FCM register settings for soft reset (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0xFF00_0000	CMD0 = 0xFF = reset command; other commands unused
FBAR	-	Unused
FPAR	-	Unused
FBCR	-	Unused
MDR	-	Unused
FIR	0x4000_0000	OP0 = CM0 = command 0; OP1-OP7 = NOP

13.5.4.2 NAND flash read status command sequence example

An example of configuring FCM to execute a status read command to large-page NAND Flash is shown in the table below.

This sequence does not require use of the shared FCM buffer RAM, but reads the NAND Flash status into register MDR[AS0].

The sequence is initiated by writing FMR[OP] = 10 and issuing a special operation to the bank. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled.

Table 13-207. FCM register settings for status read (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0x7000_0000	CMD0 = 0x70 = read status command; other commands unused
FBAR	-	Unused
FPAR	-	Unused
FBCR	-	Unused
MDR	-	Status returned in AS0
FIR	0x4B00_0000	OP0 = CM0 = command 0;

Table 13-207. FCM register settings for status read (ORn[PGS] = 1)

Register	Initial Contents	Description
		OP1 = RS = read status to MDR; OP2-OP7 = NOP

13.5.4.3 NAND flash read identification command sequence example

An example of configuring FCM to execute a status ID command to large-page NAND Flash is shown in the table below.

This sequence does not require use of the shared FCM buffer RAM, but uses MDR to set up a dummy address prior to the sequence, and then to receive the first 4 bytes of ID during the sequence.

The sequence is initiated by writing FMR[OP] = 10, and issuing a special operation to the bank. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled. MDR[AS3-AS0] then can be read to obtain the first 4 bytes of NAND Flash ID.

Table 13-208. FCM register settings for ID read (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0x9000_0000	CMD0 = 0x90 = read ID command; other commands unused
FBAR	-	Unused
FPAR	-	Unused
FBCR	-	Unused
MDR	All zeros	AS0 = 0x00 = dummy address for read ID command; AS0-AS3 return with first 4 bytes of ID code
FIR	0x43BB_BBB0	OP0 = CM0 = command 0; OP1 = UA = user address from MDR; OP2-OP6 = RS = read 4 bytes ID into MDR[AS3-AS0]; OP7 = NOP
FMR	0x0000_F002	
LSOR	0x0000_0001	Assumes NAND flash is on eLBC CS1

13.5.4.4 NAND flash page read command sequence example

An example of configuring FCM to execute a random page read command to large-page NAND Flash is shown in the table below.

This sequence reads an entire page (main and spare region) into the shared FCM buffer RAM, checking ECC as it proceeds.

The sequence is initiated by writing FMR[OP] = 11, and issuing a special operation to the bank. A few cycles before completion itself, FECCn gets updated with the ECC bytes for the main region validated by FECCn[0]. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled. Once the sequence has completed, the shared buffer (buffer 1 for page index 5) and transfer error registers (LTECCR that reports the 512 blocks with unibit /multibit errors if any) are valid.

Table 13-209. FCM register settings for page read (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0x0030_0000	CMD0 = 0x00 = random read address entry; CMD1 = 0x30 = read page
FBAR	block index (for example block 0x0001_0AB4)	BLK locates index of 128-Kbyte block
FPAR	page offset (for example 0x0000_5000 locates page 5, buffer 1)	PI locates page index in BLK; PI mod 2 indexes FCM buffer RAM; MS = 0 and CI = 0
FBCR	All zeros	BC = 0 to read entire 2112-byte page with ECC check
MDR	-	Unused
FIR	0x4125_E000	OP0 = CM0 = command 0; OP1 = CA = column address; OP2 = PA = page address; OP3 = CM1 = command 1; OP4 = RBW = wait on Flash ready and read data into FCM buffer; OP5-OP7 = NOP

13.5.4.5 NAND flash block erase command sequence example

An example of configuring FCM to execute a block erase command to large-page NAND Flash is shown in the table below.

This sequence does not require use of the shared FCM buffer RAM, but returns with the erase status in MDR[AS0].

The sequence is initiated by writing FMR[OP] = 11, and issuing a special operation to the bank. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled.

Note that operations specified by OP3 and OP4 (status read) should never be skipped while erasing a NAND Flash device, because, in case that happens, contention may arise on LGPL4. A possible case is that the next transaction from eLBC may try to use that pin as an output and since the NAND Flash device might already be driving it, contention will occur. In case OP3 and OP4 operations are skipped, it may also happen that a new command is issued to the NAND Flash device even when the device has not yet finished processing the previous request. This may also result in unpredictable behavior.

Table 13-210. FCM register settings for block erase (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0x6070_D000	CMD0 = 0x60 = block address entry; CMD1 = 0x70 = read status CMD2 = 0xD0 = erase block;
FBAR	Block Index (for example block 0x0001_0AB4)	BLK locates index of 128-Kbyte block
FPAR	All zeros	PI = 0 to locate block boundary
FBCR	-	Unused
MDR	-	Returns with AS0 holding erase status
FIR	0x426D_B000	OP0 = CM0 = command 0; OP1 = PA = page address; OP2 = CM2 = command 2; OP3 = CW1 = wait on Flash ready and issue command 1; OP4 = RS = read erase status into MDR[AS0]; OP5-OP7 = NOP

13.5.4.6 NAND flash program command sequence example

An example of configuring FCM to execute a program command to large-page NAND Flash is shown in the table below.

This sequence writes an entire page (main and spare region) from the shared FCM buffer RAM, generating ECC as it proceeds.

The shared buffer (buffer 1 for page index 5) must be initialized by software prior to starting the sequence. The sequence is initiated by writing FMR[OP] = 11, and issuing a special operation to the bank. At the conclusion of the sequence, eLBC will issue a command complete interrupt (LTESR[CC]) if interrupts are enabled. The status of the programming operation is returned in MDR[AS0].

Note that operations specified by OP5 and OP6 (status read) should never be skipped while programming a NAND Flash device, because, in case that happens, contention may arise on LGPL4. A possible case is that the next transaction from eLBC may try to use that pin as an output and since the NAND Flash device might already be driving it, contention will occur. In case OP5 and OP6 operations are skipped, it may also happen that a new command is issued to the NAND Flash device even when the device has not yet finished processing the previous request. This may also result in unpredictable behavior.

Table 13-211. FCM register settings for page program (ORn[PGS] = 1)

Register	Initial Contents	Description
FCR	0x8070_1000	CMD0 = 0x80 = page address and data entry; CMD1 = 0x70 = read status CMD2 = 0x10 = program page;
FBAR	block index (for example block 0x0001_0AB4)	BLK locates index of 128-Kbyte block
FPAR	page offset (for example 0x0000_5000 locates page 5, buffer 1)	PI locates page index in BLK; PI mod 2 indexes FCM buffer RAM; MS = 0 and CI = 0
FBCR	All zeros	BC = 0 to write entire 2112-Byte page with ECC generation
MDR	-	Returns with AS0 holding program status
FIR	0x4128_6DB0	OP0 = CM0 = command 0; OP1 = CA = column address; OP2 = PA = page address; OP3 = WB = write data from buffer; OP4 = CM2 = command 2; OP5 = CW1 = wait on Flash ready and issue command 1; OP6 = RS = read erase status into MDR[AS0]; OP7 = NOP

13.5.5 Interfacing to fast-page mode DRAM using UPM

Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device.

This section describes timing diagrams for various UPM configurations for fast-page mode DRAM, with LCRR[CLKDIV] = 4 (clock ratio of 16) or 8 (clock ratio of 32). The illustrative examples shown in [Table 13-212-Table 13-216](#) may not represent the timing necessary for any specific device used with the eLBC. Here, LGPL1 is programmed to drive R/W of the DRAM, although any LGPln signal may be used for this purpose.

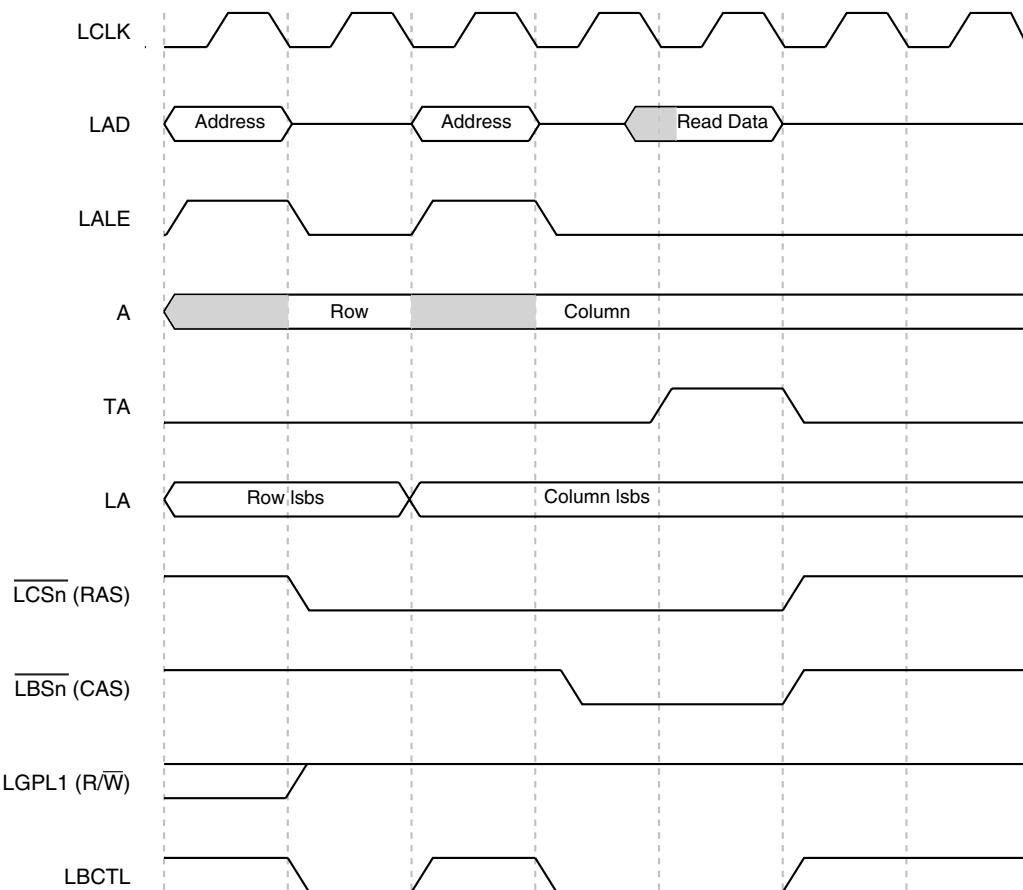


Figure 13-111. Single-beat read access to FPM DRAM

Table 13-212. Single-beat read access to FPM DRAM

cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7

Table continues on the next page...

Table 13-212. Single-beat read access to FPM DRAM (continued)

g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo[0]					Bit 22
redo[1]					Bit 23
loop	0		0	0	Bit 24
exen	0		0	0	Bit 25
amx0	1		0	0	Bit 26
amx1	0		0	0	Bit 27
na	0		0	0	Bit 28
uta	0		0	1	Bit 29
todt	0		0	1	Bit 30
last	0		0	1	Bit 31
	RSS	RSS + 1	RSS + 1	RSS + 2	

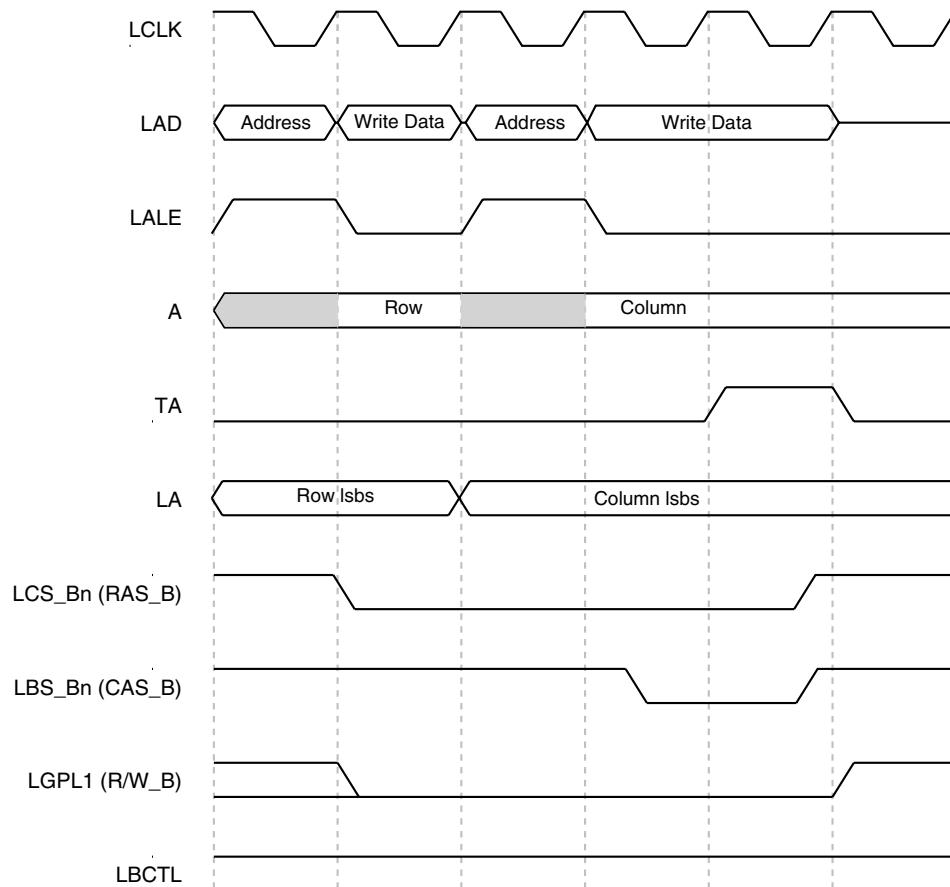


Figure 13-112. Single-beat write access to FPM DRAM

Table 13-213. Single-beat write access to FPM DRAM

cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	1	Bit 3
bst1	1		1	0	Bit 4
bst2	1		1	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	1	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	0		0	0	Bit 12
g1t3	0		0	0	Bit 13

Table continues on the next page...

Table 13-213. Single-beat write access to FPM DRAM (continued)

g2t1					Bit 14
g2t3					Bit 15
g3t1					Bit 16
g3t3					Bit 17
g4t1					Bit 18
g4t3					Bit 19
g5t1					Bit 20
g5t3					Bit 21
redo[0]					Bit 22
redo[1]					Bit 23
loop	0		0	0	Bit 24
exen	0		0	0	Bit 25
amx0	1		0	0	Bit 26
amx1	0		0	0	Bit 27
na	0		0	0	Bit 28
uta	0		0	1	Bit 29
todt	0		0	1	Bit 30
last	0		0	1	Bit 31
	WSS	WSS + 1	WSS + 1	WSS + 2	

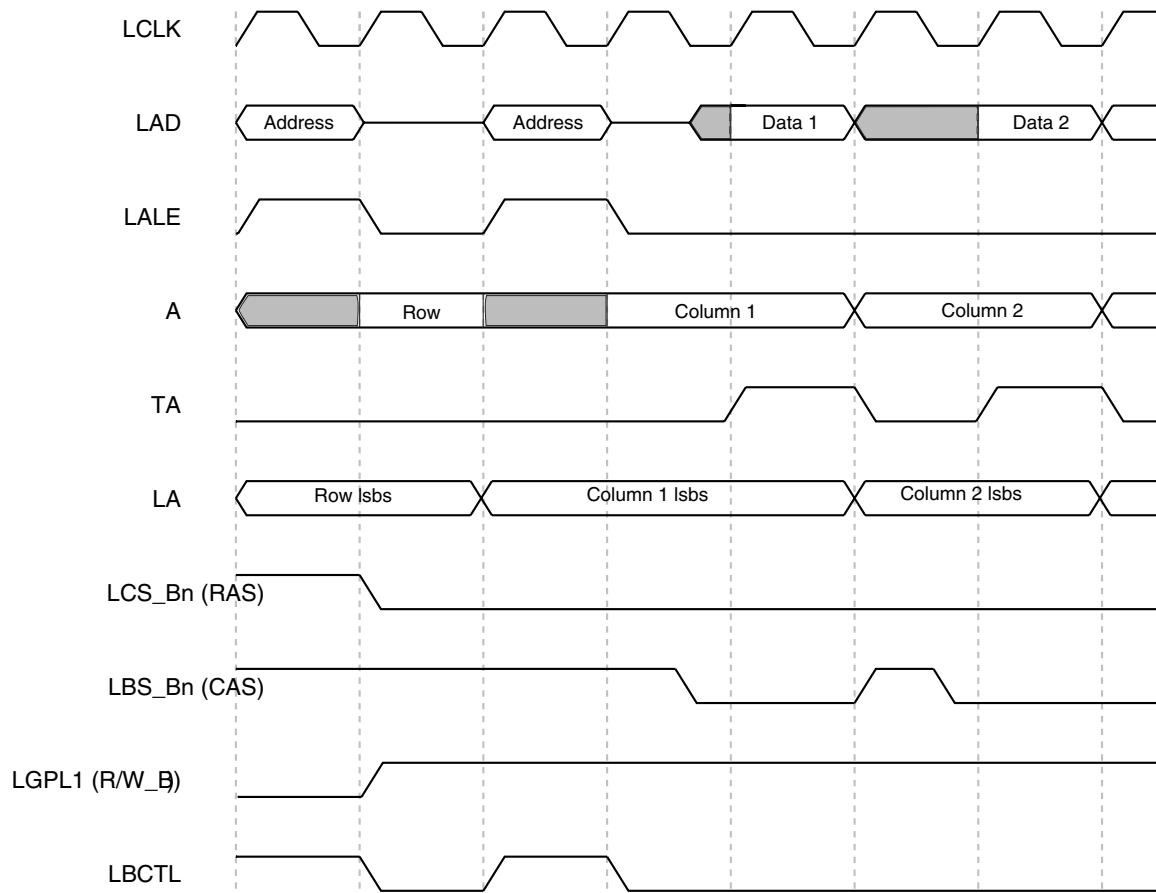


Figure 13-113. Burst read access to FPM DRAM using LOOP (two beats shown)

Table 13-214. Burst read access to FPM DRAM using LOOP (two beats shown)

cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12

Table continues on the next page...

**Table 13-214. Burst read access to FPM DRAM using LOOP (two beats shown)
(continued)**

g1t3	1		1	1	1	Bit 13
g2t1						Bit 14
g2t3						Bit 15
g3t1						Bit 16
g3t3						Bit 17
g4t1						Bit 18
g4t3						Bit 19
g5t1						Bit 20
g5t3						Bit 21
redo[0]						Bit 22
redo[1]						Bit 23
loop	0		1	1	0	Bit 24
exen	0		0	1	0	Bit 25
amx0	1		0	0	0	Bit 26
amx1	0		0	0	0	Bit 27
na	0		0	1	0	Bit 28
uta	0		0	1	0	Bit 29
todt	0		0	0	1	Bit 30
last	0		0	0	1	Bit 31
	RBS		RBS + 1	RBS + 2	RBS + 3	

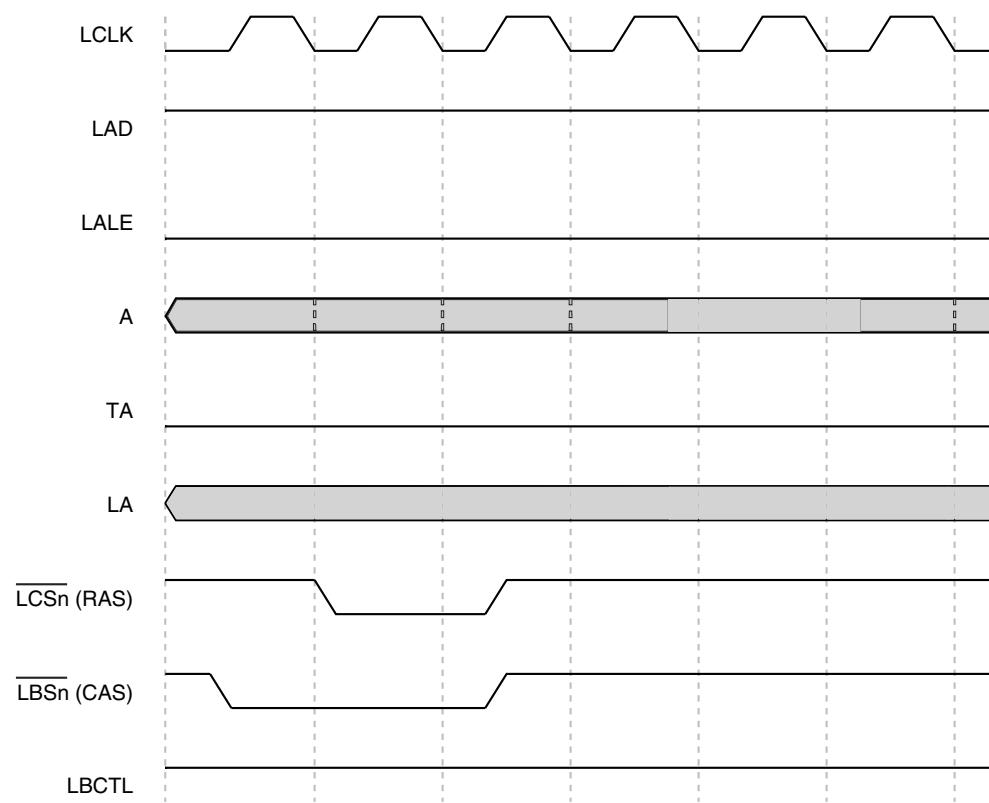


Figure 13-114. Refresh cycle (CBR) to FPM DRAM

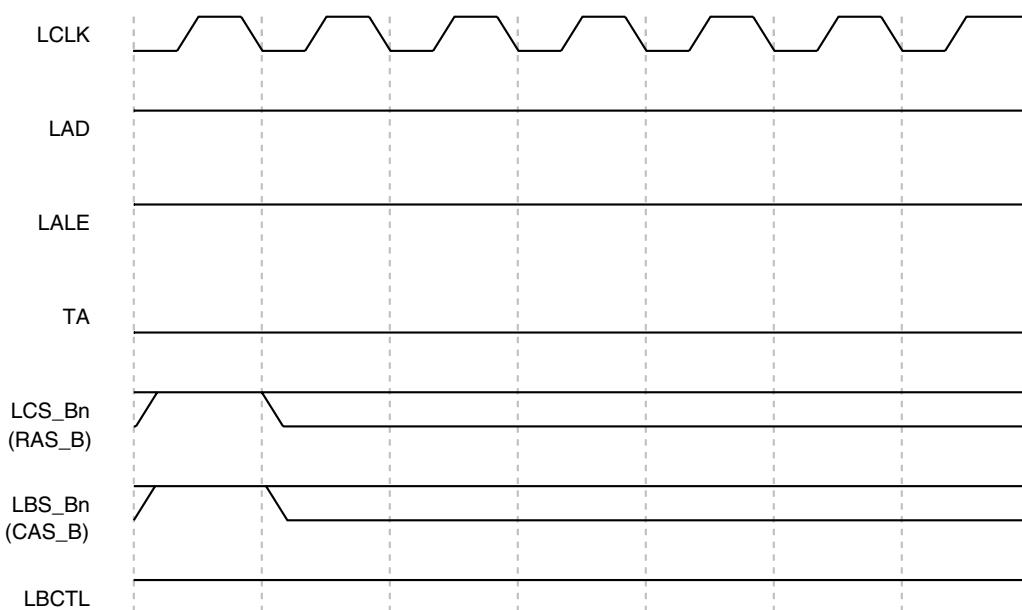
Table 13-215. Refresh cycle (CBR) to FPM DRAM

cst1	1	0	0	Bit 0
cst2	1	0	0	Bit 1
cst3	1	0	1	Bit 2
cst4	1	0	1	Bit 3
bst1	1	0	0	Bit 4
bst2	0	0	0	Bit 5
bst3	0	0	1	Bit 6
bst4	0	0	1	Bit 7
g0l0				Bit 8
g0l1				Bit 9
g0h0				Bit 10
g0h1				Bit 11
g1t1				Bit 12
g1t3				Bit 13
g2t1				Bit 14
g2t3				Bit 15

Table continues on the next page...

Table 13-215. Refresh cycle (CBR) to FPM DRAM (continued)

g3t1				Bit 16
g3t3				Bit 17
g4t1				Bit 18
g4t3				Bit 19
g5t1				Bit 20
g5t3				Bit 21
redo[0]				Bit 22
redo[1]				Bit 23
loop	0	0	0	Bit 24
exen	0	0	0	Bit 25
amx0	0	0	0	Bit 26
amx1	0	0	0	Bit 27
na	0	0	0	Bit 28
uta	0	0	0	Bit 29
todt	0	0	1	Bit 30
last	0	0	1	Bit 31
	PTS	PTS + 1	PTS + 2	

**Figure 13-115. Exception cycle****Table 13-216. Exception cycle**

cst1	1	Bit 0
------	---	-------

Table continues on the next page...

Table 13-216. Exception cycle (continued)

cst2	1	Bit 1
cst3	1	Bit 2
cst4	1	Bit 3
bst1	1	Bit 4
bst2	1	Bit 5
bst3	1	Bit 6
bst4	1	Bit 7
g0l0		Bit 8
g0l1		Bit 9
g0h0		Bit 10
g0h1		Bit 11
g1t1		Bit 12
g1t3		Bit 13
g2t1		Bit 14
g2t3		Bit 15
g3t1		Bit 16
g3t3		Bit 17
g4t1		Bit 18
g4t3		Bit 19
g5t1		Bit 20
g5t3		Bit 21
redo[0]		Bit 22
redo[1]		Bit 23
loop	0	Bit 24
exen	0	Bit 25
amx0	0	Bit 26
amx1	0	Bit 27
na	0	Bit 28
uta	0	Bit 29
todt	1	Bit 30
last	1	Bit 31
	EXS	

13.5.6 Interfacing to ZBT SRAM using UPM

ZBT SRAMs have been designed to optimize the performance of table access in networking applications.

This section describes how to interface to ZBT SRAMs. The figure below shows the connections. The UPM is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. As ZBT SRAMs are mostly used by performance-critical applications, it is assumed here that, typically, the maximum width of the local bus of 16 bits will be used.

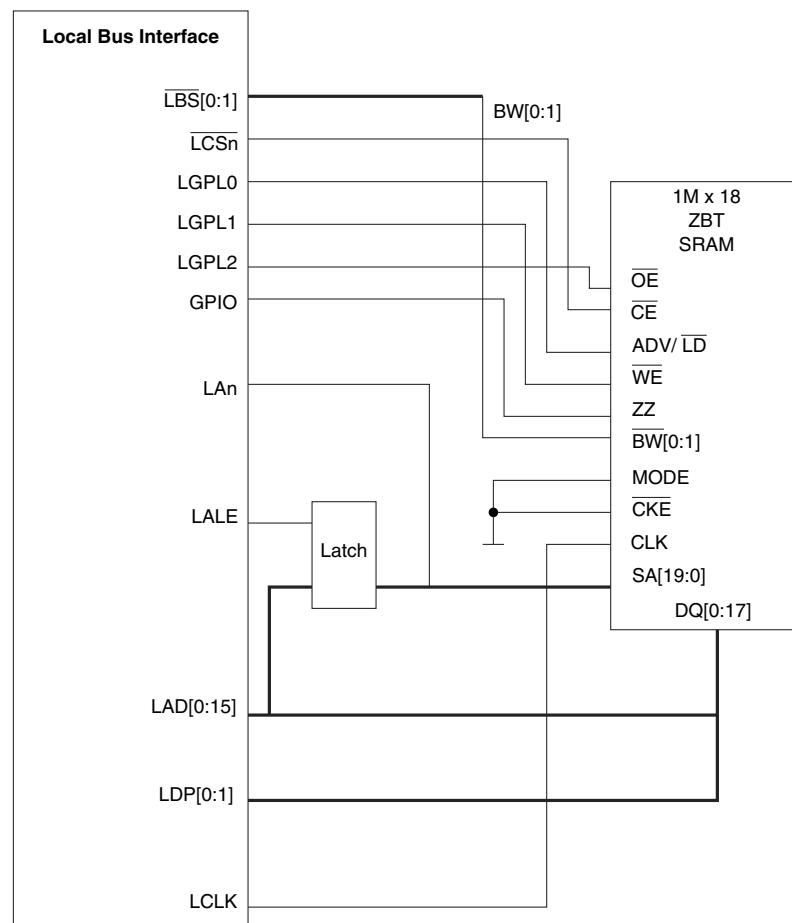


Figure 13-116. Interface to ZBT SRAM

ZBT SRAMs allow different configurations. For the local bus, the burst order should be set to linear burst order by tying the mode pin to GND. CKE_B should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the eLBC generates sixteen-beat transactions (for 16-bit ports) the UPM breaks down each burst into four consecutive four-beat bursts. The internal address generator of the eLBC generates the new LA{A21, A22} for the second, third, and fourth burst. In other words, because linear burst is used on the SRAM, the device itself bursts with the burst addresses of [0:1:2:3]. The local bus always generates linear bursts and expects [0:1:2:3:4:5:6:7:....:15]. Therefore, four

consecutive linear bursts of the ZBT SRAM with { A27, A28} = {0,0} for the first burst, { A27, A28} = { 0,1} for the second burst, {A27, A28} = {1,0} for the third burst, and {A27, A28} = {1,1} for the fourth burst give the desired burst pattern.

Chapter 14

Enhanced Serial Peripheral Interface

14.1 Introduction

The enhanced serial peripheral interface (eSPI) allows the device to exchange data with peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.

The eSPI is a full-duplex, synchronous, character-oriented channel that supports a simple interface (receive, transmit, clock and chip selects). The eSPI consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the eSPI baud rate generator in master mode. During an eSPI transfer, data is sent and received simultaneously.

The eSPI receiver and transmitter each have a FIFO of 32 bytes, as shown below. When the eSPI is disabled in the eSPI mode register (`SPMODE[EN] = 0`), it consumes little power.

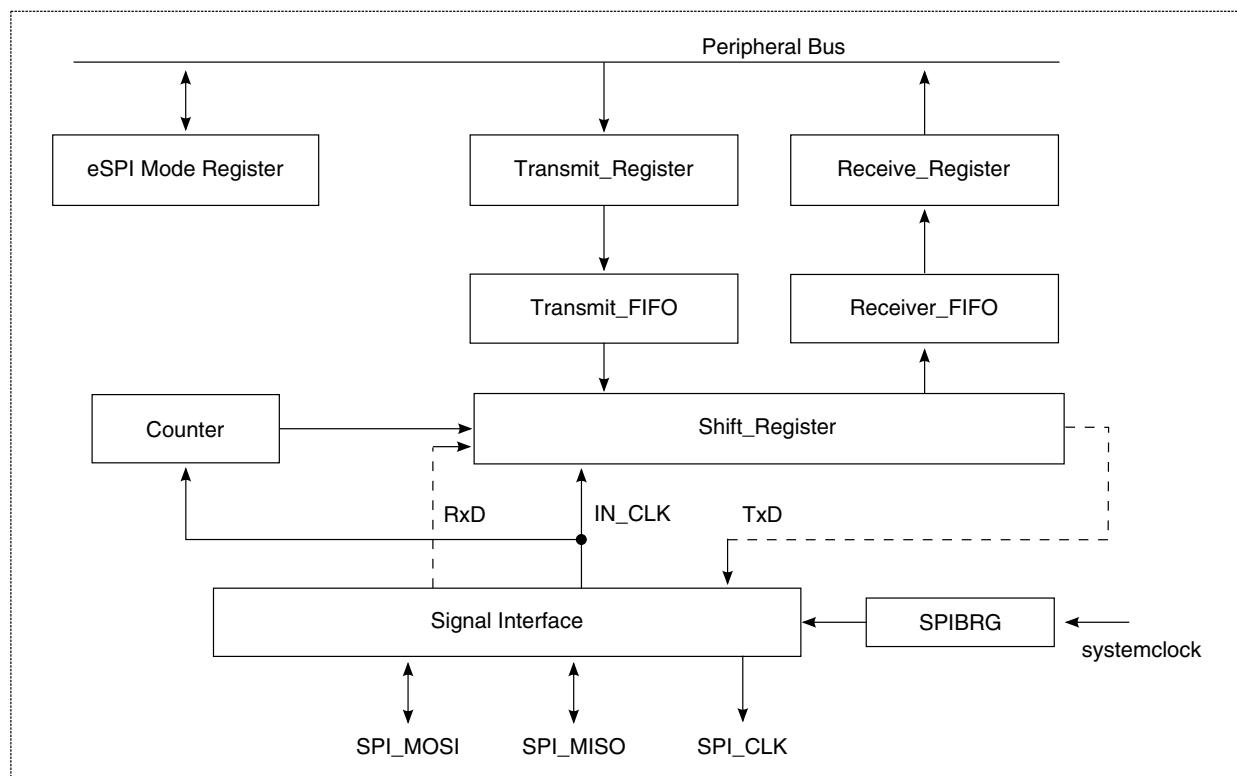


Figure 14-1. eSPI block diagram

14.1.1 Features

The major features of the eSPI are listed as follows:

- Interface contains SPI_MOSI, SPI_MISO, SPI_CLK, and 4 chip selects
- Supports eSPI master
- Supports RapidS™ full clock cycle operation
- Full-duplex or half-duplex master operation
- Supports Winbond dual output read
- Command in transaction level-easier for accessing eSPI devices
- Works with a range from 4-bit to 16-bit data characters
- Supports back-to-back character transmission and reception
- Supports reverse data mode for 8/16 bits data characters
- Supports single-master environment
- Maximum clock rate possible is (platform clock rate/2)
- Independent programmable baud rate generator
- Programmable clock phase and polarity.
- Supports four different configurations per chip select
- Local loopback capability for testing

14.1.2 eSPI transmission and reception process

As the eSPI is a character-oriented communication unit, the core is responsible for packing and unpacking the receive and transmit frames.

A frame consists of all of the characters transmitted or received during a completed eSPI transmission session, from the first character written to the eSPI transmit FIFO access register (SPITF) register to the last character transmitted with the total number as indicated in the command written to the [eSPI command register \(ESPI_SPCOM\)](#) register.

The core receives data by reading the eSPI receive FIFO access register (SPIRF) when the RNE ("not empty") bit in the eSPI event register (SPIE) is set.

The core transmits data by writing it into the SPITF register. After the core writes the final character to SPITF it waits for DON bit in the SPIE register to be set indicating frame was fully transmitted. It might then write a new command to SPCOM register.

The eSPI sets the TNF ("not full") bit in SPIE register whenever its transmit FIFO is not full.

The eSPI core handshake protocol can be implemented by using a polling or interrupt mechanism. When using a polling mechanism, the core reads the SPIE in a predefined frequency and acts according to the value of the SPIE bits. The polling frequency depends on the eSPI serial channel frequency. When using the interrupt mechanism, setting either the TNF (not full) or RNE (not empty) bits of the SPIE causes an interrupt to the core. The core then reads the SPIE and acts appropriately.

14.1.3 Modes of operation

The eSPI can be programmed to work in a single master environment. This section describes eSPI master operation in a single-master configuration.

In master mode, the eSPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single master with multiple slaves uses up to four chip select signals to selectively enable slaves, as shown below.

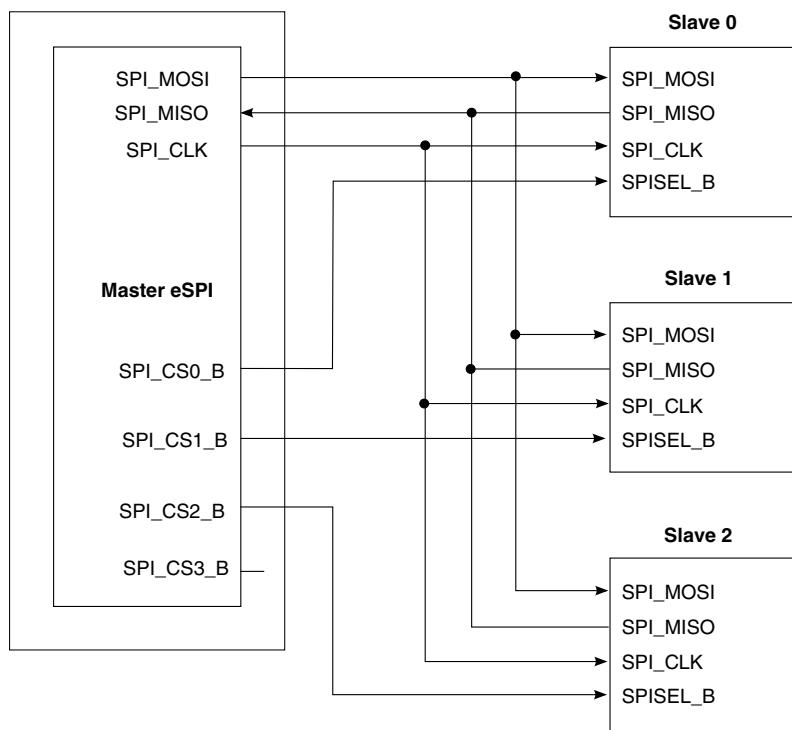


Figure 14-2. Single-master/multi-slave configuration

To start exchanging data, the core writes the data to be sent into the SPITF register. The eSPI then generates programmable clock pulses on SPI_CLK for each character. It shifts Tx data out on the "eSPI master-out slave-in" (SPI_MOSI) and Rx data in on the eSPI "master-in slave-out" (SPI_MISO) simultaneously. During the transmission process the core is responsible for supplying the data whenever the eSPI requests it to ensure smooth operation.

The maximum sustained data rate that the eSPI supports depends on the software latency. However, the eSPI can transfer a single character at very high rates—a maximum (up to system clock / 2) specified by the device (where system clock is defined as platform clock divided by 2). Gaps might be inserted between multiple frames.

14.2 External signal descriptions

The eSPI interface consists of transmit, receive, clock and chip selects.

The following table provides an overview of eSPI signal properties.

Table 14-1. Signal properties

Name	Function
SPI_MISO	Master input slave output
SPI_MOSI	Master output slave input or second master input slave output for Winbond dual output read.
SPI_CLK	Output serial clock connected to the other SPI_CLK.
SPI_CS_B [0:3]	eSPI slave select outputs

14.2.1 eSPI detailed signal descriptions

Table 14-2. ESPI detailed signal descriptions

Signal	I/O	Description	
SPI_MISO	I	Master input slave output	
		State meaning	Asserted-The data that has been received from the eSPI is high Negated-The data that has been received from the eSPI is low
		Timing	Assertion-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register) Negation-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register)
SPI_MOSI	I/O	Master output slave input or second master input slave output for Winbond dual output read	
		State meaning	Asserted-The data that has been transmitted from/to the eSPI is high Negated-The data that has been transmitted from/to the eSPI is low
		Timing	Assertion-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register). Negation-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register).
SPI_CLK	O	Serial clock out	
		State meaning	Assertion/negation-According to SPMODEx[PM,DIV16] register rate configuration
		Timing	Assertion/negation-During frame reception/transmission
SPI_CS_B [0:3]	O	eSPI slave select outputs	
		State meaning	Asserted- Slave 0, 1, 2, 3 is selected and master controls transmission/reception Negated-idle state
		Timing	Assertion-A predefined time before frame starts, during frame transmission/reception, a predefined time after frame ends Negation-When master is idle or controls another slave

The eSPI can be configured as a master in single master environment. The master eSPI generates the transfer clock SPI_CLK using the eSPI baud rate generator (BRG). The eSPI BRG takes as its input the platform clock divided by two.

SPI_CLK is a gated clock, active only during data transfers. Four combinations of SPI_CLK phase and polarity can be configured with the clock invert SPMODEEx[CIx] and clock phase SPMODEEx[CPx] register bits.

The eSPI master-in slave-out SPI_MISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPI_MOSI signal is an output for master devices and an input for slave devices. However, it also acts as a second input for master devices and as a second output for slave devices when using Winbond dual output read.

SPI_CLK is the clock output signal that shifts received data in from SPI_MISO and transmitted data out to SPI_MOSI. eSPI masters must output a slave select signal to enable eSPI slave devices.

14.3 Enhanced serial peripheral interface (eSPI) memory map

The table below shows the memory mapped registers of the eSPI and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address comprises CCSRBAR together with the SPI block base address and offset listed in the table below.

eSPI memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
11_0000	eSPI mode register (ESPI_SPMODE)	32	R/W	0000_0000h	14.3.1/795
11_0004	eSPI event register (ESPI_SPIE)	32	R/W	0020_0000h	14.3.2/797
11_0008	eSPI mask register (ESPI_SPIM)	32	R/W	0000_0000h	14.3.3/799
11_000C	eSPI command register (ESPI_SPCOM)	32	W	0000_0000h	14.3.4/801
11_0010	eSPI transmit FIFO access register (ESPI_SPITF)	32	W	0000_0000h	14.3.5/803
11_0014	eSPI receive FIFO access register (ESPI_SPIRF)	32	R	0000_0000h	14.3.6/805
11_0020	eSPI CS0 mode register (ESPI_SPMODE0)	32	R/W	0000_0000h	14.3.7/806
11_0024	eSPI CS1 mode register (ESPI_SPMODE1)	32	R/W	0010_0000h	14.3.8/808
11_0028	eSPI CS2 mode register (ESPI_SPMODE2)	32	R/W	0010_0000h	14.3.9/810
11_002C	eSPI CS3 mode register (ESPI_SPMODE3)	32	R/W	0010_0000h	14.3.10/812

14.3.1 eSPI mode register (ESPI_SPMODE)

The eSPI mode register (SPMODE) controls eSPI general operation mode.

Address: 11_0000h base + 0h offset = 11_0000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	LOOP							Reserved							
W																HO_ADJ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								Reserved							RXTHR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE field descriptions

Field	Description
0 EN	Enable eSPI. Any bits in SPMODE must not change when EN is set. 0 The eSPI is disabled. The eSPI is in a idle state and consumes minimal power. The eSPI BRG is not functioning and the input clock is disabled. 1 The eSPI is enabled.
1 LOOP	Loop mode. Enables local loopback operation. 0 Normal operation. 1 Loopback mode. Used to test the eSPI controller internal functionality, the transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that received data is ignored.
2–12 -	This field is reserved. Reserved
13–15 HO_ADJ	Data output hold adjustment. This field can be used for RapidS. 000 Output data is delayed by an extra 3 platform clock cycles 001 Output data is delayed by an extra 1 platform clock cycles 010 Output data is delayed by an extra 2 platform clock cycles 011 Output data is delayed by an extra 3 platform clock cycles 100 Output data is delayed by an extra 4 platform clock cycles 101 Output data is delayed by an extra 5 platform clock cycles 110 Output data is delayed by an extra 6 platform clock cycles 111 Output data is delayed by an extra 7 platform clock cycles

Table continues on the next page...

ESPI_SPMODE field descriptions (continued)

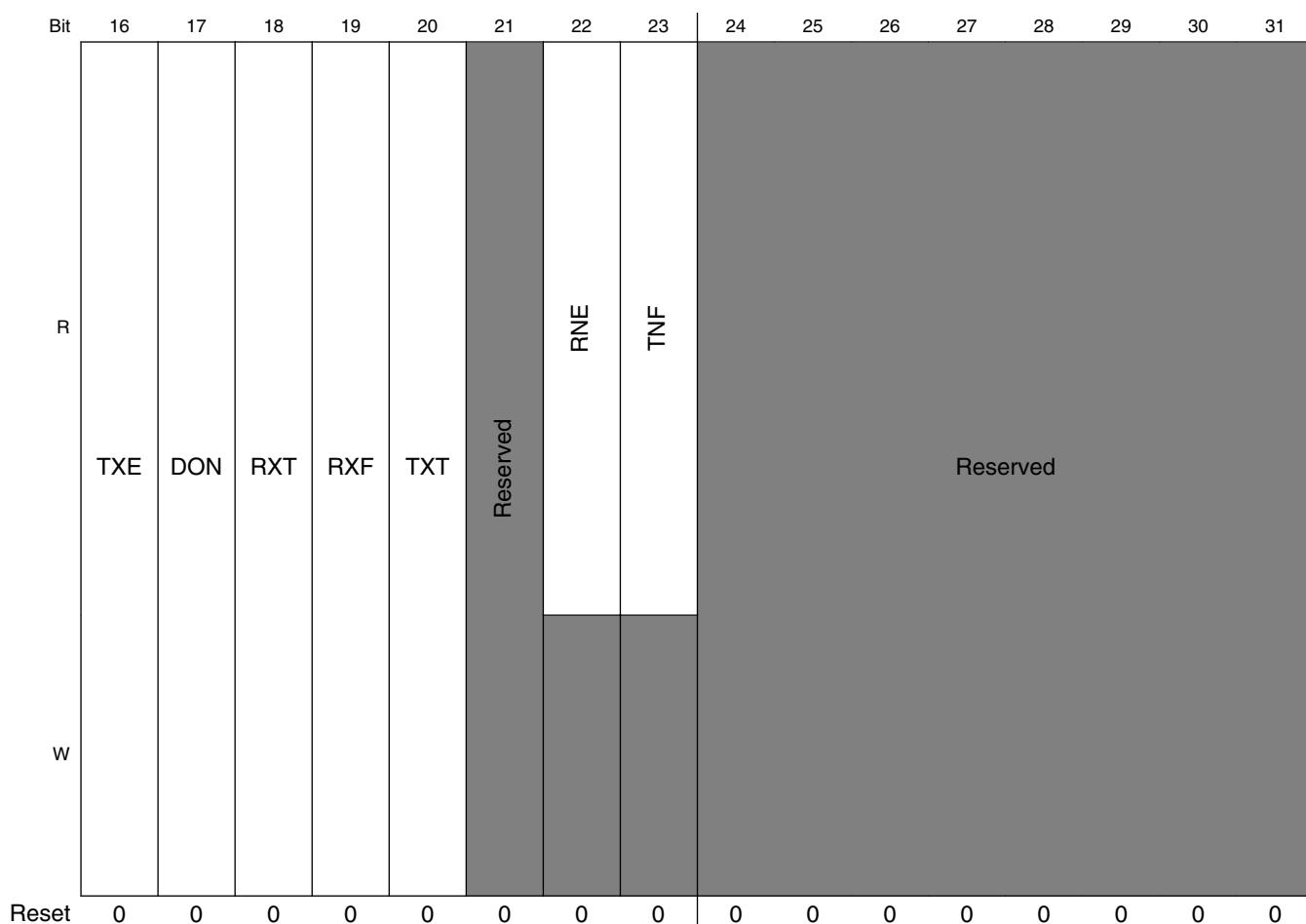
Field	Description
16–17 -	This field is reserved. Reserved
18–23 TXTHR	Tx FIFO threshold-if Tx FIFO has less than TXTHR bytes, an interrupt can be issued to the core. Valid values: 1-32
24–26 -	This field is reserved. Reserved
27–31 RXTHR	Rx FIFO threshold-if Rx FIFO has more than RXTHR bytes, an interrupt can be issued to the core. Valid values: 0-31

14.3.2 eSPI event register (ESPI_SPIE)

The eSPI event register (SPIE) generates interrupts and reports events recognized by the eSPI. When an event is recognized, the eSPI sets the corresponding SPIE bit. Clear SPIE bits by writing a 1-writing 0 has no effect. Setting a bit in the eSPI mask register (SPIM) enables interrupt and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the core clears internal interrupt requests. Bits RNE and TNF are status bits. Fields RXCNT and TXCNT hold Rx and Tx FIFO's statuses. They are not cleared as a result of writing to SPIE.

Address: 11_0000h base + 4h offset = 11_0004h

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



ESPI_SPIE field descriptions

Field	Description
0–1 -	This field is reserved. Reserved, should be cleared.
2–7 RXCNT	The current number of full Rx FIFO bytes NOTE: For character lengths of 9 to 16 bits-each character occupies 2 bytes in Rx/Tx FIFO.
8–9 -	This field is reserved. Reserved, should be cleared.
10–15 TXCNT	The current number of free Tx FIFO bytes NOTE: For character lengths of 9 to 16 bits-each character occupies 2 bytes in Rx/Tx FIFO
16 TXE	Tx FIFO is empty
17 DON	Last character was transmitted. The last character was transmitted and a new command can be written for the next frame
18 RXT	Rx FIFO has more than RXTHR bytes, that is, at least RXTHR + 1 bytes
19 RXF	Rx FIFO is full

Table continues on the next page...

ESPI_SPIE field descriptions (continued)

Field	Description
20 TXT	Tx FIFO has less than TXTHR bytes, that is, at most TXTHR - 1 bytes
21 -	This field is reserved. Reserved, should be cleared.
22 RNE	Not empty. Indicates that the Rx FIFO register contains a received character. 0 The Rx FIFO is empty 1 The Rx FIFO has a received character. The core can read the content of Rx FIFO through SPIRF.
23 TNF	Tx FIFO not full. 0 The transmitter FIFO is full. 1 The transmitter FIFO is not full.
24–31 -	This field is reserved. Reserved, should be cleared.

14.3.3 eSPI mask register (ESPI_SPIM)

The eSPI mask register (SPIM) enables/masks interrupts for events recognized by the eSPI. When an event is recognized, the eSPI sets the corresponding SPIE bit. Setting a bit in the eSPI mask register (SPIM) enables and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the core clears internal interrupt requests.

Bits RNE and TNF in SPIM are status bits. They are not cleared as a result of writing to SPIM.

Enhanced serial peripheral interface (eSPI) memory map

Address: 11_0000h base + 8h offset = 11_0008h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
TXE	DON	RXT	RXF	TXT	Reserved	RNE	TNF									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPIM field descriptions

Field	Description
0–15 -	This field is reserved. Reserved, should be cleared.
16 TXE	Tx FIFO empty interrupt mask 0 TXE event will not cause eSPI Interrupt 1 TXE event will cause eSPI Interrupt
17 DON	Last character transmitted mask 0 DON event will not cause eSPI Interrupt 1 DON event will cause eSPI Interrupt
18 RXT	Rx threshold interrupt mask 0 RXT event will not cause eSPI Interrupt 1 RXT event will cause eSPI Interrupt
19 RXF	Rx FIFO full interrupt mask 0 RXF event will not cause eSPI Interrupt 1 RXF event will cause eSPI Interrupt
20 TXT	Tx threshold interrupt mask 0 TXT event will not cause eSPI Interrupt 1 TXT event will cause eSPI Interrupt
21 -	This field is reserved. Reserved, should be cleared.
22 RNE	Rx not empty interrupt mask 0 Not Empty event will not cause eSPI Interrupt 1 Not Empty event will cause eSPI Interrupt
23 TNF	Tx not full interrupt mask 0 Not full event will not cause eSPI Interrupt 1 Not full event will cause eSPI Interrupt
24–31 -	This field is reserved. Reserved, should be cleared.

14.3.4 eSPI command register (ESPI_SPCOM)

The eSPI command register (SPCOM) is used by the host to supply information on the new frame.

After SPCOM has been written to initiate the first transaction after startup, commands can be executed only after SPIE[DON] is set. Otherwise they are ignored.

A transaction can be full duplex (regular eSPI) or half duplex. Half duplex can be used for example for write accesses to a flash (only transmit) or for a read access from a flash (first part is transmit without receive, while the second part is receive without transmit).

Enhanced serial peripheral interface (eSPI) memory map

Address: 11_0000h base + Ch offset = 11_000Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	CS	RxDelay	DO	TO	HLD	Reserved		RxSKIP								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	TRANLEN															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPCOM field descriptions

Field	Description
0–1 CS	Chip select-chip select for which transaction is destined 00 SPI_CS0_B 01 SPI_CS1_B 10 SPI_CS2_B 11 SPI_CS3_B
2 RxDelay	RxDelay 0 Normal eSPI operation 1 Rx data should be sampled a bit later than regular eSPI (used for full cycle operation such as with Atmel RapidS devices)
3 DO	This mode is useful only for character lengths of 4,6,8. DO and RapidS should not be set simultaneously. 0 Normal eSPI operation 1 Winbond dual output read-when eSPI master reads data 2 data bits are available (on MISO and MOSI)
4 TO	Transmit only 1 No reception is done for the frame (useful for write transactions) 0 Normal operation
5 HLD	HLD 0 Normal operation 1 Mask first generated SPI_CLK. Should be used only for RapidS mode0

Table continues on the next page...

ESPI_SPCOM field descriptions (continued)

Field	Description
6–7 -	This field is reserved. Reserved, should be cleared.
8–15 RxSKIP	If ($\text{RXSKIP} \neq 0$)–Number of characters skipped for reception from frame start. Non-zero values of RxSKIP force the eSPI to half-duplex mode, and therefore this causes TRANLEN-RxSKIP characters to be skipped for transmission. RXSKIP is useful for reads of SPI flash memories where the first valid read data is received several characters after the transmission begins (after the eSPI has transmitted an instruction opcode and address). NOTE: If TO = 1, RxSKIP must be set to 0. NOTE: If RXSKIP = 0 and TO = 0, the eSPI changes to full duplex mode.
16–31 TRANLEN	Transaction length - (number of characters in the frame - 1)

14.3.5 eSPI transmit FIFO access register (ESPI_SPITF)

The 32-bit write-only eSPI transmit FIFO access register (SPITF) holds the characters to be written to the transmit FIFO. The number of bits in each character is specified by SPMODEEx[LENx]. Each time SPIE[TNF] is set, the core can write more data to the SPITF register, if there is no error indication in the SPIE.

For character lengths of 4 to 8 bits, SPITF contains up to 4 characters (unless end of frame). The lsbs are in bits 7, 15, 23, and 31 of SPITF.

For character lengths of 9 to 16 bits, SPITF contains up to 2 characters (unless end of frame). For 16 bits with SPMODEEx[REVx] = 1, the lsb is in bits 15 and 31 of SPITF. For other options, lsbs are in bits 7 and 23 while msbs are in bits (23-LENx) and (39-LENx) of SPITF.

For example: REV = 0, LEN = 10 (0xA), SPITF[0-15] = 0xFB05-bitstream is: (lsb first) 1101111101 (msb last).

NOTE

The user must write N bytes of SPITF ($1 \leq N \leq 4$) that do not exceed the number of free bytes in the transmit FIFO. It is valid for the user to write only 1 or 2 bytes of SPITF (at offset 0x010) if the user wishes to write fewer characters than the maximum supported by SPITF for the particular character length in use.

The following figures show examples of the contents of SPITF with various parameters set.

SPITF Example - SPMODEx[REVx]=0, SPMODEx[LENx]=3, LSB Sent First:

	0	3	4	5	6	7	8	11	12	13	14	15		16	19	20	21	22	23	24	27	28	29	30	31
R																									
W	-	MSB 0	Data 0	LSB0	-	MSB 1	Data 1	LSB1	-	MSB 2	Data 2	LSB2	-	MSB 3	Data 3	LSB3									

SPITF Example - SPMODEx[REVx]=x, SPMODEx[LENx]=7:

	0	1	6	7	8	9		14	15		16	17		22	23	24			30	31			
R																							
W	MSB0	Data 0	LSB 0	MSB1	Data 1	LSB 1	MSB2	Data 2	LSB 2	MSB3	Data 3	LSB 3											

SPITF Example - SPMODEx[REVx]=0, SPMODEx[LENx]=10, LSB Sent First:

	0		6	7	8		12	13	14	15	16		22	23	24		28	29	30	31	
R																					
W	Data 0 LS Byte	LSB0	-	MSB 0	Data 0 MS Byte		Data 1 LS Byte	LSB1	-	MSB 1	Data 1 MS Byte										

SPITF Example - SPMODEx[REVx]=1, SPMODEx[LENx]=15, MSB Sent First:

	0	1		7	8		14	15		16	17		23	24		30	31
R																	
W	MSB0	Data 0 MS Byte		Data 0 LS Byte	LSB0	MSB1		Data 1 MS Byte		Data 1 LS Byte	LSB1						

Address: 11_0000h base + 10h offset = 11_0010h

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

ESPI_SPITF field descriptions

Field	Description
0–31 DATA	Varies as parameters set

14.3.6 eSPI receive FIFO access register (ESPI_SPIRF)

The 32-bit read-only eSPI receive data register (SPIRF) is used to hold characters read from the receive FIFO. Each time SPIE[RNE] is set, the core can read the SPIRF register.

For character lengths of 4 to 8 bits, SPIRF contains up to 4 characters. The msbs are in bits 0, 8, 16, and 24. For character lengths of 9 to 16 bits, SPIRF contains up to 2 characters. The msbs are in bits 0 and 16. SPMODEx[REVx] does not affect the msb or lsb bit positions when reading the SPIRF register.

The user must read N bytes of SPIRF ($1 \leq N \leq 4$) that do not exceed the amount of data in the receive FIFO. The user can read less bytes than the amount of data in the receive FIFO. For example, a 1-byte read of SPIRF when configured for 8-bit characters with 4 characters of data in the receive FIFO results in the 3 unread characters shuffling down to the lower 24 bits of SPIRF in preparation for the following SPIRF read.

SPIRF Example - SPMODEx[LENx]=3

	0	1	2	3	4	7	8	9	10	11	12	15	16	17	18	19	20	23	24	25	26	27	28	31
R	MSB0	Data 0	LSB0	-	MSB1	Data 1	LSB1	-		MSB2	Data 2	LSB2	-	MSB3	Data 3	LSB3	-							
W																								

SPIRF Example - SPMODEx[LENx]=10:

	0	1	7	8	9	10	11	15	16	17	23	24	25	26	27	31
R	MSB0	Data 0 MS Byte		Data 0 LS Byte	LSB0	-		MSB1	Data 1 MS Byte		Data 1 LS Byte	LSB1	-			
W																

SPIRF Example - SPMODEx[LENx]=15:

	0	1	7	8	14	15	16	17	23	24	30	31
R	MSB0	Data 0 MS Byte		Data 0 LS Byte	LSB0	MSB1		Data 1 MS Byte		Data 1 LS Byte	LSB1	
W												

Address: 11_0000h base + 14h offset = 11_0014h

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

ESPI_SPIRF field descriptions

Field	Description
0–31 DATA	Varies as parameters set

14.3.7 eSPI CS0 mode register (ESPI_SPMODE0)

The eSPI CS0 mode register (SPMODE0) controls eSPI master operation with chip select 0.

Address: 11_0000h base + 20h offset = 11_0020h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CI0	CP0	REV0	DIV160	PM0				ODD0	Reserved		POL0	LENO			
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CS0BEF				CS0AFT				CS0CG				Reserved			
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE0 field descriptions

Field	Description
0 CI0	Clock invert. Inverts eSPI clock polarity. See Figure 14-13 and Figure 14-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP0	Clock phase. Selects the transfer format. See Figure 14-13 and Figure 14-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV0	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV160	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. NOTE: System clock as used here is defined to be platform clock divided by 2.

Table continues on the next page...

ESPI_SPMODE0 field descriptions (continued)

Field	Description
	<p>0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.</p>
4–7 PM0	<p>Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by $2 \times ([PM] + 1)$, a range from 2 to 32. For example, if the prescale modulus is set to PM=0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be $16 \times (2 \times (0x0011 + 1)) = 128$</p> <p>NOTE: System clock as used here is defined to be platform clock divided by 2</p>
8 ODD0	<p>ODD0</p> <p>0 Even division - $2 \times (PM + 1) \times (15 \times DIV16 + 1)$ - 50% duty cycle 1 Odd division - $(2 \times PM + 1) \times (15 \times DIV16 + 1)$ (except for PM = 0 where it divides by $2 \times (7 \times DIV16 + 1)$); duty cycle is $(PM + 1) \div (2 \times PM + 1)$ for DIV16 = 0; duty cycle is 50% for DIV16 = 1.</p>
9–10 -	This field is reserved. Reserved
11 POL0	<p>CS0 Polarity.</p> <p>1 Asserted Low, Negated High 0 Asserted High, Negated Low.</p>
12–15 LENO	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS0BEF	<p>CS assertion time in bits before frame start (that is, before clock toggles)</p> <p>Example: CS0BEF = 0010 inserts 2 bits time gap between CS0 assertion to clock toggle</p>
20–23 CS0AFT	<p>CS assertion time in bits after frame end (that is, after clock finishes toggling)</p> <p>Example: CS0AFT = 0010 inserts 2 bits time gap between clock stop to CS0 negation</p>
24–28 CS0CG	<p>Clock gap</p> <p>insert gaps between transmitted frames according to this size (during this time, chip select is negated).</p> <p>Chip select is negated minimum time of 1 bit time.</p> <p>Example: CS0CG = 00101 inserts $5 + 1 = 6$ bits time gap between every two consecutive frames</p>
29–31 -	This field is reserved. Reserved

14.3.8 eSPI CS1 mode register (ESPI_SPMODE1)

The eSPI CS1 mode register (SPMODE1) controls eSPI master operation with chip select 1.

Address: 11_0000h base + 24h offset = 11_0024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CI1	CP1	REV1	DIV161	PM1				ODD1	Reserved		POL1	LEN1			
W	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CS1BEF				CS1AFT				CS1CG				Reserved			
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE1 field descriptions

Field	Description
0 CI1	Clock invert. Inverts eSPI clock polarity. See Figure 14-13 and Figure 14-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP1	Clock phase. Selects the transfer format. See Figure 14-13 and Figure 14-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV1	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV161	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. NOTE: System clock as used here is defined to be platform clock divided by 2. 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM1	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by $2 \times ([PM] + 1)$, a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be $16 \times (2 \times (0x0011 + 1)) = 128$ NOTE: System clock as used here is defined to be platform clock divided by 2.

Table continues on the next page...

ESPI_SPMODE1 field descriptions (continued)

Field	Description
8 ODD1	ODD1 0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1.
9–10 -	This field is reserved. Reserved
11 POL1	CS1 Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN1	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS1BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS1BEF = 0010 inserts 2 bits time gap between CS1 assertion to clock toggle.
20–23 CS1AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS1AFT = 0010 inserts 2 bits time gap between clock stop to CS1 negation.
24–28 CS1CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames.
29–31 -	This field is reserved. Reserved

14.3.9 eSPI CS2 mode register (ESPI_SPMODE2)

The eSPI CS2 mode register (SPMODE2) controls the eSPI master operation with chip select 2.

Address: 11_0000h base + 28h offset = 11_0028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CI2	CP2	REV2	DIV162	PM2				ODD2	Reserved		POL2	LEN2			
W	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CS2BEF				CS2AFT				CS2CG				Reserved			
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE2 field descriptions

Field	Description
0 CI2	Clock invert. Inverts eSPI clock polarity. See Figure 14-13 and Figure 14-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP2	Clock phase. Selects the transfer format. See Figure 14-13 and Figure 14-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV2	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV162	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. NOTE: System clock as used here is defined to be platform clock divided by 2. 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM2	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by $2 \times ([PM] + 1)$, a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the system clock/SPI_CLK ratio will be $16 \times (2 \times (0x0011 + 1)) = 128$ NOTE: System clock as used here is defined to be platform clock divided by 2.

Table continues on the next page...

ESPI_SPMODE2 field descriptions (continued)

Field	Description
8 ODD2	ODD2 0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1
9–10 -	This field is reserved. Reserved
11 POL2	CS2 Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN2	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS2BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS2BEF = 0010 inserts 2 bits time gap between CS2 assertion to clock toggle
20–23 CS2AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS2AFT = 0010 inserts 2 bits time gap between clock stop to CS2 negation
24–28 CS2CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames
29–31 -	This field is reserved. Reserved

14.3.10 eSPI CS3 mode register (ESPI_SPMODE3)

The eSPI CS3 mode register (SPMODE3) controls the eSPI master operation with chip select 3.

Address: 11_0000h base + 2Ch offset = 11_002Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CI3	CP3	REV3	DIV163	PM3				ODD3	Reserved		POL3	LEN3			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CS3BEF				CS3AFT				CS3CG				Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE3 field descriptions

Field	Description
0 CI3	Clock invert. Inverts eSPI clock polarity. See Figure 14-13 and Figure 14-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP3	Clock phase. Selects the transfer format. See Figure 14-13 and Figure 14-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV3	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first - for 8/16 bits data character only
3 DIV163	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. NOTE: System clock as used here is defined to be platform clock divided by 2 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM3	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by $2 \times ([PM] + 1)$, a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be $16 \times (2 \times (0x0011 + 1)) = 128$ NOTE: System clock as used here is defined to be platform clock divided by 2

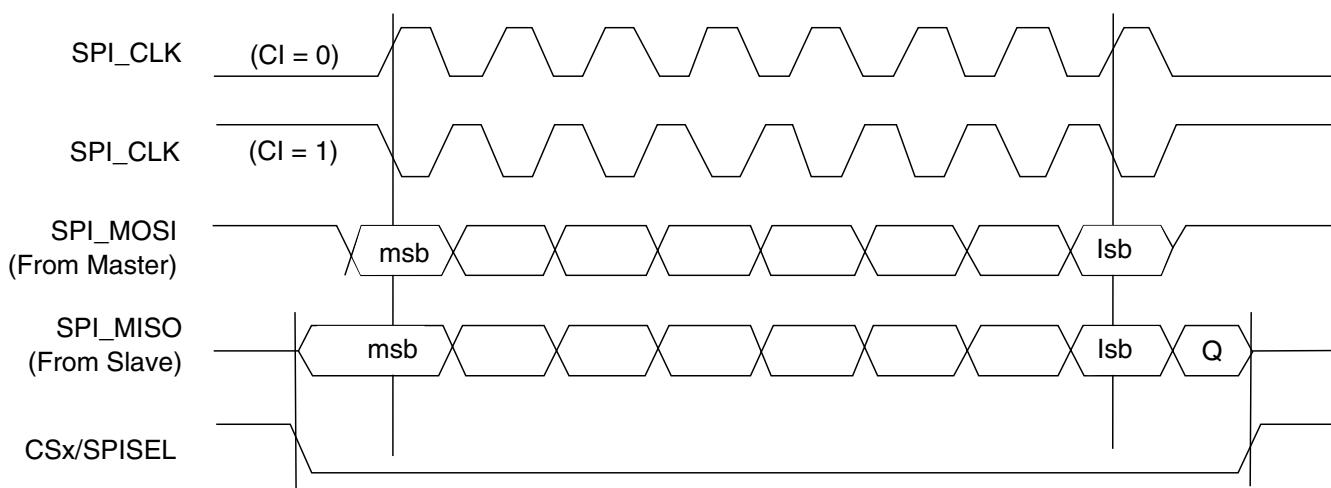
Table continues on the next page...

ESPI_SPMODE3 field descriptions (continued)

Field	Description
8 ODD3	ODD3 0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1
9–10 -	This field is reserved. Reserved
11 POL3	CS Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN3	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS3BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS3BEF = 0010 inserts 2 bits time gap between CS3 assertion to clock toggle
20–23 CS3AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS3AFT = 0010 inserts 2 bits time gap between clock stop to CS3 negation
24–28 CS3CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames
29–31 -	This field is reserved. Reserved

14.4 eSPI transfer formats

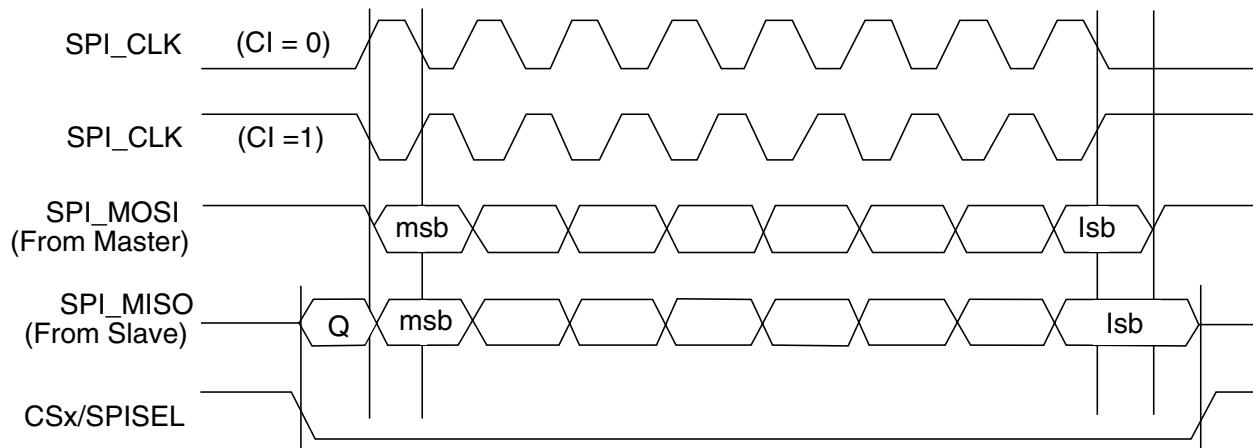
Figure below shows the eSPI transfer format in which SPI_CLK starts toggling in the middle of the transfer (SPMODEEx[CPx] = 0).

CI and CP values for various eSPI devices

NOTE: Q = Undefined Signal.

Figure 14-13. eSPI transfer format with SPMODEEx[CPx] = 0

Figure below shows the eSPI transfer format in which SPI_CLK starts toggling at the beginning of the transfer (SPMODEEx[CPx] = 1).



NOTE: Q = Undefined Signal.

Figure 14-14. eSPI transfer format with SPMODEEx[CPx] = 1

14.5 CI and CP values for various eSPI devices

1. Regular devices-eSPI mode0-CI = CP = 0
2. Regular devices-eSPI mode3-CI = CP = 1

For Winbond devices DO should also be set for dual output read command.

3. RapidS mode0-CI = 0, CP = 1, HLD = 1
4. RapidS mode3-CI = 1, CP = 0

14.6 eSPI programming examples

This section provides eSPI programming examples for 24-bit address memory and 16-bit address memory.

14.6.1 24-bit address example

The following sequence initializes the eSPI to read 36 bytes from 24-bit address memory, start address = 0x00_0040:

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write 0xFFFF_FFFF to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 0x8000_100F to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 0x2417_1108-REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1.
5. Configure SPITF = 0x0300_0040-0x03 is read opcode while 0x00_0040 is the 24-bit start address.
6. Configure SPCOM = 0x0004_0027 so 4 bytes are skipped (1 for opcode and 3 for 24-bit address), TRANLEN = $36 + 4 - 1 = 39 = 27h$.

14.6.2 16-bit address example

The following sequence initializes the eSPI to read 36 bytes from 16-bit address memory, start address = 0x0040:

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write 0xFFFF_FFFF to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 0x8000_100F to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 0x2417_1108-REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1.
5. Configure SPITF = 0x0300_40xx (xx is don't care)-0x03 is read opcode while 0x0040 is the 16-bit start address.
6. Configure SPCOM = 0x0003_0026 so 3 bytes are skipped (1 for opcode and 2 for 16-bit address), TRANLEN = $36 + 3 - 1 = 38 = 26h$.

Chapter 15

Enhanced Secure Digital Host Controller

15.1 eSDHC overview

The enhanced secure digital host controller (eSDHC) provides an interface between the host system and these types of memory cards:

- MultiMediaCard (MMC)

MMC is a universal low-cost data storage and communication medium designed to cover a wide area of applications including mobile video and gaming, which are available from either pre-loaded MMC cards or downloadable from cellular phones, WLAN, or other wireless networks. Old MMC cards are based on a seven-pin serial bus with a single data pin, while the new high-speed MMC communication is based on an advanced 11-pin serial bus designed to operate in a low voltage range.

- Secure digital (SD) card

The secure digital (SD) card is an evolution of old MMC technology. It is specifically designed to meet the security, capacity, performance, and environment requirements inherent in the emerging audio and video consumer electronic devices. The physical form factor, pin assignments, and data transfer protocol are forward-compatible with the old MMC.

The eSDHC acts as a bridge, passing host bus transactions to SD/MMC cards by sending commands and performing data accesses to or from the cards. It handles the SD/MMC protocol at the transmission level.

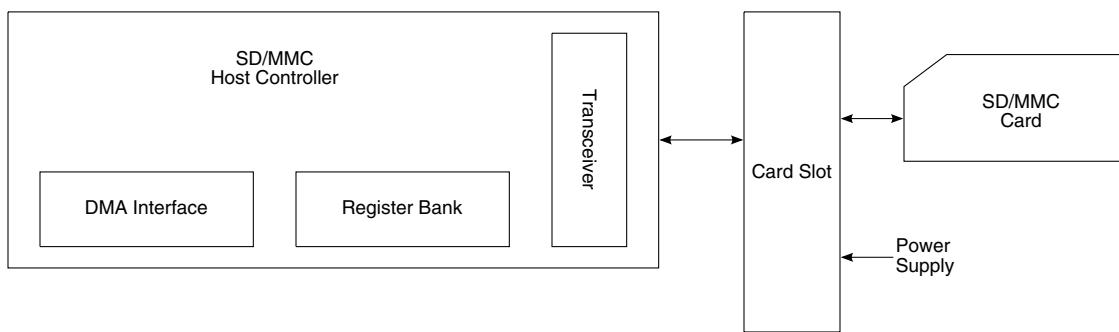


Figure 15-1. System connection of the eSDHC

This figure shows the eSDHC block diagram.

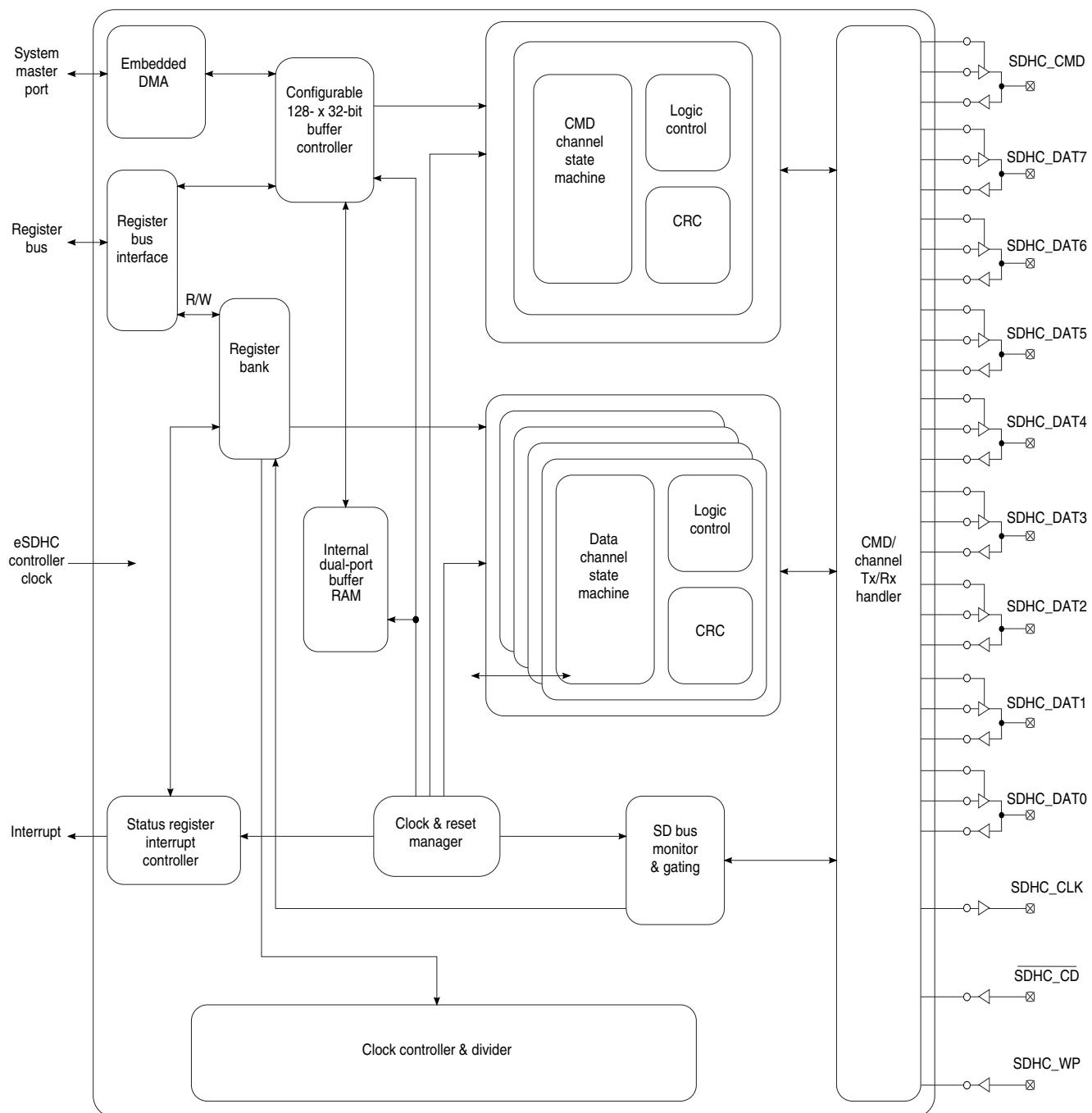


Figure 15-2. eSDHC block diagram

15.2 eSDHC features summary

The eSDHC includes the following features:

- Compatible with the following specifications:

- *SD Host Controller Standard Specification, Version 2.0* (<http://www.sdcards.org>) with test event register support
- *MultiMediaCard System Specification, Version 4.2* (<http://www.mmca.org>)
- *SD Memory Card Specification, Version 2.0* (<http://www.sdcards.org>)
- Designed to work with SD Memory, SDHC memory, miniSD Memory, MMC, MMCplus, and RS-MMC cards
- Card bus clock frequency up to 52 MHz
- Supports 1-/4-bit SD mode, 1-/4-/8-bit MMC modes
 - Up to 200 Mbps data transfer for SD/MMC cards using four parallel data lines
 - Up to 416 Mbps data transfer for MMC using 8 parallel data lines
- Single- and multi-block read and write
- Write-protection switch for write operations
- Synchronous abort
- Pause during the data transfer at a block gap
- Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer commands while the data transfer is in progress
- Fully configurable 128 x 32-bit FIFO for read/write data
- Internal DMA capabilities

15.2.1 Data transfer modes

The eSDHC can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit
- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification mode (up to 400 kHz)
- MMC full-speed mode (up to 20 MHz)
- MMC high-speed mode (up to 52 MHz)
- SD full-speed mode (up to 25 MHz)
- SD high-speed mode (up to 50 MHz)

15.3 eSDHC external signal description

The eSDHC has six chip I/O signals.

- SDHC_CLK is the internally generated clock signal that drives the MMC or SD card.
- SDHC_CMD I/O sends commands and receives responses from the card.

- SDHC_DAT7 -SDHC_DAT0 perform data transfers between the eSDHC and the card. If the eSDHC is desired to support a 4-bit data transfer, SDHC_DAT7-SDHC_DAT4 can also be optional and tied high.
- SDHC_CD_B and SDHC_WP are card detection and write protection signals from the socket.

Table 15-1. Signal properties

Name	Port	Function	Reset State	Pull up/Pull down Required
SDHC_CLK	O	Clock for MMC/SD card	0	N/A
SDHC_CMD	I/O	Command line to card	High impedance	Pull up
SDHC_DAT7	I/O	8-bit mode: DAT7 line not used in other modes	High impedance	Pull up
SDHC_DAT6	I/O	8-bit mode: DAT6 line not used in other modes	High impedance	Pull up
SDHC_DAT5	I/O	8-bit mode: DAT5 line not used in other modes	High impedance	Pull up
SDHC_DAT4	I/O	8-bit mode: DAT4 line in not used in other modes	High impedance	Pull up
SDHC_DAT3	I/O	4-/8-bit mode: DAT3 line	High impedance	Pull up. Do not use DAT3 pin as a CD pin.
SDHC_DAT2	I/O	4-/8-bit mode: DAT2 line	High impedance	Pull up
SDHC_DAT1	I/O	8-bit mode: DAT1 line 4-bit mode: DAT1 line	High impedance	Pull up
SDHC_DAT0	I/O	8-bit mode: DAT0 line 4-bit mode: DAT0 line	High impedance	Pull up
SDHC_WP	I	Card write protect detect; if not used, tied low	N/A	N/A
SDHC_CD_B	I	Card detection pin; if not used, tie high. Low Card present High No card present	N/A	N/A

15.4 Enhanced Secure Digital Host Controller (eSDHC) Memory Map

The table below shows the memory mapped registers of the eSDHC module and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the eSDHC block base address and offset listed in the following table. Undefined 4-byte address spaces within offset 0x0000-0xFFFF are reserved.

NOTE

All eSDHC registers must be accessed as aligned 4-byte quantities. Accesses to the eSDHC registers that are less than 4-bytes are not supported.

NOTE

The addresses following 0x044, except 0x050, 0x0FC and 0x40C, are reserved and read as all 0s. Writes to these registers are ignored.

eSDHC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_4000	DMA system address (eSDHC_DSADDR)	32	R/W	0000_0008h	15.4.1/823
11_4004	Block attributes (eSDHC_BLKATTR)	32	R/W	0000_0008h	15.4.2/823
11_4008	Command argument (eSDHC_CMDARG)	32	R/W	0000_0000h	15.4.3/824
11_400C	Command transfer type (eSDHC_XFERTYP)	32	R/W	0000_0000h	15.4.4/825
11_4010	Command response n (eSDHC_CMDRSP0)	32	R	0000_0000h	15.4.5/828
11_4014	Command response n (eSDHC_CMDRSP1)	32	R	0000_0000h	15.4.5/828
11_4018	Command response n (eSDHC_CMDRSP2)	32	R	0000_0000h	15.4.5/828
11_401C	Command response n (eSDHC_CMDRSP3)	32	R	0000_0000h	15.4.5/828
11_4020	Data buffer access port (eSDHC_DATPORT)	32	R/W	0000_0000h	15.4.6/829
11_4024	Present state (eSDHC_PRSSTAT)	32	R	See section	15.4.7/830
11_4028	Protocol control (eSDHC_PROCTL)	32	R/W	0000_0020h	15.4.8/835
11_402C	System control (eSDHC_SYSCTL)	32	R/W	0000_8008h	15.4.9/838
11_4030	Interrupt status (eSDHC_IRQSTAT)	32	R/W	0000_0000h	15.4.10/841
11_4034	Interrupt status enable (eSDHC_IRQSTATEN)	32	R/W	117F_013Fh	15.4.11/846
11_4038	Interrupt signal enable (eSDHC IRQSIGEN)	32	R/W	0000_0000h	15.4.12/849
11_403C	Auto CMD12 status (eSDHC_AUTOC12ERR)	32	R	0000_0000h	15.4.13/851
11_4040	Host controller capabilities (eSDHC_HOSTCAPBLT)	32	R	0163_0000h	15.4.14/854
11_4044	Watermark level (eSDHC_WML)	32	R/W	0810_0810h	15.4.15/856

Table continues on the next page...

eSDHC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_4050	Force event (eSDHC_FEVT)	32	W	0000_0000h	15.4.16/ 857
11_40FC	Host controller version (eSDHC_HOSTVER)	32	R	0000_1201h	15.4.17/ 859
11_440C	DMA control register (eSDHC_DCR)	32	R/W	0000_0000h	15.4.18/ 860

15.4.1 DMA system address (eSDHC_DSADDR)

The DMA system address register contains the lower 32-bits of the system memory address used for DMA transfers. Only access this register when no transactions are executing (after transactions have stopped). The host driver should wait until PRSSTAT[DLA] is cleared.

Address: 11_4000h base + 0h offset = 11_4000h

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

eSDHC_DSADDR field descriptions

Field	Description
0–31 DS_ADDR	DMA system address , lower 32 bits . When the eSDHC stops a DMA transfer, this register points to the system address of the next contiguous data position. NOTE: The DS_ADDR must be aligned to a four-byte boundary; the two least-significant bits must be cleared.

15.4.2 Block attributes (eSDHC_BLKATTR)

The block attributes register configures the number of data blocks and the number of bytes in each block. Only access this register when no transactions are executing (after transactions have stopped). The host driver should wait until PRSSTAT[DLA] is cleared.

Address: 11_4000h base + 4h offset = 11_4004h

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

eSDHC_BLKATTR field descriptions

Field	Description																						
0–15 BLKCNT	<p>Block count for current transfer. This field is enabled when XFERTYP[BCEN] is set and is valid only for multiple block transfers. The host driver should set this field to a value between 1 and the maximum block count. The eSDHC decrements the block count after each block transfer and stops when the count reaches zero. Clearing this field results in no data blocks being transferred.</p> <table> <tr><td>0000</td><td>Stop count</td></tr> <tr><td>0001</td><td>1 block</td></tr> <tr><td>0002</td><td>2 blocks</td></tr> <tr><td>...</td><td></td></tr> <tr><td>FFFF</td><td>65,535 blocks</td></tr> </table>	0000	Stop count	0001	1 block	0002	2 blocks	...		FFFF	65,535 blocks												
0000	Stop count																						
0001	1 block																						
0002	2 blocks																						
...																							
FFFF	65,535 blocks																						
16–18 -	This field is reserved.																						
19–31 BLKSZE	<p>Transfer block size. Specifies the block size for block data transfers. Values can range from one byte up to the maximum buffer size.</p> <p>The DMA always writes at least four bytes to memory. Thus, software should allocate a buffer space rounded up to a 4-byte aligned size in order to avoid data corruption.</p> <table> <tr><td>0000</td><td>No data transfer</td></tr> <tr><td>0001</td><td>1 byte</td></tr> <tr><td>0002</td><td>2 bytes</td></tr> <tr><td>0003</td><td>3 bytes</td></tr> <tr><td>0004</td><td>4 bytes</td></tr> <tr><td>...</td><td></td></tr> <tr><td>01FF</td><td>511 bytes</td></tr> <tr><td>0200</td><td>512 bytes</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0800</td><td>2048 bytes</td></tr> <tr><td>1000</td><td>4096 bytes</td></tr> </table>	0000	No data transfer	0001	1 byte	0002	2 bytes	0003	3 bytes	0004	4 bytes	...		01FF	511 bytes	0200	512 bytes	...		0800	2048 bytes	1000	4096 bytes
0000	No data transfer																						
0001	1 byte																						
0002	2 bytes																						
0003	3 bytes																						
0004	4 bytes																						
...																							
01FF	511 bytes																						
0200	512 bytes																						
...																							
0800	2048 bytes																						
1000	4096 bytes																						

15.4.3 Command argument (eSDHC_CMDARG)

The command argument register contains the SD/MMC command argument.

Address: 11_4000h base + 8h offset = 11_4008h

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

eSDHC_CMDARG field descriptions

Field	Description
0–31 CMDARG	Command argument. The SD/MMC command argument is specified as bits 39–8 of the command format in the SD or MMC Specification . If PRSSTAT[CIHB] is set, this register is write-protected.

15.4.4 Command transfer type (eSDHC_XFERTYP)

The transfer type register controls the operation of data transfers. The host driver should set this register before issuing a command followed by a data transfer. To prevent data loss, the eSDHC prevents a write to the bits that are involved in the data transfer of this register while the data transfer is active.

The host driver should check PRSSTAT[CDIHB] and PRSSTAT[CIHB] before writing to this register.

- If PRSSTAT[CDIHB] is set, any attempt to send a command with data by writing to this register is ignored.
- If PRSSTAT[CIHB] is set, any write to this register is ignored.

Table 15-6. Determination of transfer type

Multi/Single Block Select XFERTYP[MSBSEL]	Block Count Enable XFERTYP[BCEN]	Block Count BLKATTR[BLKCNT]	Function
0	X	X	Single Transfer
1	0	X	Infinite Transfer
1	1	Positive Number	Multiple Transfer
1	1	Zero	No Data Transfer

The table below shows how the response type can be determined by the command index check enable, command CRC check enable, and response type bits.

Table 15-7. Relation between parameters and name of response type

Response Type XFERTYP[RSPTYP]	Index Check Enable XFERTYP[CICEN]	CRC Check Enable XFERTYP[CCCEN]	Response type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6, R7
11	1	1	R1b, R5b

NOTE

The CRC field for R3 and R4 is expected to be all 1s. The CRC check should be disabled for these response types.

Enhanced Secure Digital Host Controller (eSDHC) Memory Map

Address: 11_4000h base + Ch offset = 11_400Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		CMDINX		CMDTYP		DPSEL		CICEN		CCCEN		Reserved		RSPTYP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										MSBSEL		DTDSEL		AC12EN	
W															BCEN	DMAEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSDHC_XFERTYP field descriptions

Field	Description								
0–1 -	This field is reserved.								
2–7 CMDINX	Command index. These bits should be set to the command number (CMD0-63, ACMD0-63) that is specified in bits 45-40 of the command format in the <i>SD Memory Card Physical Layer Specification</i> .								
8–9 CMDTYP	Command type. There is one type of special command-abort. Clear this bit field for all other commands. <ul style="list-style-type: none"> Abort command. If this command is set when executing a read transfer, the eSDHC stops reads to the buffer. If this command is set when executing a write transfer, the eSDHC stops driving the SDHC_DAT line. After issuing the abort command, the host driver should issue a software reset. (Abort transaction) <table> <tr><td>00</td><td>Normal-other commands</td></tr> <tr><td>01</td><td>Reserved</td></tr> <tr><td>10</td><td>Reserved</td></tr> <tr><td>11</td><td>Abort-CMD12, CMD52 for writing I/O abort in CCCR</td></tr> </table>	00	Normal-other commands	01	Reserved	10	Reserved	11	Abort-CMD12, CMD52 for writing I/O abort in CCCR
00	Normal-other commands								
01	Reserved								
10	Reserved								
11	Abort-CMD12, CMD52 for writing I/O abort in CCCR								
10 DPSEL	Data present select. Set to indicate that data is present and should be transferred using the SDHC_DAT line. It is cleared for the following: <ul style="list-style-type: none"> Commands using only the SDHC_CMD line (for example, CMD52) Commands with no data transfer but using busy signal on the SDHC_DAT[0] line (R1b or R5b, for example, CMD38) <table> <tr><td>0</td><td>No data present</td></tr> <tr><td>1</td><td>Data present</td></tr> </table>	0	No data present	1	Data present				
0	No data present								
1	Data present								
11 CICEN	Command index check enable.								

Table continues on the next page...

eSDHC_XFERTYP field descriptions (continued)

Field	Description
	<p>0 Disable. The index field is not checked.</p> <p>1 Enable. The eSDHC checks the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a command index error.</p>
12 CCCEN	<p>Command CRC check enable. The number of bits checked by the CRC field value changes according to the length of the response. (See RSPTYP[1:0] and Table 15-6.)</p> <p>0 Disable. The CRC field is not checked.</p> <p>1 Enable. The eSDHC checks the CRC field in the response if it contains the CRC field. If an error is detected, it is reported as a command CRC error.</p>
13 -	This field is reserved.
14–15 RSPTYP	<p>Response type select.</p> <p>00 No response</p> <p>01 Response length 136</p> <p>10 Response length 48</p> <p>11 Response length 48 check busy after response</p>
16–25 -	This field is reserved.
26 MSBSEL	<p>Multi/single block select. Enables multiple block SDHC_DAT line data transfers. For any other commands, this bit should be cleared. If this bit is cleared, it is not necessary to set the block count register. (See Table 15-7.)</p> <p>0 Single block</p> <p>1 Multiple blocks</p>
27 DTDSEL	<p>Data transfer direction select. Defines the direction of SDHC_DAT line data transfers. The bit is set by the host driver to transfer data from the SD card to the eSDHC and it is cleared for all other commands.</p> <p>0 Write (host to card)</p> <p>1 Read (card to host)</p>
28 -	This field is reserved.
29 AC12EN	<p>Auto CMD12 enable. Multiple block transfers for memory require CMD12 to stop the transaction. If this bit is set, the eSDHC issues CMD12 automatically when the last block transfer is completed. The host driver should not set this bit to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in the Part 3 File Security specification do not require CMD12. In a single block transfer, the eSDHC ignores this bit.</p> <p>0 Disable</p> <p>1 Enable</p>
30 BCEN	<p>Block count enable. Enables the block attributes register, which is only relevant for multiple block transfers. When this bit is cleared, the block attributes register is disabled, which is useful in executing an infinite transfer.</p> <p>0 Disable</p> <p>1 Enable</p>
31 DMAEN	DMA enable. Enables DMA functionality as described in DMA interface . If this bit is set, a DMA operation should begin when the host driver writes to the CMDINX field of the transfer type register.

Table continues on the next page...

eSDHC_XFERTYP field descriptions (continued)

Field	Description
0	Disable
1	Enable

15.4.5 Command response n (eSDHC_CMDRSPn)

The command response registers store the four parts of the response bits from the card.

The table below describes the mapping of command responses from the SD bus to the command response registers for each response type. In the table, R[] refers to a bit range within the response data as transmitted on the SD bus.

Table 15-11. Response bit definition for each response type

Response type	Meaning of response	Response field	Response register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0
R1b (Auto CMD12 response)	Card status for Auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R6 (publish RCA)	New published RCA[31:16] and card status[15:0]	R[39:8]	CMDRSP0

This table shows the following:

- Most responses with a length of 48 (R[47:0]) have 32 bits of the response data (R[39:8]) stored in the CMDRSP0 register.
- Responses of type R1b (Auto CMD12 responses) have response data bits R[39:8] stored in the CMDRSP3 register.
- Responses with length 136 (R[135:0]) have 120 bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, the eSDHC only stores part of the response data in the command response registers. This enables the host driver to efficiently read 32 bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the index field, and the CRC are checked by the eSDHC (as specified by XFERTYP[CICEN, CCCEN]) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, the eSDHC checks R[47:1], and if the response length is 136, the eSDHC checks R[119:1].

Since the eSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT (Command without data) command, the eSDHC stores the Auto CMD12 response in the CMDRSP3 register and the CMD_wo_DAT response is stored in CMDRSP0. This allows the eSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the eSDHC modifies part of the command response registers it preserves the unmodified bits.

Address: 11_4000h base + 10h offset + (4d × i), where i=0d to 3d

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

eSDHC_CMDSRSPn field descriptions

Field	Description
0–31 CMDSRSPn	Command response. See Command Response 0–3 for the mapping of command responses from the SD bus to this register for each response type.

15.4.6 Data buffer access port (eSDHC_DATPORT)

The buffer data port register is a 32-bit data port register used to access the internal buffer.

NOTE

When the internal DMA is not enabled and a write transaction is in operation, DATPORT must not be read. DATPORT also must not be used to read (or write) data by the CPU or external DMA if the data will be written (or read) by the eSDHC internal DMA.

Address: 11_4000h base + 20h offset = 11_4020h

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

eSDHC_DATPORT field descriptions

Field	Description
0–31 DATCONT	Data content. The buffer data port register is for 32-bit data access by the CPU or an external DMA . When the internal DMA is enabled, any write to this register is ignored, and a read from this register always yields 0.

15.4.7 Present state (eSDHC_PRSSTAT)

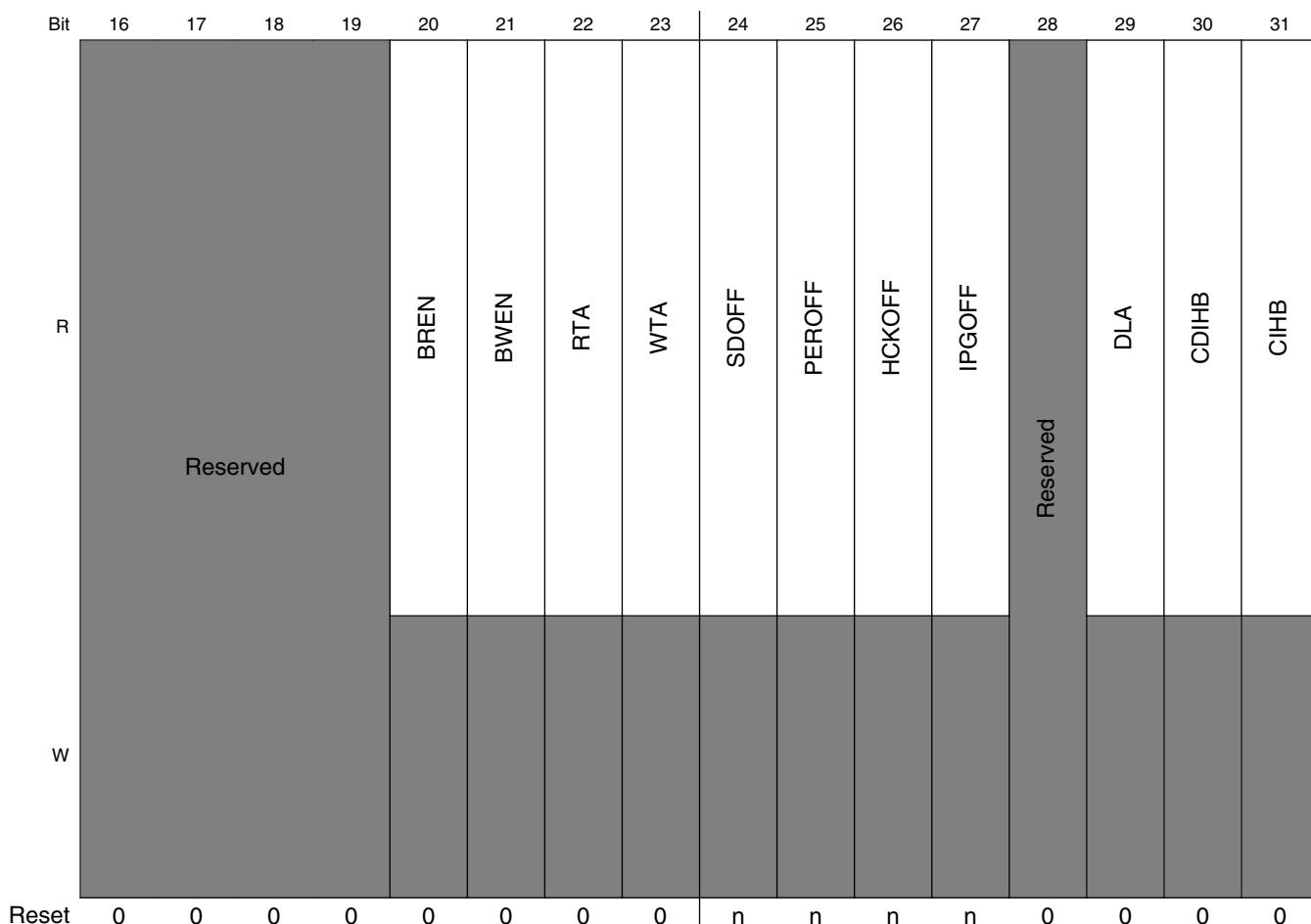
The present state register indicates the status of the eSDHC to the host driver.

NOTE

The host driver can issue CMD0, CMD12, CMD13 (for memory) when the SDHC_DAT lines are busy during a data transfer. These commands can be issued when PRSSTAT[CIHB] is cleared. Other commands should be issued when PRSSTAT[CDIHB] is cleared. Possible changes to the SD physical specification may add other commands to this list in the future.

Address: 11_4000h base + 24h offset = 11_4024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R				DLSL					CLSL			Reserved	WPSPL	CDPL	Reserved	CINS
W																
Reset	1	1	1	1	1	1	1	1	1	0	0	0	n	n	0	n



eSDHC_PRSSTAT field descriptions

Field	Description																
0–7 DLSL	<p>SDHC_DAT[7:0] line signal level. These bits are used to check the SDHC_DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from SDHC_DAT[0]. The reset value is affected by the external pull resistors. By default, read value of this bit field after reset is 0b11110111, when SDHC_DAT[3] is pull-down and other lines are pull-up.</p> <p>The meaning of PRSSTAT bits is as follows:</p> <p>PRSSTAT SDHC_DATn</p> <table> <tr><td>0</td><td>7</td></tr> <tr><td>1</td><td>6</td></tr> <tr><td>2</td><td>5</td></tr> <tr><td>3</td><td>4</td></tr> <tr><td>4</td><td>3</td></tr> <tr><td>5</td><td>2</td></tr> <tr><td>6</td><td>1</td></tr> <tr><td>7</td><td>0</td></tr> </table>	0	7	1	6	2	5	3	4	4	3	5	2	6	1	7	0
0	7																
1	6																
2	5																
3	4																
4	3																
5	2																
6	1																
7	0																
8 CLSL	SDHC_CMD line signal level. This status is used to check the SDHC_CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull resistor, by default, read value of this bit after reset is 1, when the command line is pull-up.																

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
9–11 -	This field is reserved.
12 WPSPL	Write protect switch pin level. The write protect switch is supported for memory. This bit reflects the inverse of the SDHC_WP pin of the card socket. A software reset does not affect this bit. The reset value is affected by the external write protect switch. If the SDHC_WP pin is not used, it should be tied to 0 so that the reset value of this bit is 1 and write is enabled. 0 Write protected (SDHC_WP = 1) 1 Write enabled (SDHC_WP = 0)
13 CDPL	Card detect pin level. This bit reflects the inverse value of the SDHC_CD_B pin for the card socket. Debouncing is not performed on this bit. This bit may be valid, but it is not guaranteed because of a propagation delay. Use of this bit is limited to testing since it must be debounced by software. A software reset does not affect this bit. Write to the force event register does not affect this bit. The reset value is affected by the external card detection pin. If this bit is not used, it should be tied to 0. 0 No card present (SDHC_CD_B = 1) 1 Card present (SDHC_CD_B = 0)
14 -	This field is reserved.
15 CINS	Card inserted. Indicates if a card has been inserted. The eSDHC debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from 0 to 1 generates a card-insertion interrupt in the interrupt status register and changing from 1 to 0 generates a card removal interrupt in the interrupt status register. A write to the force event register does not affect this bit. The software reset for all in the system control register does not affect this bit. A software reset does not affect this bit. 0 Power-on-reset or no card 1 Card inserted
16–19 -	This field is reserved.
20 BREN	Buffer read enable. This status is used for non-DMA read transfers. The eSDHC may implement multiple buffers to transfer data efficiently. This read-only flag indicates that a burst-length of valid data exists in the host-side buffer. When the buffer is read, this bit is cleared. When a burst length of data is ready in the buffer, this bit is set and a buffer read ready interrupt is generated (if the interrupt is enabled). 0 Buffer read disable 1 Buffer read enable
21 BWEN	Buffer write enable. This status is used for non-DMA write transfers. The eSDHC can implement multiple buffers to transfer data efficiently. This read-only flag indicates if space is available for a burst length of write data. When the buffer is written, this bit is cleared. When a burst length of data is written to the buffer, this bit is set and a buffer write ready interrupt is generated (if the interrupt is enabled). 0 Buffer write disable 1 Buffer write enable
22 RTA	Read transfer active. This status is used for detecting completion of a read transfer. This bit is set for either of the following conditions:

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
	<ul style="list-style-type: none"> • After the end bit of the read command • When writing a 1 to PROCTL[CREQ] to restart a read transfer <p>This bit is cleared for either of the following conditions:</p> <ul style="list-style-type: none"> • When the last data block as specified by block length is transferred to the system • When all valid data blocks have been transferred to the system and no current block transfers are being sent as a result of PROCTL[SABGREQ] being set. A transfer complete interrupt is generated when this bit changes to 0. <p>0 No valid data 1 Transferring data</p>
23 WTA	<p>Write transfer active. This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in eSDHC.</p> <p>This bit is set in either of the following cases:</p> <ul style="list-style-type: none"> • After the end bit of the write command. • When writing a 1 to PROCTL[CREQ] to restart a write transfer. <p>This bit is cleared in either of the following cases:</p> <ul style="list-style-type: none"> • After getting the CRC status of the last data block, as specified by the transfer count (single and multiple) • After getting the CRC status of any block where data transmission is about to be stopped by a stop-at-block-gap request. <p>During a write transaction, a IRQSTAT[BGE] interrupt is generated when this bit is changed to 0, as result of PROCTL[SABGREQ] being set. This status is useful for the host driver in determining when to issue commands during write busy.</p> <p>0 No valid data 1 Transferring data</p>
24 SDOFF	<p>SD clock gated off internally. Indicates the SD clock is internally gated off because of a buffer overrun, buffer underrun, or a read pause without read-wait assertion. This bit is for the host driver to debug data transaction on SD bus.</p> <p>This status bit resets to 0, but reflects the value of the automatic clock gating and may transition to 1 if the eSDHC is idle.</p>
25 PEROFF	<p>The internal bus clock gated off internally. This status bit indicates the internal bus clock is internally gated off. This bit is for the host driver to debug a transaction on SD bus.</p> <p>This status bit resets to 0, but reflects the value of the automatic clock gating and may transition to 1 if the eSDHC is idle.</p>
26 HCKOFF	<p>Master clock gated off internally. This status bit indicates master clock is internally gated off. This bit is for the host driver to debug a data transfer.</p> <p>This status bit resets to 0, but reflects the value of the automatic clock gating and may transition to 1 if the eSDHC is idle.</p>
27 IPGOFF	<p>Controller clock gated off internally. Indicates that the controller clock is internally gated off. This bit is for the host driver to debug.</p> <p>This status bit resets to 0, but reflects the value of the automatic clock gating and may transition to 1 if the eSDHC is idle.</p>
28 -	This field is reserved.

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
29 DLA	<p>Data line active. Indicates whether one of the SDHC_DAT line on SD bus is in use.</p> <p>For read transactions, this bit indicates if a read transfer is executing on the SD bus. Clearing this bit from 1 to 0 between data blocks generates a block gap event interrupt.</p> <p>This bit is set in either of the following cases:</p> <ul style="list-style-type: none"> • After the end bit of the read command • When writing a 1 to PROCTL[CREQ] to restart a read transfer <p>This bit is cleared in either of the following cases:</p> <ul style="list-style-type: none"> • When the end bit of the last data block is sent from the SD bus to the eSDHC • When beginning a read wait transfer initiated by a stop at block gap request <p>The eSDHC waits at the next block gap by driving read wait at the start of the interrupt cycle. If the read-wait signal is already driven (data buffer cannot receive data), the eSDHC can wait for current block gap by continuing to drive the read-wait signal. It is necessary to support read wait in order to use the suspend/resume function.</p> <p>For write transactions, this bit indicates that a write transfer is executing on the SD bus. Clearing this bit from 1 to 0 generates a transfer complete interrupt.</p> <p>This bit is set in any of the following cases:</p> <ul style="list-style-type: none"> • After the end bit of the write command • When writing a 1 to PROCTL[CREQ] to continue a write transfer <p>This bit is cleared in any of the following cases:</p> <ul style="list-style-type: none"> • When the SD card releases write-busy of the last data block, the eSDHC also detects if output is not busy. If the SD card does not drive the busy signal after CRC status is received, the eSDHC should consider the card drive not busy. • When the SD card releases write-busy prior to waiting for write transfer as a result of a stop at block gap request <p>0 SDHC_DAT line inactive 1 SDHC_DAT line active</p>
30 CDIHB	<p>Command inhibit (SDHC_DAT). This bit is set if the SDHC_DAT line is active, the read transfer active is set. If this bit is cleared, it indicates the eSDHC can issue the next SD/MMC command. Commands with busy signal belong to command inhibit (SDHC_DAT) (e.g. R1b and R5b type). Clearing from 1 to 0 generates a transfer complete interrupt.</p> <p>0 Can issue command which uses the SDHC_DAT line 1 Cannot issue command which uses the SDHC_DAT line</p>
31 CIHB	<p>Command inhibit (SDHC_CMD). This bit is cleared, if the SDHC_CMD line is not in use and the eSDHC can issue a SD/MMC command using the SDHC_CMD line.</p> <p>This bit is set immediately after the XFERTYP register is written. This bit is cleared when the command response is received. Even if the CDIHB bit is set, commands using only the SDHC_CMD line can be issued if this bit is cleared. Clearing from 1 to 0 generates a command complete interrupt.</p> <p>If the eSDHC cannot issue the command because of a command conflict error (see the command CRC error) or because of a command not issued by Auto CMD12 error, this bit remains set and IRQSTAT[CC] is not set. Status issuing Auto CMD12 is not read from this bit.</p> <p>0 Can issue command using only SDHC_CMD line 1 Cannot issue command</p>

15.4.8 Protocol control (eSDHC_PROCTL)

To restart the transfer after a stop at the block gap, the continue request should be used to restart the transfer.

Any time PROCTL[SABGREQ] stops the data transfer, the host driver should wait for IRQSTAT[TC] before attempting to restart the transfer. When restarting the data transfer by continue request, the host driver should clear PROCTL[SABGREQ] before or simultaneously.

Address: 11_4000h base + 28h offset = 11_4028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved					WECRM	WECINS		Reserved					RWCTL	CREQ	SABGREQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	Reserved								CDSS	CDTL	EMODE	D3CD	DTW			Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

eSDHC_PROCTL field descriptions

Field	Description
0–4 -	This field is reserved.
5 WECRM	Wake-up event enable on SD card removal. This bit enables wakeup event via card removal assertion in the IRQSTAT register. FN_WUS (wake-up support) in CIS does not affect this bit. 0 Disable 1 Enable
6 WECINS	Wake-up event enable on SD card insertion. This bit enables wakeup event via card insertion assertion in the IRQSTAT register. FN_WUS (wake-up support) in CIS does not affect this bit.

Table continues on the next page...

eSDHC_PROCTL field descriptions (continued)

Field	Description
	<p>NOTE: When this bit is set, the card insertion status and the eSDHC interrupt can be asserted without SD_CLK toggling. When the wakeup feature is not enabled, the SD_CLK must be active in order to assert the card insertion status and the eSDHC interrupt.</p> <p>0 Disable 1 Enable</p>
7–12 -	This field is reserved.
13 RWCTL	<p>Read wait control. Not supported on this device.</p> <p>If the card supports read wait, set this bit to enable the read wait protocol to stop read data using the SDHC_DAT[2] line. Otherwise, the eSDHC has to stop the SD clock to hold read data, which restricts command generation.</p> <p>If the card does not support read wait, this bit should never be set otherwise an SDHC_DAT line conflict may occur. If this bit is cleared, a stop-at-block-gap-during-read operation is also supported, but the eSDHC stops the SD clock to pause the reading operation.</p> <p>0 Disable read-wait control, and stop SD clock at block gap when the SABGREQ bit is set 1 Enable read-wait control, and assert read wait without stopping the SD clock at block gap when PROCTL[SABGREQ] is set</p>
14 CREQ	<p>Continue request. Restarts a transaction which was stopped using the stop-at-block-gap request. To cancel the request, clear SABGREQ and set this bit to restart the transfer.</p> <p>The eSDHC automatically clears this bit in either of the following cases:</p> <ul style="list-style-type: none"> • For a read transaction, the PRSSTAT[DLA] bit changes from 0 to 1 as a read transaction restarts. • For a write transaction, the PRSSTAT[WTA] bit changes from 0 to 1 as the write transaction restarts. <p>Therefore, it is not necessary for the host driver to clear. If SABGREQ and this bit are set, the continue request is ignored.</p> <p>0 No effect 1 Restart</p>
15 SABGREQ	<p>Stop at block gap request. Stops executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the TC bit is set, indicating a transfer completion, the host driver should leave this bit set. Clearing SABGREQ and CREQ does not cause the transaction to restart.</p> <p>The eSDHC stops the SD bus clock to pause the read operation during the block gap.</p> <p>For write transfers in which the host driver writes data to the data port register, the host driver should set this bit after all block data is written. If this bit is set, the host driver should not write data to the DATPORT register after a block is sent. When this bit is set, the host driver should not clear this bit before IRQSTAT[TC] is set. Otherwise, the eSDHC behavior is undefined. Confirm that IRQSTAT[TC] is enabled.</p> <p>This bit affects PRSSTAT[RTA, WTA, DLA, CIHB].</p> <p>0 Transfer 1 Stop</p>
16–23 -	This field is reserved.
24 CDSS	<p>Card detect signal selection. Selects the source for card detection.</p> <p>0 Card detection level is selected (for normal purpose) 1 Card detection test level is selected (for test purpose)</p>

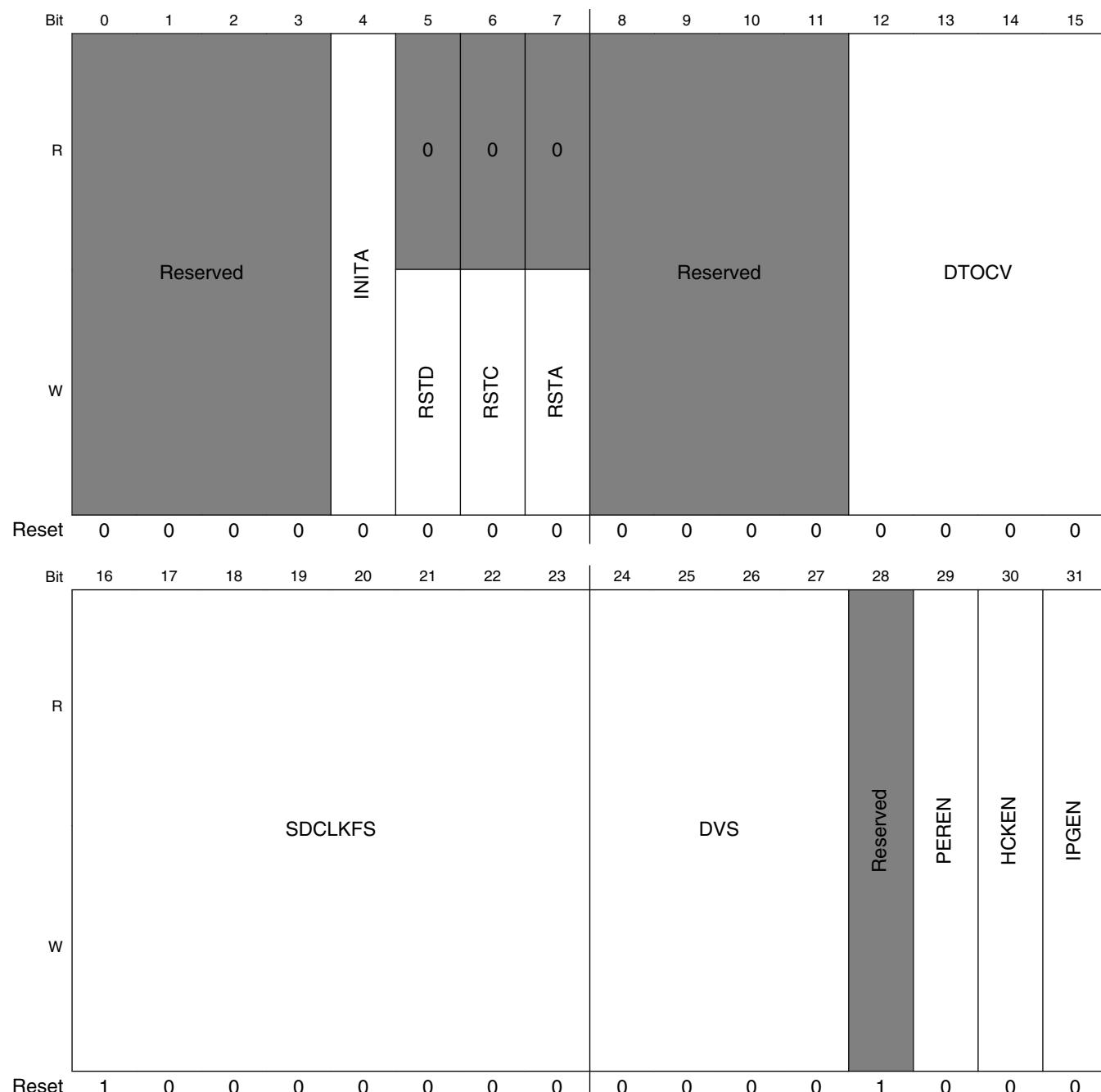
Table continues on the next page...

eSDHC_PROCTL field descriptions (continued)

Field	Description
25 CDTL	Card detect test level. Determines card insertion status when CDSS is set. 0 No card in the slot 1 Card is inserted
26–27 EMODE	Endian mode. eSDHC supports only address-invariant mode in data transfer. 00 Reserved 01 Reserved 10 Address-invariant mode. Each byte location in the main memory is mapped to the same byte location in the MMC/SD card. 11 Reserved
28 D3CD	SDHC_DAT3 as card detection pin. If this bit is set, SDHC_DAT3 should be pulled down to act as a card detection pin. 0 SDHC_DAT3 does not monitor card insertion 1 SDHC_DAT3 is card detection pin
29–30 DTW	Data transfer width. Selects the data width of the SD bus. The host driver should set it to match the data width of the card. 00 1-bit mode 01 4-bit mode 10 8-bit mode 11 Reserved
31 -	This field is reserved.

15.4.9 System control (eSDHC_SYSCTL)

Address: 11_4000h base + 2Ch offset = 11_402Ch



eSDHC_SYSCTL field descriptions

Field	Description
0–3 -	This field is reserved.

Table continues on the next page...

eSDHC_SYSCTL field descriptions (continued)

Field	Description
4 INITA	Initialization active. When this bit is written '1', 80 SD clocks are sent to the card. After the 80 clocks are sent, this bit is self-cleared. This bit is very useful during the card power-up period when 74 SD clocks are needed and clock auto-gating feature is enabled. Writing one to this bit when it is already set has no effect. Clearing this bit at any time does not affect it. When PRSSTAT[CIHB] or PRSSTAT[CDIHB] is set, writing a one to this bit is ignored. That is, when the command line or data line is active, writing to this bit is not allowed.
5 RSTD	Software reset for SDHC_DAT line. The DMA and part of the data circuit are reset. The following registers and bits are cleared by this bit: <ul style="list-style-type: none"> • DATPORT register • Buffer is cleared and initialized; PRSSTAT register • PRSSTAT[BREN, BWEN, RTA, WTA, DLA, CDIHB] • PROCTL[CREQ, SABGREQ] • IRQSTAT[BRR, BWR, DINT, BGE, TC] 0 Work 1 Reset
6 RSTC	Software reset for SDHC_CMD line. Only part of the command circuit is reset. The following bits are cleared by this bit: <ul style="list-style-type: none"> • PRSSTAT[CIHB] • IRQSTAT[CC] 0 Work 1 Reset
7 RSTA	Software reset for all. This reset affects the entire host controller except for the card-detection circuit. Register bits of type ROC, RW, RW1C, and RWAC are cleared. During its initialization, the host driver should set this bit to reset the eSDHC. The eSDHC should clear this bit when capabilities registers are valid and the host driver can read them. Additional use of the this bit does not affect the value of the capabilities registers. After this bit is set, it is recommended the host driver reset the external card and re-initialize it. 0 Work 1 Reset
8–11 -	This field is reserved.
12–15 DTOCV	Data timeout counter value. Determines the interval by which SDHC_DAT line timeouts are detected. See the data timeout error Interrupt status (eSDHC_IRQSTAT) , for information on factors that dictate timeout generation. Timeout clock frequency is generated by dividing the base clock SDHC_CLK value by this value. When setting this register, prevent inadvertent timeout events by clearing IRQSTATEN[DTOESEN]. 0000 SDHC_CLK x 2 ¹³ 0001 SDHC_CLK x 2 ¹⁴ ... 1110 SDHC_CLK x 2 ²⁷ 1111 Reserved
16–23 SDCLKFS	SDHC_CLK frequency select. This field, together with DVS, selects the frequency of SDHC_CLK pin. This bit holds the prescaler of the base clock frequency. Only the following settings are allowed: Multiple bits must not be set or the behavior of this prescaler is undefined.

Table continues on the next page...

eSDHC_SYSCTL field descriptions (continued)

Field	Description
	<p>According to the MMC Electrical Standard, High Capacity 4.2, (JESD84-B42), the maximum SD clock frequency is 52 MHz, and should never exceed this limit. The frequency of SDHC_CLK is set by the following formula:</p> $\text{clock frequency} = (\text{base clock}) \div [(\text{SDCLKFS} \times 2) \times (\text{DVS} + 1)]$ <p>For example, if the base clock frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 0x1 and divisor value of 0x1 yields 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 0x8 and divisor value of 0xE yields the exact clock value of 400 kHz.</p> <p>The reset value of this bit field is 0x80. So, if the input base clock is about 96 MHz, the default SD clock after reset is 375 kHz.</p> <p>NOTE: The base clock frequency equals the platform clock/2.</p> <ul style="list-style-type: none"> 0x01 Base clock divided by 2 0x02 Base clock divided by 4 0x04 Base clock divided by 8 0x08 Base clock divided by 16 0x10 Base clock divided by 32 0x20 Base clock divided by 64 0x40 Base clock divided by 128 0x80 Base clock divided by 256
24–27 DVS	<p>Divisor. Provides a more exact divisor to generate the desired SD clock frequency. The settings are as follows:</p> <ul style="list-style-type: none"> 0x0 Divide by 1 0x1 Divide by 2 ... 0xE Divide by 15 0xF Divide by 16
28 -	This field is reserved.
29 PEREN	<p>Peripheral clock enable. If set, the peripheral clock is always active and no automatic gating is applied, thus SDHC_CLK is active only except auto gating-off during buffer danger. If cleared, the peripheral clock is automatically off when no transaction is on the SD bus. Clearing this bit does not stop SDHC_CLK immediately. The peripheral clock will be internally gated off, if none of the following factors are met:</p> <ul style="list-style-type: none"> • Command part is reset • Data part is reset • Soft reset • Command is about to send • Clock divisor is just updated • Continue request is just set • This bit is set • Card insertion is detected • Card removal is detected • Card external interrupt is detected • 80 clocks for initialization phase is ongoing <p>0 The peripheral clock is internally gated off 1 The peripheral clock is not automatically gated off</p>

Table continues on the next page...

eSDHC_SYSCTL field descriptions (continued)

Field	Description
30 HCKEN	<p>Master clock enable. If set, master clock is always active and no automatic gating is applied. If cleared, master clock is automatically off when no data transfer is on SD bus.</p> <p>NOTE: Master clock is the clock to the DMA engine and to the interface logic.</p> <p>0 Master clock is internally gated off 1 Master clock is not automatically gated off</p>
31 IPGEN	<p>Controller clock enable. If this bit is set, the controller clock is always active and no automatic gating is applied. The controller clock is internally gated off, if neither the following factors is met:</p> <ul style="list-style-type: none"> • Command part is reset • Data part is reset • Soft reset • Command is about to send • Clock divisor is just updated • Continue request is just set • This bit is set • Card insertion is detected • Card removal is detected • Card external interrupt is detected • The internal bus clock is not gated off <p>NOTE: The controller clock is not auto-gated off if the peripheral clock is not gated off. So, clearing this bit only does not take effect if SYSCTL[PEREN] is not cleared.</p> <p>0 The controller clock is internally gated off 1 The controller clock is not automatically gated off</p>

15.4.10 Interrupt status (eSDHC_IRQSTAT)

An interrupt is generated when one of the status bits and its corresponding interrupt enable bit are set. For all bits (except BGE), writing one to a bit clears it, while writing zero keeps the bit unchanged. More than one status can be cleared with a single register write.

The table below shows that command timeout error has higher priority than command complete. If both bits are set, it can be assumed that the response was not received correctly.

Table 15-23. Relation between command timeout error and command complete status

Command complete	Command timeout error	Meaning of the status
0	0	-
X	1	Response not received within 64 SDHC_CLK cycles
1	0	Response received

The table below shows that transfer complete has higher priority than the data timeout error. If both bits are set, the data transfer can be considered complete.

Table 15-24. Relation between data timeout error and transfer complete status

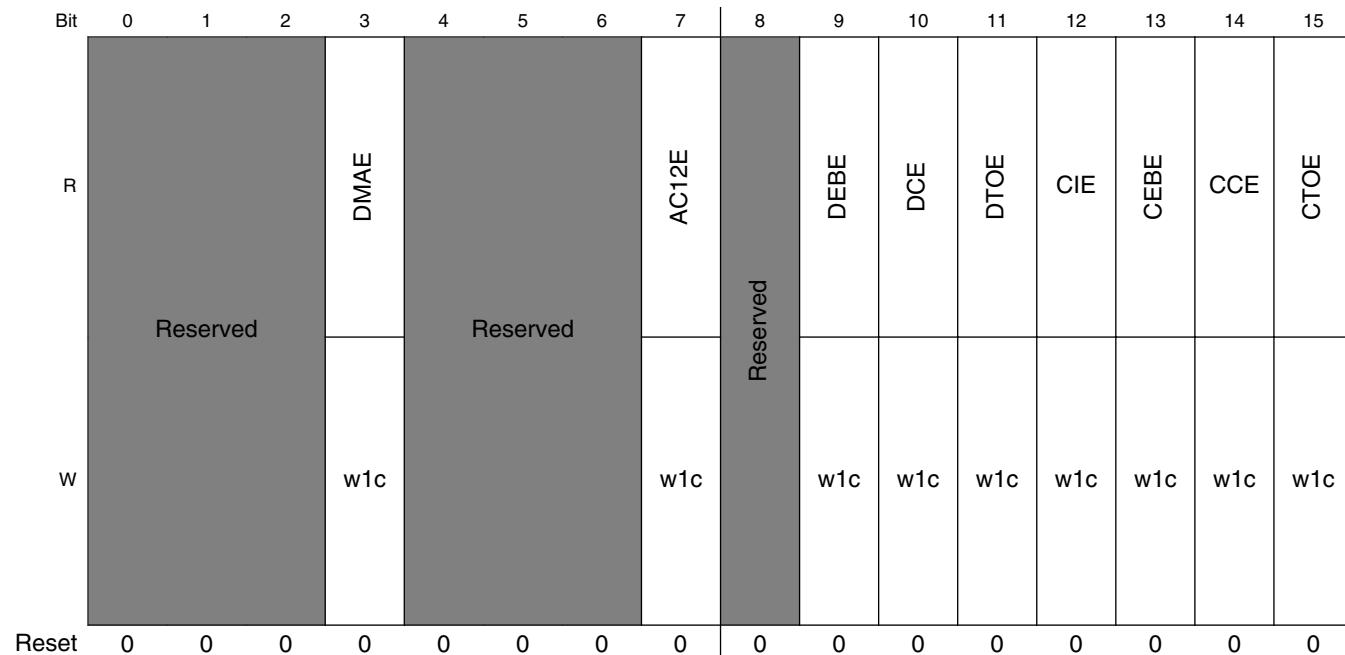
Transfer complete	Data timeout error	Meaning of the status
0	0	-
0	1	Timeout occur during transfer
1	X	Data transfer complete

The relation between command CRC error and command timeout error is shown below.

Table 15-25. Relation between command CRC error and command timeout error

Command CRC error	Command timeout error	Meaning of the status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	SDHC_CMD line conflict

Address: 11_4000h base + 30h offset = 11_4030h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									CRM	CINS	BRR	BWR	DINT		TC	CC
	Reserved													BGE		
W									w1c	w1c	w1c	w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSDHC_IRQSTAT field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAE	DMA error. Occurs when internal DMA transfer failed. This bit is set when some error occurs in the data transfer. The value in the DMA system address register is the next fetch address where the error occurs. Since any error corrupts the entire data block, the host driver should restart the transfer from the corrupted block boundary. The address of the block boundary can be calculated from the current DS_ADDR value or the remaining number of blocks and the block size. 0 No Error 1 Error
4–6 -	This field is reserved.
7 AC12E	Auto CMD12 error. Occurs when one of the bits in AUTOC12ERR is set. This bit is also set when Auto CMD12 is not executed due to a previous command error. 0 No Error 1 Error
8 -	This field is reserved.
9 DEBE	Data end bit error. Occurs when detecting 0 at the end bit position of read data on the SDHC_DAT line or at the end bit position of the CRC. 0 No Error 1 Error
10 DCE	Data CRC error. Occurs when detecting CRC error when transferring read data on the SDHC_DAT line or when detecting the write CRC status having a value other than 0b010.

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
	0 No Error 1 Error
11 DTOE	Data timeout error. Occurs during one of following timeout conditions: <ul style="list-style-type: none"> • Busy timeout for R1b and R5b types • Busy timeout after write CRC status • Read data timeout 0 No error 1 Timeout
12 CIE	Command index error. Occurs if a command index error occurs in the command response. NOTE: Under rare conditions, CIE may be set for a command without data while a command with data is in progress. On detecting a command CRC error (IRQSTAT[CCE]) or command index error (IRQSTAT[CIE]), perform error recovery and re-issue the command without data. If Auto CMD12 is enabled for data transfer then Auto CMD12 won't be issued by hardware, so software needs to issue it after data transfer completion. 0 No error 1 Error
13 CEBE	Command end bit error. Occurs when the end bit of a command response is 0. 0 No error 1 End bit error generated
14 CCE	Command CRC error. A command CRC error is generated in two cases: <ul style="list-style-type: none"> • If a response is returned and IRQSTAT[CTOE] is cleared (indicating no timeout), this bit is set when detecting a CRC error in the command response. • The eSDHC detects a SDHC_CMD line conflict by monitoring the SDHC_CMD line when a command is issued. If the eSDHC drives the SDHC_CMD line to 1, but detects 0 on the SDHC_CMD line at the next SDHC_CLK edge, then the eSDHC aborts the command (stop driving SDHC_CMD line) and sets this bit. The CTOE bit is also set to distinguish the SDHC_CMD line conflict. NOTE: Under rare conditions, CCE may be set for a command without data while a command with data is in progress. On detecting a command CRC error (IRQSTAT[CCE]) or command index error (IRQSTAT[CIE]), perform error recovery and re-issue the command without data. If Auto CMD12 is enabled for data transfer then Auto CMD12 won't be issued by hardware, so software needs to issue it after data transfer completion. 0 No error 1 CRC error generated
15 CTOE	Command timeout error. Occurs if no response is returned within 64 SDHC_CLK cycles from the end bit of the command. Also, if eSDHC detects a SDHC_CMD line conflict, this bit is set along with IRQSTAT[CCE] as shown in Table 15-35 . 0 No error 1 Time out
16–23 -	This field is reserved.
24 CRM	Card removal. This bit is set if PRSSTAT[CINS] changes from 1 to 0. When the host driver writes 1 to this bit to clear it, the status of PRSSTAT[CINS] should be confirmed. Because the card-detect state may be changed when the host driver clears this bit, an interrupt event may not be generated.

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
	<p>When this bit is cleared, it is set again if no card is inserted. To leave it cleared, clear IRQSTATEN[CRMSEN].</p> <p>0 Card state unstable or inserted 1 Card removed</p>
25 CINS	<p>Card insertion. This bit is set if PRSSTAT[CINS] changes from 0 to 1. When the host driver writes 1 to this bit to clear it, the status of PRSSTAT[CINS] should be confirmed. Because the card-detect state may be changed when the host driver clears this bit, an interrupt event may not be generated.</p> <p>When this bit is cleared, it is set again if a card has been inserted. To leave it cleared, clear IRQSTATEN[CINSEN].</p> <p>0 Card state unstable or removed 1 Card inserted</p>
26 BRR	<p>Buffer read ready. This bit is set if PRSSTAT[BREN] changes from 0 to 1.</p> <p>0 Not ready to read buffer 1 Ready to read buffer</p>
27 BWR	<p>Buffer write ready. This bit is set if PRSSTAT[BWEN] changes from 0 to 1.</p> <p>0 Not ready to write buffer 1 Ready to write buffer</p>
28 DINT	<p>DMA interrupt. Occurs when the internal DMA finishes the data transfer successfully. If errors occur during data transfer, this bit is not set. Instead, the DMAE bit is set.</p> <p>0 No DMA interrupt 1 DMA interrupt is generated</p>
29 BGE	<p>Block gap event. If PROCTL[SABGREQ] is set, this bit is set when a read or write transaction is stopped at a block gap. If PROCTL[SABGREQ] is cleared, this bit is not set.</p> <p>During a read transaction, this bit is set at the falling edge of the SDHC_DAT line active status (when the transaction is stopped at SD bus timing). Read wait must be supported to use this function.</p> <p>During a write transaction, this bit is set at the falling edge of PRSSTAT[WTA] (after reading the CRC status at SD bus timing).</p> <p>0 No block gap event 1 Transaction stopped at block gap</p>
30 TC	<p>Transfer complete. This bit is set when a read or write transfer is completed.</p> <p>For a read transaction, this bit is set at the falling edge of PRSSTAT[RTA]. There are two cases in which this interrupt is generated:</p> <ul style="list-style-type: none"> • When a data transfer completes, as specified by the data length (after the last data has been read to the host system). • When data has stopped at the block gap and completed the data transfer by setting PROCTL[SABGREQ] (after valid data has been read to the host system). <p>For a write transaction, this bit is set at the falling edge of PRSSTAT[DLA]. There are two cases in which this interrupt is generated:</p> <ul style="list-style-type: none"> • When the last data is written to the SD card, as specified by data length and the busy signal is released. • When data transfers are stopped at the block gap by setting PROCTL[SABGREQ] and data transfers have completed (after valid data is written to the SD card and the busy signal is released).

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
31 CC	Command complete. This bit is set when the end bit of the command response is received (except Auto CMD12). See PRSSTAT[CIHB]. 0 No command complete 1 Command complete

15.4.11 Interrupt status enable (eSDHC_IRQSTATEN)

Setting the bits of IRQSTATEN enables the corresponding interrupt status bit to be set by the specified event. If any bit is cleared, the corresponding IRQSTAT bit is also cleared and is never set.

NOTE

The eSDHC may sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. As a result of synchronization, there is a delay in the card interrupt (which is asserted from the card) to the time the host system is informed.

NOTE

To detect a SDHC_CMD line conflict, the host driver must set both CTOESEN and CCESEN bits.

Address: 11_4000h base + 34h offset = 11_4034h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved		DMAESEN		Reserved		AC12ESEN		Reserved	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									CRMSEN	CINSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	1

eSDHC_IRQSTATEN field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAESEN	DMA error status enable 0 Masked 1 Enabled
4–6 -	This field is reserved.
7 AC12ESEN	Auto CMD12 error status enable 0 Masked 1 Enabled
8 -	This field is reserved.
9 DEBESEN	Data end bit error status enable 0 Masked 1 Enabled
10 DCESEN	Data CRC error status enable 0 Masked 1 Enabled
11 DTOESEN	Data timeout error status enable 0 Masked 1 Enabled
12 CIESEN	Command index error status enable 0 Masked 1 Enabled
13 CEBESEN	Command end bit error status enable 0 Masked 1 Enabled
14 CCESEN	Command CRC error status enable

Table continues on the next page...

eSDHC_IRQSTATEN field descriptions (continued)

Field	Description
	0 Masked 1 Enabled
15 CTOESEN	Command timeout error status enable 0 Masked 1 Enabled
16–23 -	This field is reserved.
24 CRMSEN	Card removal status enable 0 Masked 1 Enabled
25 CINSEN	Card insertion status enable 0 Masked 1 Enabled
26 BRRSEN	Buffer read ready status enable 0 Masked 1 Enabled
27 BWRSEN	Buffer write ready status enable 0 Masked 1 Enabled
28 DINTSEN	DMA interrupt status enable 0 Masked 1 Enabled
29 BGESSEN	Block gap event status enable 0 Masked 1 Enabled
30 TCSEN	Transfer complete status enable 0 Masked 1 Enabled
31 CCSEN	Command complete status enable 0 Masked 1 Enabled

15.4.12 Interrupt signal enable (eSDHC_IRQSIGEN)

IRQSIGEN selects which interrupt status is indicated to the host system as the interrupt. These status bits all share the same interrupt line. Setting any of these bits enables an interrupt generation. The corresponding status register bit generates an interrupt when the corresponding interrupt signal enable bit is set.

Address: 11_4000h base + 38h offset = 11_4038h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			DMAEEN	Reserved		AC12EEN	Reserved	DEBEIEN		DCEIEN		DTOEIN	CIEIEN	CCEIEN	CTOEIN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								CRMEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSDHC_IRQSIGEN field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAEEN	DMA error interrupt enable 0 Masked 1 Enabled
4–6 -	This field is reserved.
7 AC12EEN	Auto CMD12 error interrupt enable

Table continues on the next page...

eSDHC_IRQSIGEN field descriptions (continued)

Field	Description
	0 Masked 1 Enabled
8 -	This field is reserved.
9 DEBEIEN	Data end bit error interrupt enable 0 Masked 1 Enabled
10 DCEIEN	Data CRC error interrupt enable 0 Masked 1 Enabled
11 DTOEIEN	Data timeout error interrupt enable 0 Masked 1 Enabled
12 CIEIEN	Command index error interrupt enable 0 Masked 1 Enabled
13 CEBEIEN	Command end bit error interrupt enable 0 Masked 1 Enabled
14 CCEIEN	Command CRC error interrupt enable 0 Masked 1 Enabled
15 CTOEIEN	Command timeout error interrupt enable 0 Masked 1 Enabled
16–23 -	This field is reserved.
24 CRMIEN	Card removal interrupt enable 0 Masked 1 Enabled
25 CINSIEN	Card insertion interrupt enable 0 Masked 1 Enabled
26 BRRIEN	Buffer read ready interrupt enable 0 Masked 1 Enabled
27 BWRIEN	Buffer write ready interrupt enable

Table continues on the next page...

eSDHC_IRQSIGEN field descriptions (continued)

Field	Description
	0 Masked 1 Enabled
28 DINTIEN	DMA interrupt enable 0 Masked 1 Enabled
29 BGEIEN	Block gap event interrupt enable 0 Masked 1 Enabled
30 TCIEN	Transfer complete interrupt enable 0 Masked 1 Enabled
31 CCIEN	Command complete interrupt enable 0 Masked 1 Enabled

15.4.13 Auto CMD12 status (eSDHC_AUTOC12ERR)

When IRQSTAT[AC12E] is set, the host driver checks this register to identify what kind of error Auto CMD12 indicated. This register is valid only when IRQSTAT[AC12E] is set.

Table 15-29. Relationship between command CRC error and command timeout error for Auto CMD12

Auto CMD12 CRC error	Auto CMD12 timeout error	Types of error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	SDHC_CMD line conflict

There are three scenarios when AUTOC12ERR can be changed:

1. When eSDHC is going to issue Auto CMD12
 - Set AC12NE if Auto CMD12 cannot be issued due to an error in the previous command.
 - Clear AC12NE if Auto CMD12 is issued.
2. At the end bit of an Auto CMD12 response
 - Check received responses by checking the error bits 1-4.

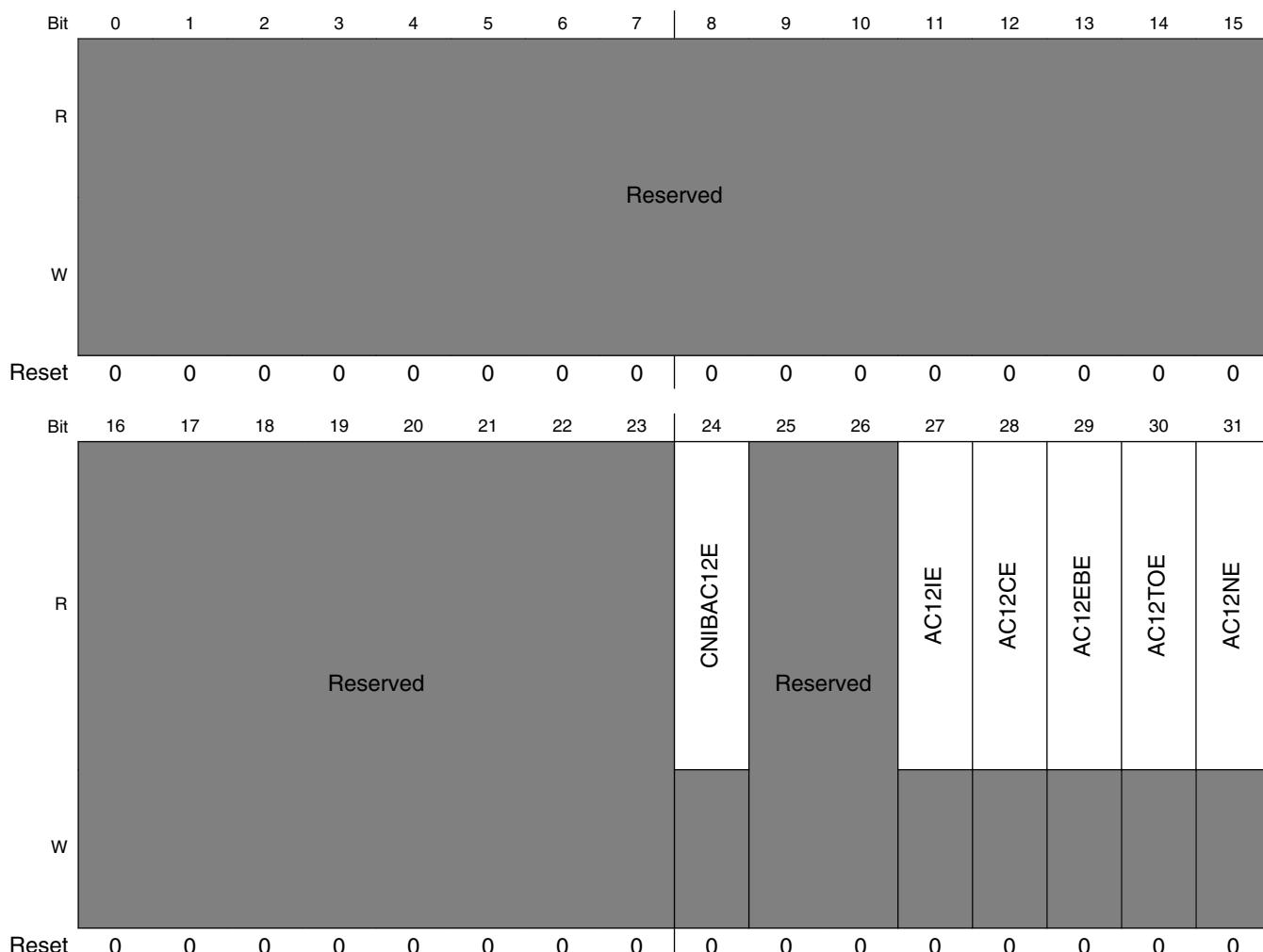
- Set if error is detected.
- Clear if error is not detected.

3. Before reading AUTOC12ERR[CNIBAC12E]

- Set CNIBAC12E if there is a command that cannot be issued
- Clear CNIBAC12E if there is no command to issue

The timing of generating the Auto CMD12 error and writing to the command register is asynchronous. The command may be blocked by any Auto CMD12 error causing CNIBAC12E to be set. Therefore, it is suggested to read this register only when IRQSTAT[AC12E] is set. An Auto CMD12 error interrupt is generated when one of the error bits 0-4 is set. The CNIBAC12E error bit does not generate an interrupt.

Address: 11_4000h base + 3Ch offset = 11_403Ch



eSDHC_AUTOC12ERR field descriptions

Field	Description
0-23 -	This field is reserved.

Table continues on the next page...

eSDHC_AUTOC12ERR field descriptions (continued)

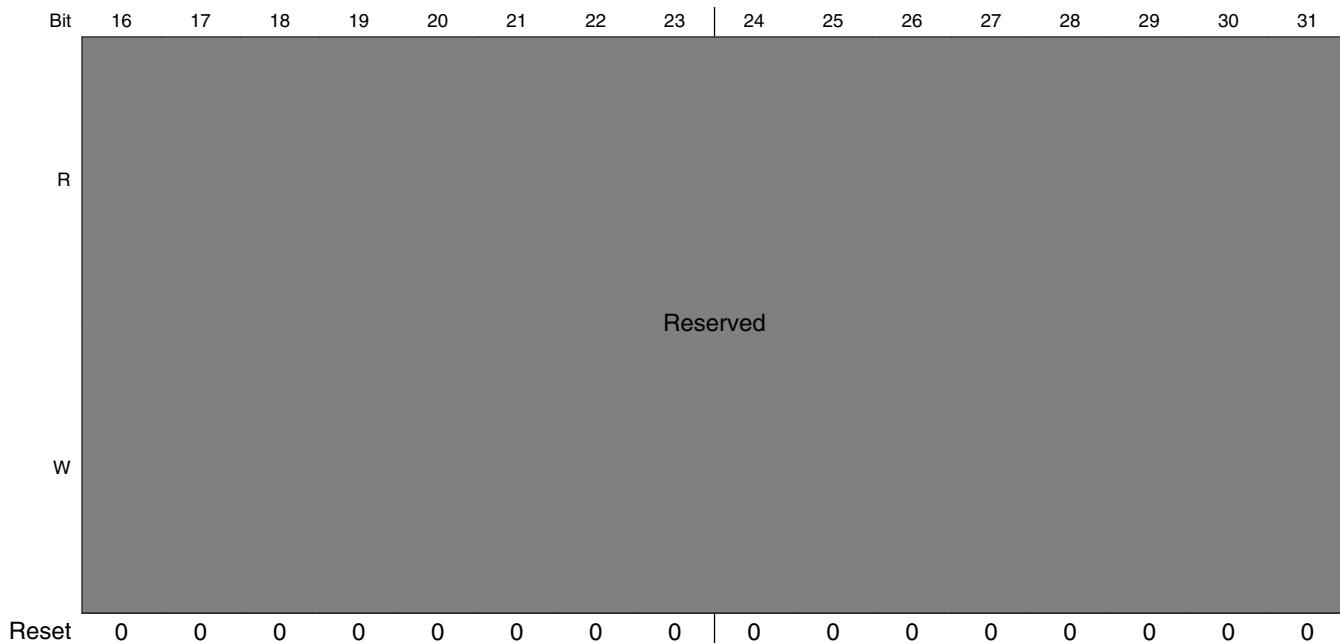
Field	Description
24 CNIBAC12E	Command not issued by Auto CMD12 error. This bit is set when CMD_wo_DAT is not executed due to an Auto CMD12 error (D04-D01). 0 No error 1 Not Issued
25–26 -	This field is reserved.
27 AC12IE	Auto CMD12 index error. Occurs if the command index error occurs in response to a command. 0 No error 1 Error, the CMD index in response is not CMD12
28 AC12CE	Auto CMD12 CRC error. Occurs when detecting a CRC error in the command response. 0 No CRC error 1 CRC error met in Auto CMD12 response
29 AC12EBE	Auto CMD12 end bit error. Occurs when detecting that the end bit of command response is 0 when it should be 1. 0 No error 1 End bit error generated
30 AC12TOE	Auto CMD12 timeout error. Occurs if no response is returned within 64 SDHC_CLK cycles from the end bit of the command. If this bit is set, the other error status bits (2-4) are meaningless. 0 No error 1 Time out
31 AC12NE	Auto CMD12 not executed. If a memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit means eSDHC cannot issue Auto CMD12 to stop the memory multiple block data transfer due to some error. If this bit is set, the other error status bits (1-4) are meaningless. 0 Executed 1 Not executed

15.4.14 Host controller capabilities (eSDHC_HOSTCAPBLT)

The host controller capabilities provides the host driver with information specific to the eSDHC implementation. The value in this register does not change in a software reset, and any write to this register is ignored.

Address: 11_4000h base + 40h offset = 11_4040h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					VS18	VS30	VS33	Reserved	DMAS	HSS	Reserved			MBL	
W	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1



eSDHC_HOSTCAPBLT field descriptions

Field	Description
0–4 -	This field is reserved.
5 VS18	Voltage support 1.8 V. This bit depends on the host system ability. Note that on P4080 Rev 2, the reset value indicates the device supports 1.8 V operation, when in reality, it does not. Software running on P4080, Rev 2, should ignore this bit when reading from this register. 0 1.8 V not supported 1 1.8 V supported
6 VS30	Voltage support 3.0 V. This bit depends on the host system ability. Note that on P4080, Rev 2, the reset value indicates the device supports 3.0 V operation, when in reality, it does not. Software running on P4080, Rev 2, should ignore this bit when reading from this register. 0 3.0 V not supported 1 3.0 V supported
7 VS33	Voltage support 3.3 V. This bit depends on the host system ability. 0 3.3 V not supported 1 3.3 V supported
8 -	This field is reserved. Note that on P4080, Rev 2, the reset value for bit 8 indicates the device supports suspend/resume operation, when in reality, it does not. Software running on P4080, Rev 2, should ignore this bit when reading from this register.
9 DMAS	DMA support. Indicates if eSDHC is capable of using internal DMA to transfer data between system memory and the data buffer directly. 0 DMA not supported 1 DMA supported

Table continues on the next page...

eSDHC_HOSTCAPBLT field descriptions (continued)

Field	Description
10 HSS	High speed support. Indicates if the eSDHC supports high speed mode and the host system can supply the SD clock frequency from 25 to 50 MHz. 0 High speed supported 1 High speed supported
11–12 -	This field is reserved. Note that on P4080 Rev 2, the reset value for bit 11 indicates the device supports ADMA1 capability, when in reality, it does not. Software running on P4080, Rev 2, should ignore this bit when reading from this register.
13–15 MBL	Max block length. Indicates the maximum block size that the host driver can read and write to the buffer in the eSDHC. The buffer should transfer block size without wait cycles. 000 512 bytes 001 1024 bytes 010 2048 bytes 011 4096 bytes
16–31 -	This field is reserved.

15.4.15 Watermark level (eSDHC_WML)

Both write and read watermark levels are configurable. The value can be any number from 1-128 words.

Address: 11_4000h base + 44h offset = 11_4044h

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

eSDHC_WML field descriptions

Field	Description
0–7 -	This field is reserved.
8–15 WR_WML	Write watermark level. Number of 32-bit words of watermark level in DMA write operation. Also, the number of words of write burst length.
16–23 -	This field is reserved.
24–31 RD_WML	Read watermark level. Number of 32-bit words of watermark level in DMA read operation. Also, the number of words of read burst length. NOTE: On P4080 Rev 2, the maximum value for RD_WML is 0x10, which means 16 words (64 bytes). Setting RD_WML to a higher value results in non-predicted behavior.

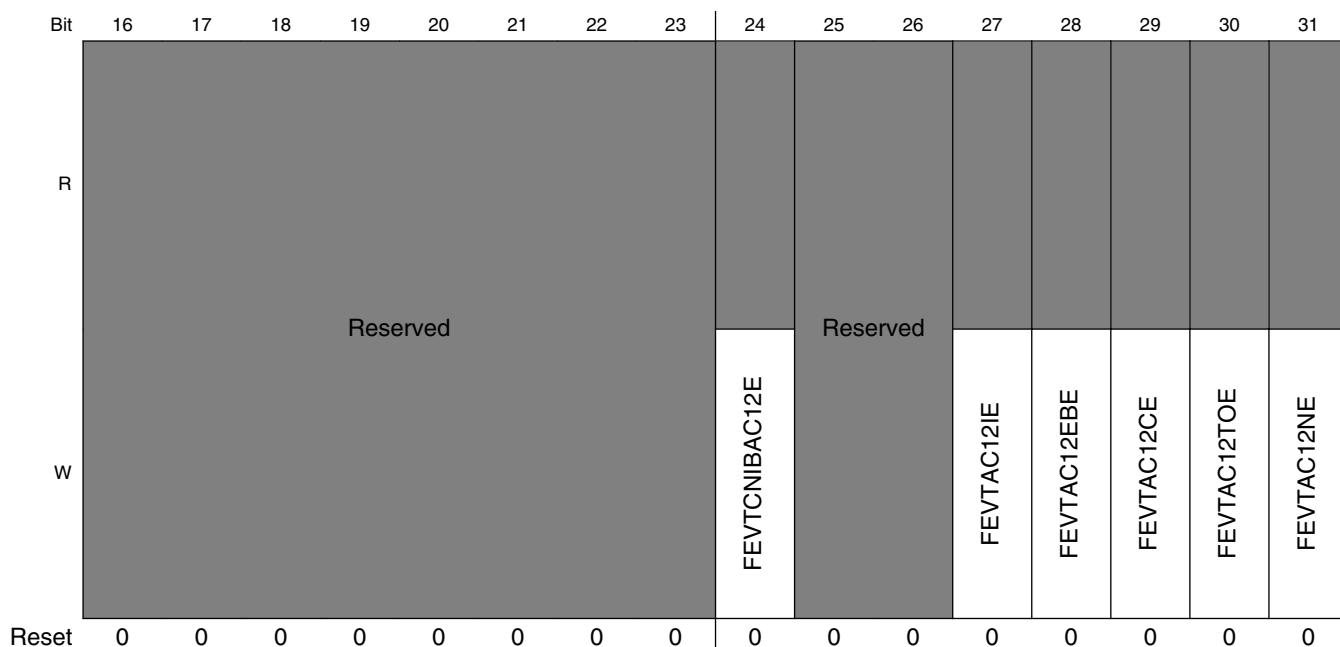
15.4.16 Force event (eSDHC_FEVT)

The force event register is not a physically implemented register. Rather, it is an address to which the IRQSTAT register can be written if the corresponding bit of IRQSTATEN is set. Therefore, this register is a write-only register and writing zero has no effect. Writing 1 to this register sets the corresponding bit of IRQSTAT. Reading from this register always returns zeroes.

Forcing a card interrupt generates a short pulse on the SDHC_DAT[1] line, and the driver may treat this interrupt as normal. The interrupt service routine may skip polling the card-interrupt source as the interrupt is self-cleared.

Address: 11_4000h base + 50h offset = 11_4050h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved				Reserved				Reserved							
W					FEVTDMAE				FEVTDEBE		FEVTDCE		FEVTDTOE		FEVTCIE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**eSDHC_FEVT field descriptions**

Field	Description
0–2 -	This field is reserved.
3 FEVTDMAE	Force event DMA error. Forces IRQSTAT[DMAE] to set.
4–6 -	This field is reserved.
7 FEVTAC12E	Force event Auto CMD12 error. Forces IRQSTAT[AC12E] to set.
8 -	This field is reserved.
9 FEVTDEBE	Force event data end bit error. Forces IRQSTAT[DEBE] to set.
10 FEVTDCE	Force event data CRC error. Forces IRQSTAT[DCE] to set.
11 FEVTDTOE	Force event data time out error. Forces IRQSTAT[DTOE] to set.
12 FEVTCIE	Force event command index error. Forces IRQSTAT[CCE] to set.
13 FEVTCEBE	Force event command end bit error. Forces IRQSTAT[CEBE] to set.
14 FEVTCCE	Force event command CRC error. Forces IRQSTAT[CCE] to set.
15 FEVTCTOE	Force event command time out error. Forces IRQSTAT[CTOE] to set.
16–23 -	This field is reserved.

Table continues on the next page...

eSDHC_FEVT field descriptions (continued)

Field	Description
24 FEVTCNIBAC12E	Force event command not executed by Auto CMD12 error. Forces AUTOC12ERR[CNIBAC12E] to set.
25–26 -	This field is reserved.
27 FEVTAC12IE	Force event Auto CMD12 index error. Forces AUTOC12ERR[AC12IE] to set.
28 FEVTAC12EBE	Force event Auto CMD12 end bit error. Forces AUTOC12ERR[AC12EBE] to set.
29 FEVTAC12CE	Force event Auto CMD12 CRC error. Forces AUTOC12ERR[AC12CE] to set.
30 FEVTAC12TOE	Force event Auto CMD12 time out error. Forces AUTOC12ERR[AC12TOE] to set.
31 FEVTAC12NE	Force event Auto CMD12 not executed. Forces AUTOC12ERR[AC12NE] to set.

15.4.17 Host controller version (eSDHC_HOSTVER)

The host controller version register contains the version for the vendor and the host controller. All the bits are read-only.

Address: 11_4000h base + FCh offset = 11_40FCh

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

eSDHC_HOSTVER field descriptions

Field	Description
0–15 -	This field is reserved.
16–23 VVN	Vendor version number. The host driver should not use this status. The upper and the lower 4-bits indicate the version. Settings not shown are reserved. 0x00 Freescale eSDHC version 1.0 0x01 Freescale eSDHC version 1.1> 0x10 Freescale eSDHC version 2.0 0x11 Freescale eSDHC version 2.1 0x12 Freescale eSDHC version 2.2

Table continues on the next page...

eSDHC_HOSTVER field descriptions (continued)

Field	Description
24–31 SVN	Specification version number. Indicates for the host controller specification version. The upper and the lower 4-bits indicate the version. Settings not shown are reserved. 0x00 SD Host Specification Version 1.0 0x01 SD Host Specification Version 2.0, supports the test event register .

15.4.18 DMA control register (eSDHC_DCR)

The DMA control register controls various settings that affect the system response to DMA operations.

Address: 11_4000h base + 40Ch offset = 11_440Ch

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

eSDHC_DCR field descriptions

Field	Description
0–24 -	This field is reserved.
25 SNOOP	Snoop attribute. 0 DMA transactions are not snooped by the CPU data cache 1 DMA transactions are snooped by the CPU data cache
26–28 -	This field is reserved.
29 RD_SAFE	Read safe. This bit should be set only if the target of a read-DMA operation is a well-behaved memory that is not affected by the read operation and returns the same data if read again from the same location. This means that unaligned reading operation can be rounded up to enable more efficient read operations.

Table continues on the next page...

eSDHC_DCR field descriptions (continued)

Field	Description
	0 It is not safe to read more bytes than were intended 1 It is safe to read more bytes than were intended
30 RD_PFE	Read prefetch enable. This bit should be set if the target of read-DMA operation is a well-behaved memory that is not affected by the read operation and returns the same data if read again from the same location. This means that prefetch of data can be done by the internal bus units and it results in faster read completion. 0 It is not allowed to prefetch data on DMA read operation 1 It is allowed to prefetch data on DMA read operation
31 RD_PF_SIZE	Read prefetch size. Determines the prefetch byte count to be used if RD_PFE is set. 0 64 bytes prefetch 1 32 bytes prefetch

15.5 eSDHC functional description

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA interface, register bank, register bus interface, dual-port memory wrapper, data/command controller, clock and reset manager, and clock generator.

15.5.1 Data buffer

The eSDHC uses one configurable data buffer so that data can be transferred between the internal system bus (register bus or bus) and the SD card in an optimized manner to maximize throughput between the two clock domains (the IP peripheral clock and the master clock).

See the figure below for an illustration of the buffer scheme.

The buffer is used as temporary storage for data being transferred between the host system and the card. The burst lengths for read and write are both configurable and can be any value between 1 and 128 words.

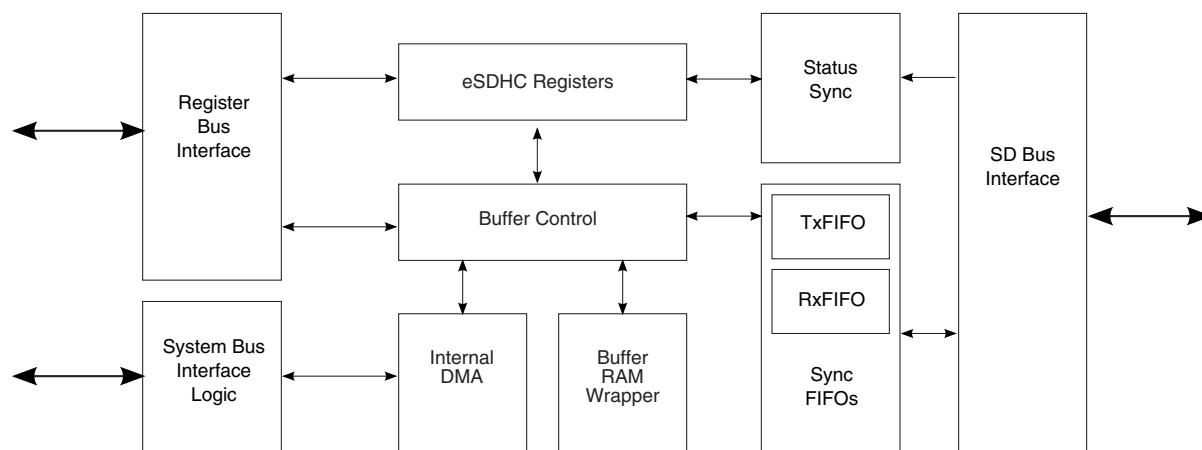


Figure 15-25. eSDHC buffer scheme

For a host read operation, when the amount of data exceeds the RD_WML value, the eSDHC sets PRSSTAT[BREN] and either:

- Issues a DMA request to inform the system to read the data
- Issues a DMA interrupt to inform the system to read the data
- When granted access permission, the internal DMA burst-reads RD_WML number of words

Conversely, for a host write operation, when the amount of buffer spaces exceeds the WR_WML value, the eSDHC sets PRSSTAT[BWEN] and either:

- Issues a DMA request to inform the system to write data to the buffer
- Issues a DMA interrupt to inform the system to write data to the buffer
- When granted crossbar access permission, the internal DMA burst-writes WR_WML number of words into the buffer

15.5.1.1 Write operation sequence

There are two ways to write data into the buffer when the user transfers data to the card:

- Processor core polling IRQSTAT[BWR] (interrupt or polling)
- Internal DMA

When the internal DMA is not used (XFERTYP[DMAEN] is not set when the command is sent), the eSDHC asserts an external DMA request when more than WML[WR_WML] number of empty buffer word slots are available and ready for receiving new data. At the same time, the eSDHC sets IRQSTAT[BWR]. The buffer write ready interrupt is generated if it is enabled by software.

When the internal DMA is used, the eSDHC does not inform the system before all the required number of bytes are transferred and no error is encountered. When an error occurs during the data transfer, the eSDHC aborts the data transfer and abandons the current block. The host driver should read the content of the DMA system address register to obtain the start address of abandoned data block. If the current data transfer is in multi-block mode, the eSDHC does not automatically send CMD12 even though XFERTYP[AC12EN] is set. Therefore, in this scenario, the host driver should send CMD12 and restart the write operation from that address. It is recommended that a software reset for data is applied before the transfer is restarted after error recovery.

The eSDHC does not start data transmission until the WML[WR_WML] number of words of data can be held in the buffer. If the buffer is empty and the host system does not write data in time, the eSDHC stops the SDHC_CLK to avoid a data buffer underrun situation.

15.5.1.2 Read operation sequence

There are two ways to read data from the buffer when transferring data to the card:

- Processor core polling IRQSTAT[BRR] (interrupt or polling)
- Internal DMA

When the internal DMA is not used (that is, XFERTYP[DMAEN] is not set when the command is sent), the eSDHC asserts a DMA request when more than WML[RD_WML] number of words are available and ready for the system to fetch the data. At the same time, the eSDHC sets the IRQSTAT[BRR] bit. The buffer read ready interrupt is generated if it is enabled by software.

When the internal DMA is used, the eSDHC does not inform the system before all the required number of bytes are transferred and no error is encountered. When an error occurs during the data transfer, the eSDHC aborts the data transfer and abandons the current block. The host driver should read the content of the DMA system address register to obtain the start address of the abandoned data block. If the current data transfer is in multiblock mode, the eSDHC does not automatically send CMD12, even though XFERTYP[AC12EN] is set. Therefore, in this scenario, the host driver should send CMD12 and restart the read operation from that address. It is recommended that a software reset for data is applied before the transfer is restarted after error recovery.

The eSDHC does not start data transmission until the WML[RD_WML] number of words of data are in the buffer. If the buffer is full and the host system does not read the data in time, the eSDHC stops the SDHC_CLK to avoid a data buffer overrun situation.

15.5.1.3 Data buffer size

To optimize use of the buffer, the buffer size must be known.

In the eSDHC, the data buffer can hold up to 128 32-bit words, and the read and write watermark levels are each configurable from one to 128 words. The host driver may configure the values according to the system situation and requirements.

During multiblock data transfer, the maximum block length is 4096 bytes, which can satisfy all the requirements from MMC and SD cards. Any block length less than this value is also allowed. The only restriction is from the external card since it may not support such a large block or a partial block access that is not an integer multiple of 512 bytes.

15.5.2 DMA interface

The internal DMA implements a DMA engine and master.

When the internal DMA is enabled (that is, XFERTYP[DMAEN] is set), the buffer interrupt status bits remain set if they are enabled. To avoid setting them, clear IRQSTATEN[BWRSEN, BRRSEN]. See [Figure 15-26](#) for an illustration of the DMA interface block. Do not use the internal DMA to read (or write) data when the CPU or an external DMA writes (or reads) the data through the DATPORT register.

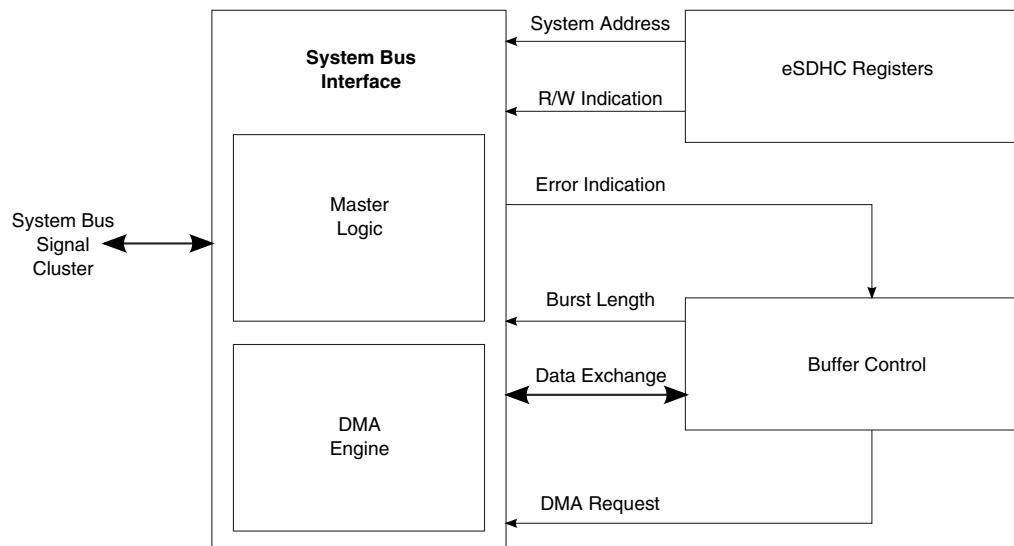


Figure 15-26. DMA system bus interface block

15.5.2.1 Internal DMA request

When the watermark level is met in the data transfer and the internal DMA is enabled, the data buffer block sends a DMA request to this block.

Meanwhile, the external DMA request is disabled. The delay of response from the internal DMA engine depends on the system bus loading and the priority assigned to the eSDHC. The DMA engine does not respond to the request during its burst transfer, and it is available as soon as the burst is over. When an access on the buffer is made, the data buffer deasserts the request. When the internal DMA accesses the buffer, the data buffer updates its internal buffer pointer. When the watermark level is satisfied, another DMA request is sent.

The data transfer is in the block unit and the last watermark level is always set to the remaining number of words. For example, for a multi-block data read with each block the size of 31 bytes, the burst length is set at six words. After the first burst transfer, if there are more than seven bytes in the buffer, which might include some data from the next block, another DMA read request is sent. This is because the remaining number of words to send for the current block is $(31 - 6 \times 4) \div 4 = 2$, and the eSDHC reads two words out of the buffer, with seven valid bytes and one stuff byte automatically added by the eSDHC.

15.5.2.2 DMA burst length

Maximum size is the only restriction on DMA burst length for the internal DMA engine.

The burst length for read or write can be one to 128 words. The actual DMA burst length depends on which is smaller: the configured watermark level or the remaining words of the current block.

Consider the example in [Internal DMA request](#). After six words are read, the burst length is two words to complete the 31-byte block. Then the burst length changes back to six words to prepare for the next 31-byte block. The host driver writer may take this variable burst length into account. For a constant burst length, the user may choose to configure the burst length as the divisor of block size.

15.5.2.3 Master interface

It is possible that the internal DMA engine fails during data transfer. When an error occurs, the DMA engine stops the transfer and goes into idle state; at the same time, the internal data buffer stops working. IRQSTAT[DMAE] is set to inform the driver.

Once the IRQSTAT[DMAE] interrupt is received, software should send CMD12 to abort the current transfer and read DSADDR[DS_ADDR] to obtain the start address of the corrupted block. After the DMA error is fixed, the software applies a data reset and restarts the transfer from this address to recover the corrupted block.

15.5.3 SD protocol unit

In addition to handling all SD protocol affairs, as well as other miscellaneous functions, the SD protocol unit performs the following:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data and its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the SDHC_DAT[0] line
- Gates off the SD clock when the buffer announces a danger status
- Detects the write-protect state

The SD protocol unit consists of the following four submodules:

- [SD transceiver](#)
- [SD clock and monitor](#)
- [Command agent](#)
- [Data agent](#)

15.5.3.1 SD transceiver

In the SD protocol unit, the transceiver is the main control module.

It consists of an FSM and the control module, from which the control signals for all other three modules are generated.

15.5.3.2 SD clock and monitor

This module monitors the signal level on all eight data lines and the command lines, directly route the level values into the register bank for the driver to debug with.

The transceiver reports the card insertion state according to the SDHC_CD_B state, or signal level on SDHC_DAT[3] line when PROCTL[D3CD] is set.

The module detects the SDHC_WP (write protect) line. With the information of SDHC_WP state, the register bank ignores the command accompanied by write operation, when the SD_WP switch is on.

If the internal data buffer is in danger and the SD clock must be gated off to avoid buffer over/underrun, this module asserts the gate of output SD clock to shut the clock off. When the buffer danger is eliminated when system access of the buffer catches up, the clock gate of this module is open and the SD clock is active again.

15.5.3.3 Command agent

The command agent deals with the transactions on the SDHC_CMD line.

See the figure below for an illustration of the structure for the command CRC shift register.

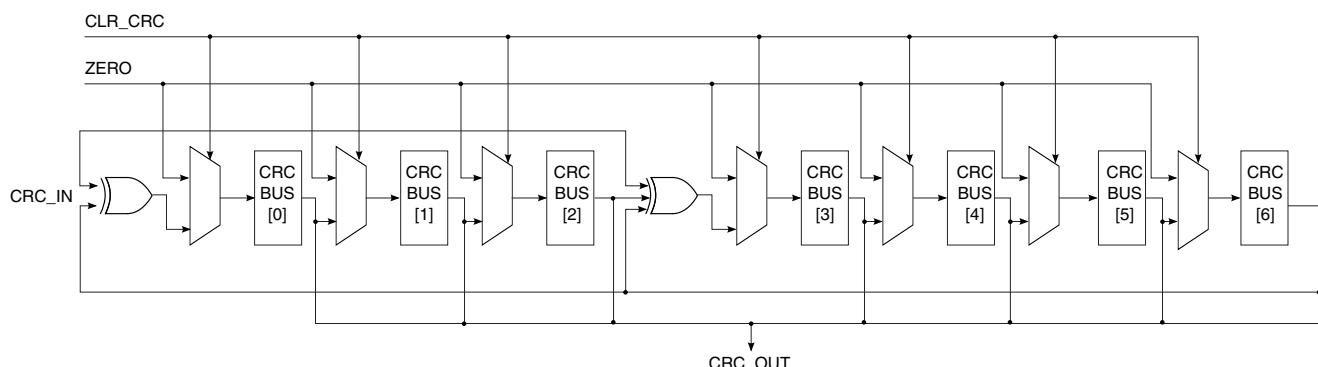


Figure 15-27. Command CRC shift register

The CRC polynomials for the SDHC_CMD are as follows:

1. Generator polynomial: $G(x) = x^7 + x^3 + 1$

$$M(x) = (\text{first bit}) x^{n-1} + (\text{second bit}) x^{n-2} + \dots + (\text{last bit}) x^0$$

$$\text{CRC}[6:0] = \text{Remainder } [(M(x) x^{n-1}) \div G(x)]$$

15.5.3.4 Data agent

The data agent handles the transactions on the eight data lines.

The CRC polynomials for the SDHC_DAT are as follows:

1. Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$

$$M(x) = (\text{first bit}) x x^n + (\text{second bit}) x x^{n-1} + \dots + (\text{last bit}) x x^0$$

$$\text{CRC}[15:0] = \text{Remainder } [(M(x) x x^{16}) \div G(x)]$$

15.5.4 Clock and reset manager

This module controls the reset signals within the eSDHC.

Within eSDHC, there are four types of reset signals, all of which are fed into the clock and reset manager module: hardware reset, software reset for all, software reset for data, and software reset for command. Stable signals are generated inside the module in order to reset all other modules.

In addition, the clock and reset manager module:

- gates off all inside signals,
- monitors the activities of all other modules,
- supplies the clocks for other modules,
- and, when enabled, automatically gates off the corresponding clocks.

15.5.5 Clock generator

The clock generator generates the SDHC_CLK by dividing the internal bus clock into two stages.

Refer to [Figure 15-28](#) for the structure of the divider, in which the term base represents the frequency of the internal bus clock . Refer to SYSCTL[SDCLKFS] and SYSCTL[DVS] (see [System control \(eSDHC_SYSCTL\)](#)) to select the divisor values.

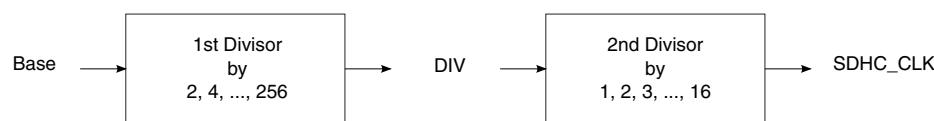


Figure 15-28. Two stages of clock divider

The first stage is a prescaler. The frequency of clock output from this stage, DIV, can be base/2, base/4, ..., or base/256.

The second stage outputs the actual clock, SDHC_CLK, as the driving clock for all sub-modules of SD protocol unit, and the sync FIFOs in [Figure 15-25](#) to synchronize with the data rate from the internal data buffer. It can be div, div/2, div/3,..., or div/16. Thus, the highest frequency of SDHC_CLK generated by the internal bus clock is base while the lowest frequency is base/4096.

15.5.6 Card insertion and removal detection

The eSDHC uses the SDHC_DAT[3] pin or the SDHC_CD_B pin to detect card insertion or removal.

- When the SDHC_DAT[3] pin is used for card detection, the user needs to pull down this pad as a default state. When there is no card on the MMC/SD bus, the SDHC_DAT[3] is pulled to a low voltage level by default. When any card is inserted to or removed from the socket, the eSDHC detects the logic value changes on the SDHC_DAT[3] pin and generates an interrupt.
- When the SDHC_DAT[3] pin is not used for card detection, SDHC_CD_B must be connected for card detection. It may be implemented by a GPIO. Whether or not SDHC_DAT[3] is configured for card detection, SDHC_CD_B is always a reference for card detection, either SDHC_DAT[3] or SDHC_CD_B reports the card inserted, the eSDHC informs the host system that a card is inserted, and the interrupt is sent if it is enabled.

15.5.7 Power management and wake-up events

When there is no operation between the eSDHC and the card through the SD bus, the user can completely disable the internal clocks in the chip-level clock-control module to save power. When using the eSDHC to communicate with the card, it can enable the clock and start the operation. This can be done by clearing the SCCR[SDHCCM] bits.

In some circumstances, when the clocks to eSDHC are disabled, or when the system is in low-power mode, there are events that require the user to enable the clock and handle the event. These events are called wake-up interrupts. The eSDHC can generate these interrupts even when there are no clocks enabled. The two interrupts that can be used as wake-up events are as follows:

- Card removal interrupt
- Card insertion interrupt

These two wake-up events (or wake-up interrupts) can also be used to wake the system from low-power modes.

NOTE

To make the interrupt as a wake-up event when all clocks to the eSDHC are disabled or when the entire system is in low-power mode, the corresponding wake-up enable bit must be set. See [Protocol control \(eSDHC_PROCTL\)](#) for more information on the eSDHC PROCTL register.

15.5.7.1 Setting wake-up events

For the eSDHC to respond to a wake-up event, the software must set the respective wake-up enable bit before the CPU enters sleep mode.

See [Protocol control \(eSDHC_PROCTL\)](#), for more information on the wake-up enable bits.

Before the software disables the host clock, it should ensure that all of the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

15.6 Initialization/application information

All communication between the system and cards is controlled by the host.

The host sends the following types of commands: broadcast and addressed (point-to-point) commands.

Broadcast commands, such as GO_IDLE_STATE, SEND_OP_COND or ALL_SEND_CID, are intended for all cards. In broadcast mode, all cards are in the open-drain mode to avoid bus contention. See [Commands for MMC/SD](#), for the bc and bcr category commands.

After the broadcast command CMD3 is issued, the cards enter standby mode. From this point, addressed commands are used. In this mode, the SDHC_CMD/SDHC_DAT I/O pads turn to push-pull mode in order to provide the driving capability for maximum frequency operation. See [Commands for MMC/SD](#), for the ac and adtc category commands.

15.6.1 Command send and response receive basic operation

Assuming the data type WORD is an unsigned 32-bit integer, the following flow acts as a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
    WORD wCmd; // 32-bit integer to make up the data to write into the XFERTYP register, it is
                // recommended to implement in a bit-field manner
    wCmd = (<cmd_index> & 0x3f) << 24; // set the first 8 bits as '00'+<cmd_index>
    set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, and DTDSEL according to the command index;
                // XFERTYP register bits
    if (internal DMA is used) wCmd |= 0x1;
    if (multi-block transfer) {
        set XFERTYP[MSBSEL] bit;
        if (finite block number) {
            set XFERTYP[BCEN] bit;
            if (auto12 command is to use) set XFERTYP[AC12EN] bit;
        }
    }
    write_reg(CMDARG, <cmd_arg>); // configure the command argument
    write_reg(XFERTYP, wCmd); // set XFERTYP register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
    while (IRQSTAT[CC] is not set); // wait until command complete bit is set
    read IRQSTAT register and check if any error bits about command are set;
    if (any error bits are set) report error;
    write 1 to clear IRQSTAT[CC] and all command error bits;
}
```

For the sake of simplicity, the function `wait_for_response` is implemented here by means of polling. For an effective and formal way, the response is usually checked after the command complete interrupt is received. This ensures the corresponding interrupt status bits are enabled.

For some scenarios, the response timeout is expected. For instance, after all cards respond to CMD3 and enter standby state, there is no response to the host when CMD2 is sent. The host driver should manage false errors similar to this with caution.

15.6.2 Card identification mode

When a card is inserted into the socket or the card is reset by the host, the host needs to validate the operation voltage range, identify the cards, and request the cards to publish the relative card address (RCA) or to set the RCA for the MMCs.

15.6.2.1 Card detect

This figure shows a flow diagram of the detection of MMC and SD cards using the eSDHC.

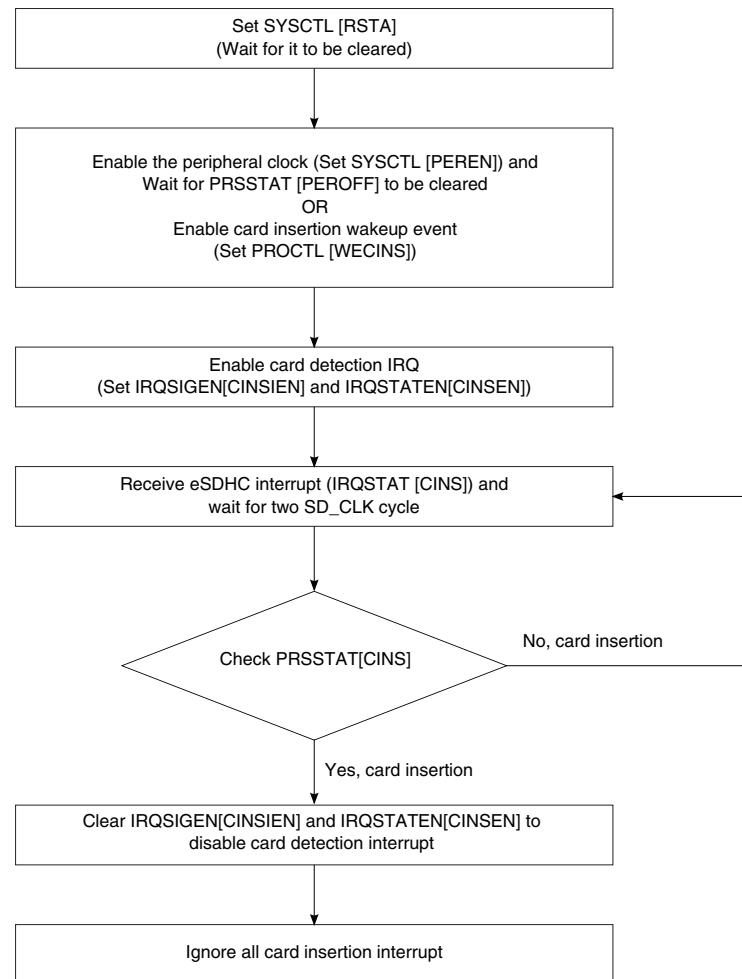


Figure 15-29. Flow diagram for card detection

1. Set SYSCTL[RSTA] and wait for it to be cleared.
2. Set SYSCTL[PEREN] to enable the clocks and wait for PRSSTAT[PEROFF] to be cleared or enable wakeup event (set PROCTL[WECINS]).
3. Set IRQSIGEN[CINSIEN] and IRQSTATEN[CINSEN] to enable card detection interrupt.
4. When an IRQSTAT[CINS] interrupt from eSDHC is received, wait for two SD_CLK cycle and then check PRSSTAT[CINS] to see if the interrupt is caused by card insertion.
5. Clear the IRQSIGEN[CINSIEN] and IRQSTATEN[CINSEN] to disable card detection interrupt and ignore all card insertion interrupt afterwards.

15.6.2.2 Reset

The host consists of the following types of reset:

- Hardware reset (card and host), which is driven by POR (power-on reset).
- Software reset (host only), which is proceeded by the write operation on the SYSCTL[RSTD], SYSCTL[RSTC], or SYSCTL[STA] bits to reset the data part, command part, or all parts of the host controller, respectively.
- Card reset (card only). The command CMD0, GO_IDLE_STATE, is the software reset command for all types of MMCs and SD memory cards. This command sets each card into idle state regardless of the current card state. The cards are initialized with a default relative card address (RCA = 0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host must validate the voltage range of the card. See [Figure 15-30](#) for the software flow chart.

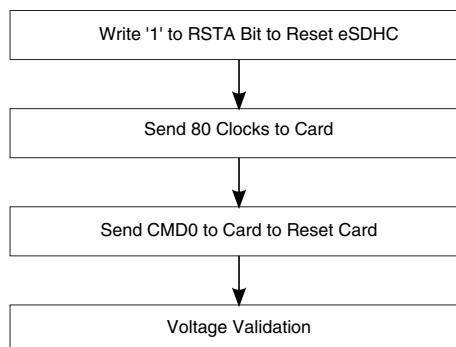


Figure 15-30. Flow chart for reset of eSDHC card

The following pseudocode shows how to initialize the eSDHC host controller and the memory card.

```

software_reset()
{
    Configure the I/O muxes to select SD signals;
    set_bit(SYSCtrl, RSTA);                                // software reset the host
    set_SYSCTL[DTOCV and SDCLKFS];                         // get the SDHC_CLK of frequency
around 400 kHz
    poll PRSSTAT[CIHB and CDIHB];                          // wait until both bits are
cleared
    set_bit(SYSCtrl, INTIA);                                // send 80 clock ticks for
card to power-up
    If the card is SD/MMC
        send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with
CMD0
}
  
```

15.6.2.3 Voltage validation

All cards should be able to establish communication with the host by using any operation voltage in the maximum allowed voltage range, as specified in this standard.

However, the supported minimum and maximum values for V_{DD} are defined in the operation conditions register (OCR) and may not cover the whole range. Cards that store the CID (card identification) and CSD data in the preloaded memory are only able to communicate this information under data transfer V_{DD} conditions. This means that if the host and card have different V_{DD} ranges, the card is not able to complete the identification cycle, nor is it able to send CSD data.

Therefore, a special command is available, as follows:

- SEND_OP_CONT (CMD1 for MMC),
- SD_SEND_OP_CONT (ACMD41 for SD Memory)

The voltage validation procedure is designed to provide a mechanism to identify and reject cards that do not match the V_{DD} range(s) desired by the host. This is accomplished when the host sends the desired V_{DD} voltage window as the operand of this command. Cards that cannot perform data transfer in the specified range must discontinue any further bus operations and enter the inactive state.

By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the inactive state. This query should be used if the host is able to select a common voltage range or if a notification should be sent to the system when a non-usuable card in the stack is detected.

15.6.2.4 Card registry

Card registry on the MMC and SD cards differs.

SD card registry is as follows:

NOTE

The identification process starts at a clock rate lower than 400 kHz and a power voltage higher than 2.7 V, as defined by the card specification. At this time, the SDHC_CMD line output drivers are push-pull drivers instead of open-drain.

1. After the bus is activated, the host requests the cards to send their valid operation conditions. The response to ACMD41 is the operation condition register of the card.
2. The host should send the same command to all of the new cards in the system.
Incompatible cards are placed into the inactive state.

3. The host then issues the command, ALL_SEND_CID (CMD2), to each card to get its CID. Cards that are currently unidentified (that is, in ready state), send their CID number as the response.
4. After the CID is sent by the card, the card goes into the identification state.
5. The host then issues Send_Relative_Addr (CMD3), requesting the card publish a new relative card address (RCA) that is shorter than CID. This RCA is used to address the card for future data transfer operations.
6. Once the RCA is received, the card changes to standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.
7. The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from the cards in system.

MMC card registry is as follows:

NOTE

For MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open-drain driver stages on the SDHC_CMD line allow for parallel card operation during card identification.

1. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the wired-OR operation on the condition restrictions of all cards in the system. Incompatible cards are sent into inactive state.
2. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique CID number. All unidentified cards (the cards in ready state) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods stop sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can successfully send its full CID to the host. This card then goes into identification state.
3. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to this card a relative card address (RCA).
4. Once the RCA is received, the card state changes to standby state, the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull.
5. The host repeats the process, namely CMD2 and CMD3, until the host receives a time-out condition to recognize completion of the identification process.

15.6.3 Card access

The following sections describe the supported access modes with external cards.

15.6.3.1 Block write

This section describes the process of writing data to external cards in block mode.

15.6.3.1.1 Normal write

During block write (CMD24-27), one or more blocks of data are transferred from the host to the card; the host appends a CRC to the end of each block.

If the CRC fails, the card should indicate the failure on the SDHC_DAT line (see below). The transferred data is discarded and not written, and all further transmitted blocks (in multi-block write mode) are ignored.

When the host uses partial blocks whose accumulated length is not block-aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, as defined in the MMC/SD specification, and then waits in the receive-data state for a stop command, ignoring all further data transfers. The write operation is also aborted when the host attempts to write over a write-protected area.

For MMC and SD cards, programming the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC-protected. If a part of the CSD or CID register is stored in the ROM, this unchangeable section must match the corresponding section of the receive buffer. If this match fails, the card reports an error and does not change any register content.

Some cards may require a long and unpredictable period of time to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing. When its write buffer is full and unable to accept new data from a new WRITE_BLOCK command, the card holds the SDHC_DAT line low. The host may poll the status of the card with a SEND_STATUS command (CMD13) cards, at any time and the card responds with its status. The card status indicates whether the card can accept new data or if the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card) to change the card into the standby state and release the

SDHC_DAT line without interrupting the write operation. When re-selecting the card, the host reactivates the busy indication by pulling SDHC_DAT low when programming is still in progress and the write buffer is unavailable.

For the sake of simplicity, the software flow described below incorporates the internal DMA, and the write operation is a multi-block write with Auto CMD12 enabled. For the CPU polling status method and different transfer natures, remove the internal DMA part of the procedure and insert alternative steps.

1. Check the card status and wait until the card is ready for data.
2. Set the card block length.
 - For MMC/SD cards, use SET_BLOCKLEN (CMD16).
3. Set eSDHC BLKATTR[BLKSIZE] to the same block length set in the card in step 2.
4. Set eSDHC BLKATTR[BLKCNT] with the number of blocks to send.
5. Disable the buffer write ready interrupt, configure the DMA setting, and enable the eSDHC DMA when sending the command with data transfer. Set XFERTYP[AC12EN].
6. Wait for the transfer complete interrupt.
7. Check the status bit to see if a read CRC error or any other errors occurred between sending Auto CMD12 and receiving the response.

15.6.3.1.2 Write with pause

The write operation can be paused during the transfer.

Instead of stopping the SDHC_CLK at any time to pause all the operations which is also inaccessible to the host driver, the driver can set PROCTL[SABGREQ] to pause the transfer between the data blocks. Because there is no timeout condition in a write operation during the data blocks, a write operation to the cards can be paused in this way and if line SDHC_DAT0 is not required to de-assert to release busy state.

Similar to the flow described in [Normal write](#), the write with pause is shown with the same type of write operations:

1. Check the card status and wait until card is ready for data.
2. Set the card block length.
 - For MMC/SD cards, use SET_BLOCKLEN (CMD16)
3. Set the eSDHC BLKATTR[BLKSIZE] to the same as the block length set in the card in step 2.
4. Set eSDHC BLKATTR[BLKCNT] with the number of blocks to send.
5. Disable the buffer write ready interrupt, configure the DMA setting, and enable the eSDHC DMA when sending the command with data transfer. Set XFERTYP[AC12EN].
6. Set PROCTL[SABGREQ].

7. Wait for the transfer complete interrupt.
8. Clear PROCTL[SABGREQ].
9. Check the status bit to see if a read CRC error occurred.
10. Set PROCTL[CREQ] to continue the read operation.
11. Wait for the transfer complete interrupt.
12. Check the status bit to see if a read CRC error or any other errors occurred between sending Auto CMD12 and receiving the response.

The number of blocks left during the data transfer is accessible by reading the content of BLKATTR[BLKCNT]. Due to the data transfers and setting PROCTL[SABGREQ] are concurrent, along with the delay of register read and the register setting, the actual number of blocks left may not be the same as the value read earlier. The driver should read the value of BLKATTR[BLKCNT] after the transfer is paused and the transfer complete interrupt is received.

It is also possible that the transfer of the last block begins when the stop-at-block-gap request is sent to the buffer. In this case, the next block gap is the actual end of the transfer, and therefore, the request is ignored. The driver should treat this as a non-pause transfer and a common write operation.

When the write operation is paused, the data transfer inside the host system does not stop and the transfer remains active until the data buffer is full.

15.6.3.2 Block read

This section discusses normal read and read with pause.

15.6.3.2.1 Normal read

For block reads, the basic unit of a data transfer is a block whose maximum size is stored in areas defined in corresponding card specifications.

A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17, CMD18, CMD53, and so on, can initiate a block read. After completing the transfer, the card returns to the transfer state.

For multiblock reads, data blocks are continuously transferred until a stop command is issued. If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card that does not support partial block length, should detect the block misalignment at the beginning of the first misaligned block and report the error, depending on its card type.

For simplicity, the software flow described below incorporates the internal DMA, and the read operation is a multiblock read with Auto CMD12 enabled. For the other method (CPU polling status) and different transfer nature, the internal DMA part should be removed and the alternative steps are straightforward.

1. Check the card status and wait until the card is ready for data.
2. Set the card block length.
 - For MMC/SD cards, use SET_BLOCKLEN (CMD16)
3. Set eSDHC BLKATTR[BLKSIZE] to the same as the block length set in the card in step 2.
4. Set eSDHC BLKATTR[BLKCNT] with the number of blocks to send.
5. Disable the buffer read ready interrupt, configure the DMA setting, and enable the eSDHC DMA when sending the command with data transfer. Set XFERTYP[AC12EN].
6. Wait for the transfer complete interrupt.
7. Check the status bit to see if a read CRC error or any other errors occurred between sending Auto CMD12 and receiving the response.

15.6.3.2.2 Read with pause

In general, the read operation is not able to pause.

Similarly to the flow described in [Normal read](#), the read with pause is shown with the following read operations:

1. Set PROCTL[RWCTL].
2. Check the card status and wait until the card is ready for data.
3. Set the card block length.
 - For MMC/SD cards, use SET_BLOCKLEN (CMD16)
4. Set eSDHC BLKATTR[BLKSIZE] to the same as the block length set in the card in Step 2.
5. Set eSDHC BLKATTR[BLKCNT] with the number of blocks to send.
6. Disable the buffer read ready interrupt, configure the DMA setting, and enable the eSDHC DMA when sending the command with data transfer. Set XFERTYP[AC12EN].
7. Set PROCTL[SABGREQ].
8. Wait for the transfer complete interrupt.
9. Clear PROCTL[SABGREQ].
10. Check the status bit to see if a read CRC error occurred.
11. Set PROCTL[CREQ] to continue the read operation.
12. Wait for the transfer complete interrupt.
13. Check the status bit to see if a read CRC error or any other errors occurred between sending Auto CMD12 and receiving the response.

Similar to the write operation, it is possible to meet the ending block of the transfer when paused. In this case, the eSDHC ignores the stop-at-block-gap request and treats it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause is transferred to the host system; the internal data buffer is not flushed.

15.6.3.3 Transfer error

This section discusses CRC errors, internal DMA errors, and Auto CMD12 errors.

15.6.3.3.1 CRC error

At the end of a block transfer, a write CRC status error or read CRC error may occur.

For this type of error, the last block received should be discarded because the integrity of the data block is not guaranteed. It is recommended that the following data blocks be discarded and the block be retransferred from the corrupted one. For a multiblock transfer, the host driver should issue CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the XFERTYP[AC12EN, BCEN] are set, the eSDHC does not automatically send CMD12 because the last block is not transferred. On the other hand, if it is within the last block that CRC error occurs, Auto CMD12 is sent by the eSDHC. In this case, the driver should resend or re-obtain the last block with a single block transfer.

15.6.3.3.2 Internal DMA error

During the data transfer with the internal DMA, if the DMA engine encounters an error on the platform bus, the DMA operation is aborted and a DMA error interrupt is sent to the host system.

When acknowledged by such an interrupt, the driver should calculate the start address of the data block where the error occurred. The start address can be calculated by either of the following methods:

- Read the DSADDR[DSADDR] field. The error occurs during the previous burst. Therefore, by taking the block size, the previous burst length, and the start address of the next burst transfer into account, one can obtain the start address of the corrupted block.
- Read the BLKATTR[BLKCNT] field. The start address of the corrupted block can be calculated by the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block. However, if BCEN is not set, the

contents of the block attribute register does not change and this method does not work.

When a DMA error occurs, it is recommended to abort the current transfer by means of CMD12 (for multi-block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover the error.

15.6.3.3 Auto CMD12 error

After the last block of a multi-block transfer is sent or received and XFERTYP[AC12EN] is set when the data transfer is initiated by the data command, the eSDHC automatically sends CMD12 to the card to stop the transfer.

When an error occurs at this point, the host driver should respond as follows:

Auto CMD12 response timeout	It is not certain whether the command has been accepted by the card or not. The driver should clear the Auto CMD12 error status bits and resend CMD12 until it is accepted by the card.
Auto CMD12 response CRC error	Because CMD12 has been received by the card, the card aborts the transfer. The driver may ignore the error and clear the error status bit.
Auto CMD12 conflict error or not sent	The command was not sent. Therefore, the driver should send CMD12 manually.

15.6.3.4 Card interrupt

The external cards can inform the host controller through the use of special signals.

15.6.4 Switch function

MMCs transferring data with a bus width other than one-bit wide is a new feature added to the MMC specification.

The high-speed timing mode for all card devices is also newly-defined in recent various card specifications. To enable these new features, a type of switch command should be issued by the host driver.

For SD cards, the high-speed mode is queried and enabled by CMD6 (with the mnemonic symbol as SWITCH_FUNC); for MMCs, the high-speed mode is queried by CMD8 and enabled by CMD6 (with the mnemonic symbol as SWITCH).

The 4-bit and 8-bit bus width of MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by software reset, but such manner of restoring to normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

15.6.4.1 Query, enable and disable SD high speed mode

```
enable_sd_high_speed_mode(void)
{
    set BLKATTR[BLKCNT] to 1 (block), set BLKATTR[BLKSIZE] to 64 (bytes);
    send CMD6, with argument 0xFFFFFFF1 and read 64 bytes of data accompanying the R1

        response;
    wait data transfer done bit is set;
    check if the bit 401 of received 512 bit is set;
    if (bit 401 is '0') report the SD card does not support high speed mode and return;
    send CMD6, with argument 0x80FFFFFF1 and read 64 bytes of data accompanying the R1

        response;
    check if the bit field 379~376 is 0xF;
    if (the bit field is 0xF) report the function switch failed and return;
    change clock divisor value or configure the system clock feeding into eSDHC to
generate

        the card_clk of around 50MHz;
    (data transactions like normal peers)
}

disable_sd_high_speed_mode(void)
{
    set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
    send CMD6, with argument 0x80FFFFFF0 and read 64 bytes of data accompanying the R1

        response;
    check if the bit field 379~376 is 0xF;
    if (the bit field is 0xF) report the function switch failed and return;
    change clock divisor value or configure the system clock feeding into eSDHC to
generate

        the card_clk of the desired value below 25MHz;
    (data transactions like normal peers)
}
```

15.6.4.2 Query, enable and disable MMC high speed mode

```
enable_mmc_high_speed_mode(void)
{
    send CMD9 to get CSD value of MMC;
    check if the value of SPEC_VER field is 4 or above;
    if (SPEC_VER value is less than 4) report the MMC does not support high speed mode
and

    return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is

    26MHz or 52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
```

```

change clock divisor value or configure the system clock feeding into eSDHC to
generate

        the card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
        (data transactions like normal peers)
}

disable_mmc_high_speed_mode(void)
{
    send CMD6 with argument 0x2B90100;
    set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
    send CMD8 to get EXT_CSD value of MMC;
    check if HS_TIMING byte (byte number 185) is 0;
    if (HS_TIMING is not 0) report the function switch failed and return;
    change clock divisor value or configure the system clock feeding into eSDHC to
generate

        the card_clk of the desired value below 20MHz;
        (data transactions like normal peers)
}

```

15.6.4.3 Set MMC bus width

```

change_mmc_bus_width(void)
{
    send CMD9 to get CSD value of MMC;
    check if the value of SPEC_VER field is 4 or above;
    if (SPEC_VER value is less than 4) report the MMC does not support multiple bit
width

        and return;
    send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
    send CMD13 to wait card ready (busy line released);
    (data transactions like normal peers)
}

```

15.6.5 Commands for MMC/SD

See the table below for a list of commands for the MMC/SD cards.

See the corresponding specifications for details about the command information.

Four kinds of commands control the MMC, as follows:

1. Broadcast commands (bc)-no response
2. Broadcast commands with response (bcr)-response from all cards simultaneously
3. Addressed (point-to-point) commands (ac)-no data transfer on SDHC_DAT
4. Addressed (point-to-point) data transfer commands (adtc)

Table 15-35. Commands for MMC/SD

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMCs and SD memory cards in idle state to send their operation conditions register contents in the response on the SDHC_CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the SDHC_CMD line.
CMD3 ⁽¹⁾	ac	[31:6] RCA [15:0] stuff bits	R1	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD6 ⁽²⁾	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to SD Physical Specification version 2.0 for details.
CMD6 ⁽³⁾	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to the MultiMediaCard System Specification version 4.0 final draft 2 for details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address; address 0 deselects all.
CMD8	bcr	[31:12] reserved bits [11:8] supply voltage(VHS) [7:0] check pattern	R7	SEND_IF_COND	Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to '0'.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the SDHC_CMD line.

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the SDHC_CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card starting at the given address until STOP_TRANSMISSION is received.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host starting at the given address until the STOP_TRANSMISSION command is received.
CMD21-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until the STOP_TRANSMISSION command is received.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command should be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write-protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card-specific data (WP_GRP_SIZE).

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write-protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write-protection features, this command asks the card to send the status of the write-protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last write block of the continuous range to be erased.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_STAR	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command address a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43-51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128 Kbytes of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Access a multiple I/O register with a single command, it allows the reading or writing of a large number of I/O registers.

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹					
CMD54	Reserved									
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.					
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general-purpose or application-specific commands. The size of the data block is set by the SET_BLOCK_LEN command.					
CMD57-63	Reserved									
ACMDs should be preceded with the APP_CMD command										
(Commands listed below are for SD cards only. Other SD commands not listed below are not supported by this module)										
ACMD6	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width (00 = 1 bit or 10 = 4 bit bus) to be used for data transfer. The allowed data bus widths are given in DCR register.					
ACMD13	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD memory card status.					
ACMD18	adtc	[31:0] stuff bits	R1	SECURE_READ_MULTI_BLOCK	Protected Area Access Command: Reads continuously transfer data blocks from Protected Area of SD Memory Card. Refer to Security Specification Version 2.00 for more details.					
ACMD22	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written (without errors) sectors. Responds with 32 bit + CRC data block.					
ACMD23	ac	[31:23] stuff bits [22:0] number of blocks	R1	SET_WR_BLK_ERASE_COUNT	-					
ACMD25	adtc	[31:0] stuff bits	R1	SECURE_WRITE_MULTI_BLOCK	Protected Area Access Command: Writes continuously transfer data blocks to Protected Area of SD memory card. Refer to Security Specification Version 2.00 for more details.					
ACMD26	adtc	[31:0] stuff bits	R1	SECURE_WRITE_MKB	System Area Access Command: Overwrite the existing Media Key Block (MKB) on System Area of SD Memory Card with new MKB. This command is used in dynamic update MKB scheme. Refer to Security Specification Version 2.00 for more details.					
ACMD38	ac	[31:0] stuff bits	R1b	SECURE_ERASE	Protected Area Access Command:					

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
					Erase a specified region of the Protected Area of SD Memory Card. Refer to Security Specification Version 2.00 for more details.
ACMD41	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) content in the response on the SDHC_CMD line.
ACMD42	ac	-	R1	SET_CLR_CARD_DETECT	-
ACMD43	adtc	[31:24]Unit_Count: [23:16] MKB_ID: [15:0]Unit_Offset:	R1	GET_MKB	Reads Media Key Block from System Area of SD Memory Card. <ul style="list-style-type: none"> • Unit_Count specifies the Number of units to read. (Here, a unit = 512 bytes (fixed).) • MKB_ID specifies the application unique number. • Unit_Offset specifies the start address(offset) to read. Refer to Security Specification Version 2.00 for more details.
ACMD44	adtc	[31:0] stuff bits	R1	GET_MID	Reads Media ID from the System Area of SD Memory Card. Refer to Security Specification Version 2.00 for more details.
ACMD45	adtc	[31:0] stuff bits	R1	SET_CER_RN1	AKE Command: Writes random number RN1 as challenge1 in AKE process. Refer to Security Specification Version 2.00 for more details.
ACMD46	adtc	[31:0] stuff bits	R1	GET_CER_RN2	AKE Command: Reads random number RN2 as challenge2 in AKE process. Refer to Security Specification Version 2.00 for more details.
ACMD47	adtc	[31:0] stuff bits	R1	SET_CER_RES2	AKE Command: Writes RES2 as response2 to RN2 in AKE process. Refer to Security Specification Version 2.00 for more details.
ACMD48	adtc	[31:0] stuff bits	R1	GET_CER_RES1	AKE Command: Reads RES1 as response1 to RN1 in AKE process. Refer to Security Specification Version 2.00 for more details.
ACMD49	ac	[31:0] stuff bits	R1b	CHANGE_SECURE_AREA	Protected Area Access Command:

Table continues on the next page...

Table 15-35. Commands for MMC/SD (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description ¹
					Change size of Protected Area. Refer to Security Specification Version 2.00 for more details.
ACMD51	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR)

- Registers mentioned in this table are SD card registers.

NOTE

- CMD3 differs for MMC and SD cards. For MMC cards, CMD3 is referred to as SET_RELATIVE_ADDR and has a response type R1. For SD cards, CMD3 is referred to as SEND_RELATIVE_ADDR and has a response type R6, with RCA inside.
- CMD6 differs completely between high-speed MMC cards and high-speed SD cards. Command SWITCH_FUNC is used for high speed SD cards.
- Command SWITCH is for high-speed MMC cards. The index field can contain any value from 0-255, but only values 0-191 are valid. If the index value is in the 192-255 range, the card does not perform any modification and the status bit EXT_CSD[SWITCH_ERROR] is set.

Table 15-36. EXT_CSD access modes

Bits	Access name	Operation
00	Command set	The command set is changed according to the command set field of the argument
01	Set bits	The bits in the pointed byte are set, according to the set bits in the value field.
10	Clear bits	The bits in the pointed byte are cleared, according to the set bits in the value field.
11	Write byte	The value field is written into the pointed byte.

[other]

- CMD8 differs for MMC and SD cards. For SD cards, CMD8 is referred to as SEND_IF_COND. For MMC cards, CMD8 is referred to as SEND_EXT_CSD.

15.6.6 Software restrictions

When polling read or write, once the software begins a buffer read or write, it must access exactly the number of times as set in the watermark level register, as if a DMA burst occurred.

When the internal DMA is not enabled and a write transaction is in operation, DATPORT (described in [Data buffer access port \(eSDHC_DATPORT\)](#)) must not be read.

Chapter 16

Universal Serial Bus Interface

16.1 Overview

This chapter describes the universal serial bus (USB) interface of the chip.

The USB interface implements many industry standards. However, it is beyond the scope of this document to document the intricacies of these standards. Instead, it is left to the reader to refer to the governing specifications.

The following documents are available from the USB Implementers Forum web page at <http://www.usb.org/developers/docs/>.

- *Universal Serial Bus Revision 2.0 Specification*

The following documents are available from the Intel USB Specifications web page at <http://www.intel.com/technology/usb/spec.htm>.

- *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, Revision 1.0*

The following documents are available from the ULPI web page at <http://www.ulpi.org/>

- *UTMI+ Specification, Revision 1.0*
- *UTMI Low Pin-Count Interface (ULPI) Specification, Revision 1.0*

The device implements a dual-role (DR) USB module. This module may be connected to an external port. Collectively the module and external port are called the USB interface.

The USB DR module is a USB 2.0-compliant serial interface engine for implementing a USB interface. The registers and data structures are based on the *Enhanced Host Controller Interface Specification for Universal Serial Bus* (EHCI) from Intel Corporation. Each DR module is either a device or host controller.

The DR module supports the required signaling for transceivers (PHYs).

The module supports the required signaling for external UTMI low pin count interface (ULPI) transceivers (PHYs).

The USB interface is shown in the figure below.

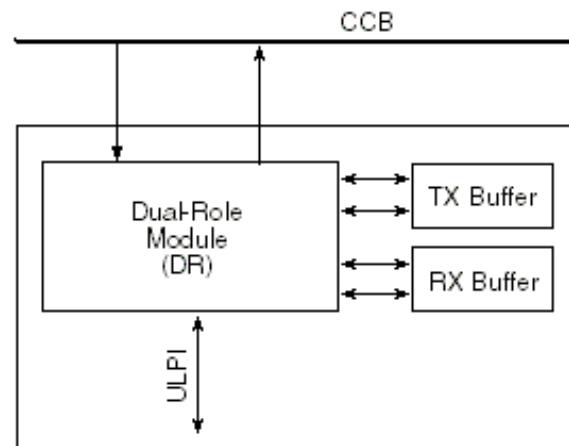


Figure 16-1. USB interface block diagram

16.1.1 USB features summary

The USBDR module includes the following features:

- Complies with *USB Specification Rev 2.0*
- Supports operation as a standalone USB host controller
 - Supports enhanced host controller interface (EHCI)
- Supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operation. Low speed is only supported in host mode
- Supports external PHY with ULPI (UTMI + low-pin interface)
- Supports operation as a standalone USB device
 - Supports one upstream facing port
 - Supports six bidirectional USB endpoints
- Host and device support

16.1.2 Modes of operation

The USB DR module operates in two modes:

- Host
- Device

NOTE

Only high-speed and full-speed operations are supported in device mode.

16.2 USB External Signals

This section contains detailed descriptions of all the USB controller signals. The table below describes the signals, indicating which interface supports each signal.

Table 16-1. USB External Signals

Signal	I/O	Description
USB_D0	I/O	ULPI-Use as USB_D0
USB_D1	I/O	ULPI-Use as USB_D1
USB_D2	I/O	ULPI-Use as USB_D2
USB_D3	I/O	ULPI-Use as USB_D3
USB_D4	I/O	ULPI-Use as USB_D4
USB_D5	I/O	ULPI-Use as USB_D5
USB_D6	I/O	ULPI-Use as USB_D6
USB_D7	I/O	ULPI-Use as USB_D7
USB_NXT	I	ULPI-Use as USB_NXT
USB_DIR	I	ULPI-Use as USB_DIR
USB_STP	O	ULPI-Use as USB_STP
USB_CLK	I	ULPI-Use as USB_CLK

16.2.1 ULPI

The ULPI (UTMI+ low pin interface) is a reduced pin-count (12 signals) extension of the UTMI+ specification.

Pin count is reduced by converting relatively static signals to register bits, and providing a bidirectional, generic data bus that carries USB and register data. This interface minimizes pin count requirements for external PHYs. The table below describes the signals for the ULPI interface.

Table 16-2. ULPI signal descriptions

Signal	I/O	Description	
USBn_DIR	I	Direction. USBn_DIR controls the direction of the data bus. When the PHY has data to transfer to USB port, it drives USBn_DIR high to take ownership of the bus. When the PHY has no data to transfer it drives USBn_DIR low and monitors the bus for link activity. The PHY pulls USBn_DIR high whenever the interface cannot accept data from the link.	
		State Meaning	Asserted-PHY has data to transfer to the link. Negated-PHY has no data to transfer.
		Timing	Synchronous to PHY_CLK.
USBn_NXT	I	Next data. The PHY asserts USBn_NXT to throttle the data. When USB port is sending data to the PHY, USBn_NXT indicates when the current byte has been accepted by the PHY. The USB port places the next byte on the data bus in the following clock cycle. When the PHY is sending data to USB port, USBn_NXT indicates when a new byte is available for USB port to consume.	
		State Meaning	Asserted-PHY is ready to transfer byte. Negated-PHY is not ready.
		Timing	Synchronous to PHY_CLK.
USBn_STP	O	Stop. USBn_STP indicates the end of a transfer on the bus.	
		State Meaning	Asserted-USB asserts this signal for 1 clock cycle to stop the data stream currently on the bus. If USB port is sending data to the PHY, USBn_STP indicates the last byte of data was previously on the bus. If the PHY is sending data to USB port, USBn_STP forces the PHY to end its transfer, negate USBn_DIR and relinquish control of the data bus to the USB port. Negated-Indicates normal operation.
		Timing	Synchronous to PHY_CLK.
USBn_D[7:0]	I/O	Data bit n . USBn_D n is bit n of the 8-bit (USBn_D7-USBn_D0), uni-directional data bus used to carry USB, register, and interrupt data between the PHY and the USB controller.	
		State Meaning	Asserted-Data bit n is 1. Negated-Data bit n is 0.
		Timing	Synchronous to PHY_CLK.

16.2.2 PHY clocks

The USBnCLK input provides the clocking signal for the ULPI PHY interface .

NOTE

A write to registers in the USB controller memory map may cause the system to hang if PORTSC[PHCD]=0 when no USB PHY clock is applied.

16.3 USB memory map/register definition

This section provides the memory map and detailed descriptions of all USB interface registers.

The following sections provide details about the registers in the USB memory map.

NOTE

Memory may be viewed from either a big-endian or little-endian byte ordering perspective depending on the processor configuration. In big-endian mode, the most-significant byte of word 0 is located at address 0 and the least-significant byte of word 0 is located at address 3. In little-endian mode, the least-significant byte of word 0 is located at address 0 and the most-significant byte of word 0 is located at address 3. Within registers, bits are numbered within a word starting with bit 31 as the most-significant bit. By convention, USB registers use little-endian byte ordering. In the USB DR module, these are the registers from offsets 0x00 to 0x1FF. The registers associated with the internal system interface (0x400 and above) use big-endian byte ordering.

USB memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Identification register (USB_USB_ID)	32	R	See section	16.3.1/897
80	General purpose timer load n (USB_GPTIMER0LD)	32	R/W	0000_0000h	16.3.2/898
84	General purpose timer control n (USB_GPTIMER0CTRL)	32	R/W	0000_0000h	16.3.3/899
88	General purpose timer load n (USB_GPTIMER1LD)	32	R/W	0000_0000h	16.3.2/898
8C	General purpose timer control n (USB_GPTIMER1CTRL)	32	R/W	0000_0000h	16.3.3/899
100	Capability Register Length (USB_CAPLENGTH)	8	R	40h	16.3.4/900
102	Host Controller Interface Version Number (USB_HCIVERSION)	16	R	0100h	16.3.5/900
104	Host Controller Structural Parameters (USB_HCSPARAMS)	32	R	0001_0011h	16.3.6/901
108	Host Controller Capability Parameters (USB_HCCPARAMS)	32	R	0000_0006h	16.3.7/903
120	Device Controller Interface Version Number (USB_DCIVERSION)	16	R	0001h	16.3.8/905
124	Device Controller Capability Parameters (USB_DCCPARAMS)	32	R	0000_0186h	16.3.9/906

Table continues on the next page...

USB memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
140	USB Command (USB_USBCMD)	32	R/W	0008_0000h	16.3.10/ 906
144	USB Status (USB_USBSTS)	32	R/W	0000_0000h	16.3.11/ 911
148	USB Interrupt Enable (USB_USBINTR)	32	R/W	0000_0000h	16.3.12/ 915
14C	USB Frame Index (USB_FRINDEX)	32	R/W	0000_0000h	16.3.13/ 917
154	Periodic Frame List Base Address [host mode] (USB_PERIODICLISTBASE)	32	R/W	0000_0000h	16.3.14/ 918
154	USB Device Address [device mode] (USB_DEVICEADDR)	32	R/W	0000_0000h	16.3.15/ 919
158	Next Asynchronous List Addr [host mode] (USB_ASYNCLISTADDR)	32	R/W	0000_0000h	16.3.16/ 919
158	Address at Endpoint List [device mode] (USB_ENDPOINTLISTADDR)	32	R/W	0000_0000h	16.3.17/ 920
160	Master Interface Data Burst Size (USB_BURSTSIZE)	32	R	0000_1010h	16.3.18/ 920
164	Transmit FIFO Tuning Controls (USB_TXFILLTUNING)	32	R	See section	16.3.19/ 921
170	ULPI Register Access (USB_ULPI_VIEWPORT)	32	R/W	0000_0000h	16.3.20/ 923
178	Endpoint NAK Indication Register (USB_ENDPTNAK)	32	R/W	0000_0000h	16.3.21/ 926
17C	Endpoint NAK Indication Enable Register (USB_ENDPTNAKEN)	32	R/W	0000_0000h	16.3.22/ 926
180	Configured Flag Register (USB_CONFIGFLAG)	32	R	0000_0001h	16.3.23/ 927
184	Port Status/Control (USB_PORTSC)	32	R/W	See section	16.3.24/ 928
1A8	USB Device Mode (USB_USBMODE)	32	R/W	0000_5000h	16.3.25/ 935
1AC	Endpoint Setup Status (USB_ENDPTSETUPSTAT)	32	w1c	0000_0000h	16.3.26/ 936
1B0	Endpoint Initialization (USB_ENDPOINTPRIME)	32	R/W	0000_0000h	16.3.27/ 937
1B4	Endpoint Flush (USB_ENDPTFLUSH)	32	R/W	0000_0000h	16.3.28/ 938
1B8	Endpoint Status (USB_ENDPTSTATUS)	32	R	0000_0000h	16.3.29/ 938
1BC	Endpoint Complete (USB_ENDPTCOMPLETE)	32	w1c	0000_0000h	16.3.30/ 939
1C0	Endpoint Control 0 (USB_ENDPTCTRL0)	32	R/W	0080_0080h	16.3.31/ 941

Table continues on the next page...

USB memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C4	Endpoint Control n (USB_ENDPTCTRL1)	32	R/W	0000_0000h	16.3.32/ 943
1C8	Endpoint Control n (USB_ENDPTCTRL2)	32	R/W	0000_0000h	16.3.32/ 943
1CC	Endpoint Control n (USB_ENDPTCTRL3)	32	R/W	0000_0000h	16.3.32/ 943
1D0	Endpoint Control n (USB_ENDPTCTRL4)	32	R/W	0000_0000h	16.3.32/ 943
1D4	Endpoint Control n (USB_ENDPTCTRL5)	32	R/W	0000_0000h	16.3.32/ 943
400	Snoop n (USB_SNOOP1)	32	R/W	0000_0000h	16.3.33/ 945
404	Snoop n (USB_SNOOP2)	32	R/W	0000_0000h	16.3.33/ 945
408	Age Count Threshold (USB_AGE_CNT_THRESH)	32	R/W	0000_0000h	16.3.34/ 946
40C	Priority Control (USB_PRI_CTRL)	32	R/W	0000_0000h	16.3.35/ 947
410	System Interface Control (USB_SI_CTRL)	32	R/W	0000_0000h	16.3.36/ 948
500	Control (USB_CONTROL)	32	R/W	0000_0000h	16.3.37/ 949

16.3.1 Identification register (USB_USB_ID)

The identification register is used to declare the slave interface presence and include table of the hardware configuration parameters. It is not defined by the EHCI specification.

The ID register provides a simple way to determine if the USB DR module is provided in the system. The ID register identifies the USB DR and its revision (0000_0000) .

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				VERSION				REVISION				TAG			
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				NID				Reserved		ID					
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

USB_USB_ID field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–25 VERSION	Identifies the version of the module.
24–21 REVISION	Identifies the revision number of the module.
20–16 TAG	Identifies the tag of the USB core.
15–14 -	This field is reserved. Reserved
13–8 NID	Ones complement version of ID bit.
7–6 -	This field is reserved. Reserved
ID	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB_DR.

16.3.2 General purpose timer load n (USB_GPTIMERnLD)

Host/device controller drivers can measure time related activities using these timer registers. These registers are not part of standard EHCI controller.

Address: 0h base + 80h offset + (8d × i), where i=0d to 1d

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

Reset 0

USB_GPTIMERnLD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value . This field is the value loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus one for the timer duration. Note: Maximum value is 0xFFFFFFF or 16.777215 seconds.

16.3.3 General purpose timer control n (USB_GPTIMERnCTRL)

Host/device controller drivers can measure time related activities using these timer registers. These registers are not part of standard EHCI controller.

Address: 0h base + 84h offset + (8d × i), where i=0d to 1d

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

USB_GPTIMERnCTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run . This bit enables the general purpose timer to run. Setting or clearing this bit does not have any effect on GPTCNT value. 0 Timer Stop 1 Timer Run
30 GPTRST	General Purpose Timer Reset . Writing 1 to this bit reloads GPTCNT with value in GPTLD. 0 No Action 1 Reload Counter
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode . This bit selects between single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt and stops until software resets the counter. In repeat mode, the timer counts down to zero, generates an interrupt and automatically reloads the counter to begin again.

Table continues on the next page...

USB_GPTIMERnCTRL field descriptions (continued)

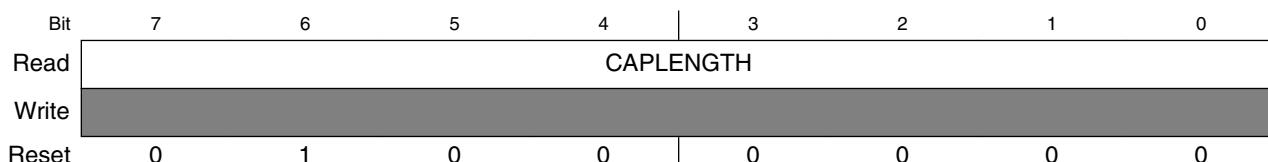
Field	Description
0	One-shot
1	Repeat
GPTCNT	General Purpose Timer Counter . Contains the value of running counter.

16.3.4 Capability Register Length (USB_CAPLENGTH)

The capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation. Most of these registers are defined by the EHCI specification. Registers that are not defined by the EHCI specification are noted in their descriptions.

CAPLENGTH is used as an offset to add to the register base address to find the beginning of the operational register space, that is, the location of the USBCMD register.

Address: 0h base + 100h offset = 100h



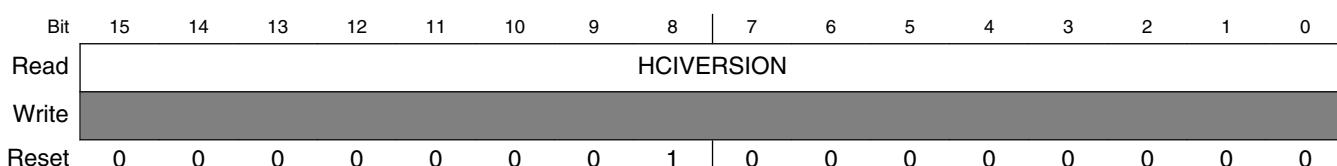
USB_CAPLENGTH field descriptions

Field	Description
CAPLENGTH	Capability registers length. It indicates which offset to add to the register base address at the beginning of the operational register. Value is 0x40.

16.3.5 Host Controller Interface Version Number (USB_HCIVERSION)

HCIVERSION contains a BCD encoding of the EHCI revision number supported by this host controller. The most-significant byte of the register represents major revision and the least-significant byte is minor revision.

Address: 0h base + 102h offset = 102h



USB_HCIVERSION field descriptions

Field	Description
HCIVERSION	EHCI revision number. Value is 0x0100 indicating version 1.0.

16.3.6 Host Controller Structural Parameters (USB_HCSPARAMS)

HCSPARAMS contains structural parameters such as the number of downstream ports.

Address: 0h base + 104h offset = 104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				N_TT				N_PTT				Reserved			PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				Reserved			PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

USB_HCSPARAMS field descriptions

Field	Description
31–28 -	This field is reserved. Reserved, should be cleared.
27–24 N_TT	Number of transaction translators. This is a non-EHCI field. This field indicates the number of embedded transaction translators associated with the module. Always 1. See Embedded transaction translator function .
23–20 N_PTT	Ports per transaction translator. This is a non-EHCI field. The number of ports assigned to each transaction translator. This is equal to N_PORTS.
19–17 -	This field is reserved. Reserved, should be cleared.
16 PI	Port indicators. Indicates whether the ports support port indicator control. Always 1. 1 The port status and control registers include a R/W field for controlling the state of the port indicator.
15–12 N_CC	Number of companion controllers associated with the DR controller. Always 0.
11–8 N_PCC	Number ports per CC. This field indicates the number of ports supported per internal companion controller. Always 0.
7–5 -	This field is reserved. Reserved, should be cleared.
4 PPC	Power port control. Indicates whether the host controller supports port power control. ULPI Mode: 0 USBDR will write 0 for DRVBUS bit of OTG Control register in PHY.

Table continues on the next page...

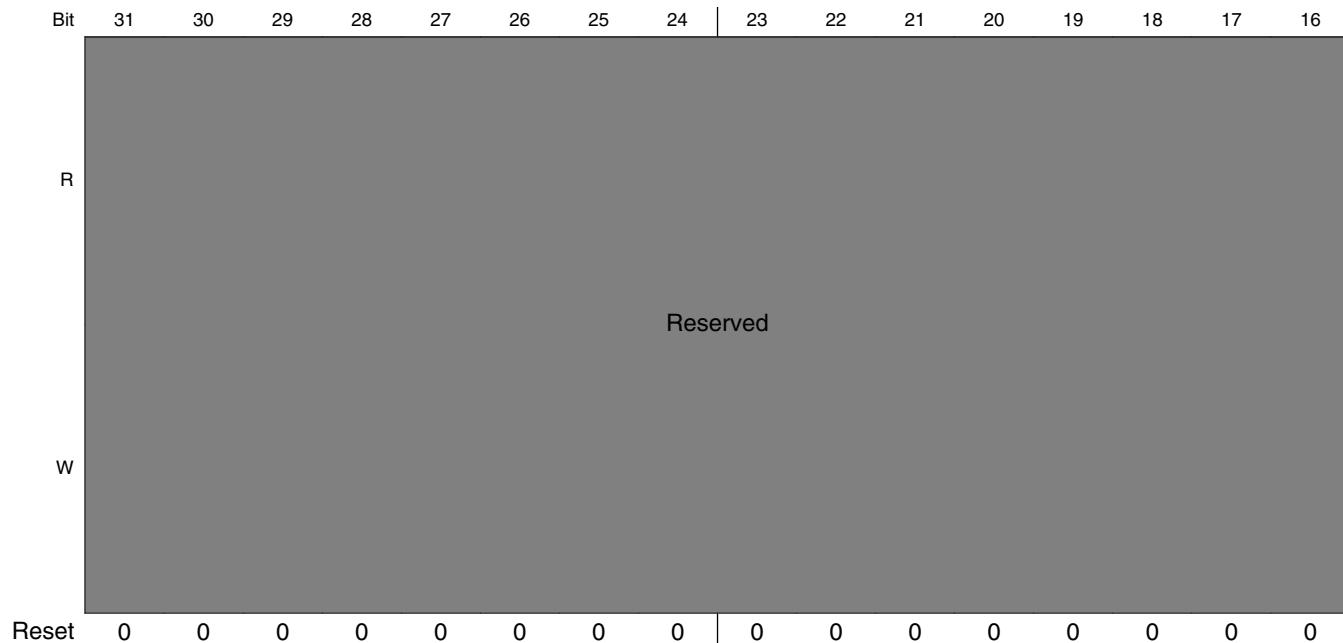
USB_HCSPARAMS field descriptions (continued)

Field	Description
	1 USBDR will write 1 for DRVVBUS bit of OTG Control register in PHY. The OTG control register is defined in ULPI specification.
N_PORTS	Number of ports. Number of physical downstream ports implemented for host applications. The value of this field determines how many port registers are addressable in the operational register. Always 1.

16.3.7 Host Controller Capability Parameters (USB_HCCPARAMS)

HCCPARAMS identifies multiple mode control (time-base bit functionality) addressing capability.

Address: 0h base + 108h offset = 108h



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

USB_HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved, should be cleared.
15–8 EECP	EHCI extended capabilities pointer. Indicates the existence of a capabilities list. A value of 0x00 indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device. This field is always 0.
7–4 IST	Isochronous scheduling threshold. Indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit 7 is zero, the value of the least significant 3 bits indicates the number of microframes a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit 7 is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field is always 0.
3 -	This field is reserved. Reserved, should be cleared.
2 ASP	Asynchronous schedule park capability. Indicates whether the USB DR module supports the park feature for high-speed queue heads in the asynchronous schedule. The feature can be disabled or enabled and set to a specific level by using the <i>asynchronous schedule park mode enable</i> and <i>asynchronous schedule park mode count</i> fields in the USBCMD register.

Table continues on the next page...

USB_HCCPARAMS field descriptions (continued)

Field	Description
	This field is always 1 (park feature supported).
1 PFL	Programmable frame list flag. Indicates whether system software can specify and use a frame list length less than 1024 elements. Frame list size is configured via the USBCMD register frame list size field. The frame list must always be aligned on a 4-K page boundary. This requirement ensures that the frame list is always physically contiguous. This field is always 1.
0 ADC	64-bit addressing capability. Always 0; 64-bit addressing is not supported. 0 Data structures use 32-bit address memory pointers

16.3.8 Device Controller Interface Version Number (USB_DCIVERSION)

This register is not defined in the EHCI specification. DCIVERSION is a two-byte register containing a BCD encoding of the device controller interface. The most-significant byte of the register represents major revision and the least-significant byte is minor revision.

Address: 0h base + 120h offset = 120h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read									DCIVERSION							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USB_DCIVERSION field descriptions

Field	Description
DCIVERSION	Device interface revision number.

16.3.9 Device Controller Capability Parameters (USB_DCCPARAMS)

This register is not defined in the EHCI specification. This register describes the overall host/device capability of the DR module.

Address: 0h base + 124h offset = 124h

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

USB_DCCPARAMS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved, should be cleared.
8 HC	Host capable. Always 1, indicating the USB DR controller can operate as an EHCI compatible USB 2.0 host
7 DC	Device capable. Always 1, indicating the USB DR controller can operate as an USB 2.0 device. 1 Device capability. 0 No device capability (host only).
6–5 -	This field is reserved. Reserved, should be cleared.
DEN	Device endpoint number. Indicates the number of endpoints built into the device controller. Always 0x 6 .

16.3.10 USB Command (USB_USBCMD)

The operational registers are comprised of dynamic control or status registers that may be read-only, read/write, or read/write-1-to-clear. The following registers define the operational registers.

The module executes the command indicated in this register.

Address: 0h base + 140h offset = 140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
FS2	ATDTW	SUTW	Reserved	ASPE	Reserved	ASP		LR		IAA	ASE	PSE	FS	RST	RS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, should be cleared.
23–16 ITC	Interrupt threshold control. The system software uses this field to set the maximum rate at which the USB DR module issues interrupts. ITC contains the maximum interrupt interval measured in microframes. Valid values are shown below. 0x00 Immediate (no threshold)

Table continues on the next page...

USB_USBCMD field descriptions (continued)

Field	Description
	0x01 1 microframe 0x02 2 microframes 0x04 4 microframes 0x08 8microframes 0x10 16 microframes 0x20 32 microframes 0x40 40 microframes
15 FS2	See bits 3-2 below. This is a non-EHCI bit.
14 ATDTW	Add dTD TripWire. This is a non-EHCI bit. Used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software. This bit shall also be cleared by hardware when its state machine is in hazard region where adding a dTD to a primed endpoint may go unrecognized. More information on the use of this bit is described in Device operational model .
13 SUTW	Setup tripwire. This is a non-EHCI bit. Used as a semaphore when the 8 bytes of setup data read extracted from a QH by the DCD. If the setup lockout mode is off (See USB Device Mode (USB_USBMODE)) then there exists a hazard when new setup data arrives and the DCD is copying setup from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. More information on the use of this bit is described in Device operational model .
12 -	This field is reserved. Reserved, should be cleared.
11 ASPE	Asynchronous schedule park mode enable. This bit defaults to a 1 and is R/W. Software uses this bit to enable or disable park mode. 0 Disabled 1 Enabled
10 -	This field is reserved. Reserved, should be cleared.
9–8 ASP	Asynchronous schedule park mode count. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1H to 0x3H. Software must not write a zero to this field when ASPE is set as this results in undefined behavior.
7 LR	Light host/device controller reset (OPTIONAL). Not implemented. Always 0.
6 IAA	Interrupt on async advance doorbell. Used as a doorbell by software to tell the USB DR controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the controller has evicted all appropriate cached schedule states, it sets USBSTS[AAI]. If USBINTR[AAE] is set, the host controller asserts an interrupt at the next interrupt threshold. The controller clears this bit after it has set USBSTS[AAI]. Software should not set this bit when the asynchronous schedule is inactive. Doing so yields undefined results. This bit is only used in host mode. Setting this bit when the USB DR module is in device mode is selected results in undefined results.
5 ASE	Asynchronous schedule enable. Controls whether the controller skips processing the asynchronous schedule. Only used in host mode. 0 Do not process the asynchronous schedule 1 Use the ASYNCLISTADDR register to access the asynchronous schedule.

Table continues on the next page...

USB_USBCMD field descriptions (continued)

Field	Description
4 PSE	Periodic schedule enable. Controls whether the controller skips processing the periodic schedule. Only used in host mode. 0 Do not process the periodic schedule. 1 Use the PERIODICLISTBASE register to access the periodic schedule.
3–2 FS	Frame list size. Together with bit 15 these bits make the FS[2:0] field. This field is read/write only if programmable frame list flag in the HCCPARAMS registers is set to 1. This field specifies the size of the frame list that controls which bits in FRINDEX should be used for the frame list current index. Only used in host mode. Note that values below 256 elements are not defined in the EHCI specification. 000 1024 elements (4096 bytes) 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
1 RST	Controller reset. Software uses this bit to reset the controller. This bit is cleared by the controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. <ul style="list-style-type: none">• Host Mode: When software sets this bit, the host controller resets its internal pipelines, timers, counters, state machines, and so on to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit when USBSTS[HCH] is a zero. Attempting to reset an actively running host controller results in undefined behavior.• Device Mode: When software writes a one to this bit, the USB_DR resets its internal pipelines, timers, counters, state machines, and so on to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a controller reset, all primed endpoints should be flushed and the USBCMD[RS] bit should be set to 0. <p>NOTE:</p> <ul style="list-style-type: none">• Once set, this bit is high for 63 clock cycles of system clock and then it transitions back to 0. Software lets the previous reset complete before setting this bit again. On setting this bit, the USBDR also resets the PHY. Software ensures that PHY also comes out reset before setting this bit again.
0 RS	Run/Stop. Host mode: <ul style="list-style-type: none">• When this bit is set, the controller proceeds with the execution of the schedule. The controller continues execution as long as this bit is set. When this bit is set to 0, the host controller completes the current transaction on the USB and then halts. The USBSTS[HCH] bit indicates when the USB DR controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the halted state (that is, USBSTS[HCH] is a one). Device mode: <ul style="list-style-type: none">• Setting this bit causes the USB DR controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up is disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Clearing this bit causes a detach event.

Table continues on the next page...

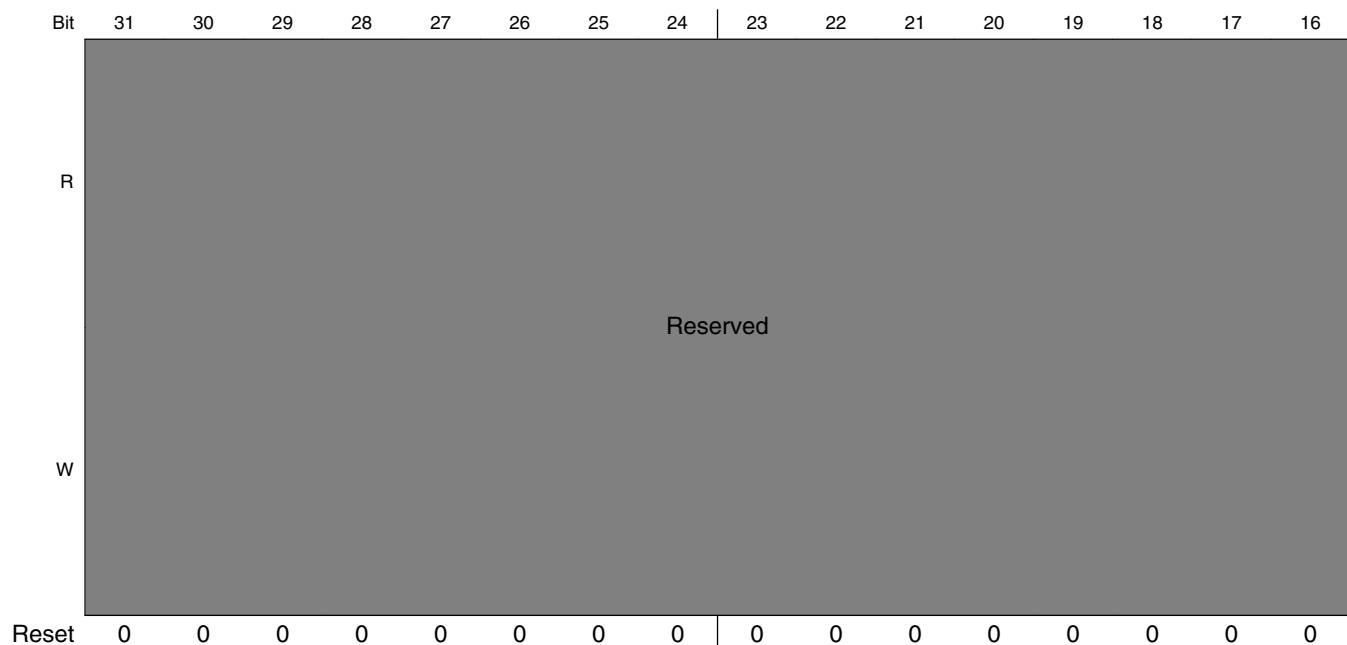
USB_USBCMD field descriptions (continued)

Field	Description
0	Stop
1	Run

16.3.11 USB Status (USB_USBSTS)

The USB status register indicates various states of the USB DR module and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them (indicated by a w1c in the bit's W cell).

Address: 0h base + 144h offset = 144h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	Reserved	ULPII	Reserved	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	
W	0	0	0	0	0	0	0	w1c								

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

USB_USBSTS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved, should be cleared.
15 AS	Asynchronous schedule status. Reports the current real status of the asynchronous schedule. The USB DR controller is not required to immediately disable or enable the asynchronous schedule when software transitions USBCMD[ASE]. When this bit and USBCMD[ASE] have the same value, the asynchronous schedule is either enabled (1) or disabled (0). Only used in host mode. 0 Disabled 1 Enabled
14 PS	Periodic schedule status. Reports the current real status of the periodic schedule. The USB DR controller is not required to immediately disable or enable the periodic schedule when software transitions USBCMD[PSE]. When this bit and USBCMD[PSE] have the same value, the periodic schedule is either enabled (1) or disabled (0). Only used in host mode. 0 Disabled 1 Enabled
13 RCL	Reclamation. Used to detect an empty asynchronous schedule. Only used by the host mode. 0 Non-empty asynchronous schedule 1 Empty asynchronous schedule

Table continues on the next page...

USB_USBSTS field descriptions (continued)

Field	Description
12 HCH	HC halted. This bit is a zero whenever USBCMD[RS] is a one. The USB DR controller sets this bit to one after it has stopped executing because of USBCMD[RS] being cleared, either by software or by the host controller hardware (for example, internal error). Only used in host mode. 0 Running 1 Halted
11 -	This field is reserved. Reserved, should be cleared.
10 ULPII	ULPI interrupt. An event completion to the viewport register sets this bit. If the ULPI enables the USBINTR[ULPIE] to be set, the USB interrupt (UI) occurs.
9 -	This field is reserved. Reserved, should be cleared.
8 SLI	DC Suspend. This is a non-EHCI bit. When a device controller enters a suspend state from an active state, this bit is set. This bit is only cleared by software by writing a 1 to it. Only used by the device controller. 0 Active 1 Suspended
7 SRI	Host mode: <ul style="list-style-type: none"> This is a non-EHCI status bit. In host mode, this bit is set every 125 µs, provided the PHY clock is present and running (for example, the port is NOT suspended), and can be used by the host controller driver as a time base. Device mode: <ul style="list-style-type: none"> SOF received. When the USB DR controller detects a Start Of (Micro)Frame, this bit is set. When a SOF is extremely late, the DR controller automatically sets this bit to indicate that an SOF was expected. Therefore, this bit is set roughly every 1- ms in device FS mode and every 125 µs in HS mode and is synchronized to the actual SOF that is received. Because the controller is initialized to FS before connect, this bit is set at an interval of 1 ms during the prelude to the connect and chirp. Software writes a 1 to this bit to clear it.
6 URI	USB reset received. This is a non-EHCI bit. When the USB DR controller detects a USB reset and enters the default state, this bit is set. Software can write a 1 to this bit to clear the USB reset received status bit. Only used by the device mode. 0 No reset received 1 Reset received
5 AAI	Interrupt on async advance. System software can force the controller to issue an interrupt the next time the USB DR controller advances the asynchronous schedule by writing a one to USBCMD[IAA]. This status bit indicates the assertion of that interrupt source. Only used by the host mode. 0 No async advance interrupt 1 Async advance interrupt
4 SEI	System error. This bit is set whenever an error is detected on the system bus. If USBINTR[SEE] is set, an interrupt is generated. The interrupt and status bits remain asserted until cleared by writing a 1 to this bit. Additionally, when in host mode, USBCMD[RS] is cleared, effectively disabling the USB DR controller. For the USB DR controller in device mode, an interrupt is generated, but no other action is taken. 0 Normal operation 1 Error

Table continues on the next page...

USB_USBSTS field descriptions (continued)

Field	Description
3 FRI	Frame list rollover. The controller sets this bit to a one when the frame list index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in USBCMD[FS]) is 1024, FRINDEX rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the USB DR controller sets this bit to a one every time FRINDEX [12] toggles. Only used by the host mode.
2 PCI	<p>Port change detect.</p> <p>Host mode:</p> <ul style="list-style-type: none"> The controller sets this bit when a connect status occurs on any port, a port enable/disable change occurs, an over current change occurs, or PORTSC[FPR] is set as the result of a J-K transition on the suspended port. <p>Device mode:</p> <ul style="list-style-type: none"> The USB DR controller sets this bit when it enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to reset or suspend events, the notification mechanisms are USBSTS[URI] and USBSTS[SLI], respectively. <p>The controller also sets this bit when the VBUS de-asserts (occurs when the host disables port power or the device has been disconnected from the bus). This bit is not EHCI compatible.</p>
1 UEI	USB error interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the controller. This bit is set along with the UI, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1 in EHCI for a complete list of host error interrupt conditions. Also see Table 16-113 in this chapter for more information on device error matrix. For the USB DR controller in device mode, only resume signaling is detected, all others are ignored.
	<p>0 No error</p> <p>1 Error detected</p>
0 UI	USB interrupt (USBINT). This bit is set by the controller when the cause of an interrupt is a completion of a USB transaction where the transfer descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

16.3.12 USB Interrupt Enable (USB_USBINTR)

The interrupts to software are enabled with the USB interrupt enable register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

Address: 0h base + 148h offset = 148h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
R	Reserved								Reserved								
W	ULPIE								SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

USB_USBINTR field descriptions

Field	Description
31–11 -	This field is reserved. Reserved, should be cleared.
10 ULPIE	ULPI interrupt enable. An event completion to the viewport register sets the USBSTS[ULPII]. If the ULPI enables ULPIE bit to be set, then the USBINT (USBSTS[UI]) occurs. 0 Disable 1 Enable
9 -	This field is reserved. Reserved, should be cleared.
8 SLE	Sleep enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SLI] transitions, the USB DR controller issues an interrupt. The interrupt is acknowledged by software writing a one to USBSTS[SLI]. Only used in device mode.

Table continues on the next page...

USB_USBINTR field descriptions (continued)

Field	Description
	0 Disable 1 Enable
7 SRE	SOF received enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SRI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[SRI]. 0 Disable 1 Enable
6 URE	USB reset enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[URI] is a one, the device controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[URI] bit. Only used in device mode. 0 Disable 1 Enable
5 AAE	Interrupt on async advance enable. When this bit is a one, and USBSTS[AAI] is a one, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[AAI]. Only used in host mode. 0 Disable 1 Enable
4 SEE	System error enable. When this bit is a one, and USBSTS[SEI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[SEI]. 0 Disable 1 Enable
3 FRE	Frame list rollover enable. When this bit is a one, and USBSTS[FRI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[FRI]. Only used by the host mode. 0 Disable 1 Enable
2 PCE	Port change detect enable. When this bit is a one, and USBSTS[PCI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[PCI]. 0 Disable 1 Enable
1 UEE	USB error interrupt enable. When this bit is a one, and USBSTS[UEI] is a one, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UEI]. 0 Disable 1 Enable
0 UE	USB interrupt enable. When this bit is a one, and USBSTS[UI] is a one, the DR controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UI]. 0 Disable 1 Enable

16.3.13 USB Frame Index (USB_FRINDEX)

In host mode, the frame index register is used by the controller to index the periodic frame list. The register updates every 125 microseconds (once each microframe). Bits N-3 are used to select a particular entry in the periodic frame list during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in USBCMD[FS].

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the USB DR controller is in the Halted state as indicated by the USBSTS[HCH]. A write to this register while USBCMD[RS] is set produces undefined results. Writes to this register also affect the SOF value.

In device mode, this register is read-only and the USB DR controller updates the FRINDEX[13-3] register from the frame number indicated by the SOF marker.

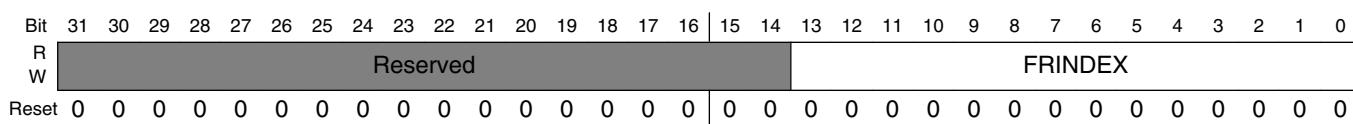
Whenever a SOF is received by the USB bus, FRINDEX[13-3] is checked against the SOF marker. If FRINDEX[13-3] is different from the SOF marker, FRINDEX[13-3] is set to the SOF value and FRINDEX[2-0] is cleared (that is, SOF for 1 ms frame). If FRINDEX[13-3] is equal to the SOF value, FRINDEX[2-0] is incremented (that is, SOF for 125- μ sec microframe).

The table below illustrates values of N based on the value of the Frame List Size in the USBCMD register, when used in host mode.

Table 16-20. FRINDEX N Values

USBCMD[FS]	Frame List Size	FRINDEX N value
000	1024 elements (4096 bytes)	12
001	512 elements (2048 bytes)	11
010	256 elements (1024 bytes)	10
011	128 elements (512 bytes)	9
100	64 elements (256 bytes)	8
101	32 elements (128 bytes)	7
110	16 elements (64 bytes)	6
111	8 elements (32 bytes)	5

Address: 0h base + 14Ch offset = 14Ch



USB FRINDEX field descriptions

Field	Description
31–14 - -	This field is reserved. Reserved, should be cleared.
FRINDEX	Frame index. The value in this register increments at the end of each time frame (for example, microframe). Bits N-3 are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or microframes) before moving to the next index. In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2-0 indicate the current microframe.

16.3.14 Periodic Frame List Base Address [host mode] (USB_PERIODICLISTBASE)

This register contains the beginning address of the periodic frame list in the system memory. The host controller driver loads this register prior to starting the schedule execution by the controller. The memory structure referenced by this physical memory pointer is assumed to be 4 KB aligned. The contents of this register are combined with the frame index register (FRINDEX) to enable the controller to step through the periodic frame list in sequence.

This register is shared between the host and device mode functions. In host mode, it is the PERIODICLISTBASE register; in device mode, it is the DEVICEADDR register. See [USB Device Address \[device mode\] \(USB_DEVICEADDR\)](#), for more information.

Address: 0h base + 154h offset = 154h

USB_PERIODICLISTBASE field descriptions

Field	Description
31-12 PERBASE	Base address. Correspond to memory address signal [31:12]. Only used in the host mode.
-	This field is reserved. Reserved, should be cleared.

16.3.15 USB Device Address [device mode] (USB_DEVICEADDR)

The device address register is not defined in the EHCI specification. In device mode, the upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software reprograms the address after receiving a SET_ADDRESS descriptor.

This register is shared between the host and device mode functions. In device mode, it is the DEVICEADDR register; in host mode, it is the PERIODICLISTBASE register. See [Periodic Frame List Base Address \[host mode\] \(USB_PERIODICLISTBASE\)](#), for more information.

Address: 0h base + 154h offset = 154h

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

USB_DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device address. This field corresponds to the USB device address.
-	This field is reserved. Reserved, should be cleared.

16.3.16 Next Asynchronous List Addr [host mode] (USB_ASYNCLISTADDR)

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits 4-0 of this register cannot be modified by the system software and always return zeros when read.

Note that this register is shared between the host and device mode functions. In host mode, it is the ASYNCLISTADDR register; in device mode, it is the ENDPOINTLISTADDR register. See [Address at Endpoint List \[device mode\] \(USB_ENDPOINTLISTADDR\)](#), for more information.

Address: 0h base + 158h offset = 158h

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

USB ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link pointer low (LPL). These bits correspond to memory address signals [31:5]. This field may only reference a queue head (QH). Only used by the host controller.
-	This field is reserved. Reserved, should be cleared.

16.3.17 Address at Endpoint List [device mode] (USB_ENDPOINTLISTADDR)

The endpoint list address register is not defined in the EHCI specification. In device mode, this register contains the address of the top of the endpoint list in system memory. Bits 10-0 of this register cannot be modified by the system software and always return zeros when read. The memory structure referenced by this physical memory pointer is assumed to be 64-bytes. The queue head is actually a 48-byte structure, but must be aligned on 64-byte boundary. However, the ENDPOINTLISTADDR[EPBASE] has a granularity of 2 KB, so in practice the queue head should be 2 KB aligned.

This register is shared between the host and device mode functions. In device mode, it is the ENDPOINTLISTADDR register; in host mode, it is the ASYNCLISTADDR register. See [Next Asynchronous List Addr \[host mode\] \(USB_ASYNCLISTADDR\)](#), for more information.

Address: 0h base + 158h offset = 158h

USB_ENDPOINTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint list address. Address of the top of the endpoint list.
-	This field is reserved. Reserved, should be cleared.

16.3.18 Master Interface Data Burst Size (USB_BURSTSIZE)

The master interface data burst size register is not defined in the EHCI specification. This register is used to control and dynamically change the burst size used during data movement on the initiator (master) interface.

Note that , on P4080, Rev 2, BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] are effectively write-only fields . The actual value written in the register is used correctly for the burst length, but the fields always return 10h. On P4080, Rev 3, BURSTSIZE returns the actual value when read.

Address: 0h base + 160h offset = 160h

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

USB_BURSTSIZE field descriptions

Field	Description
31–16 -	This field is reserved. Reserved, should be cleared.
15–8 TXPBURST	Programable TX burst length. This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus. Must not be set to greater than 16.
RXPBURST	Programable RX burst length. This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory. Must not be set to greater than 16.

16.3.19 Transmit FIFO Tuning Controls (USB_TXFILLTUNING)

The transmit FIFO tuning controls register is not defined in the EHCI specification. This register is used to control and dynamically change the burst size used during data movement on device DMA transfers. It is only used in host mode.

The fields in this register control performance tuning associated with how the USB DR module posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_s = Total Packet Flight Time (send-only) packet ($T_s = T_0 + T_1$)

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_p = Total Packet Time (fetch and send) packet ($T_p = T_{ff} + T_s$)

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at any time during the pre-fill operation the time remaining the [micro]frame is $< T_s$ then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the module eventually recovers, a mark is made in the scheduler health counter to note the occurrence of a back-off event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (T_{ff}) parameter described below.

NOTE

This register is read only and its default value is 0x0000_0000.
The register reads 0x0002_0000 when read in host mode.

Address: 0h base + 164h offset = 164h

USB TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved.
	Reserved, should be cleared.
21–16 TXFIFOTHRES	FIFO burst threshold. Control the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be as low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory.
	This value is ignored if USBMODE[SDIS] (stream disable bit) is set. When USBMODE[SDIS] is set, the host controller behaves as if TXFIFOTHRES is set to the maximum value.
15–13 -	This field is reserved.
	Reserved, should be cleared.
12–8 TXSCHHEALTH	Scheduler health counter. Increment when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame.
	This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register clears the counter and this counter stops counting after reaching the maximum of 31.
TXSCHOH	Scheduler overhead. These bits add an additional fixed offset to the schedule time estimator described above as T_{ff} . As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization.

Table continues on the next page...

USB_TXFILLTUNING field descriptions (continued)

Field	Description
	<ul style="list-style-type: none"> The time unit represented in this register is $1.267\mu s$ when a device is connected in high-speed mode. The time unit represented in this register is $6.333\mu s$ when a device is connected in low-/full-speed mode. <p>For most applications, TXSCHOH can be set to 4 or less. A good value to begin with is: TXFIFOTHRES \times (16 \times 4 bytes-per-word) \div (40 \times TimeUnit), always rounded to the next higher integer. TimeUnit is either 1.267 or 6.333 as noted earlier in this description. If this value of TXSCHOH results in a TXSCHHEALTH count of 0 per second, try lowering the value by 1 if optimizing performance is desired. If TXSCHHEALTH exceeds 10 per second, try raising the value by 1.</p> <p>If streaming mode is disabled via the USBMODE register, treat TXFIFOTHRES as the maximum value for purposes of the TXSCHOH calculation.</p>

16.3.20 ULPI Register Access (USB_ULPI_VIEWPORT)

The ULPI register access provides indirect access to the ULPI PHY register set. Although the controller modules perform access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access. Be advised that writes to the ULPI through the ULPI viewport can substantially harm standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI. Note that executing read operations through the ULPI viewport should have no harmful side effects to standard USB operations. Also note that if the ULPI interface is not enabled, this register will always read zeros.

There are two operations that can be performed with the ULPI viewport, wakeup and read /write operations. The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync state bit (ULPISS). If this bit is set, then the ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS is cleared, then read/write operations will not be able execute. Undefined behavior results if a read or write operation is performed when ULPISS is cleared. To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is set and the ULPİRUN bit is cleared. Poll the ULPI Viewport until ULPIWU is cleared for the operation to complete.

To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPİRUN bit is set. Poll the ULPI Viewport until ULPİRUN is cleared for the operation to complete. For read operations, ULPIDATRD is valid once ULPİRUN is cleared.

The polling method above can be replaced with interrupts using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation completes, the ULPI interrupt is set.

NOTE

Read or write to the ULPI PHY extended register set (address > 3Fh) is not supported.

Address: 0h base + 170h offset = 170h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

USB ULPI VIEWPORT field descriptions

Field	Description
31 ULPIWU	ULPI Wake Up. Writing 1 to this bit begins the wakeup operation. This bit automatically transitions to 0 after the wakeup is complete. Once this bit is set, it cannot be cleared by software.
	NOTE: The driver must never execute a wakeup and a read/write operation at the same time.
30 ULPIRUN	ULPI Run. Writing 1 to this bit begins a read/write operation. This bit automatically transitions to 0 after the read/write is complete. Once this bit is set, it can not be cleared by software.
	NOTE: The driver must never execute a wakeup and a read/write operation at the same time.
29 ULPIRW	This bit selects between running a read or write operation to the ULPI. 0 Read 1 Write
28 -	This field is reserved. Reserved, should be cleared.
27 ULPISS	This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0.
	 0 Any other state (that is, carkit, serial, low power). 1 Normal Sync State.
26–24 ULPIPORT	For wakeup or read/write operations this value selects the port number to which the ULPI PHY is attached. Valid values are 0 and 1.
23–16 ULPIADDR	When a read or write operation is commanded, the address of the operation is written to this field.
15–8 ULPIDATRD	After a read operation completes, the result is placed in this field.
ULPIDTWR	When a write operation is commanded, the data to be sent is written to this field.

16.3.21 Endpoint NAK Indicator Register (USB_ENDPTNAK)

This register is only used in device mode.

Address: 0h base + 178h offset = 178h

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

USB_ENDPTNAK field descriptions

Field	Description
31–22 -	This field is reserved. Reserved.
21–16 EPTN	TX Endpoint NAK . Each TX endpoint has one bit in this field. This bit is set when controller sends a NAK handshake on a received IN token for the corresponding endpoint. EPTN[5] - Endpoint #5 ... EPTN[1] - Endpoint #1 EPTN[0] - Endpoint #0 Only used in device mode.
15–6 -	This field is reserved. Reserved.
EPRN	RX Endpoint NAK . Each RX endpoint has one bit in this field. This bit is set when controller sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. EPRN[5] - Endpoint #5 ... EPRN[1] - Endpoint #1 EPRN[0] - Endpoint #0 Only used in device mode.

16.3.22 Endpoint NAK Indication Enable Register (USB_ENDPTNAKEN)

This register is only used in device mode.

Address: 0h base + 17Ch offset = 17Ch

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

USB_ENDPTNAKEN field descriptions

Field	Description
31–22 -	This field is reserved. Reserved.
21–16 EPTNE	TX Endpoint NAK Enable . Each bit is an enable bit for corresponding TX Endpoint NAK bit. If this bit is set and corresponding TX Endpoint NAK bit is set, the NAK interrupt bit is set. EPTNE[5] - Endpoint #5 ... EPTNE[1] - Endpoint #1 EPTNE[0] - Endpoint #0 Only used in device mode.
15–6 -	This field is reserved. Reserved.
EPRNE	RX Endpoint NAK Enable . Each bit is an enable bit for corresponding RX Endpoint NAK bit. If this bit is set and corresponding RX Endpoint NAK bit is set, the NAK interrupt bit is set. EPRNE[5] - Endpoint #5 ... EPRNE[1] - Endpoint #1 EPRNE[0] - Endpoint #0 Only used in device mode.

16.3.23 Configured Flag Register (USB_CONFIGFLAG)

This EHCI register is not used in this implementation. A read from this register returns a constant of a 0x0000_0001 to indicate that all port routings default to this host controller.

Address: 0h base + 180h offset = 180h

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

USB_CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved.
0 CF	Configure flag. Always 1. Indicating all port routings default to this host.

16.3.24 Port Status/Control (USB_PORTSC)

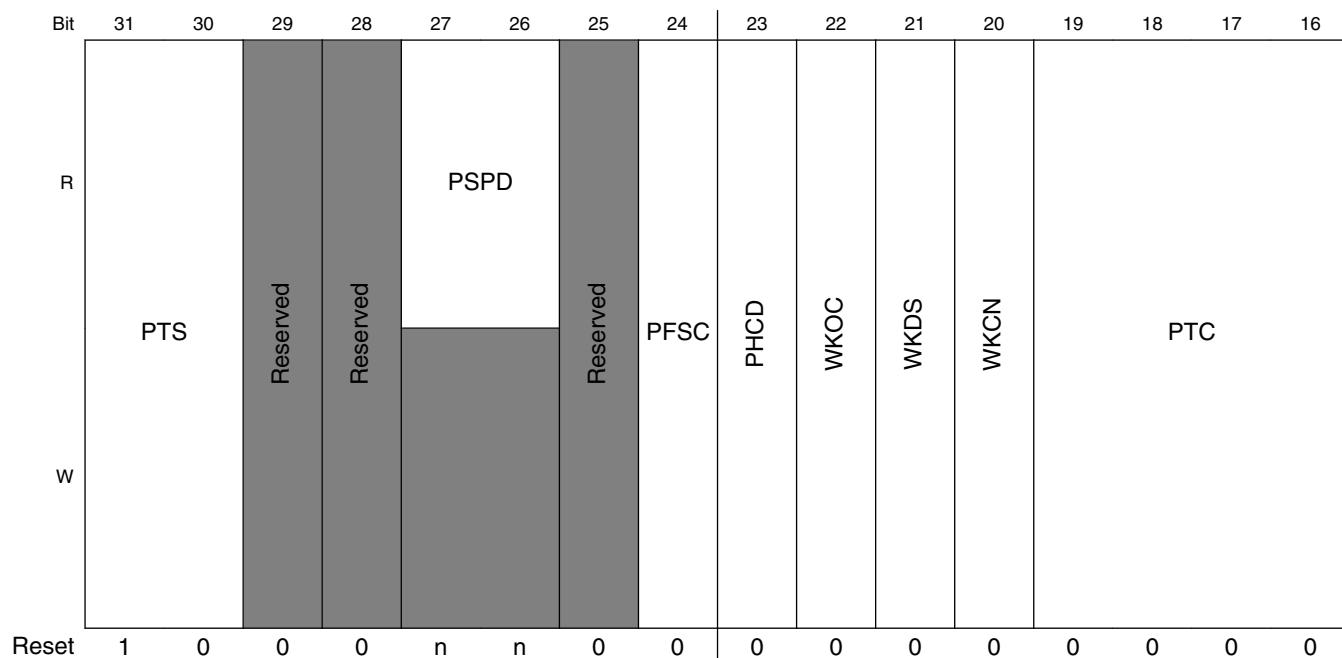
The port status and control (PORTSC) register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

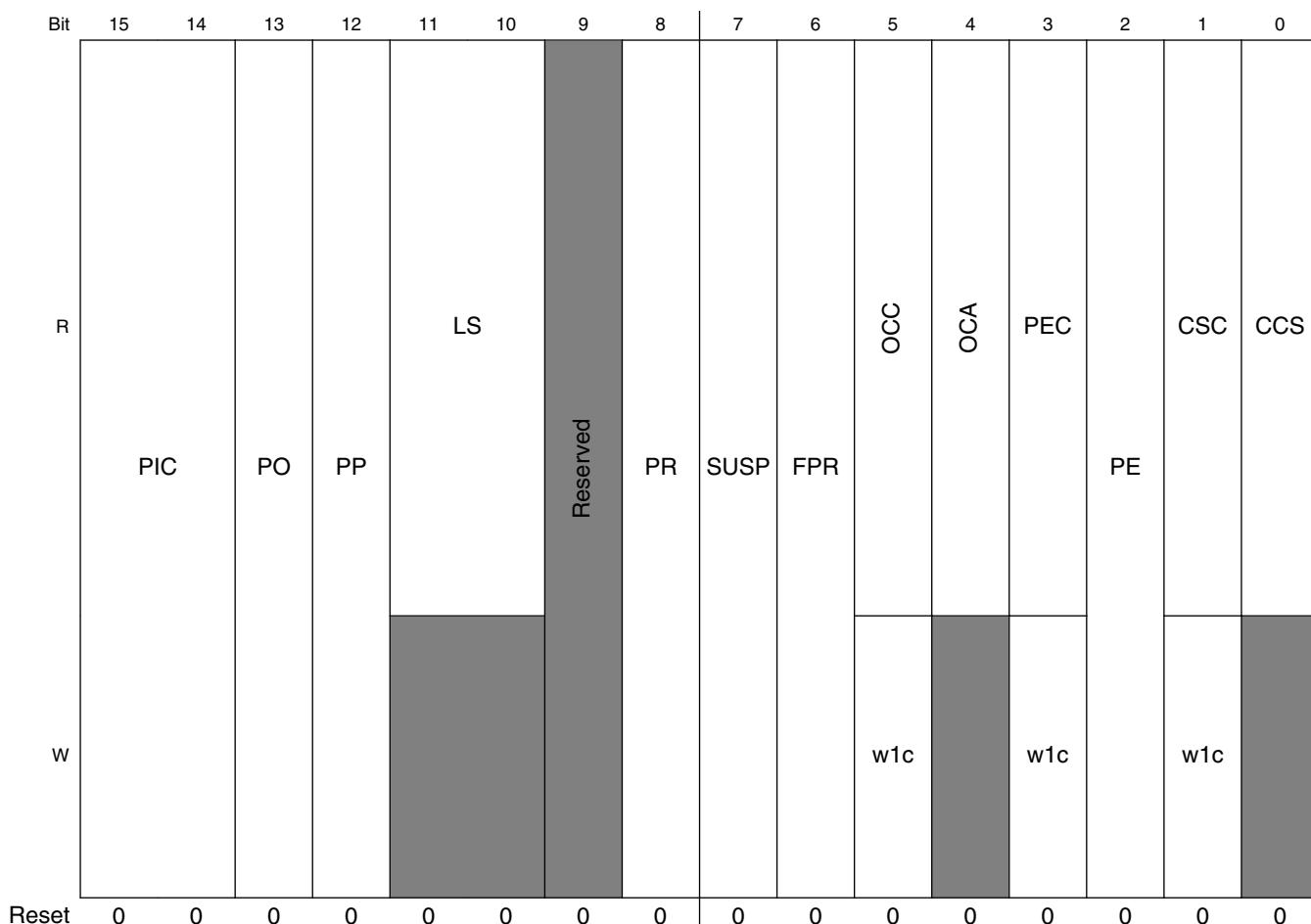
- No device connected
- Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power to one.

In device mode, the USB DR controller does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: 0h base + 184h offset = 184h





USB_PORTSC field descriptions

Field	Description
31–30 PTS	Port transceiver select. This register bit is used to control which parallel transceiver interface is selected. This bit is not defined in the EHCI specification. 00 Reserved 01 Reserved, should be cleared 10 ULPI parallel interface 11 Reserved
29 -	This field is reserved. Reserved, should be cleared
28 -	This field is reserved. Reserved, should be cleared
27–26 PSPD	Port speed. This read-only register field indicates the speed at which the port is operating. This bit is not defined in the EHCI specification. 00 Full-speed 01 Low-speed 10 High-speed 11 Undefined

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description								
25 -	This field is reserved. Reserved, should be cleared								
24 PFSC	<p>Port force full-speed connect. Used to disable the chirp sequence that allows the port to identify itself as a HS port. This is useful for testing FS configurations with a HS host, hub or device.</p> <p>This bit is not defined in the EHCI specification.</p> <p>This bit is for debugging purposes.</p> <p>0 Allow the port to identify itself as high speed. 1 Force the port to only connect at full speed.</p>								
23 PHCD	<p>PHY low power suspend. This bit is not defined in the EHCI specification.</p> <p>Host mode:</p> <ul style="list-style-type: none"> The PHY can be put into low power suspend - when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. <p>Device mode:</p> <ul style="list-style-type: none"> The PHY can be put into low power suspend - when the device is not running (USBCMD[RS] = 0b) or suspend signaling is detected on the USB. Low power suspend is cleared automatically when the resume signaling has been detected or when forcing port resume. <p>Reading this bit indicates the status of the PHY.</p> <p>NOTE: If there is no clock connected to the $USBn_CLK$ signals, PHCD must be set and the following registers should not be written: DEVICE_ADDR/PERIODICLISTBASE, PORTSC, ENDPTCTRL0, ENDPTCTRL1, ENDPTCTRL2, ENDPTCTRL3, ENDPTCTRL4, ENDPTCTRL5 .</p> <p>0 Normal PHY operation. 1 Signal the PHY to enter low power suspend mode</p>								
22 WKOC	<p>Wake on over-current enable. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events.</p> <p>This field is zero if Port Power (PP) is zero.</p> <p>This bit is used only in Host mode and used by an external power control circuit.</p>								
21 WKDS	<p>Wake on disconnect enable. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events.</p> <p>This field is zero if Port Power(PP) is zero or in device mode.</p> <p>This bit is used only in Host mode and used by an external power control circuit.</p>								
20 WKCN	<p>Wake on connect enable. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events.</p> <p>This field is zero if Port Power(PP) is zero or in device mode.</p> <p>This bit is used only in Host mode and used by an external power control circuit.</p>								
19–16 PTC	<p>Port test control. Any other value than zero indicates that the port is operating in test mode.</p> <p>Refer to Chapter 7 of the USB Specification Revision 2.0 [3] for details on each test mode.</p> <table style="margin-left: 20px;"> <tr> <td>0000</td> <td>Not Enabled</td> </tr> <tr> <td>0001</td> <td>J_STATE</td> </tr> <tr> <td>0010</td> <td>K_STATE</td> </tr> <tr> <td>0011</td> <td>SEQ_NAK</td> </tr> </table>	0000	Not Enabled	0001	J_STATE	0010	K_STATE	0011	SEQ_NAK
0000	Not Enabled								
0001	J_STATE								
0010	K_STATE								
0011	SEQ_NAK								

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	0100 Packet 0101 FORCE_ENABLE 0110-1111 Reserved, should be cleared
15–14 PIC	Port indicator control. Control the link indicator signals. These signals are valid for host mode only. This field is output from the controller for use by an external LED driving circuit. 00 Off 01 Amber 10 Green 11 Undefined
13 PO	Port owner. Unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero. System software uses this field to release ownership of the port to a selected module (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner hand-off is not implemented in this design, therefore this bit is always 0.
12 PP	Port power. Represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, and so on. When an over-current condition is detected on a powered port, the PP bit in each affected port is transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host controller (PPC = 1). In a device-only implementation port power control is not necessary, thus PPC and PP = 0. 0 USBDR writes 0 for DRVVBUS bit of OTGControl register in PHY. 1 USBDR writes 1 for DRVVBUS bit of OTGControl register in PHY. The OTG Control register is defined in ULPI specification.
11–10 LS	Line status. Reflect the current logical levels of the USB D+ (bit 11) and D- (bit 10) signal lines. The use of line status by the host controller driver is not necessary (unlike EHCI), because the connection of FS and LS is managed by hardware. 00 SE0 10 J-state 01 K-state 11 Undefined
9 -	This field is reserved. Reserved, should be cleared
8 PR	Port reset. Host mode: <ul style="list-style-type: none"> When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. Device mode: <ul style="list-style-type: none"> This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register.

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	<p>This field is zero if Port Power(PP) is zero.</p> <p>1 Port is in reset. 0 Port is not in reset.</p>
7 SUSP	<p>Suspend.</p> <p>Host mode:</p> <ul style="list-style-type: none"> The port enabled bit (PE) and suspend (SUSP) bit define the port states as follows: <p>0x Disable 10 Enable 11 Suspend</p> <ul style="list-style-type: none"> When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The module unconditionally sets this bit to zero when software clears the FPR bit. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (that is, port enabled bit is a zero) the results are undefined. This field is zero if Port Power (PP) is zero in host mode. <p>Device mode:</p> <p>1 Port in suspend state. 0 Port not in suspend state. Default.</p> <p>In device mode this bit is a read-only status bit.</p>
6 FPR	<p>Force port resume. This bit is not-EHCI compatible.</p> <ul style="list-style-type: none"> Software sets this bit to one to drive resume signaling. The controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one a J-to-K transition is detected, USBSTS[PCI] (port change detect) is also set. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no affect because the port controller will time the resume operation and clear this bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. <p>Device mode:</p> <ul style="list-style-type: none"> After the device has been in Suspend State for 5 ms or more, software can set this bit to one to drive resume signaling before clearing. The USB DR controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit is cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, USBSTS[PCI] is also set. <p>1 Resume detected/driven on port. 0 No resume (K-state) detected/driven on port.</p>
5 OCC	Over-current change. This bit gets set when there is a change to over-current active. Software clears this bit by writing a one to this bit position.

Table continues on the next page...

USB_PORTSC field descriptions (continued)

Field	Description
	<p>Host mode:</p> <ul style="list-style-type: none"> The user can provide over-current detection to the USB <i>n</i>_PWRFAULT signal for this condition. <p>Device mode:</p> <ul style="list-style-type: none"> This bit must always be 0. <p>1 Over current detect. 0 No over current.</p>
4 OCA	<p>Over-current active. This bit will automatically transition from one to zero when the over current condition is removed.</p> <p>Host mode:</p> <ul style="list-style-type: none"> The user can provide over-current detection to the USB <i>n</i>_PWRFAULT signal for this condition. <p>Device mode:</p> <ul style="list-style-type: none"> This bit must always be 0. <p>1 Port currently in over-current condition. 0 Port not in over-current condition.</p>
3 PEC	<p>Port enable/disable change.</p> <p>For the root hub, this bit gets set only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it.</p> <p>In device mode:</p> <ul style="list-style-type: none"> The device port is always enabled. (This bit is zero.) <p>This field is zero if Port Power(PP) is zero.</p> <p>1 Port disabled. 0 No change.</p>
2 PE	<p>Port enabled/disabled.</p> <p>Host mode:</p> <ul style="list-style-type: none"> Ports can only be enabled by the controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host and bus events. When the port is disabled, (0) downstream propagation of data is blocked except for reset. This field is zero if Port Power(PP) is zero in host mode. <p>Device mode:</p> <ul style="list-style-type: none"> The device port is always enabled. (This bit is one.)
1 CSC	<p>Connect change status.</p> <p>Host mode:</p> <ul style="list-style-type: none"> This bit indicates a change has occurred in the port's Current Connect Status. the controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system

Table continues on the next page...

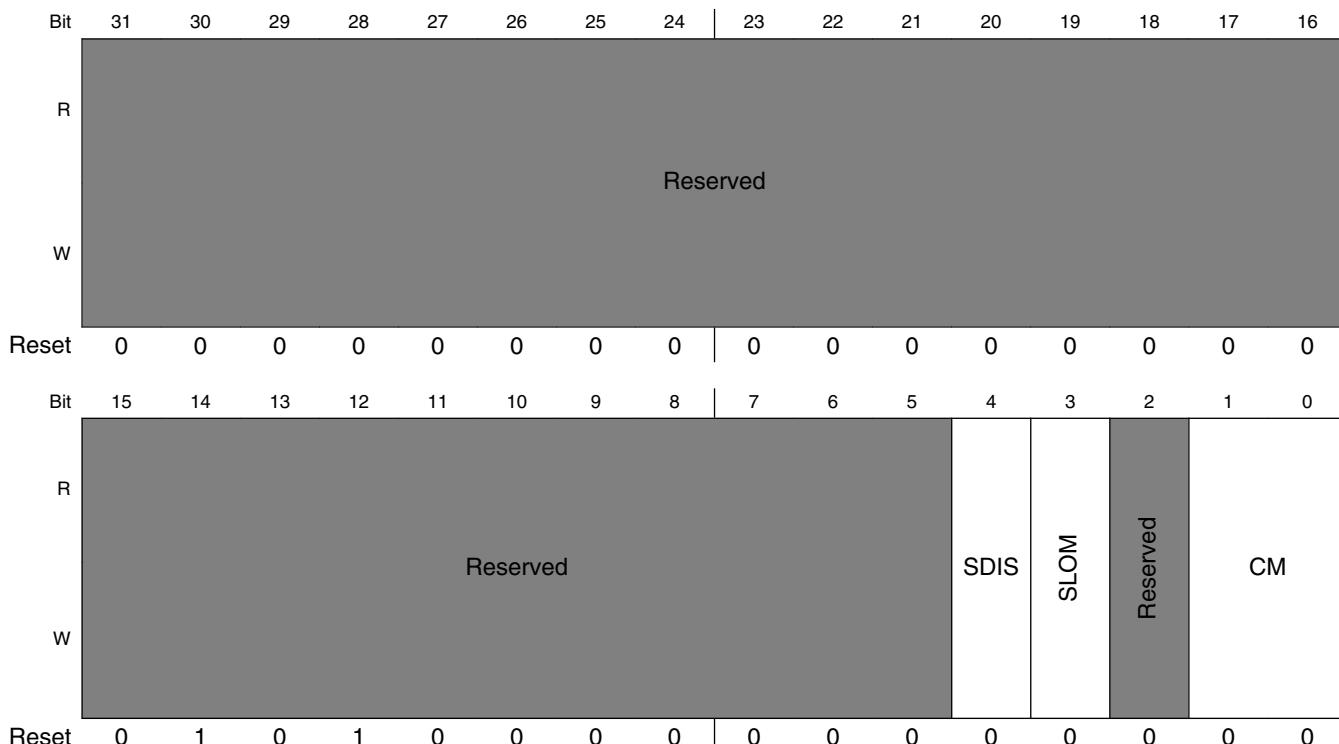
USB_PORTSC field descriptions (continued)

Field	Description
	<p>software has cleared the changed condition, hub hardware is 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it.</p> <p>1 Connect Status has changed. 0 No change.</p> <ul style="list-style-type: none"> • This field is zero if Port Power(PP) is zero. <p>Device mode:</p> <ul style="list-style-type: none"> • This bit is undefined.
0 CCS	<p>Current Connect Status.</p> <p>Host Mode:</p> <p>1 Device is present on port 0 No device is present</p> <p>This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit to be set.</p> <p>This field is zero if Port Power(PP) is zero in host mode.</p> <p>Device Mode:</p> <p>1 Attached 0 Not attached</p> <p>A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the PSPD bits in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to USBCMD[RS] (run bit). It does not state the device being disconnected or suspended.</p> <p>A value of zero also occurs if vbus de-asserts either when the host disables port power or the device has been disconnected from the bus.</p>

16.3.25 USB Device Mode (USB_USBMODE)

The USB mode register is not defined in the EHCI specification. This register controls the operating mode of the module.

Address: 0h base + 1A8h offset = 1A8h



USB_USBMODE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved, should be cleared.
4 SDIS	Stream disable In host mode, setting this bit ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity or a complete packet is stored in FIFO before the packet is launched onto the USB. Note that time duration to pre-fill the FIFO becomes significant when stream disable is active. See Transmit FIFO Tuning Controls (USB_TXFILLTUNING) to characterize the adjustments needed for the scheduler when using this feature. Also note that in systems with high system bus utilization, setting this bit will ensure no overruns or underruns during operation, at the expense of link utilization. For those who desire optimal link performance, SDIS can be left clear, and the rules used under the description of the TXFILLTUNING register to limit underruns/overruns.

Table continues on the next page...

USB_USBMODE field descriptions (continued)

Field	Description
	In device mode, setting this bit disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note that in high-speed mode, all packets received are responded to with a NYET handshake when stream disable is active. 1 Active. 0 Inactive.
3 SLOM	Setup lockout mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control endpoint operation model . 1 Setup lockouts off. DCD requires use of setup data buffer tripwire in USBCMD (SUTW). 0 Setup lockouts on
2 -	This field is reserved. Reserved, should be cleared.
CM	Controller mode This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to USBCMD[RST] before reprogramming this register. Defaults to the idle state and needs to be initialized to the desired operating mode after reset. 00 Idle (default). 01 Reserved, should be cleared. 10 Device controller. 11 Host controller.

16.3.26 Endpoint Setup Status (USB_ENDPTSETUPSTAT)

The endpoint setup status register is not defined in the EHCI specification. This register contains the endpoint setup status. It is only used in device mode.

Address: 0h base + 1ACh offset = 1ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																					ENDPTSETUPSTAT										
W	w1c																					w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

USB_ENDPTSETUPSTAT field descriptions

Field	Description
31–6 -	This field is reserved. Reserved, should be cleared.

Table continues on the next page...

USB_ENDPTSETUPSTAT field descriptions (continued)

Field	Description
ENDPTSETUPSTAT	<p>Setup endpoint status. For every setup transaction that is received, a corresponding bit in this register is set. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lockout mechanism is engaged.</p> <p>This register is only used in device mode.</p>

16.3.27 Endpoint Initialization (USB_ENDPOINTPRIME)

The endpoint initialization register is not defined in the EHCI specification. This register is used to initialize endpoints. It is only used in device mode.

Address: 0h base + 1B0h offset = 1B0h

USB_ENDPOINTPRIME field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared.
21–16 PETB	Prime endpoint transmit buffer. For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint.
	Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. PETB[5] (bit 21 of the register) corresponds to endpoint 5.
	Note that these bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
15–6 -	This field is reserved. Reserved, should be cleared.
PERB	Prime endpoint receive buffer. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation in order to respond to a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed. PERB[5] corresponds to endpoint 5.
	Note that these bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.

16.3.28 Endpoint Flush (USB_ENDPTFLUSH)

The endpoint flush register is not defined in the EHCI specification. This register is only used in device mode.

Address: 0h base + 1B4h offset = 1B4h

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

USB_ENDPTFLUSH field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared.
21–16 FETB	Flush endpoint transmit buffer. Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[5] (bit 21 of the register) corresponds to endpoint 5.
15–6 -	This field is reserved. Reserved, should be cleared.
FERB	Flush endpoint receive buffer. Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FERB[5] corresponds to endpoint 5.

16.3.29 Endpoint Status (USB_ENDPTSTATUS)

The endpoint status register is not defined in the EHCI specification. This register is only used in device mode.

Address: 0h base + 1B8h offset = 1B8h

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

USB-ENDPTSTATUS field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared
21–16 ETBR	Endpoint transmit buffer ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. ETBR[5] (bit 21 of the register) corresponds to endpoint 5. Note that these bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.
15–6 -	This field is reserved. Reserved, should be cleared
ERBR	Endpoint receive buffer ready. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. ERBR[5] corresponds to endpoint 5. Note that these bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.

16.3.30 Endpoint Complete (USB-ENDPTCOMPLETE)

The endpoint complete register is not defined in the EHCI specification. This register is only used in device mode.

Address: 0h base + 1BCh offset = 1BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										ETCE	Reserved										ERCE	w1c									
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

USB-ENDPTCOMPLETE field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared
21–16 ETCE	Endpoint transmit complete event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register. ETCE[5] (bit 21 of the register) corresponds to endpoint 5.

Table continues on the next page...

USB_ENDPTCOMPLETE field descriptions (continued)

Field	Description
15–6 -	This field is reserved. Reserved, should be cleared
ERCE	Endpoint receive complete event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register. ERCE[5] corresponds to endpoint 5.

16.3.31 Endpoint Control 0 (USB_ENDPTCTRL0)

Endpoint control register 0 is not defined in the EHCI specification. Every device will implement endpoint 0 as a control endpoint.

Address: 0h base + 1C0h offset = 1C0h

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

USB_ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, should be cleared.
23 TXE	TX endpoint enable. Endpoint zero is always enabled. 0 Disable 1 Enable
22–20 -	This field is reserved. Reserved, should be cleared.
19–18 TXT	TX endpoint type. Endpoint zero is always a control endpoint (00).
17 -	This field is reserved. Reserved, should be cleared.
16 TXS	TX endpoint stall. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 1 Endpoint stalled 0 Endpoint OK
15–8 -	This field is reserved. Reserved, should be cleared.
7 RXE	RX endpoint enable. Endpoint zero is always enabled. 0 Disabled 1 Enabled
6–4 -	This field is reserved. Reserved, should be cleared.
3–2 RXT	RX endpoint type. Endpoint zero is always a control endpoint (00).
1 -	This field is reserved. Reserved, should be cleared.
0 RXS	RX endpoint stall Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 1 Endpoint stalled 0 Endpoint OK

16.3.32 Endpoint Control *n* (USB_ENDPTCTRL*n*)

The endpoint control *n* registers are not defined in the EHCI specification. There is an ENDPTCTRL *n* register of each endpoint in a device.

Address: 0h base + 1C4h offset + (4d × i), where i=0d to 4d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved		TXT	TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE	RXR	RXI	Reserved		RXT	RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ENDPTCTRL*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, should be cleared
23 TXE	TX endpoint enable 0 Disabled 1 Enabled
22 TXR	TX data toggle reset. Whenever a configuration event is received for this endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX data toggle inhibit. Used only for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 PID sequencing enabled 1 PID sequencing disabled
20 -	This field is reserved. Reserved, should be cleared

Table continues on the next page...

USB_ENDPTCTRL n field descriptions (continued)

Field	Description
19–18 TXT	<p>TX endpoint type</p> <p>NOTE: When only one endpoint (RX or TX, but not both) of an endpoint pair is used, the unused endpoint should be configured as a bulk type endpoint.</p> <ul style="list-style-type: none"> 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX endpoint data source. This bit should always be written as 0, which selects the dual port memory/DMA engine as the source.
16 TXS	<p>TX endpoint stall. This bit is set automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It is cleared automatically upon receipt of a SETUP request if this endpoint is configured as a control endpoint.</p> <p>Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above.</p> <ul style="list-style-type: none"> 0 Endpoint OK 1 Endpoint stalled
15–8 -	This field is reserved. Reserved, should be cleared
7 RXE	<p>RX endpoint enable</p> <ul style="list-style-type: none"> 0 Disabled 1 Enabled
6 RXR	RX data toggle reset. Whenever a configuration event is received for this endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
5 RXI	<p>RX data toggle inhibit. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packets regardless of their data PID.</p> <ul style="list-style-type: none"> 1 PID sequencing enabled 0 PID sequencing disabled
4 -	This field is reserved. Reserved, should be cleared
3–2 RXT	<p>RX endpoint type.</p> <p>NOTE: When only one endpoint (RX or TX, but not both) of an endpoint pair is used, the unused endpoint should be configured as a bulk type endpoint.</p> <ul style="list-style-type: none"> 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX endpoint data sink. This bit should always be written as 0, which selects the dual port memory/DMA engine as the sink.
0 RXS	RX endpoint stall. This bit is set automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It is cleared automatically upon receipt a SETUP request if this endpoint is configured as a control endpoint,

Table continues on the next page...

USB_ENDPTCTRL*n* field descriptions (continued)

Field	Description				
	<p>Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above,</p> <table> <tr> <td data-bbox="352 255 380 259">1</td><td data-bbox="380 255 554 259">Endpoint stalled</td></tr> <tr> <td data-bbox="352 273 380 277">0</td><td data-bbox="380 273 514 277">Endpoint OK</td></tr> </table>	1	Endpoint stalled	0	Endpoint OK
1	Endpoint stalled				
0	Endpoint OK				

16.3.33 Snoop n (USB SNOOPn)

These registers use big-endian byte ordering and are not defined in the EHCI specification. The SNOOP1 and SNOOP2 registers provide snooping control and address range selection function. Transactions that hit a snooping window will generate cache coherent transactions on the internal system bus. When the five lower bits (SNOOP n [27-31]) are equal to 00000, snooping is always disabled on the system bus for all DMA transfers. When SNOOP n [27-31] is 01011 through 11110, the twenty upper bits (SNOOP n [0-19]) provide the starting base address for which transactions are snooped. These twenty bits are compared to the twenty upper bits of the address provided by the DMA block of the USB controller. When a match occurs, the five lower bits are decoded as shown below. This provides a snooping region of 4 KB to 2 GB within each starting base address that is programmed by the core. The SNOOP n [20-26] are not used.

Address: 0h base + 400h offset + (4d x i), where i=0d to 1d

USB SNOOP n field descriptions

Field	Description
0-19 Snoop_address	The starting base address for which transactions are snooped.
20-26 -	This field is reserved. Reserved, should be cleared
27-31 Snoop_Enables	Snoop_Enables
	0x00 Snooping disabled
	0x0B 4 KB snoop range starting at the value defined by SNOOP n [0-19]
	0x0C 8 KB snoop range starting at the value defined by SNOOP n [0-18]
	0x0D 16 KB snoop range starting at the value defined by SNOOP n [0-17]
	0x0E 32 KB snoop range starting at the value defined by SNOOP n [0-16]
	0x0F 64 KB snoop range starting at the value defined by SNOOP n [0-15]
	0x10 128 KB snoop range starting at the value defined by SNOOP n [0-14]

Table continues on the next page...

USB_SNOOPn field descriptions (continued)

Field	Description
0x11	256 KB snoop range starting at the value defined by SNOOP n [0-13]
0x12	512 KB snoop range starting at the value defined by SNOOP n [0-12]
0x13	1 MB snoop range starting at the value defined by SNOOP n [0-11]
0x14	2 MB snoop range starting at the value defined by SNOOP n [0-10]
0x15	4 MB snoop range starting at the value defined by SNOOP n [0-9]
0x16	8 MB snoop range starting at the value defined by SNOOP n [0-8]
0x17	16 MB snoop range starting at the value defined by SNOOP n [0-7]
0x18	32 MB snoop range starting at the value defined by SNOOP n [0-6]
0x19	64 MB snoop range starting at the value defined by SNOOP n [0-5]
0x1A	128 MB snoop range starting at the value defined by SNOOP n [0-4]
0x1B	256 MB snoop range starting at the value defined by SNOOP n [0-3]
0x1C	512 MB snoop range starting at the value defined by SNOOP n [0-2]
0x1D	1 GB snoop range starting at the value defined by SNOOP n [0-1]
0x1E	2 GB snoop range starting at the value defined by SNOOP n [0]

16.3.34 Age Count Threshold (USB_AGE_CNT_THRESH)

Note that this register uses big-endian byte ordering and is not defined in the EHCI specification. The age count threshold (AGE_CNT_THRESH) register provides the aging counter threshold value used to determine the priority state of the USB controller's internal system interface. The threshold value is in units of platform clock cycles. This register should be written during system initialization or during normal system operation when the system bus interface is idle. It can be read at any time.

If the aging counter is less than the AGE_CNT_THRESH value, If the aging counter is greater than or equal to the AGE_CNT_THRESH value

The aging counter begins to count from zero when a bus access is requested. It increments every bus cycle until the bus transaction completes. At the completion of a bus transaction, the counter is synchronously reset to zero. If there are any outstanding bus requests, the aging counter will then begin counting immediately.

The AGE_CNT_THRESH is compared against the value of the aging counter during each clock cycle of the current transaction. If AGE_CNT_THRESH is equal to zero, . If the aging counter is less than the AGE_CNT_THRESH value, . If the aging counter is greater than or equal to the AGE_CNT_THRESH value .

The setting of AGE_CNT_THRESH is highly dependent on both the mix of other controllers operating on the system bus as well as the kind of traffic moving through the USB controller. A recommended approach is first to try leaving the aging mechanism disabled and see if the USB meets performance requirements. If USB performance does not meet application requirements, try the following setting :

- Set AGE_CNT_THRESH to .

Raising AGE_CNT_THRESH benefits the other controllers on the system bus by reducing the frequency that this USB controller raises its priority to the arbiter.

Address: 0h base + 408h offset = 408h

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

USB_AGE_CNT_THRESH field descriptions

Field	Description
0–17 - 	This field is reserved. Reserved, should be cleared
18–31 Threshold	Aging counter threshold value.

16.3.35 Priority Control (USB_PRI_CTRL)

This register uses big-endian byte ordering and is not defined in the EHCI specification. The priority control (PRI_CTRL) register sets the priority level for each of two priority states. The priority state is determined by the value programmed in the AGE_CNT_THRESH register and the number of platform cycles that a particular transaction takes to complete .

Address: 0h base + 40Ch offset = 40Ch

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

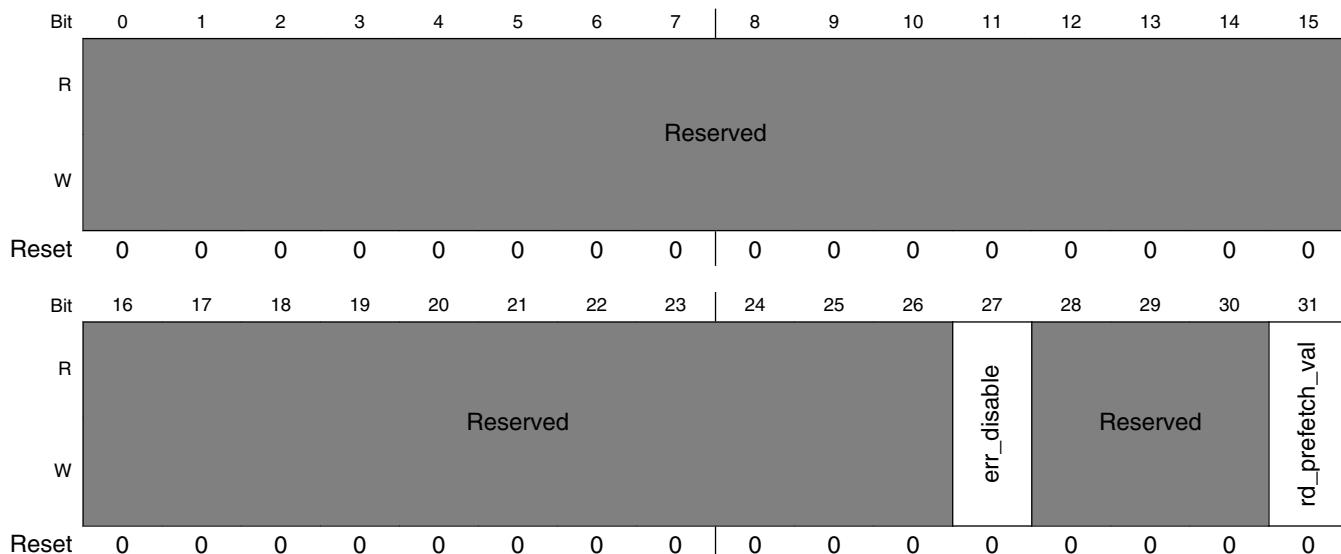
USB_PRI_CTRL field descriptions

Field	Description
0–27 - 	This field is reserved. Reserved, should be cleared
28–29 pri_lvl1	Priority level for priority state 1. The highest priority is 2'h3 and the lowest priority is 2'b0.
30–31 pri_lvl0	Priority level for priority state 0. The highest priority is 2'h3 and the lowest priority is 2'b0.

16.3.36 System Interface Control (USB_SI_CTRL)

This register uses big-endian byte ordering and is not defined in the EHCI specification. The system interface control register (SI_CTRL) controls various functions pertaining to the internal system interface.

Address: 0h base + 410h offset = 410h



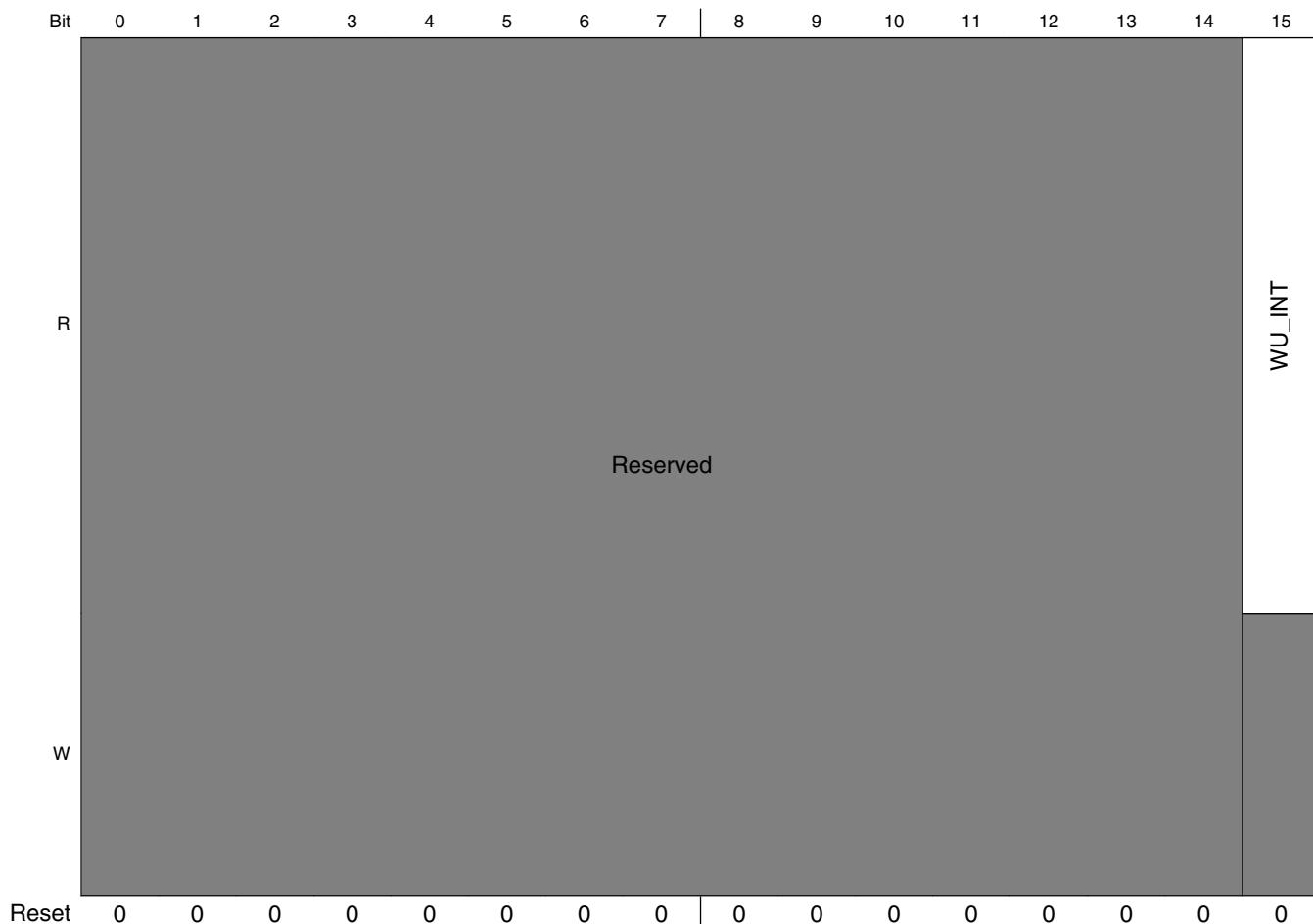
USB_SI_CTRL field descriptions

Field	Description
0–26 -	This field is reserved. Reserved, should be cleared
27 err_disable	When this bit is set, it causes the controller to ignore system bus errors. If cleared the controller responds according to the values set in USBSTS[SEI] and USBINT[SEE]. 0 enable 1 disable
28–30 -	This field is reserved. Reserved, should be cleared
31 rd_prefetch_val	Selects whether 32 bytes or 64 bytes are fetched during burst read transactions at the system interface. When this input is ZERO, 64 bytes are fetched, and when it is ONE, 32 bytes are fetched. NOTE: 32 byte fetch mode is not supported, this value must be set to 0. 0 64-byte fetch 1 32-byte fetch

16.3.37 Control (USB_CONTROL)

This register uses big-endian byte ordering and is not defined in the EHCI specification. The USB general purpose (CONTROL) register contains the general-purpose IP control register outputs.

Address: 0h base + 500h offset = 500h



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

USB_CONTROL field descriptions

Field	Description
0–14 -	This field is reserved. Reserved
15 WU_INT	Reflects the state of the wake up interrupt. The wake up interrupt signal is asserted when a wake-up event occurs while in a low-power suspend state. If WU_INT_EN is set, this WU_INT signal generates an interrupt to the system to indicate wake up servicing is required. WU_INT will remain set until the USB controller is exited from the low power by clearing the PORTSC[PHCD] bit. 0 Normal operation or Low Power mode waiting for wakeup event 1 Low power wakeup event has occurred
16–18 -	This field is reserved.
22 -	This field is reserved.
26–27 -	This field is reserved.
28 -	This field is reserved.
29 USB_EN	ULPI mode: In safe mode, all USB interface signals are put into input mode or driven inactive, except for SUSPEND_STP, which is driven high. Also, the input signal DIR is forced to appear high to the controller. This prevents any start-up problems that otherwise could occur if the PHY and the controller take significantly different times to complete power-on reset. 1 Normal operation 0 Safe mode
30 WU_INT_EN	This bit is used to mask/unmask the system wakeup interrupt signal NOTE: PORTSC[PHCD] bit must be set for the system wakeup interrupt generation.

Table continues on the next page...

USB_CONTROL field descriptions (continued)

Field	Description
	0 System wakeup interrupt disabled 1 System wakeup interrupt enabled
31 ULPI_INT_EN	Used to enable the ULPI low power wakeup interrupt from the PHY when the PHY is in low power mode only. NOTE: PORTSC[PHCD] bit must be set. 0 ULPI low power wakeup interrupt disabled 1 ULPI low power wakeup interrupt enabled

16.4 Functional description

The USB DR module can be broken down into functional sub-blocks, which are described below.

16.4.1 System interface

The system interface block contains all the control and status registers that allow a processor to interface to the USB DR module.

These registers allow the processor to control the configuration of the module, ascertain the capabilities of the module, and control the module's operation. It also has registers to control snoopability and priority of the DMA interface.

16.4.2 DMA engine

The module contains a local DMA engine.

The DMA engine interfaces internally to the system memory bus. It is responsible for moving all of the data to be transferred over the USB between the module and buffers in system memory.

Like the system interface block, the DMA engine block uses a simple synchronous bus signaling protocol that eases connections to a number of different standard buses.

The DMA controller must access both control information and packet data from system memory. The control information is contained in link list-based queue structures. The DMA controller has state machines that are able to parse data structures defined in the EHCI specification. In host mode, the data structures are EHCI compliant and represent queues of transfers to be performed by the host controller, including the split-transaction

requests that allow an EHCI controller to direct packets to FS and LS devices. In device mode, the data structures are designed to be similar to those in the EHCI specification and are used to allow device responses to be queued for each of the active pipes in the device.

16.4.3 FIFO RAM controller

The FIFO RAM controller is used for context information and to control FIFOs between the protocol engine and the DMA controller.

These FIFOs decouple the system processor/memory bus requests from the extremely tight timing required by USB.

The use of the FIFO buffers differs between host and device mode operation. In host mode, a single data channel is maintained in each direction through the buffer memory. In device mode, multiple FIFO channels are maintained for each of the active endpoints in the system.

In host mode, the USB DR module uses a 512-byte Tx buffer and a 512-byte Rx buffer. Device operation uses a single 512-byte Rx buffer and a 512-byte Tx buffer for each endpoint. The 512-byte buffers allow the module to buffer a complete HS bulk packet.

16.4.4 PHY interface

The ULPI interface connects to an external PHY. The USB DR module interfaces to any ULPI-compatible PHY.

The primary function of the port controller block is to isolate the rest of the module from the transceiver, and to move all of the transceiver signaling into the primary clock domain of the module.

This allows the module to run synchronously with the system processor and its associated resources.

Due to pincount limitations the module only supports certain combinations of PHY interfaces and USB functionality. Refer to the table below for more information.

Table 16-52. Supported PHY interfaces

PHY	Function
ULPI	Host/Device

16.5 Host data structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware).

The data structure definitions in this section support a 32-bit memory buffer address space. The interface consists of a periodic schedule, periodic frame list, asynchronous schedule, isochronous transaction descriptors, split-transaction isochronous transfer descriptors, queue heads, and queue element transfer descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. Isochronous data streams are managed using isochronous transaction descriptors. Isochronous split-transaction data streams are managed with split-transaction isochronous transfer descriptors. All interrupt, control, and bulk data streams are managed with queue heads and queue element transfer descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

NOTE

Software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writable fields. The host controller must preserve the read-only fields on all data structure writes.

16.5.1 Periodic frame list

The figure below shows the organization of the periodic schedule. This schedule is for all periodic transfers (isochronous and interrupt).

The periodic schedule is referenced from the operational registers space using the PERIODICLISTBASE address register and the FRINDEX register. The periodic schedule is based on an array of pointers called the periodic frame list. The PERIODICLISTBASE address register is combined with the FRINDEX register to produce a memory pointer into the frame list. The periodic frame list implements a sliding window of work over time.

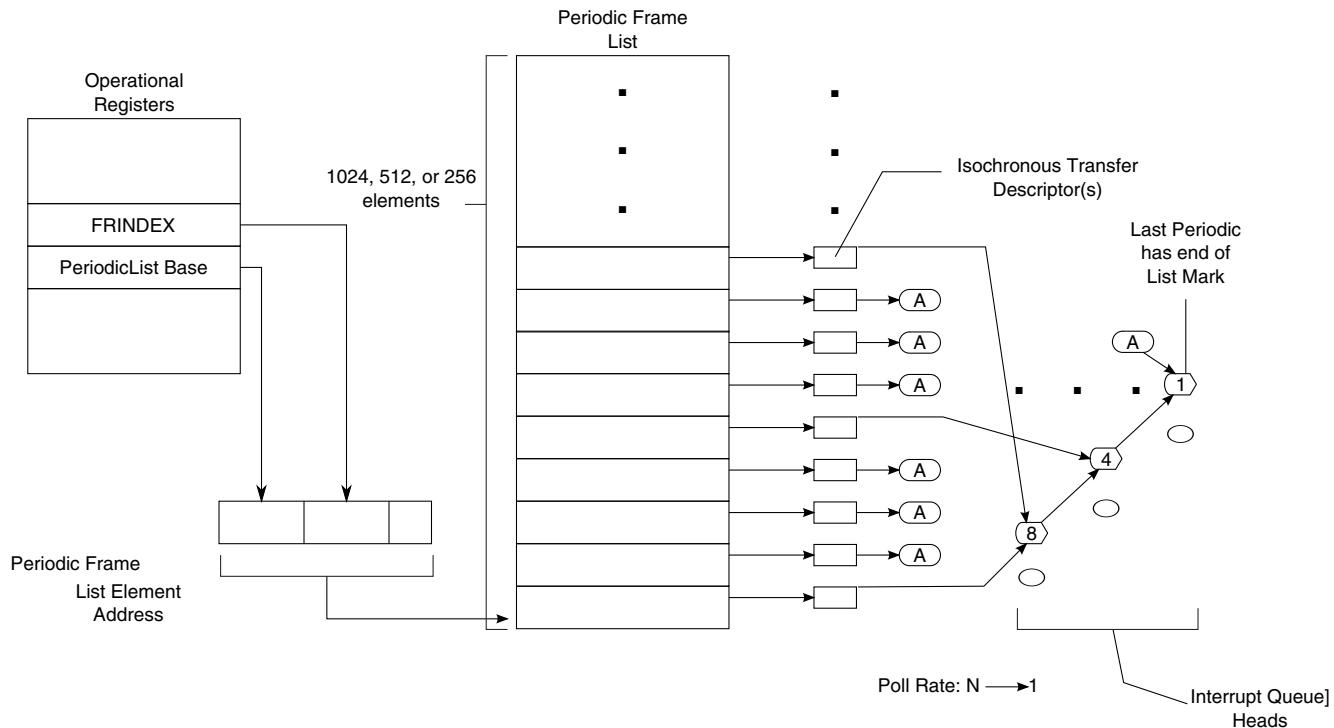


Figure 16-50. Periodic schedule organization

The periodic frame list is a 4K-page aligned array of Frame List Link pointers. The length of the frame list is programmable. The programmability of the periodic frame list is exported to system software through the HCCPARAMS register. The length can be selected by system software as one of 8, 16, 32, 64, 128, 256, 512 or 1024 elements. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USBCMD register.

Frame list link pointers direct the host controller to the first work item in the frame's periodic schedule for the current microframe. The link pointers are aligned on DWord boundaries within the frame list. The table below shows the format for the frame list link pointer.

Table 16-53. Frame list link pointer format

Frame list link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or

a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least-significant bits in a frame list pointer are used to key the host controller in as to the type of object the pointer is referencing.

The least-significant bit is the T bit (bit 0). When this bit is set, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field indicates the exact type of data structure being referenced by this pointer. The value encodings for the Typ field are given in the table below.

Table 16-54. Typ field encodings

Typ	Description
00	Isochronous transfer descriptor
01	Queue head
10	Split transaction isochronous transfer descriptor
11	Frame span traversal node

16.5.2 Asynchronous list queue head pointer

The asynchronous transfer list (based at the ASYNCLISTADDR register) is where all the control and bulk transfers are managed.

Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty. The figure below shows the asynchronous schedule organization.

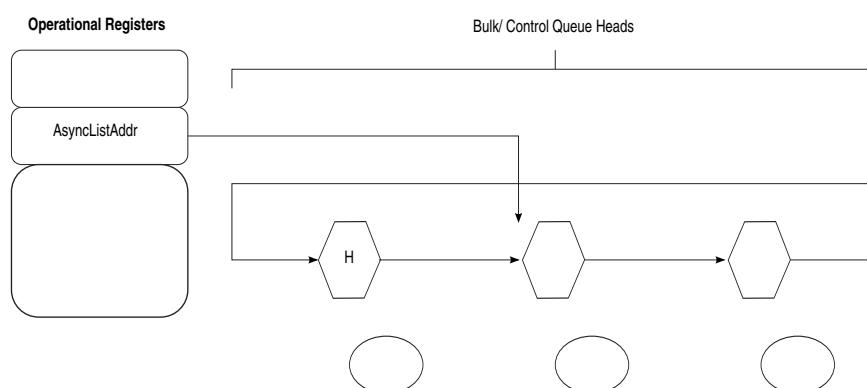


Figure 16-51. Asynchronous schedule organization

The asynchronous list is a simple circular list of queue heads. The ASYNCLISTADDR register is simply a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

16.5.3 Isochronous (high-speed) transfer descriptor (iTД)

The table below shows the format of an isochronous transfer descriptor.

This structure is used only for high-speed isochronous endpoints.

All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 16-55. Isochronous transaction descriptor (iTД)

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	15 4	1 4	1 3	1 2	11 0	1 0	9 9	8 8	7 7	6 6	5 5	4 4	3 3	2 2	1 1	0 0	offset							
Next link pointer																								00	Typ	T	0x00												
Status ¹	Transaction 0 Length ¹								ioc	PG ²		Transaction 0 Offset ²								0x04																			
Status ¹	Transaction 1 length ¹								ioc	PG ²		Transaction 1 offset ²								0x08																			
Status ¹	Transaction 2 length ¹								ioc	PG ²		Transaction 2 offset ²								0x0C																			
Status ¹	Transaction 3 length ¹								ioc	PG ²		Transaction 3 offset ²								0x10																			
Status ¹	Transaction 4 length ¹								ioc	PG ²		Transaction 4 offset ²								0x14																			
Status ¹	Transaction 5 length ¹								ioc	PG ²		Transaction 5 offset ²								0x18																			
Status ¹	Transaction 6 length ¹								ioc	PG ²		Transaction 6 offset ²								0x1C																			
Status ¹	Transaction 7 length ¹								ioc	PG ²		Transaction 7 offset ²								0x20																			
Buffer pointer (page 0)												EndPt	R	Device address								0x24																	
Buffer pointer (page 1)												I/O	Maximum packet size								0x28																		
Buffer pointer (Page 2)												Reserved								Mult	0x2C																		
Buffer pointer (page 3)												Reserved								0x30																			
Buffer pointer (page 4)												Reserved								0x34																			
Buffer pointer (page 5)												Reserved								0x38																			
Buffer pointer (page 6)												Reserved								0x3C																			

1. Host controller read/write; all others read-only.
2. These fields may be modified by the host controller if the I/O field indicates an OUT.

16.5.3.1 Next link pointer-iTD

The first DWord of an iTD is a pointer to the next schedule data structure, as shown in the table below.

Table 16-56. Next schedule element pointer

Bits	Name	Description
31-5	Link Pointer	Correspond to memory address signals [31:5], respectively. This field points to another isochronous transaction descriptor (iTД/siTД) or queue head (QH).
4-3	-	Reserved, should be cleared. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2-1	Typ	Indicates to the host controller whether the item referenced is an iTД, siТД or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. Value encodings are: 00 iTД (isochronous transfer descriptor) 01 QH (queue head) 10 siТД (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate 1 Link Pointer field is not valid. 0 Link Pointer field is valid.

16.5.3.2 iTD transaction status and control list

DWords 1-8 constitute eight slots of transaction control and status.

Each transaction description includes the following:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction *n* Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description, plus the endpoint information contained in the first three DWords of the buffer page pointer list, to execute a transaction on the USB.

The table below shows the iTD transaction status and control fields.

Table 16-57. iTD transaction status and control

Bits	Name	Description
31-28	Status	<p>Records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:</p> <ul style="list-style-type: none"> 31 Active. Set by software to enable the execution of an isochronous transaction by the host controller. When the transaction associated with this descriptor is completed, the host controller clears this bit indicating that a transaction for this element should not be executed when it is next encountered in the schedule. 30 Data buffer error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (underrun). If an overrun condition occurs, no action is necessary. 29 Babble detected. Set by the host controller during status update when "babble" is detected during the transaction generated by this descriptor. 28 Transaction error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, and so on). This bit may only be set for isochronous IN transactions.
27-16	Transaction <i>n</i> Length	For an OUT, this field is the number of data bytes the host controller will send during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (for example, 0 zero length data, 1 one byte, 2 two bytes, and so on). The maximum value this field may contain is 0xC00 (3072).
15	ioc	Interrupt on complete. If this bit is set, it specifies that when this transaction completes, the host controller should issue an interrupt at the next interrupt threshold.
14-12	PG	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.
11-0	Transaction <i>n</i> Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent PG field to produce the starting buffer address for this transaction.

16.5.3.3 iTD buffer page pointer list (plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K aligned) to the data buffer for this transfer descriptor.

This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous. Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) x 1024 (maximum packet size) x 8 (transaction records) = 24 576 bytes to be moved with this data structure, regardless of the alignment offset of the first page.

Since each pointer is a 4 K-aligned page pointer, the least-significant 12 bits in several of the page pointers are used for other purposes.

The following tables describe buffer pointer page n .

Table 16-58. Buffer pointer page 0 (Plus)

Bits	Name	Description
31-12	Buffer pointer (Page 0)	A 4K-aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-8	EndPt	Selects the particular endpoint number on the device serving as the data source or sink.
7	-	Reserved, should be cleared. Reserved for future use and should be initialized by software to zero.
6-0	Device Address	This field selects the specific device serving as the data source or sink.

Table 16-59. iTD buffer pointer page 1 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11	I/O	Direction (I/O). This field encodes whether the high-speed transaction should use an IN or OUT PID. 0 OUT 1 IN
10-0	Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (for example, per microframe). This field is used with the Multi field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (0x400). Any value larger yields undefined results.

Table 16-60. Buffer pointer page 2 (plus)

Bits	Name	Description
31-12	Buffer pointer (page 2)	This is a 4K-aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-2	-	Reserved, should be cleared. This bit reserved for future use and should be cleared.
1-0	Mult	Indicates to the host controller the number of transactions that should be executed per transaction description (for example, per microframe). 00 Reserved, should be cleared. A zero in this field yields undefined results. 01 One transaction to be issued for this endpoint per microframe 10 Two transactions to be issued for this endpoint per microframe 11 Three transactions to be issued for this endpoint per microframe

Table 16-61. Buffer pointer page 3-6

Bits	Name	Description
31-12	Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-0	-	Reserved, should be cleared. These bits reserved for future use and should be cleared.

16.5.4 Split transaction isochronous transfer descriptor (siTD)

All full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure.

This data structure satisfies the operational requirements for managing the split transaction protocol.

The table below shows the split-transaction isochronous transfer descriptor (siTD).

Table 16-62. Split-transaction isochronous transaction descriptor (siTD)

31	3	2	2	2	2	2	2	2	2	2	1	1	1	1	15	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	offset
0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	14	3	2	1	0	9	8	7	6	5	4	3	2	1	0	offset	
Next Link Pointer																					00		Typ	T	0x00						
I/O	Port Number	0	Hub Address	0000	EndPt	0	Device Address	0x04																							
0000_0000_0000_00000				μFrame C-mask	μFrame S-mask	0x08																									
ioc	P ¹	0000	Total Bytes to Transfer ¹	μFrame C-prog-mask ¹	Status ¹	0x0C																									
Buffer Pointer (Page 0)				Current Offset ¹	0x10																										
Buffer Pointer (Page 1)				000_0000	TP ¹	T-count ¹	0x14																								
Back Pointer				0000	T	0x18																									

1. Host controller read/write; all others read-only.

16.5.4.1 Next link pointer-siTD

DWord0 of a siTD is a pointer to the next schedule data structure.

The table below describes the next link pointer fields.

Table 16-63. Next link pointer

Bits	Name	Description
31-5	Next Link Pointer	This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	-	Reserved, should be cleared. These bits must be written as zeros.
2-1	Typ	Indicates to the host controller whether the item referenced is an iTD/siTD or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. Value encodings are: 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate.

Table 16-63. Next link pointer

Bits	Name	Description
		0 Link pointer is valid.
		1 Link pointer field is not valid.

16.5.4.2 siTD endpoint capabilities/characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and microframe scheduling control.

The table below describes the endpoint and transaction translator characteristics.

Table 16-64. Endpoint and transaction translator characteristics

Bits	Name	Description
31	I/O	Direction (I/O). This field encodes whether the full-speed transaction should be an IN or OUT. 0 OUT 1 IN
30-24	Port Number	This field is the port number of the recipient transaction translator.
23	-	Reserved, should be cleared. Bit reserved and should be cleared.
22-16	Hub Address	This field holds the device address of the companion controllers' hub.
15-12	-	Reserved, should be cleared. Field reserved and should be cleared.
11-8	EndPt	Endpoint Number. Selects the particular endpoint number on the device serving as the data source or sink.
7	-	Reserved, should be cleared. Bit is reserved for future use. It should be cleared.
6-0	Device Address	Selects the specific device serving as the data source or sink.

The table below describes the microframe schedule control.

Table 16-65. Microframe schedule control

Bits	Name	Description
31-16	-	Reserved, should be cleared. This field reserved for future use. It should be cleared.
15-8	μ Frame C-mask	Split completion mask. This field (along with the Active and SplitX-state fields in the status byte) is used to determine during which microframes the host controller should execute complete-split transactions. When the criteria for using this field is met, an all-zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the μ Frame C-Mask field is a one, this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	μ Frame S-mask	Split start mask. This field (along with the Active and SplitX-state fields in the Status byte) is used to determine during which microframes the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into

Table 16-65. Microframe schedule control

Bits	Name	Description
		this bit field. If the FRINDEX register value indexes to a position where the μ Frame S-mask field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

16.5.4.3 siTD transfer state

DWords 3-6 manage the state of the transfer, as described in the table below.

Table 16-66. siTD transfer status and control

Bits	Name	Description											
31	ioc	Interrupt on complete 0 Do not interrupt when transaction is complete. 1 Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it will assert a hardware interrupt at the next interrupt threshold.											
30	P	Page select. Indicates which data page pointer should be concatenated with the CurrentOffset field to construct a data buffer pointer 0 Selects Page 0 pointer 1 Selects Page 1 pointer The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a one to a zero).											
29-26	-	Reserved, should be cleared. This field reserved for future use and should be cleared.											
25-16	Total Bytes to Transfer	This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)											
15-8	μ Frame C-prog-mask	Split complete progress mask. This field is used by the host controller to record which split-completes have been executed.											
7-0	Status	<p>This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:</p> <table border="1"> <thead> <tr> <th>Status Bits</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>Active. Set by software to enable the execution of an isochronous split transaction by the host controller.</td> </tr> <tr> <td>6</td> <td>ERR. Set by the host controller when an ERR response is received from the companion controller.</td> </tr> <tr> <td>5</td> <td>Data buffer error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the host controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.</td> </tr> <tr> <td>4</td> <td>Babble detected. Set by the host controller during status update when "babble" is detected during the transaction generated by this descriptor.</td> </tr> </tbody> </table>	Status Bits	Definition	7	Active. Set by software to enable the execution of an isochronous split transaction by the host controller.	6	ERR. Set by the host controller when an ERR response is received from the companion controller.	5	Data buffer error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the host controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.	4	Babble detected. Set by the host controller during status update when "babble" is detected during the transaction generated by this descriptor.	
Status Bits	Definition												
7	Active. Set by software to enable the execution of an isochronous split transaction by the host controller.												
6	ERR. Set by the host controller when an ERR response is received from the companion controller.												
5	Data buffer error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the host controller will transmit an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.												
4	Babble detected. Set by the host controller during status update when "babble" is detected during the transaction generated by this descriptor.												

Table continues on the next page...

Table 16-66. siTD transfer status and control (continued)

Bits	Name	Description	
		3	Transaction error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, and so on). This bit will only be set for IN transactions.
		2	Missed microframe. The host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction.
		1	Split transaction state (SplitXstate). The bit encodings are: 0 Do start split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 1 Do complete split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
		0	Reserved, should be cleared. Bit reserved for future use and should be cleared.

16.5.4.4 siTD buffer pointer list (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer.

This structure supports one physical page cross. The most-significant 20 bits of each DWord in this section are the 4K (page) aligned buffer pointers. The least-significant 12 bits of each DWord are used as additional transfer state.

The table below describes the siTD buffer pointer page 0.

Table 16-67. siTD buffer pointer page 0 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 0)	Bits 31-12 are 4K page-aligned, physical memory addresses. These bits correspond to physical address bits 31-12 respectively. The field P specifies the current active pointer
11-0	Current Offset	The 12 least-significant bits of the Page 0 pointer is the current byte offset for the current page pointer (as selected with the page indicator bit (P field)). The host controller is not required to write this field back when the siTD is retired (Active bit transitioned from a one to a zero).

The table below describes the siTD buffer pointer page 1.

Table 16-68. siTD buffer pointer page 1 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 1)	Bits 31-12 are 4K page-aligned, physical memory addresses. These bits correspond to physical address bits 31-12 respectively. The field P specifies the current active pointer
11-5	-	Reserved, should be cleared.
4-3	TP	Transaction position. This field is used with T-count to determine whether to send all, first, middle, or last with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are:

Table continues on the next page...

Table 16-68. siTD buffer pointer page 1 (plus) (continued)

Bits	Name	Description
		00 All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01 Begin. This is the first data payload for a full-speed transaction that is greater than 188 bytes. 10 Mid. This is the middle payload for a full-speed OUT transaction that is larger than 188 bytes. 11 End. This is the last payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0	T-Count	Transaction count. Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

16.5.4.5 siTD back link pointer

DWord 6 of a siTD is simply another schedule link pointer.

This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

The table below describes the siTD back link pointer.

Table 16-69. siTD back link pointer

Bits	Name	Description
31-5	Back Pointer	A physical memory pointer to an siTD
4-1	-	Reserved, should be cleared. This field is reserved for future use. It should be cleared.
0	T	Terminate 0 siTD Back Pointer field is valid 1 siTD Back Pointer field is not valid

16.5.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head.

This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20,480 (5 x 4096) bytes. The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

The table below shows the queue element transfer descriptors.

Table 16-70. Queue element transfer descriptor (qTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	offset
Next qTD Pointer																														0000	T	0x00
Alternate Next qTD Pointer																														0000	T	0x04
dt ¹	Total Bytes to Transfer ¹				ioc	C_Page ¹	Cerr ¹	PID	Status ¹																				0x08			
Buffer Pointer (Page 0)					Current Offset ¹																								0x0C			
Buffer Pointer (Page 1)					0000_0000_0000																								0x10			
Buffer Pointer (Page 2)					0000_0000_0000																								0x14			
Buffer Pointer (Page 3)					0000_0000_0000																								0x18			
Buffer Pointer (Page 4)					0000_0000_0000																								0x1C			

1. Host controller read/write; all others read-only.

Queue element transfer descriptors must be aligned on 32-byte boundaries.

16.5.5.1 Next qTD pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The table below describes the qTD next element transfer pointer.

Table 16-71. qTD next element transfer pointer (DWord 0)

Bits	Name	Description
31-5	Next qTD Pointer	This field contains the physical memory address of the next qTD to be processed and corresponds to memory address signals [31:5], respectively.
4-1	-	Reserved, should be cleared. These bits are reserved and their value has no effect on operation.
0	T	Terminate. Indicates to the host controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid transfer element descriptor) 1 Pointer is invalid

16.5.5.2 Alternate next qTD pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next client buffer on short packet.

To be more explicit the host controller always uses this pointer when the current qTD is retired due to short packet. The table below describes the alternate qTD next element transfer pointer.

Table 16-72. qTD alternate next element transfer pointer (DWord 1)

Bits	Name	Description
31-5	Alternate next qTD pointer	This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4-1	-	Reserved, should be cleared. These bits are reserved and their value has no effect on operation.
0	T	Terminate. Indicates to the host controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid transfer element descriptor) 1 Pointer is invalid

16.5.5.3 qTD token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

Note that some of the field descriptions in the table below reference fields are defined in the queue head. See [Queue head](#), for more information on these fields.

Table 16-73. qTD token (DWord 2)

Bits	Name	Description
31	dt	Data toggle. This is the data toggle sequence bit. The use of this bit depends on the setting of the Data Toggle Control bit in the queue head.
30-16	Total bytes to transfer	Total bytes to transfer. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 x 4K (0x5000). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that total bytes to transfer be an even multiple of QH[Maximum Packet Length]. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QH[Maximum Packet Length]. Although it is possible to create a transfer up to 20K this assumes the page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K. Therefore, the maximum recommended transfer is 16K (0x4000).
15	ioc	Interrupt on complete. If this bit is set, the host controller should issue an interrupt at the next interrupt threshold when this qTD is completed.
14-12	C_Page	Current range. This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0x0 to 0x4. The host controller is not required to write this field back when the qTD is retired.

Table continues on the next page...

Table 16-73. qTD token (DWord 2) (continued)

Bits	Name	Description												
11-10	Cerr	Error counter. 2-bit down counter that keeps track of the number of consecutive errors detected while executing this qTD. If this field is programmed with a non-zero value during setup, the host controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the host controller marks the qTD inactive, sets the Halted bit to a one, and error status bit for the error that caused Cerr to decrement to zero. An interrupt is generated if USBINTR[UEE] is set. If the host controller driver (HCD) software programs this field to zero during setup, the host controller will not count errors for this qTD and there is no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.												
		<table border="1"> <thead> <tr> <th>Error</th><th>Decrement Counter</th></tr> </thead> <tbody> <tr> <td>Transaction Error</td><td>Yes</td></tr> <tr> <td>Data Buffer Error</td><td>No. Data buffer errors are host problems. They don't count against the device's retries. Note that software must not program Cerr to a value of zero when the EPS field is programmed with a value indicating a full- or low-speed device. This combination could result in undefined behavior.</td></tr> <tr> <td>Stalled</td><td>No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented</td></tr> <tr> <td>Babble Detected</td><td>No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented</td></tr> <tr> <td>No Error</td><td>No. If the EPS field indicates a HS device or the queue head is in the asynchronous schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset Cerr to extend the total number of errors for this transaction. For example, Cerr should be reset with maximum value (0b11) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 0b00.</td></tr> </tbody> </table>	Error	Decrement Counter	Transaction Error	Yes	Data Buffer Error	No. Data buffer errors are host problems. They don't count against the device's retries. Note that software must not program Cerr to a value of zero when the EPS field is programmed with a value indicating a full- or low-speed device. This combination could result in undefined behavior.	Stalled	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented	Babble Detected	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented	No Error	No. If the EPS field indicates a HS device or the queue head is in the asynchronous schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset Cerr to extend the total number of errors for this transaction. For example, Cerr should be reset with maximum value (0b11) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 0b00.
Error	Decrement Counter													
Transaction Error	Yes													
Data Buffer Error	No. Data buffer errors are host problems. They don't count against the device's retries. Note that software must not program Cerr to a value of zero when the EPS field is programmed with a value indicating a full- or low-speed device. This combination could result in undefined behavior.													
Stalled	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented													
Babble Detected	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented													
No Error	No. If the EPS field indicates a HS device or the queue head is in the asynchronous schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset Cerr to extend the total number of errors for this transaction. For example, Cerr should be reset with maximum value (0b11) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 0b00.													
9-8	PID Code	<p>This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:</p> <ul style="list-style-type: none"> 00 OUT Token generates token (E1H) 01 IN Token generates token (69H) 10 SETUP Token generates token (2DH) (undefined if endpoint is an Interrupt transfer type, for example, μFrame S-mask field in the queue head is non-zero.) 11 Reserved, should be cleared 												
7-0	Status	<p>This field is used by the host controller to communicate individual command execution states back to the host controller driver (HCD) software. This field contains the status of the last transaction performed on this qTD. The bit encodings are:</p> <table border="1"> <thead> <tr> <th>Bits</th><th>Status Field Description</th></tr> </thead> <tbody> <tr> <td>7</td><td>Active. Set by software to enable the execution of transactions by the host controller.</td></tr> <tr> <td>6</td><td>Halted. Set by the host controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set, the Active bit is also cleared.</td></tr> </tbody> </table>	Bits	Status Field Description	7	Active. Set by software to enable the execution of transactions by the host controller.	6	Halted. Set by the host controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set, the Active bit is also cleared.						
Bits	Status Field Description													
7	Active. Set by software to enable the execution of transactions by the host controller.													
6	Halted. Set by the host controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set, the Active bit is also cleared.													

Table continues on the next page...

Table 16-73. qTD token (DWord 2) (continued)

Bits	Name	Description
	5	Data buffer error. Set by the host controller during status update to indicate that the host controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the host controller will force a time-out condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble detected. Set by the host controller during status update when babble is detected during the transaction. In addition to setting this bit, the host controller also sets the Halted bit to a one. Since babble is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (time-out, CRC, bad PID). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed microframe. This bit is ignored unless the QH[EPS] field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	1	<p>Split transaction state (SplitXstate). This bit is ignored by the host controller unless the QH[EPS] field indicates a full- or low-speed endpoint. When a full- or low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are:</p> <p>0 Do start split. This value directs the host controller to issue a start split transaction to the endpoint.</p> <p>1 Do complete split. This value directs the host controller to issue a Complete split transaction to the endpoint.</p>
	0	<p>Ping state (P)/ERR. If the QH[EPS] field indicates a high-speed device and the PID Code indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are:</p> <p>0 Do OUT. This value directs the host controller to issue an OUT PID to the endpoint.</p> <p>1 Do Ping. This value directs the host controller to issue a PING PID to the endpoint.</p> <p>If the QH[EPS] field does not indicate a high-speed device, then this field is used as an error indicator bit. It is set by the host controller whenever a periodic split-transaction receives an ERR handshake.</p>

16.5.5.4 qTD buffer page pointer list

The last five DWords of a queue element transfer descriptor make up an array of physical memory address pointers.

These pointers reference the individual pages of a data buffer.

System software initializes the Current Offset field to the starting offset into the current page, where current page is selected with the value in the C_Page field.

The table below describes the qTD buffer pointer.

Table 16-74. qTD buffer pointer

Bits	Name	Description
31-12	Buffer pointer (page <i>n</i>)	Each element in the list is a 4K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction via the new buffer pointer.
11-0	Current offset (Page 0) / - (pages 1-4)	Reserved in all pointers except the first one (that is, page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the reserved fields are initialized to zeros.

16.5.6 Queue head

The table below shows the queue head structure.

Table 16-75. Queue head layout

31	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	15	14	1	1	1	1	9	8	7	6	5	4	3	2	1	0	offset									
0	9	8	7	6	5	4	3	2	1	0	9	8	7	6							00		Typ	T	0x00 ¹																
RL	C	Maximum Packet Length				H	dtc	EPS	EndPt			I	Device Address																	0x04 ¹											
Mult	Port Number		Hub Addr			μFrame C-mask				μFrame S-mask																0x08															
Current qTD Pointer ²															00000					0x0C																					
Next qTD Pointer ²															0000					T ²					0x10 ^{3,4}																
Alternate Next qTD Pointer ²															NakCnt ²					T ²					0x14 ^{3,4}																
dt ²	Total Bytes to Transfer ²					ioc ²	C_Page ²	Cerr ²	PID Code ²	Status ²										0x18 ^{3,4}																					
Buffer Pointer (Page 0) ²															Current Offset ²					0x1C ^{3,4}																					
Buffer Pointer (Page 1) ²															0000					C-prog-mask ²					0x20 ^{3,4}																
Buffer Pointer (Page 2) ²															S-bytes ²					FrameTag ²					0x24 ^{3,4}																
Buffer Pointer (Page 3) ²															0000_0000_0000					0x28 ³																					

Table continues on the next page...

Table 16-75. Queue head layout (continued)

Buffer Pointer (Page 4) ²	0000_0000_0000	0x2C ³
--------------------------------------	----------------	-------------------

1. Offsets 0x00 through 0x07 contain the static endpoint state.
2. Host controller read/write; all others read-only.
3. Offsets 0x10 through 0x2F contain the transfer overlay.
4. Offsets 0x10 through 0x27 contain the transfer results.

16.5.6.1 Queue head horizontal link pointer

The first DWord of a queue head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

The table below describes the queue head.

Table 16-76. Queue head DWord 0

Bits	Name	Description
31-5	QHLP	Queue head horizontal link pointer. This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4-3	-	Reserved, should be cleared. These bits must be written as zeros.
2-1	Typ	Indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate. 1 Last QH (pointer is invalid). 0 Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

16.5.6.2 Endpoint capabilities/characteristics

The second and third DWords of a queue head specify static information about the endpoint.

This information does not change over the lifetime of the endpoint. There are three types of information in this region:

- Endpoint characteristics. These are the USB endpoint characteristics, which include addressing, maximum packet size, and endpoint speed.
- Endpoint capabilities. These are adjustable parameters of the endpoint. They affect how the endpoint data stream is managed by the host controller.
- Split transaction characteristics. This data structure manages full- and low-speed data streams for bulk, control, and interrupt with split transactions to USB 2.0 Hub transaction translator. Additional fields exist for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following tables describe the endpoint characteristics.

Table 16-77. Endpoint characteristics: Queue head DWord 1

Bits	Name	Description
31-28	RL	Nak count reload. This field contains a value, which is used by the host controller to reload the Nak Counter field.
27	C	Control endpoint flag. If the QH[EPS] field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to a zero.
26-16	Maximum packet length	This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	H	Head of reclamation list flag. This bit is set by system software to mark a queue head as being the head of the reclamation list.
14	dtc	Data toggle control (DTC). Specifies where the host controller should get the initial data toggle on an overlay transition. 0 Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1 Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.
13-12	EPS	Endpoint speed. This is the speed of the associated endpoint. 00 Full-speed (12 Mbps) 01 Low-speed (1.5 Mbps) 10 High-speed (480 Mbps) 11 Reserved, should be cleared This field must not be modified by the host controller.
11-8	EndPt	Endpoint number. Selects the particular endpoint number on the device serving as the data source or sink.
7	I	Inactivate on next transaction. This bit is used by system software to request that the host controller set the Active bit to zero. This field is only valid when the queue head is in the periodic schedule and the EPS field indicates a full- or low-speed endpoint. Setting this bit when the queue head is in the asynchronous schedule or the EPS field indicates a high-speed device yields undefined results.
6-0	Device address	Selects the specific device serving as the data source or sink.

Table 16-78. Endpoint capabilities: Queue head DWord 2

Bits	Name	Description
31-30	Mult	High-bandwidth pipe multiplier. This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). 00 Reserved, should be cleared. A zero in this field yields undefined results. 01 One transaction to be issued for this endpoint per microframe 10 Two transactions to be issued for this endpoint per microframe 11 Three transactions to be issued for this endpoint per microframe
29-23	Port number	This field is ignored by the host controller unless the EPS field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 hub (for hub at device address Hub Addr below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22-16	Hub addr	This field is ignored by the host controller unless the EPS field indicates a full- or low-speed device. The value is the USB device address of the USB 2.0 hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15-8	μ Frame C-mask	This field is ignored by the host controller unless the EPS field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the Active and SplitX-state fields) is used to determine during which microframes the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the μ Frame C- mask field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	μ Frame S-mask	Interrupt schedule mask. This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the μ Frame S-mask field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the EPS field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the PID_Code field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the EPS field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

16.5.6.3 Transfer overlay

The nine DWords in this area represent a transaction working space for the host controller.

The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the queue head horizontal link pointer to the next queue head. The host controller never follows the next transfer queue element or alternate queue element pointers unless it is actively attempting to advance the queue. For the duration of the

transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a queue head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

The table below describes the current qTD link pointer.

Table 16-79. Current qTD link pointer

Bits	Name	Description
31-5	Current qTD pointer	Current element transaction descriptor link pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4-0	-	Reserved, should be cleared. These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a queue element transfer descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves an execution cache for the transfer.

The table below describes the host-controller rules for bits in overlay.

Table 16-80. Host-controller rules for bits in overlay (DWords 5, 6, 8, and 9)

DWord	QH Offset	Bits	Name	Description
5	0x14	4-1	NakCnt	<p>Nak counter-RW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from RL before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.</p> <p>NOTE: Note that in host mode, the NAK counter is decremented after receiving a NYET to an OUT Token. Thus, the NAK counter may be lower than expected.</p> <p>On P4080, Rev 3, the NAK counter is not decremented after receiving a NYET to an OUT Token. Thus, the NAK counter more accurately reflects the actual count.</p>
6	0x18	31	dt	Data toggle. The Data toggle control controls whether the host controller preserves this bit when an overlay operation is performed.
6	0x18	15	ioc	Interrupt on complete. The ioc control bit is always inherited from the source qTD when the overlay operation is performed.

Table continues on the next page...

Table 16-80. Host-controller rules for bits in overlay (DWords 5, 6, 8, and 9) (continued)

DWord	QH Offset	Bits	Name	Description
6	0x18	11-10	Cerr	Error counter. Copied from the qTD during the overlay and written back during queue advancement.
6	0x18	0	Status[0]	Ping state (P)/ERR. If the EPS field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	0x20	7-0	C-prog-mask	Split-transaction complete-split progress. Initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	0x24	11-5	S-bytes	Software must ensure that the S-bytes field in a qTD is zero before activating the qTD. Keeps track of the number of bytes sent or received during an IN or OUT split transaction.
9	0x24	4-0	FrameTag	Split-transaction frame tag. Initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.

16.5.7 Periodic frame span traversal node (FSTN)

The periodic frame span traversal node (FSTN) data structure, shown in the table below, is to be used only for managing full- and low-speed transactions that span a host-frame boundary.

Software must not use an FSTN in the asynchronous schedule. An FSTN in the asynchronous schedule results in undefined behavior.

Table 16-81. Frame span traversal node structure

31	3	2	2	2	2	2	2	2	2	2	2	1	1	1	1	15	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0	offset	
0	9	8	7	6	5	4	3	2	1	0	9	8	7	6		4	3	2	1	0	9	8	7	6	5	4	3	2	1	0				
Normal path link pointer																															-	Typ	T	0x00
Back path link pointer																															-	Typ	T	0x04

NOTE

The host controller performs only read operations to the FSTN data structure.

16.5.7.1 FSTN normal path pointer

The first DWord of an FSTN contains a link pointer to the next schedule object.

This object can be of any valid periodic schedule data type. The table below describes the FSTN normal path pointer.

Table 16-82. FSTN normal path pointer

Bits	Name	Description
31-5	NPLP	Normal path link pointer. Contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	-	Reserved, should be cleared. These bits must be written as 0s.
2-1	Typ	Indicates to the host controller whether the item referenced is a iTD/siTD, QH, or FSTN. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate. 0 Link pointer is valid. 1 Link pointer field is not valid.

16.5.7.2 FSTN back path link pointer

The second DWord of an FSTN node contains a link pointer to a queue head.

If the T-bit in this pointer is a zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set, then this FSTN is the Restore indicator. When the T-bit is a one, the host controller ignores the Typ field.

The table below describes the FSTN back path link pointer.

Table 16-83. FSTN back path link pointer

Bits	Name	Description
31-5	BPLP	Back path link pointer. Contains the address of a queue head. This field corresponds to memory address signals [31:5], respectively.
4-3	-	Reserved, should be cleared. These bits must be written as 0s.
2-1	Typ	Software must ensure this field is set to indicate the target data structure is a Queue Head (01). Any other value in this field yields undefined results.
0	T	Terminate. 0 Link pointer is valid (that is, the host controller may use bits 31-5 as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator. 1 Link pointer field is not valid (that is, the host controller must not use bits 31-5 as a valid memory address). This value also indicates that this FSTN is a Restore indicator.

16.6 Host operations

The general operational model for the USB DR module in host mode is defined by the Enhanced Host Controller Interface (EHCI) Specification. The EHCI specification describes the register-level interface for a host controller for the USB Revision 2.0. It includes a description of the hardware/software interface between system software and host controller hardware. Information concerning the initialization of the USB module is included in the following section; however, the full details of the EHCI specification are beyond the scope of this document.

16.6.1 Host controller initialization

After initial power-on or host controller reset (hardware or through USBCMD[RST]), all of the operational registers are at their default values.

To configure the external ULPI PHY, the following initialization sequence is required:

1. The UTMI PHY should remain disabled if the ULPI is being used.
3. Wait for 10 ms.

The user can proceed to the host controller initialization phase.

In order to initialize the USB DR module, software should perform the following steps:

1. Set the controller mode to host mode. Optionally set USBMODE[SDIS] (streaming disable)
- NOTE**
- Transitioning from device mode to host mode requires a host controller reset before modifying USBMODE.
2. Program the PTS field of the PORTSC register if using a non-ULPI PHY.
3. Set CONTROL[USB_EN].
4. Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
5. Write the base address of the periodic frame list to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the periodic frame list should have their T-Bits set.
6. Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn on the controller by setting the RS bit.

At this point, the USB DR module is up and running and the port registers begin reporting device connects. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled high-speed ports, but the schedules have not yet been enabled. The EHCI host controller will not transmit SOFs to enabled Full- or Low-speed ports.

In order to communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to USBCMD[ASE]. In order to communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to USBCMD[PSE]. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

16.6.2 Power port

The HCSPARAMS[PPC] bit indicates whether the USB 2.0 host controller has port power control.

When the PPC bit is set, the host controller supports port power switches. Each available switch has an output enable. PPE is controlled based on the state of the combination bits- PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bit. The Configured Flag and Port Power Control bits are always 1'b1 in Host Mode. The PPE always follows the state of Port Power (PP) bit that is, if PP is 0, PPE will be 0 and if PP is 1, PPE will be 1.

16.6.3 Reporting over-current

Host ports by definition are power providers on USB.

Whether the ports are considered high- or low-powered is a platform implementation issue. The EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic resides outside the DR logic. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- Over-current active bit (OCA) is set. When the over-current condition goes away, the OCA will transition from a one to a zero.

- Over-current change bit (OCC) is set. On every transition of OCA, the controller will set OCC to a one. Software sets OCC to a zero by writing a one to this bit.
- Port enabled/disabled bit (PE) is cleared. When this change bit gets set, USBSTS[PCI] (the port change detect bit) is set.
- Port power (PP) bit may optionally be cleared. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When OCC transitions from a zero to a one, the controller also sets USBSTS[PCI] to a one. In addition, if the Port Change Interrupt Enable bit, USBINTR[PCE], is a one, the controller issues an interrupt to the system. Refer to [Table 16-84](#) for summary of behavior for over-current detection when the controller is halted (suspended from a device component point of view).

16.6.4 Suspend/resume

The host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 hub.

Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely through software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software-initiated resumes are called Resume Events/Actions; bus-initiated resume events are called wake-up events. The classes of wakeup events are:

- Remote-wakeup enabled device asserts resume signaling. In similar kind to USB 2.0 hubs, when in host mode the host controller responds to explicit device resume signaling and wake up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the port control bits in the PORTSC register.

Selective suspend is a feature supported by the PORTSC register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the bus, it should suspend the enabled port, then shut off the controller by setting the USBCMD[RS] to a zero.

When a wake event occurs the system will resume operation and system software must set the RS bit to a one and resume the suspended port.

16.6.4.1 Port suspend/resume

System software places the USB into suspend mode by writing a one into the appropriate PORTSC Suspend bit.

Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is a one).

The host controller may evaluate the Suspend bit immediately or wait until a microframe or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several microframes of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on the suspended port by writing a one to PORTSC[FPR]. Software should not attempt to resume a port unless the port reports that it is in the suspended state. If system software sets PORTSC[FPR] when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 milliseconds after a port indicates that it is suspended (Suspend bit is a one) before initiating a port resume through PORTSC[FPR]. When PORTSC[FPR] is set, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then clears PORTSC[FPR]. When the host controller receives the write to transition PORTSC[FPR] to zero, it completes the resume sequence as defined in the USB specification, and clears both PORTSC[FPR] and PORTSC[SUSP]. Software-initiated port resumes do not affect the port change detect bit (USBSTS[PCI]) nor do they cause an interrupt if USBINTR[PCE] (port change interrupt enable) is a one. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 μ sec. The port's PORTSC[FPR] bit is set and USBSTS[PCI] is set. If USBINTR[PCE] is a one, the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 milliseconds), then terminates the resume sequence by clearing PORTSC[FPR] in the port. The host controller receives the write of zero to PORTSC[FPR], terminates the resume sequence and clears PORTSC[FPR] and PORTSC[SUSP]. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the SUSP and FPR bits are zero. Software must ensure that the host controller is running (that is, USBSTS[HCH] is a zero), before terminating a resume by clearing the port's PORTSC[FPR] bit. If HCH is a one when PORTSC[FPR] is cleared, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

The table below summarizes the wake-up events. Whenever a resume event is detected, USBSTS[PCI] is set. If USBINTR[PCE] (port change interrupt enable) is a one, the host controller also generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the USBSTS[PCI].

Table 16-84. Behavior during wake-up events

Port status and signaling type	Signaled port response	Device state	
		D0	Not D0
Port disabled, resume K-State received	No effect	N/A	N/A
Port suspended, Resume K-State received	Resume reflected downstream on signaled port. PORTSC[FPR] is set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's PORTSC[WKDS], is set. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect (CCS) and Enable (PE) status bits are cleared, and the Connect Change status bit (CSC) is set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's PORTSC[WKDS], is cleared. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect (CCS) and Enable (PE) status bits are cleared, and the Connect Change status bit (CSC) is set. USBSTS[PCI] is set.	[1], [3]	[3]
Port is not connected and the port's PORTSC[WKCN] bit is a one. A connect is detected.	PORTSC Connect Status (CCS) and Connect Status Change (CSC) bits are set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is not connected and the port's PORTSC[WKCN] bit is a zero. A connect is detected.	PORTSC Connect Status (CCS) and Connect Status Change (CSC) bits are set. USBSTS[PCI] is set.	[1], [3]	[3]
Port is connected and the port's PORTSC[WKOC] bit is a one. An over-current condition occurs.	PORTSC Over-current Active (OCA), Over-current Change (OCC) bits are set. If Port Enable/Disable bit (PE) is a one, it is cleared. USBSTS[PCI] is set	[1], [2]	[2]
Port is connected and the port's PORTSC[WKOC] bit is a zero. An over-current condition occurs.	PORTSC Over-current Active (OCA), Over-current Change (OCC) bits are set. If Port Enable/Disable bit (PE) is a one, it is cleared. USBSTS[PCI] is set.	[1], [3]	[3]

¹ Hardware interrupt issued if USBINTR[PCE] (port change interrupt enable) is set.

² PME# asserted if enabled (Note: PME Status must always be set).

³ PME# not asserted.

16.6.5 Schedule traversal rules

The host controller executes transactions for devices using a simple, shared-memory schedule.

The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware/software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the [Periodic Frame List Base Address \[host mode\] \(USB_PERIODICLISTBASE\)](#) register. The PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in [Host data structures](#). In each microframe, if the periodic schedule is enabled (see [Periodic schedule](#)) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the PERIODICLISTBASE and the FRINDEX registers (see the figure below). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set. When the host controller encounters a T-Bit set during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the microframe.

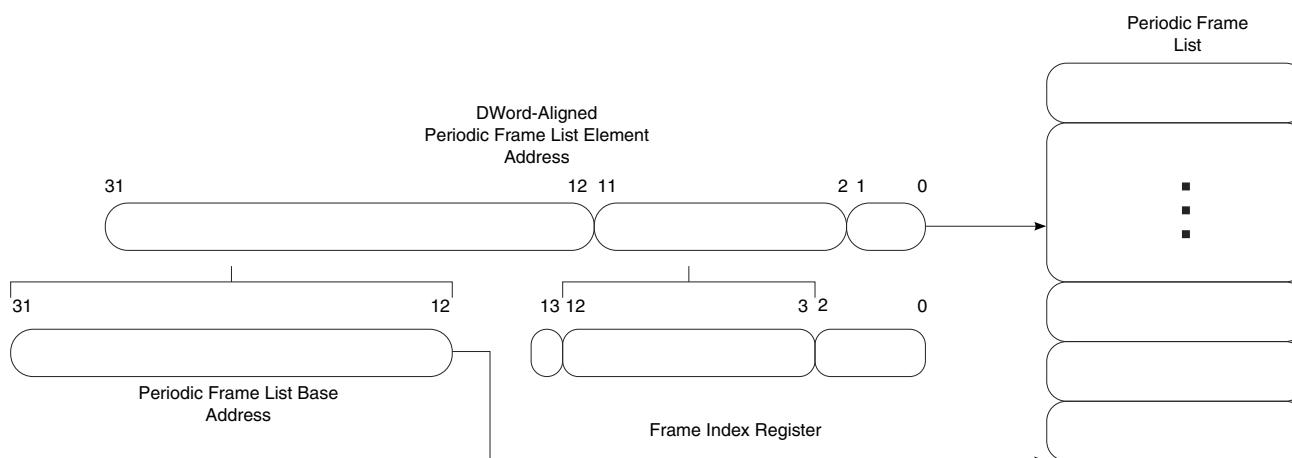


Figure 16-52. Derivation of pointer into frame list array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register ASYNCLISTADDR to access the asynchronous schedule, as shown in the figure below.

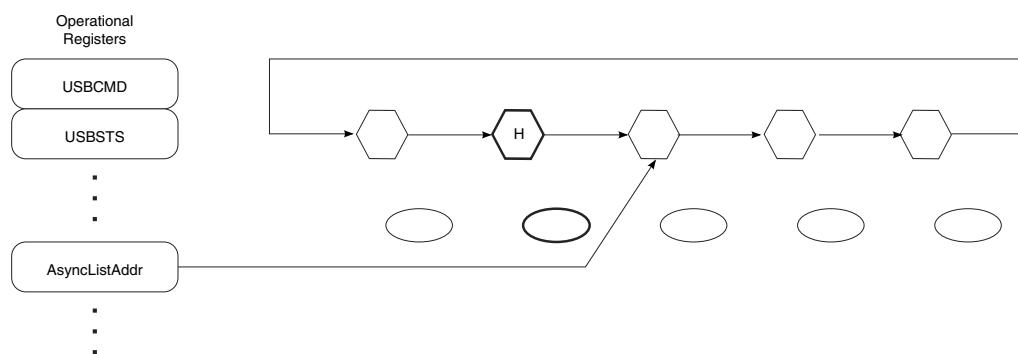


Figure 16-53. General format of asynchronous schedule list

The ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to a zero for queue heads in the asynchronous schedule.

16.6.6 Periodic schedule frame boundaries vs. bus frame boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(es) below USB 2.0 hubs be strictly aligned.

Super-imposed on this requirement is that USB 2.0 hubs manage full- and low-speed transactions via a microframe pipeline (see start- (SS) and complete- (CS) splits illustrated in the figure below). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

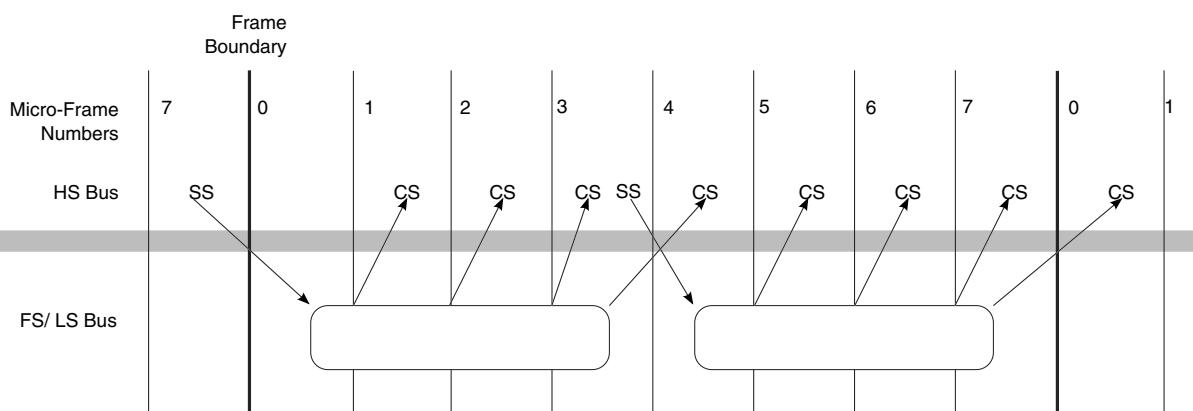


Figure 16-54. Frame boundary relationship between HS bus and FS/LS bus

The simple projection, as the figure above illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement a one microframe phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed via the Frame List Index Register (FRINDEX). Bits FRINDEX[2-0], represent the microframe number. The SOF value is coupled to the value of FRINDEX[13-3]. Both FRINDEX[13-3] and the SOF value are incremented based on FRINDEX[2-0]. It is required that the SOF value be delayed from the FRINDEX value by one microframe. The one microframe delay yields a host controller periodic schedule and bus frame boundary relationship as illustrated in the figure below. This adjustment allows software to trivially schedule the periodic start and complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface.

The figure below illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined. The host controller's view of the 1-millisecond boundaries is called H-Frames. The high-speed bus's view of the 1-millisecond boundaries is called B-Frames.

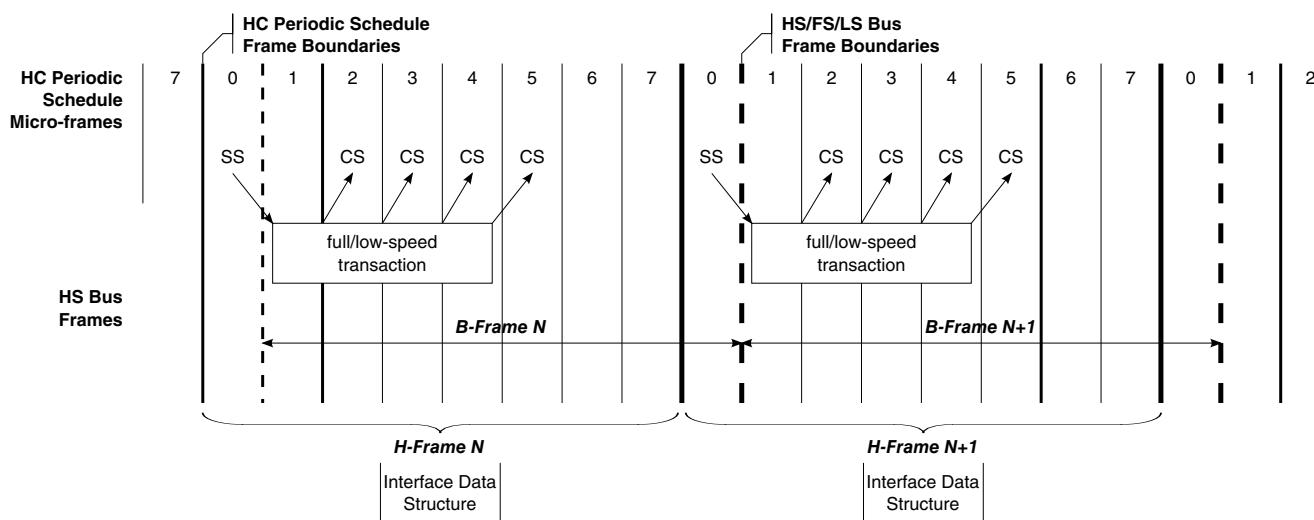


Figure 16-55. Relationship of periodic schedule frame boundaries to bus frame boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13-3]. Microframe numbers for the H-Frame are tracked by FRINDEX[2-0]. B-Frame boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Microframe numbers on the high-speed bus are only derived from the SOF token's frame number (that is, the high-speed bus will see eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (that is, B-Frames lag H-Frames by one microframe time) illustrated in the figure above. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 hub periodic pipeline. As described in [USB Frame Index \(USB_FRINDEX\)](#), the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13-3] by one microframe count. The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV. This lag behavior can be accomplished by incrementing FRINDEX[13-3] based on carry-out on the 7 to 0 increment of FRINDEX[2-0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2-0].

Software is allowed to write to FRINDEX. [USB Frame Index \(USB_FRINDEX\)](#), provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 16-85. Operation of FRINDEX and SOFV (SOF value register)

Current			Next		
FRINDEX[13-3]	SOFV	FRINDEX[2-0]	FRINDEX[13-3]	SOFV	FRINDEX[2-0]
N	N	111	N+1	N	000
N+1	N	000	N+1	N+1	001
N+1	N+1	001	N+1	N+1	010
N+1	N+1	010	N+1	N+1	011
N+1	N+1	011	N+1	N+1	100
N+1	N+1	100	N+1	N+1	101
N+1	N+1	101	N+1	N+1	110
N+1	N+1	110	N+1	N+1	111

16.6.7 Periodic schedule

The periodic schedule traversal is enabled or disabled through USBCMD[PSE] (periodic schedule enable).

If USBCMD[PSE] is cleared, then the host controller simply does not try to access the periodic frame list via the PERIODICLISTBASE register. Likewise, when USBCMD[PSE] is a one, then the host controller does use the PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to USBCMD[PSE] immediately. In order to eliminate conflicts with split transactions, the host controller evaluates USBCMD[PSE] only when FRINDEX[2-0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 0b000 microframe. These work items must be removed from the schedule before USBCMD[PSE] is cleared. USBSTS[PS] (periodic schedule status) indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by setting (or clearing) USBCMD[PSE]. Software then can poll USBSTS[PS] to determine when the periodic schedule has made the desired transition. Software must not modify USBCMD[PSE] unless the value of USBCMD[PSE] equals that of USBSTS[PS].

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions on the USB. The figure below illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/

siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

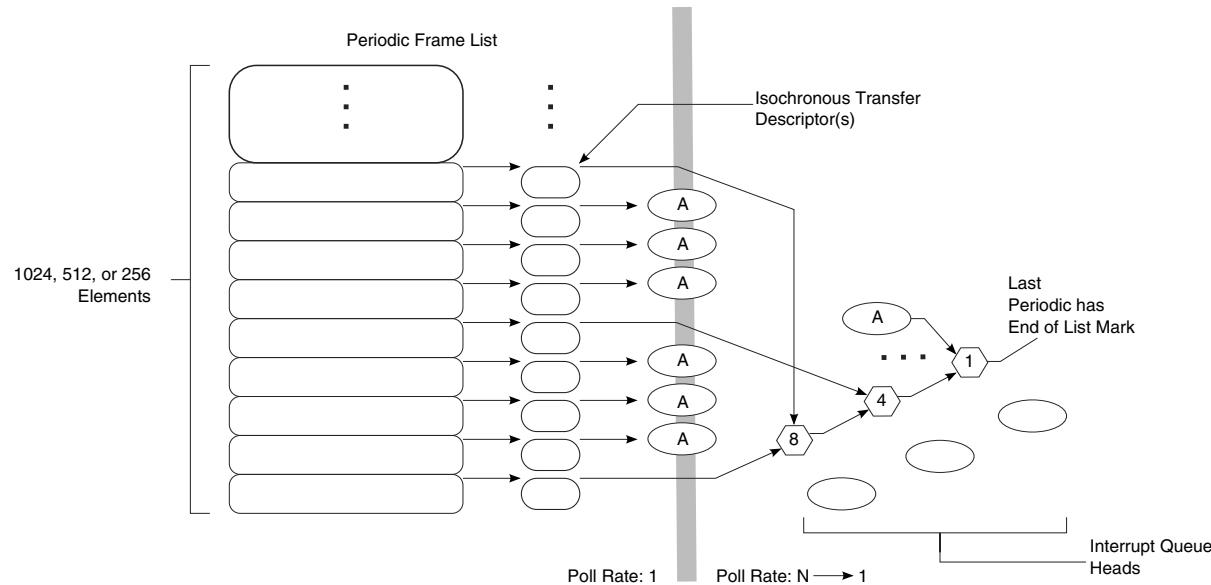


Figure 16-56. Example periodic schedule

16.6.8 Managing isochronous transfers using iTDs

The structure of an iTD is presented in isochronous (high-speed) transfer descriptor (iTID).

There are four distinct sections to an iTD:

- Next link pointer
 - This is the first field.
 - This field is for schedule linkage purposes only.
- Transaction description array
 - This is an eight-element array.
 - Each element represents control and status information for one microframe's worth of transactions for a single high-speed isochronous endpoint.
- Buffer page pointer array
 - This is a 7-element array of physical memory pointers to data buffers.
 - These are 4K aligned pointers to physical memory.
- Endpoint capabilities

- This area utilizes the unused low-order 12 bits of the buffer page pointer array.
- Its fields are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

16.6.8.1 Host controller operational model for iTDs

The host controller uses FRINDEX register bits 12-3 to index into the periodic frame list.

This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits 2-0. Each iTD can span 8 microframes worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits 2-0 to index into the transaction description array. When the first iTD in the periodic list is traversed after periodic schedule is enabled, the value of FRINDEX[2:0] may be other than 0, so the first transaction issued by the controller may be any of the eight available active transactions. If the active bit in the Status field of the indexed transaction description is cleared, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is a one the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is a 0, then the host controller will store Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (example: page 0 pointer) selected by the active transaction descriptions' PG (example value: 0b00) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When

this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (example: page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high-bandwidth pipes via the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current microframe. In other words, the Mult field represents a transaction count for the endpoint in the current microframe. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

For OUT transfers, the value of the Transaction *n* Length field represents the total bytes to be sent during the microframe. The Mult field must be set by software to be consistent with Transaction *n* Length and Maximum Packet Size. The host controller will send the bytes in Maximum Packet Sized portions. After each transaction, the host controller decrements its local copy of Transaction *n* Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction *n* Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction *n* Length field.

After all transactions for the endpoint have completed for the microframe, Transaction *n* Length contains the total bytes received. The following actions can occur:

- If the final value of Transaction *n* Length is less than the value of Maximum Packet Size, less data was received than was allowed for from the associated endpoint. This short packet condition does not set USBSTS[UI] (USB interrupt). The host controller does not detect this condition.
- If the device sends more than Transaction *n* Length or Maximum Packet Size bytes (whichever is less), the host controller sets the Babble Detected bit and clears the Active bit. Note, that the host controller does not update the iTD field Transaction *n* Length in this error scenario.
- If the Mult field is greater than one, the host controller automatically executes the value of Mult transactions. The host controller does not execute all Mult transactions in the following cases:

- The endpoint is an OUT and Transaction n Length goes to zero before all the Mult transactions have executed (ran out of data).
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed.

The end of microframe may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made; the result is written back to the iTD; and the host controller proceeds to processing the next microframe.

16.6.8.2 Software operational model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N microframes.

When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

The figure below illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is, the periodic frame list and a set of iTDs).

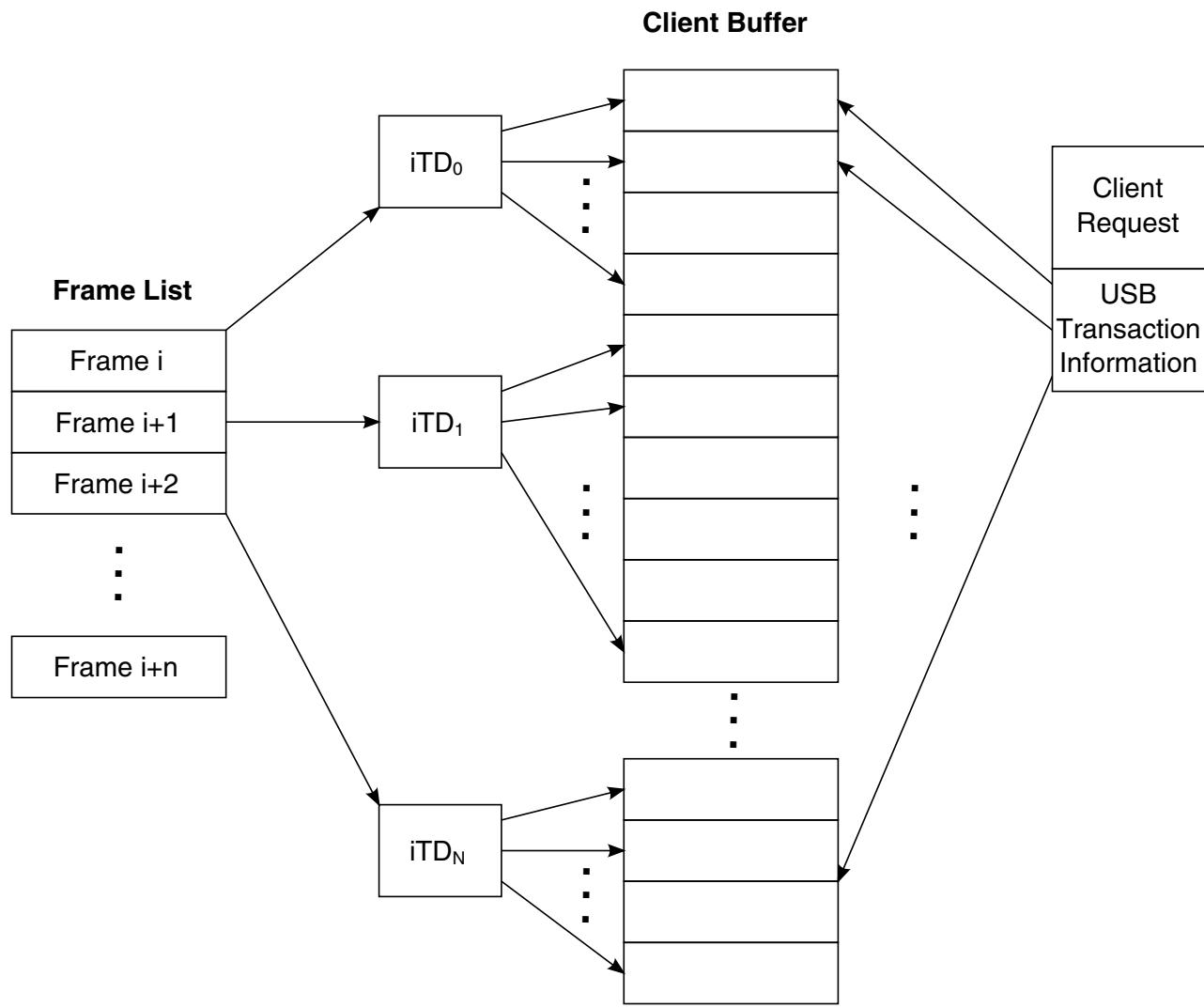


Figure 16-57. Example Association of iTDs to Client Request Buffer

On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one microframe's worth of transactions. The EHCI controller does not provide per transaction results within a microframe. It treats the per microframe transactions as a single logical transfer.

On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2-0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer will wrap a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to alias the page selector to page zero. USB 2.0 isochronous endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to a one.

16.6.8.2.1 Periodic scheduling threshold

The isochronous scheduling threshold field in the HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures.

It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 microframes worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 microframes. The three caching models are: no caching, microframe caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and microframe the host controller is currently executing. Of course, there is no information about where in the microframe the host controller is, so a constant uncertainty factor of one microframe has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the isochronous scheduling threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per microframe) but will always dump any accumulated schedule state at the end of the microframe. At the appropriate time relative to the beginning of every microframe, the host controller always begins schedule traversal from the frame list. Software can use the value of the FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 microframes in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the isochronous scheduling threshold field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 microframes). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the current microframe/frame (assume modulo 8 arithmetic in adding the constant 1 to the microframe number). For any current frame N, if the current microframe is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current microframe is 7, then software can add isochronous transactions to Frame N + 2.

Microframe caching is indicated with a non-zero value in the least-significant 3 bits of the isochronous scheduling threshold field. System software assumes the host controller caches one or more periodic data structures for the number of microframes indicated in the isochronous scheduling threshold field. For example, if the count value were 2, then the host controller keeps a window of two microframes worth of state (current microframe, plus the next) on chip. On each microframe boundary, the host controller releases the current microframe state and begins accumulating the next microframe state.

16.6.9 Asynchronous schedule

The asynchronous schedule traversal is enabled or disabled through USBCMD[ASE] (asynchronous schedule enable).

If USBCMD[ASE] is cleared, then the host controller simply does not try to access the asynchronous schedule via the ASYNCLISTADDR register. Likewise, if USBCMD[ASE] is set, the host controller does use the ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to USBCMD[ASE] are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the ASYNCLISTADDR register to get the next queue head.

USBSTS[AS] indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to USBCMD[ASE]. Software then can poll USBSTS[AS] to determine when the asynchronous schedule has made the desired transition. Software must not modify USBCMD[ASE] unless the value of USBCMD[ASE] equals that of the USBSTS[AS] (asynchronous schedule status).

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the ASYNCLISTADDR register. The default value of the ASYNCLISTADDR register after reset is undefined and the schedule is disabled when USBCMD[ASE] is cleared.

Software may only write this register with defined results when the schedule is disabled, for example, USBCMD[ASE] and the USBSTS[AS] are cleared. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting USBCMD[ASE]. The asynchronous schedule is actually enabled when USBSTS[AS] is set.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when one of the following events occurs:

- The end of a microframe occurs.
- The host controller detects an empty list condition
- The schedule has been disabled through USBCMD[ASE].

The queue heads in the asynchronous list are linked into a simple circular list as shown in [Figure 16-53](#). Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTID or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

16.6.9.1 Adding queue heads to asynchronous schedule

This is a software requirement section. There are two independent events for adding queue heads to the asynchronous schedule.

The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the ASYNCLISTADDR register, then enables the list by setting USBCMD[ASE] to a one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example qTD pointers have T-Bits set or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
--
-- Requirement: all inputs must be properly initialized.
--
-- pQHeadCurrent is a pointer to a queue head that is
-- already in the active list
-- pQueueHeadNew is a pointer to the queue head to be added
--
-- This algorithm links a new queue head into a existing
-- list
--
pQueueHeadNew.HorizontalPointer = pQueueHeadCurrent.HorizontalPointer
pQueueHeadCurrent.HorizontalPointer = physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

16.6.9.2 Removing queue heads from asynchronous schedule

This is a software requirement section.

There are two independent events for removing queue heads from the asynchronous schedule.

The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list. Software deactivates the asynchronous schedule by setting USBCMD[ASE] to a zero. Software can determine when the list is idle when USBSTS[AS] is cleared. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, and then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list using the following algorithm. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
--
-- Requirement: all inputs must be properly initialized.
--
-- pQHeadPrevious is a pointer to a queue head that
-- references the queue head to remove
-- pQHeadToUnlink is a pointer to the queue head to be
-- removed
-- pQheadNext is a pointer to a queue head still in the
-- schedule. Software provides this pointer with the
-- following strict rules:
-- if the host software is one queue head, then
-- pQHeadNext must be the same as
-- QueueheadToUnlink.HorizontalPointer. If the host
-- software is unlinking a consecutive series of
-- queue heads, QHeadNext must be set by software to
-- the queue head remaining in the schedule.
--
-- This algorithm unlinks a queue head from a circular list
--
pQueueHeadPrevious.HorizontalPointer = pQueueHeadToUnlink.HorizontalPointer
pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the H-bit set, it must select another queue head still linked into the schedule and set its H-bit. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host

controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (USBCMD[IAA]-interrupt on async advance doorbell) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (USBSTS[AAI]-interrupt on async advance) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit, it also clears the command bit. The third bit is an interrupt enable (USBINTR[AAE]-interrupt on async advance enable) that is matched with the status bit. If the status bit is set and the interrupt enable bit is set, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example where consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is, traversed beyond queue head (B) in this example).

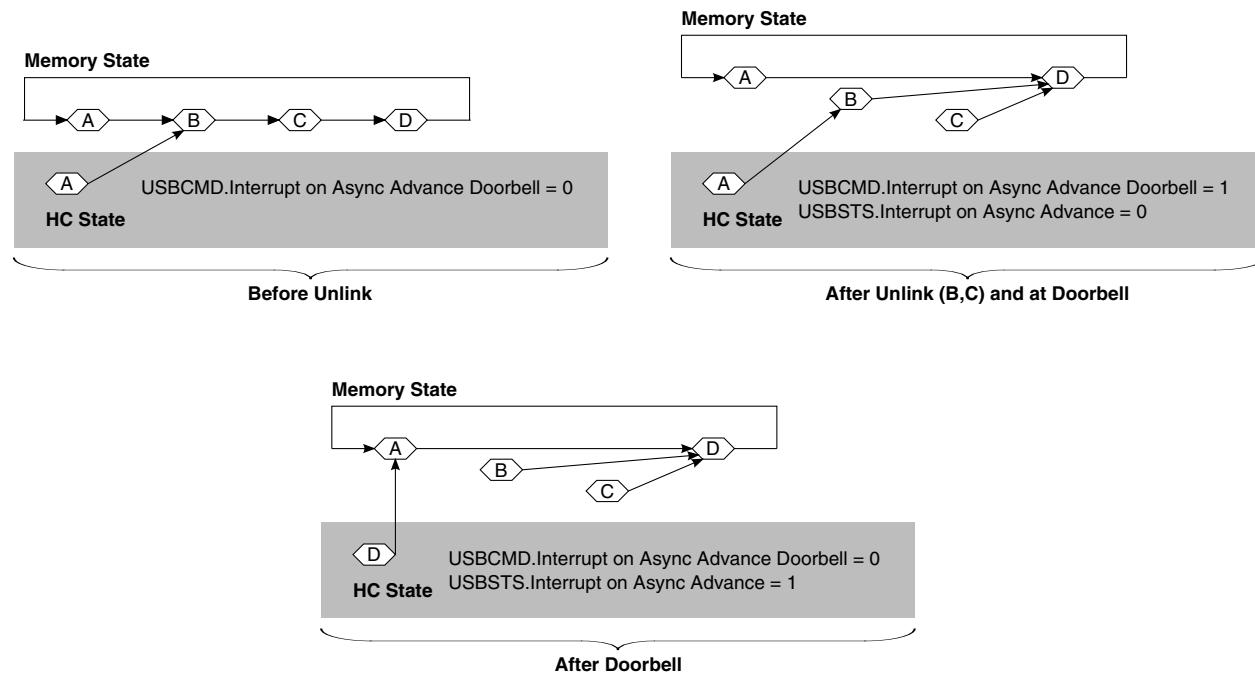


Figure 16-58. Generic queue head unlink scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting USBSTS[AAI].

Software may re-use the memory associated with the removed queue heads after it observes USBSTS[AAI] is set, following assertion of the doorbell. Software should acknowledge the interrupt on async advance status as indicated in the USBSTS register, before using the doorbell handshake again

16.6.9.3 Empty asynchronous schedule detection

EHCI uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see [Table 16-75](#)) defines an H-bit in the queue head, which allows software to mark a queue head as being the head of the reclaim list. Host controller also keeps a 1-bit flag in the USBSTS register (Reclamation) that is cleared when the host controller observes a queue head with the H-bit set. The reclamation flag in the status register is set when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see [Asynchronous schedule traversal: Start event](#)).

If the controller ever encounters an H-bit of one and a Reclamation bit of zero, the controller simply stops traversal of the asynchronous schedule.

The figure below shows an example illustrating the H-bit in a schedule.

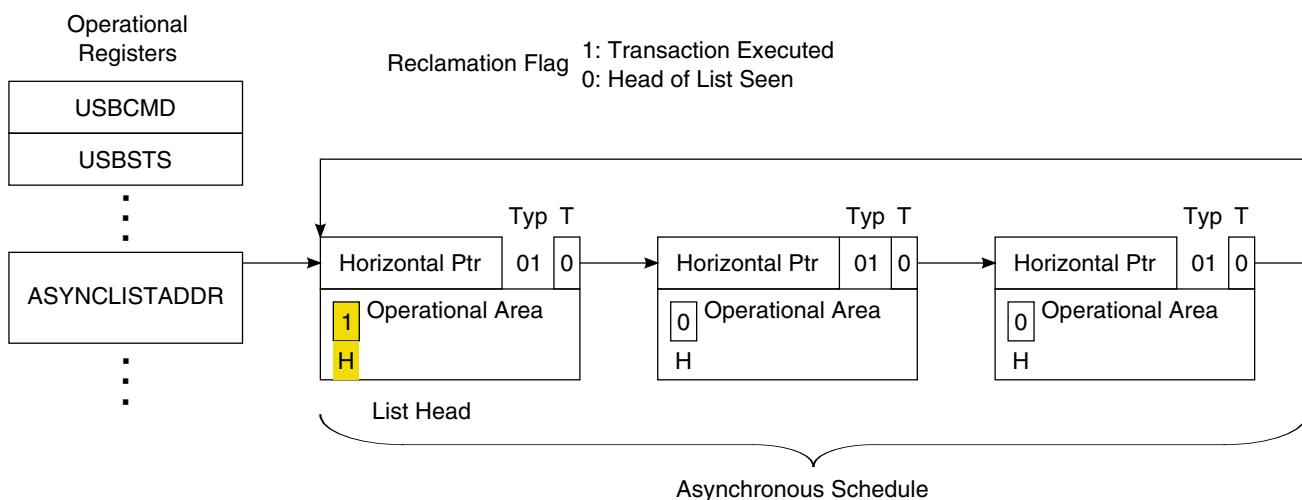


Figure 16-59. Asynchronous schedule list with annotation to mark head of list

16.6.9.4 Asynchronous schedule traversal: Start event

Once the host controller has idled itself using the empty schedule detection, it naturally activates and begins processing from the Periodic Schedule at the beginning of each microframe.

In addition, it may have idled itself early in a microframe. When this occurs (idles early in the microframe) the host controller must occasionally reactivate during the microframe and traverse the asynchronous schedule to determine whether any progress can be made. Asynchronous schedule Start Events are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the microframe is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state.

16.6.9.5 Reclamation status bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature depends on the proper management of the Reclamation bit (RCL) in the USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule. The host controller sets USBSTS[RCL] whenever an asynchronous schedule traversal Start Event occurs. USBSTS[RCL] is also set whenever the host controller executes a transaction while traversing the asynchronous schedule. The host controller clears USBSTS[RCL] whenever it finds a queue head with its H-bit set. Software should only set a queue head's H-bit if the queue head is in the asynchronous schedule. If software sets the H-bit in an interrupt queue head, the resulting behavior is undefined. The host controller may clear USBSTS[RCL] when executing from the periodic schedule.

16.6.10 Managing control/bulk/interrupt transfers via queue heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure defined in [Queue element transfer descriptor \(qTD\)](#).

One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed. Each qTD represents one or more bus transactions, which is defined in the context of the EHCI specification as a transfer.

The general processing model for the host controller's use of a queue head is simple:

- Read a queue head,
- Execute a transaction from the overlay area,
- Write back the results of the transaction to the overlay area
- Move to the next queue head.

If the host controller encounters errors during a transaction, the host controller will set one of the error reporting bits in the queue head's Status field. The Status field accumulates all errors encountered during the execution of a qTD (that is, the error bits in the queue head Status field are sticky until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions will occur for the endpoint and the host controller will not advance the queue.

16.6.10.1 Buffer pointer list use for data streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer.

The EHCI specification requires that the buffer associated with the transfer be virtually contiguous. This means that if the buffer spans more than one physical page, it must obey the following rules:

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4K chunk beyond the first page, each buffer portion matches to a full 4K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The figure below illustrates these requirements.

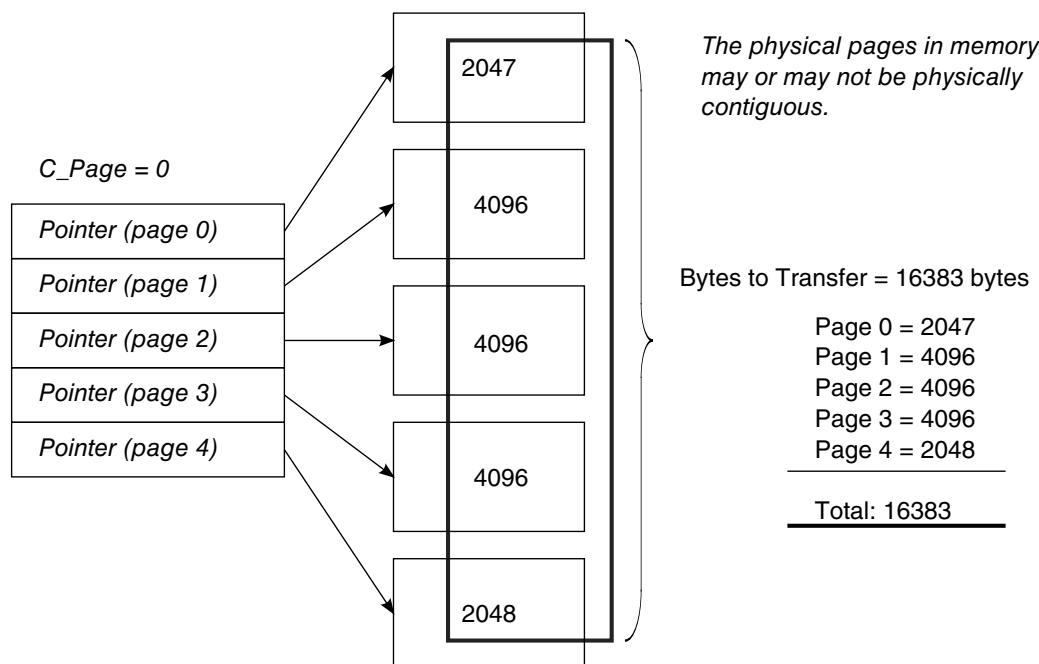


Figure 16-60. Example mapping of qTD buffer pointers to buffer pages

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the C_Page field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing C_Page and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the Bytes to Transfer field.

The figure above illustrates a nominal example of how System software would initialize the buffer pointers list and the C_Page field for a transfer size of 16383 bytes. C_Page is cleared. The upper 20-bits of Page 0 references the start of the physical page. Current Offset (the lower 12-bits of queue head Dword 7) holds the offset in the page for example, 2049 (for example, 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4K page.

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because C_Page is cleared) and concatenates the Current Offset field. The 512 bytes are moved during the transaction, the Current Offset and Total Bytes to Transfer are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller will increment C_Page (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and Current Offset has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is, C_Page) when necessary. There are three conditions for how the host controller handles C_Page.

- The current transaction does not span a page boundary. The value of C_Page is not adjusted by the host controller.
- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is, the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment C_Page before writing back status for the transaction.

Note that the only valid adjustment the host controller may make to C_Page is to increment by one.

16.6.10.2 Adding interrupt queue heads to the periodic schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head.

Queue heads are linked into the periodic schedule so they are polled at the appropriate rate. System software sets a bit in a queue head's S-Mask to indicate which microframe within a 1 millisecond period a transaction should be executed for the queue head.

Software must ensure that all queue heads in the periodic schedule have S-Mask set to a non-zero value. An S-mask with a zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and S-Mask values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in the table below.

Table 16-86. Example periodic reference patterns for interrupt transfers

Frame # reference sequence	Description
0, 2, 4, 6, 8, S- Mask = 0x01	A queue head for the bInterval of 2 milliseconds (16 microframes) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the S-Mask field in the queue head is set to 0x01, indicating that the transaction for the endpoint should be executed on the bus during microframe 0 of the frame.
0, 2, 4, 6, 8, ... S- Mask = 0x02	Another example of a queue head with a bInterval of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the S-Mask is set to 0x02 indicating that the transaction for the endpoint should be executed on the bus during microframe 1 of the frame.

16.6.10.3 Managing transfer complete interrupts from queue heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an Interrupt on Complete (IOC) bit set, or whenever a transfer (qTD) completes with a short packet.

If system software needs multiple qTDs to complete a client request (that is, like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

16.6.11 Ping control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping.

Ping is required for all USB 2.0 High-speed bulk and control endpoints. Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The Status field has a Ping State bit, which the host controller uses to determine the next actual PID it will use in the next transaction to the endpoint (see [Table 16-73](#)). The Ping State bit is only managed by the host controller for queue heads that meet all of the following criteria:

- The queue head is not an interrupt
- The EPS field equals High-Speed
- The PIDCode field equals OUT

The table below illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the *USB Specification, Revision 2.0* for detailed description on the Ping protocol.

Table 16-87. Ping control state transition table

		Event	
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr ¹	Do Ping
Do Ping	PING	Stall	N/C ²
Do OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping ³
Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr ¹	Do Ping
Do OUT	OUT	Stall	N/C ²

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example, Active cleared and Halt set). Software intervention is required to restart queue.
3. A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping.

The Ping State bit is described in [Table 16-73](#). The defined ping protocol allows the host to be imprecise on the initialization of the ping protocol (that is, start in Do OUT when there is uncertainty about the space in the device). The host controller manages the Ping State bit. System software sets the initial value in the queue head when it initializes a

queue head. The host controller preserves the Ping State bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the Ping State bit is preserved.

NOTE

For high-speed bulk and control endpoints, a host controller queries the high-speed device endpoint with a special ping token to determine whether the device has sufficient space for the next OUT transaction. The mechanism avoids using bus time to send data until the host controller knows that the endpoint has space for the data. If a timeout occurs after the data phase of an OUT transaction, the host controller fails to enter the ping state and instead retries the OUT token again.

The Ping flow control for the high-speed devices does not work under this condition. Therefore, some USB bandwidth could be wasted. However, NAK response to PING token or timeout is expected to be an unusual occurrence. A high-speed bulk/control endpoint must specify its maximum NAK rate in its endpoint descriptor. The endpoint is allowed to NAK at most one time each micro-frame period.

16.6.12 Split transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 hubs. This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below a USB 2.0 hub, utilizing the split transaction protocol. Refer to the USB 2.0 Specification for the complete definition of the split transaction protocol.

Full- and low-speed devices are enumerated identically as high-speed devices, but the transactions to the full- and low-speed endpoints use the split-transaction protocol on the high-speed bus.

The split transaction protocol is an encapsulation of (or wrapper around) the full- or low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 hub and transaction translator below which the full- or low-speed device is attached.

EHCI uses dedicated data structures for managing full-speed isochronous data streams. Control, Bulk and Interrupt are managed using the queuing data structures. The interface data structures need to be programmed with the device address and the transaction

translator number of the USB 2.0 hub operating as the low-/full-speed host controller for this link. The following sections describe the details of how the host controller processes and manages the split transaction protocol.

16.6.12.1 Split transactions for asynchronous transfers

A queue head in the asynchronous schedule with an EPS field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head.

All full-speed bulk and full-, low-speed control are managed via queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full-/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (SplitXState) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the Control Transfer Type (C) bit in the queue head to a one. If this is not a control transfer type endpoint, the C bit must be initialized by software to be a zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the C bit is a zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the C bit is a one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of *USB Specification, Revision 2.0* for details.

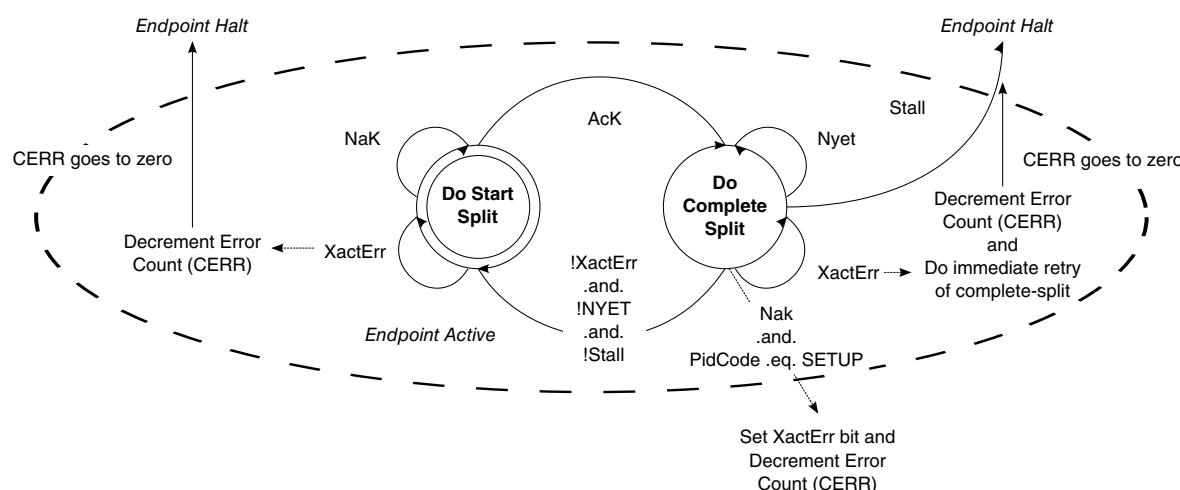


Figure 16-61. Host controller asynchronous schedule split-transaction state machine

16.6.12.1.1 Asynchronous-do-start-split

Do-start-split is the state which software must initialize a full- or low-speed asynchronous queue head.

This state is entered from the Do-Complete-Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the transaction translator. If the bus transaction completes without an error and PID Code indicates an IN or OUT transaction, then the host controller reloads the error counter (Cerr). If it is a successful bus transaction and the PID Code indicates a SETUP, the host controller will not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements Cerr and proceeds to the next queue head in the asynchronous schedule.

16.6.12.1.2 Asynchronous-do-complete-split

This state is entered from the Do-Start-Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's PID Code indicates an IN or OUT, the host controller reloads the error counter (Cerr). When a Nyet handshake is received for a complete-split bus transaction where the queue head's PID Code indicates a SETUP, the host controller must not adjust the value of Cerr.

Independent of PID Code, the following responses have the indicated effects:

- Transaction Error (XactErr). Timeout/data CRC failure. The error counter (Cerr) is decremented by one and the complete split transaction is immediately retried (if possible). If there is not enough time in the microframe to execute the retry, the host controller ensures that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the

retries must be immediate. When the host controller returns to the asynchronous schedule in the next microframe, the first transaction from the schedule will be the retry for this endpoint. If Cerr went to zero, the host controller halts the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the PID Code is a SETUP, then the Nak response is a protocol error. The XactErr status bit is set and the Cerr field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the halt bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the PID Code indicates an IN, then any of following response is expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is good, the host controller advances the state of the transfer (for example, moves the data pointer by the number of bytes received, decrements the BytesToTransfer field by the number of bytes received, and toggles the dt bit). The host controller then exits this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited.

If the PID Code indicates an OUT/SETUP, then any of following response is expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The current offset field is incremented by maximum packet length or bytes to transfer, whichever is less. The bytes to transfer field is decremented by the same amount and the data toggle bit (dt) is toggled. The host controller then exits this state.

Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue.

16.6.12.2 Split transaction interrupt

Split-transaction interrupt-IN/OUT endpoints are managed using the same data structures used for high-speed interrupt endpoints.

They both co-exist in the periodic schedule. Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer

framework. For example, for a high-speed endpoint, the host controller will visit a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the execution phase are different (that is, takes more than one bus transaction to complete), but the remainder of the operational framework is intact.

16.6.12.2.1 Split transaction scheduling mechanisms for interrupt

Full- and low-speed interrupt queue heads have an EPS field indicating full- or low-speed and have a non-zero S-mask field.

The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which microframes the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit microframes, or the data or response information in the pipeline is lost. The figure below illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and C_n labels indicate microframes where software can schedule start-splits and complete splits (respectively).

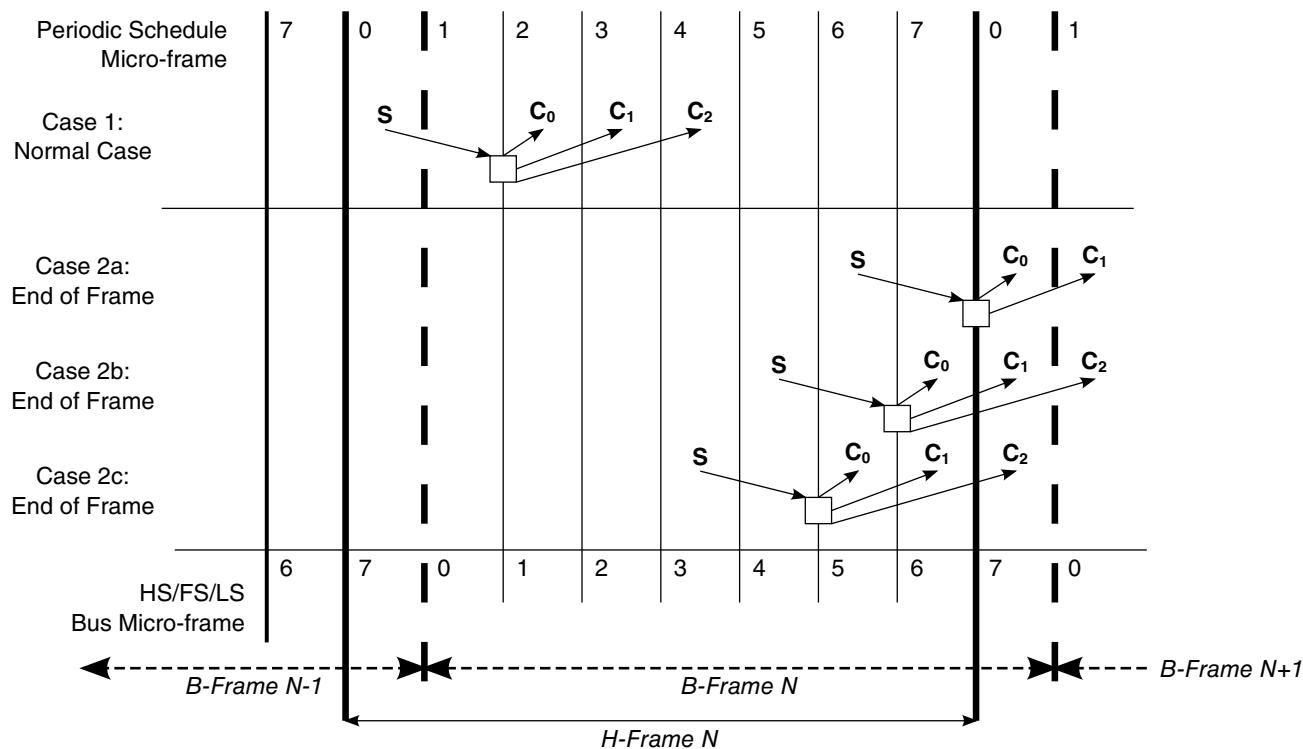


Figure 16-62. Split transaction, interrupt scheduling boundary conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (H-Frame in this case).
- Case 2a through Case 2c: The USB 2.0 hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the H-Frame boundary when the start-split is in microframe 4 or later. When this occurs, the H-Frame to B-Frame alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs.

The figure below illustrates the general layout of the periodic schedule.

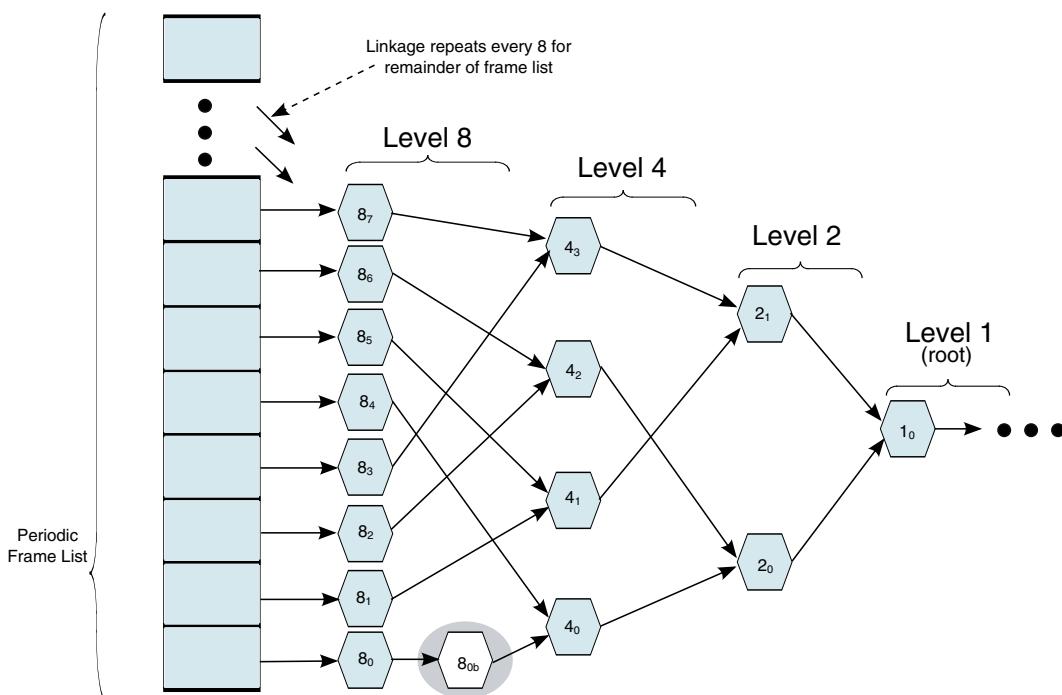


Figure 16-63. General structure of EHCI periodic schedule utilizing interrupt spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by spreading interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} were such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b} . It would then have to move 4_1 and everything linked after into the same path as 4_0 . This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. **Periodic frame span traversal node (FSTN)**, defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol.

- SplitXState. This is a single bit residing in the Status field of a queue head (Table 16-73). This bit is used to track the current state of the split transaction.

- Frame S-mask. This is a bit-field where-in system software sets a bit corresponding to the microframe (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to [Figure 16-62](#), case one, the S-mask would have a value of 0b0000_0001 indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Start, and the current microframe as indicated by FRINDEX[2-0] is 0, then execute a start-split transaction.
- Frame C-mask. This is a bit-field where system software sets one or more bits corresponding to the microframes (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to [Figure 16-62](#), case one, the C-mask would have a value of 0b0001_1100 indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Complete, and the current microframe as indicated by FRINDEX[2-0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between H-Frames and B-Frames is correctly performed when setting bits in S-mask and C-mask.

16.6.12.2.2 Host controller operational model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is, boundary cases 2a through 2c).

An FSTN is essentially a back pointer, similar in intent to the back pointer field in the siTD data structure.

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

There are four components to the use of FSTNs:

- FSTN data structure, defined in [Periodic frame span traversal node \(FSTN\)](#).
- A Save Place indicator; this is always an FSTN with its Back Path Link Pointer[T] bit cleared.
- A Restore indicator; this is always an FSTN with its Back Path Link Pointer[T] bit set.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during microframes 2 through 7 it simply follows the node's Normal Path Link Pointer to access the next schedule data structure. Note that the FSTN's Normal Path Link Pointer[T] bit may set, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a Save-Place FSTN in microframes 0 or 1, it saves the value of the Normal Path Link Pointer and sets an internal flag indicating that it is executing in Recovery Path mode. Recovery Path mode modifies the host controller's rules for how it traverses the schedule and limits which data structures are considered for execution of bus transactions. The host controller continues executing in Recovery Path mode until it encounters a Restore FSTN or it determines that it has reached the end of the microframe.

The rules for schedule traversal and limited execution while in Recovery Path mode are:

- Always follow the Normal Path Link Pointer when it encounters an FSTN that is a Save-Place indicator. The host controller must not recursively follow Save-Place FSTNs. Therefore, while executing in Recovery Path mode, it must never follow an FSTN's Back Path Link Pointer.
- Do not process an siTD or iTD data structure; simply follow its Next Link Pointer.
- Do not process a QH (Queue Head) whose EPS field indicates a high-speed device; simply follow its Horizontal Link Pointer.
- When a QH's EPS field indicates a Full/Low-speed device, the host controller only considers it for execution if its SplitXState is DoComplete (note: this applies whether the PID Code indicates an IN or an OUT). Refer to the *EHCI Specification* for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in Recovery Path mode. Refer to the *EHCI Specification* for special handling when in Recovery Path mode.
- Stop traversing the recovery path when it encounters an FSTN that is a Restore indicator. The host controller unconditionally uses the saved value of the Save-Place FSTN's Normal Path Link Pointer when returning to the normal path traversal. The host controller must clear the context of executing a Recovery Path when it restores schedule traversal to the Save-Place FSTN's Normal Path Link Pointer.

If the host controller determines that there is not enough time left in the microframe to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive microframe, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the figure below.

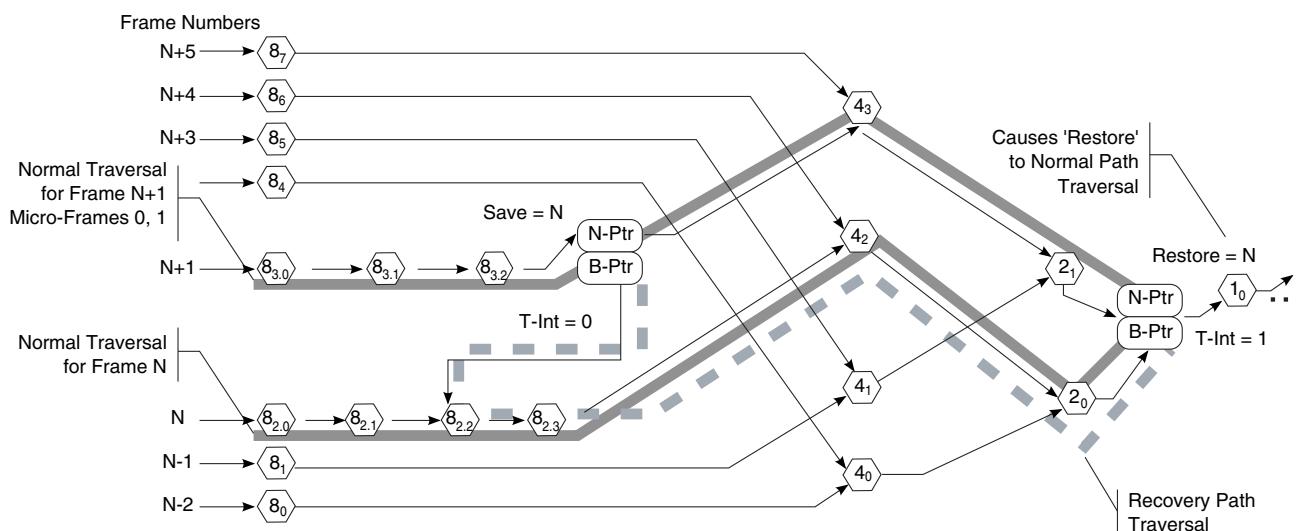


Figure 16-64. Example host controller traversal of recovery path via FSTNs

In frame N (microframes 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the Normal Path Link Pointers in any FSTNs it encounters. This is because the host controller has not yet encountered a Save-Place FSTN so it is not executing in Recovery Path mode. When it encounters the Restore FSTN, (Restore-N), during microframes 0 and 1, it uses Restore-N. Normal Path Link Pointer to traverse to the next data structure (that is, normal schedule traversal). This is because the host controller must use a Restore FSTN's Normal Path Link Pointer when not executing in a Recovery-Path mode. The nodes traversed during frame N include: {8_{2.0}, 8_{2.1}, 8_{2.2}, 8_{2.3}, 4₂, 2₀, Restore-N, 1₀ ...}.

In frame N+1 (microframes 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N. Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in Recovery Path mode (the recovery path is annotated in the figure above with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set (definition of a Restore indicator), so the host controller exits Recovery Path mode by clearing the internal Recovery Path mode flag and commences (restores) schedule traversal using the saved value of the Save-Place FSTN's Normal Path Link Pointer (for example, Save-N.Normal Path Link Pointer). The nodes traversed during these microframes include: {8_{3.0}, 8_{3.1}, 8_{3.2}, Save-A, 8_{2.2}, 8_{2.3}, 4₂, 2₀, Restore-N, 4₃, 2₁, Restore-N, 1₀ ...}.

In frame N+1 (microframes 2-7), when the host controller encounters Save-Path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these microframes include: {8_{3.0}, 8_{3.1}, 8_{3.2}, Save-A, 4₃, 2₁, Restore-N, 1₀ ...}.

16.6.12.2.3 Software operational model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse.

When using FSTNs, system software must adhere to the following rules:

- Each Save-Place indicator requires a matching Restore indicator.

The Save-Place indicator is an FSTN with a valid Back Path Link Pointer and T-bit equal to zero. Note that Back Path Link Pointer[Typ] field must be set to indicate the referenced data structure is a queue head. The Restore indicator is an FSTN with its Back Path Link Pointer[T] bit set.

A Restore FSTN may be matched to one or more Save-Place FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a Restore FSTN at the beginning of this list in order to match all possible Save-Place FSTNs.

- If the schedule does not have elements linked at a poll-rate level of one, and one or more Save-Place FSTNs are used, then System Software must ensure the Restore FSTN's Normal Path Link Pointer's T-bit is set, as this is used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a Restore FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that Recovery Path mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A Save-Place FSTN's Back Path Link Pointer must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list location. In other words, if the Save-Place FSTN is reachable from frame list offset N, then the FSTN's Back Path Link Pointer must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one Save-Place FSTN reachable in any single frame. Note there are times when two (or more, depending on the implementation) could exist as full-/low-speed footprints change with bandwidth adjustments. This could

occur, for example when a bandwidth rebalance causes system software to move the Save-Place FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

16.6.12.2.4 Tracking split transaction progress for interrupt transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost.

For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 hub and into the system before it expires from the transaction translator pipeline. When a lost data condition is detected, the queue is halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- C-prog-mask. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the transaction translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. C-prog-mask is a simple bit-vector that the host controller sets one of the C-prog-mask bits for each complete-split executed. The bit position is determined by the microframe number in which the complete-split was executed. The host controller always checks C-prog-mask before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- FrameTag. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (H-Frame number) when the next complete split must be executed.
- S-bytes. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The S-bytes field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

16.6.12.2.5 Split transaction execution state machine for interrupt

In the following section, all references to microframe are in the context of a microframe within an H-Frame.

As with asynchronous full- and low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- cMicroFrameBit. This is a single-bit encoding of the current microframe number. It is an eight-bit value calculated by the host controller at the beginning of every microframe. It is calculated from the three least significant bits of the FRINDEX register (that is, $cMicroFrameBit = (1 \text{ shifted-left}(FRINDEX[2-0]))$). The cMicroFrameBit has at most one bit asserted, which always corresponds to the current microframe number. For example, if the current microframe is 0, then cMicroFrameBit will equal 0b0000_0001.

The variable cMicroFrameBit is used to compare against the S-mask and C-mask fields to determine whether the queue head is marked for a start- or complete-split transaction for the current microframe.

The figure below illustrates how a complete interrupt split transaction is managed. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the SplitXState is at Do_Start and the single bit in cMicroFrameBit has a corresponding bit active in QH[S-mask]. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there is more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set in the QH[C-mask] field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the three-strikes-rule (for example, after the host tries the same transaction three times, and each encounters an error, the host controller stops retrying the bus transaction and halts the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

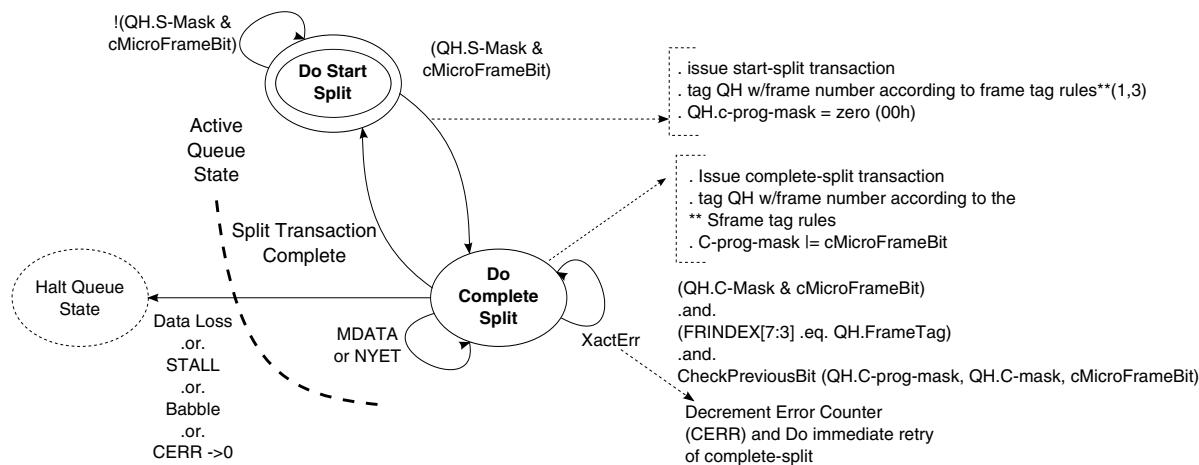


Figure 16-65. Split transaction state machine for interrupt

16.6.12.2.6 Periodic interrupt-do-start-split

This is the state when software must initialize a full- or low-speed interrupt queue head StartXState bit. This state is entered from the Do_Complete Split state only after the split transaction is complete.

This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, and so on).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see [Periodic interrupt-do-complete-split](#), for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the execute transaction state), bit-wise ANDs QH[S-mask] with cMicroFrameBit to determine whether to execute a start-split. If the result is non-zero, then the host controller issues a start-split transaction. If the PID Code field indicates an IN transaction, the host controller must zero-out the QH[S-bytes] field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the

complete-split phase of the split transaction. Specifically, it records the expected frame number into QH[FrameTag] field, sets C-prog-mask to zero (0x00), and exits this state. Note that the host controller must not adjust the value of Cerr as a result of completion of a start-split transaction.

16.6.12.2.7 Periodic interrupt-do-complete-split

This state is entered unconditionally from the Do-Start-Split state after a start-split transaction is executed on the bus.

Each time the host controller visits a queue head in this state (once within the execute transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. cMicroFrameBit is bit-wise ANDed with QH[C-mask] field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this microframe.
- Test B. QH[FrameTag] is compared with the current contents of FRINDEX[7-3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)

Begin

-- Return values:
-- TRUE - no error
-- FALSE - error
-- 

Boolean rvalue = TRUE;

previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous microframe. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.

If (previousBit bitAND QH.C-mask) then
```

```
If not(previousBit bitAND QH.C-prog-mask) then
    rvalue = FALSE;
End if
End If

-- If the C-prog-mask already has a one in this bit position,
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that is detectable here
-- should not allow
-- a transaction to be executed.

If (cMicroFrameBit bitAND QH.C-prog-mask) then
    rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this microframe. Note this is the same test performed in the Do Start Split state. Whenever it evaluates to TRUE and the controller is NOT processing in the context of a Recovery Path mode, it means a start-split should occur in this microframe. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (Test A and B and C and not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates QH[C-prog-mask] by bit-ORing with cMicroFrameBit. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets QH[FrameTag] to the expected H-Frame number. The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the Cerr will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last)

On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction

translator must have responded to all the complete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.

The test for whether this is the Last complete split can be performed by XOR QH[C-mask] with QH[C-prog-mask]. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the XactErr status bit is set and the Cerr field is decremented.

- NYET (and not Last)

See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for C-prog-mask and FrameTag) and stay in this state. The host controller must not adjust Cerr on this response.

- Transaction Error (XactErr). Timeout, data CRC failure, and so on

The Cerr field is decremented and the XactErr bit in the Status field is set. The complete split transaction is immediately retried (if Cerr is non-zero). If there is not enough time in the microframe to complete the retry and the endpoint is an IN, or Cerr is decremented to a zero from a one, the queue is halted. If there is not enough time in the microframe to complete the retry and the endpoint is an OUT and Cerr is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section on full- and low-speed interrupts) in the *USB Specification Revision 2.0* for detailed requirements on why these errors must be immediately retried.

- ACK

This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The Current Offset field is incremented by Maximum Packet Length or Bytes to Transfer, whichever is less. The field Bytes To Transfer is decremented by the same amount. And the data toggle bit (dt) is toggled. The host controller will then exit this state for this queue head. The host controller must reload Cerr with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue.

- MDATA

This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in QH[S-bytes]. The host controller must not adjust Cerr on this response.

- DATA0/1

This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in QH[S-bytes]. The state of the transfer is advanced by the result and the host controller exits this state for this queue head.

Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.

- NAK

The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload Cerr with maximum value on this response.

- ERR

There was an error during the full- or low-speed transaction. The ERR status bit is set, Cerr is decremented, the state of the transfer is not advanced, and this state is exited.

- STALL

The queue is halted (an exit condition of the Execute Transaction state). The status field bits: Active bit is cleared and the Halted bit is set and the qTD is retired.

Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. The table below lists the possible combinations and the appropriate action.

Table 16-88. Interrupt IN/OUT do complete split state execution criteria

Condition	Action	Description
not(A)	Ignore QHD	Neither a start nor complete-split is scheduled for the current microframe. Host controller should continue walking the schedule.
not(D)		

Table continues on the next page...

Table 16-88. Interrupt IN/OUT do complete split state execution criteria (continued)

Condition	Action	Description
A not(C)	If PIDCode = IN Halt QHDIf PIDCode = OUT Retry start-split	Progress bit check failed. This means a complete-split has been missed. There is the possibility of lost data. If PID Code is an IN, then the queue head must be halted. If PID Code is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect Cerr. In either case, set the Missed Microframe bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	QH.FrameTag test failed. This means that exactly one or more H-Frames have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If PID Code is an IN, then the queue head must be halted. If PID Code is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect Cerr. In either case, set the Missed Microframe bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHDIf PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If PID Code is an IN, then the Queue head must be halted. If PID Code is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect Cerr. In either case, set the Missed Microframe bit in the status field to a one. Note that when executing in the context of a Recovery Path mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a Recovery Path mode.

16.6.12.2.8 Managing the QH[FrameTag] field

The QH[FrameTag] field in a queue head is completely managed by the host controller.

The rules for setting QH[FrameTag] are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of FRINDEX[2-0] is 6, QH[FrameTag] is set to FRINDEX[7-3] + 1. This accommodates split transactions whose start-split and complete-splits are in different H-Frames (case 2a, see [Figure 16-62](#)).
- Rule 2: If the current value of FRINDEX[2-0] is 7, QH[FrameTag] is set to FRINDEX[7-3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c in [Figure 16-62](#).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of FRINDEX[2-0] is not 6, or currently in Do Complete Split and the current value of (FRINDEX[2-0]) is not 7, FrameTag is set to FRINDEX[7-3]. This accommodates all other cases in [Figure 16-62](#).

16.6.12.2.9 Rebalancing the periodic schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation.

This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that system software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the host controller provides a simple assist to system software. System software sets the Inactivate-on-next-Transaction (I) bit to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software then waits for the host controller to observe the I-bit is set and transitions the Active bit to a zero. The rules for how and when the host controller clears the Active bit are:

- If the Active bit is cleared, no action is taken. The host controller does not attempt to advance the queue when the I-bit is set.
- If the Active bit is set and the SplitXState is DoStart (regardless of the value of S-mask), the host controller simply clears the Active bit. The host controller is not required to write the transfer state back to the current qTD. Note that if the S-mask indicates that a start-split is scheduled for the current microframe, the host controller must not issue the start-split bus transaction; it must clear the Active bit.

System software must save transfer state before setting the I-bit. This is required so that it can correctly determine what transfer progress (if any) occurred after the I-bit was set and the host controller executed its final bus-transaction and cleared the Active bit.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since the Active bit and the I-bit cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped using the I-bit.

1. Set the Halted bit, then
2. Clear the I-bit, then
3. Set the Active bit and clear the Halted bit in the same write.

Setting the Halted bit inhibits the host controller from attempting to advance the queue between the time the I-bit is cleared and the Active bit is set.

16.6.12.3 Split transaction isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB 2.0 hub.

The host controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (see [Managing isochronous transfers using iTDs](#), for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

16.6.12.3.1 Split transaction scheduling mechanisms for isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline.

As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which microframes the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in [Split transaction scheduling mechanisms for interrupt](#), apply.

Figure 16-66 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The S_n and C_n labels indicate microframes where software can schedule start- and complete-splits (respectively). The H-Frame boundaries are marked with a large, solid bold vertical line. The B-Frame boundaries are marked with a large, bold, dashed line. The figure below illustrates the relationship of an siTD to the H-Frame.

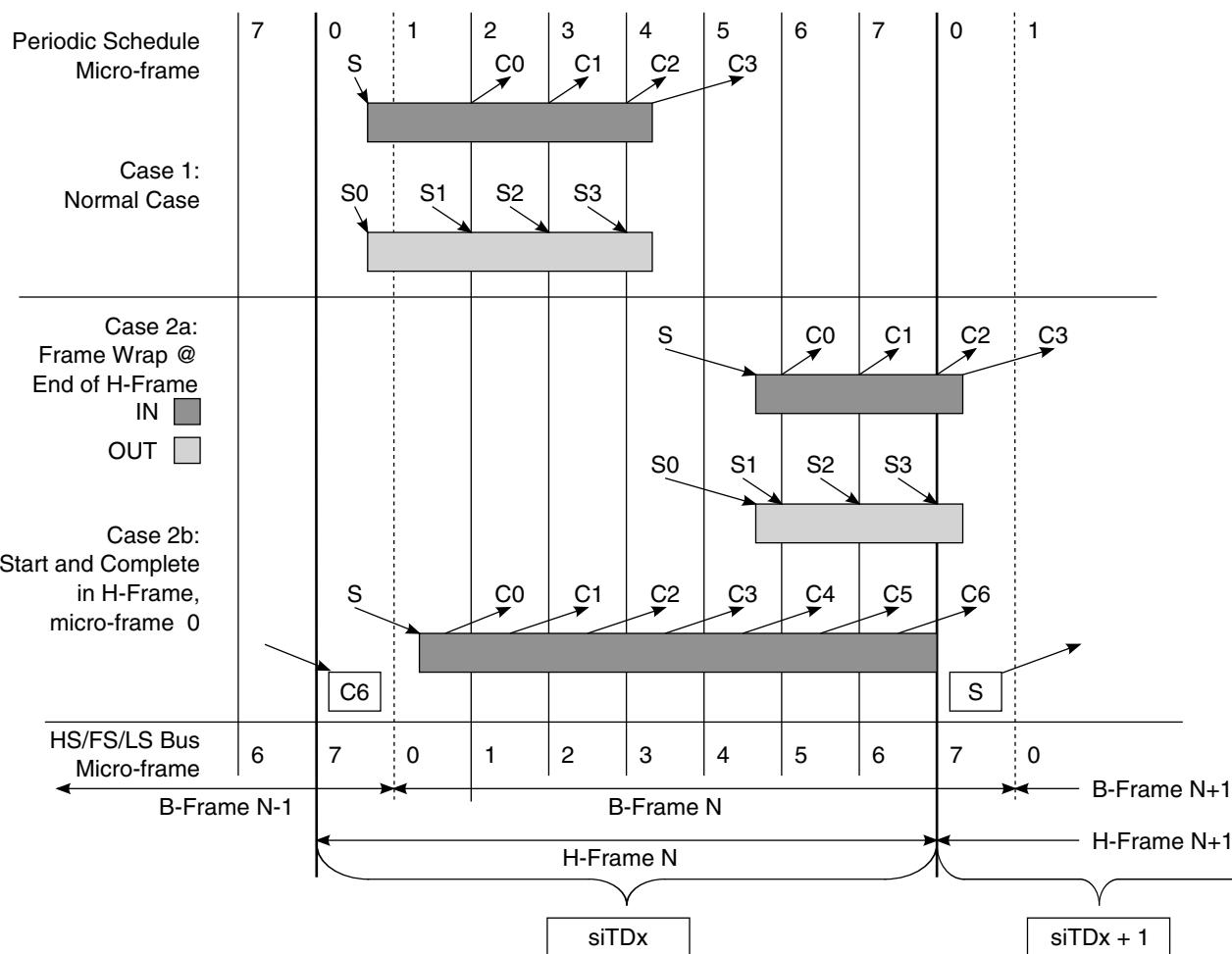


Figure 16-66. Split transaction, isochronous scheduling boundary conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to N complete-splits. The scheduling boundary cases are:

- Case 1: The entire split transaction is completely bounded by an H-Frame. For example, the start-splits and complete-splits are all scheduled to occur in the same H-Frame.
- Case 2a: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an H-Frame boundary. This can only occur when the split transaction has the possibility of moving data in B-Frame, microframes 6 or 7 (H-Frame microframe 7 or 0). When an H-Frame boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(for example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).

Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer.

Software must never schedule full-speed isochronous OUTs across an H-Frame boundary.

- Case 2b: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same microframe. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol:

- SplitXState

This is a single bit residing in the Status field of an siTD (see [Table 16-66](#)). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in [Split transaction execution state machine for isochronous](#).

- Frame S-mask

This is a bit-field wherein system software sets a bit corresponding to the microframe (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit. For example, referring to the IN example in [Figure 16-66](#), case 1, the S-mask would have a value of 0b0000_0001 indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Start Split, and the current microframe as indicated by FRINDEX[2-0] is 0, then execute a start-split transaction.

- Frame C-mask

This is a bit-field where system software sets one or more bits corresponding to the microframes (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit. For example, referring to the IN example in [Figure 16-66](#), case 1, the C-mask would have a value of 0b 0011_1100 indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Complete Split, and the current microframe as indicated by FRINDEX[2-0] is 2, 3, 4, or 5, then execute a complete-split transaction.

- Back Pointer

This field in a siTD is used to complete an IN split-transaction using the previous H-Frame's siTD. This is only used when the scheduling of the complete-splits span an H-Frame boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN and OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the H-Frame boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. The figure below illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the B-Frames (HS/FS/LS Bus) and the H-Frames. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each H-Frame corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

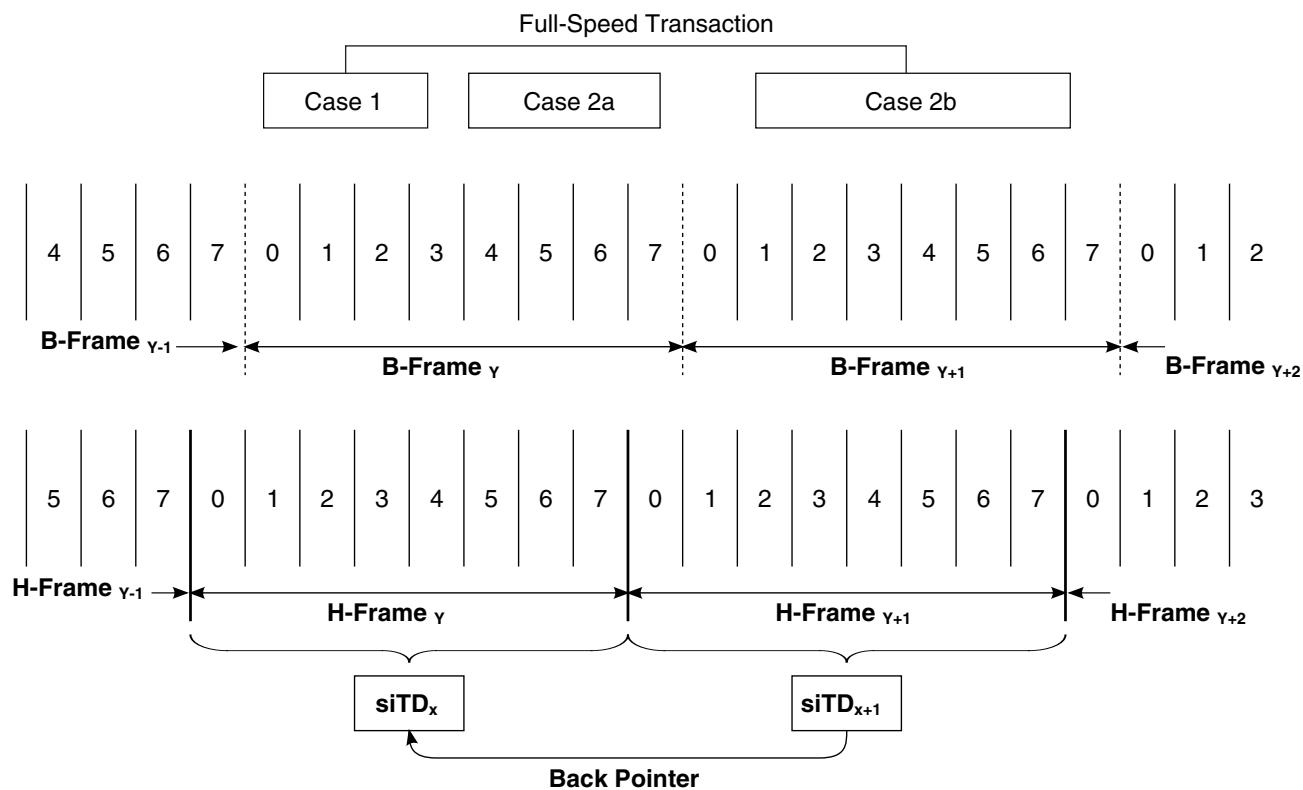


Figure 16-67. siTD scheduling boundary examples

Each case is described below:

- **Case 1:** One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single H-Frame.
- **Case 2a, 2b:** Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction. siTD_X is used to always issue the start-split and the first N complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during microframe 7 of H-Frame_{Y+1}, or microframe 0 of H-Frame_{Y+2}. The complete splits are scheduled using siTD_{X+2} (not shown). The complete-splits to extract this data must use the buffer pointer from siTD_{X+1}. The only way for the host controller to reach siTD_{X+1} from H-Frame_{Y+2} is to use siTD_{X+2}'s back pointer.

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the B-Frame.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in H-Frame, microframe 1. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in microframe 1 of H-Frame N and the last complete-split would need to occur in microframe 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

16.6.12.3.2 Tracking split transaction progress for isochronous transfers

Isochronous endpoints do not employ the concept of a halt on error, however the host controller does identify and report per-packet errors observed in the data stream.

This includes schedule traversal problems (skipped microframes), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the microframes they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs for their transfers and the data structures are only reachable using the schedule in the exact microframe in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD re-initialized (activated) before the host controller gets back to the siTD (in a future microframe).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD using the fields Transaction Position (TP) and Transaction Count (T-count). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See [Split transaction scheduling mechanisms for isochronous](#), for a description on how these fields are used during a sequence of start-split transactions.

The fields siTD[T-Count] and siTD[TP] are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction

isochronous endpoint is established, S-mask, T-Count, and TP initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- C-prog-mask. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the transaction translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. C-prog-mask is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the microframe (FRINDEX[2-0]) number in which the complete-split was executed. The host controller always checks C-prog-mask before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's Active bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. It is important to note that an IN siTD is retired based solely on the responses from the transaction translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD[Total Bytes to Transfer] field to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of Total Bytes to Transfer to zero signals the end of the transfer and results in clearing the Active bit. However, in this case, the result has not been delivered by the transaction translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a transaction translator. In summary, the periodic pipeline rules require that on a microframe boundary, the transaction translator holds the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and gives the remaining bytes to the high-speed pipeline stage. At the microframe boundary, the transaction translator could have received the entire packet (including both CRC bytes)

but not received the packet EOP. In the next microframe, the transaction translator responds with an MDATA and sends all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its Total Bytes to Transfer field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the transaction translator (for example, the transaction translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator is not consistent and the transaction translator detects and reacts to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the C-prog-mask is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (for example, state not advanced) and report the appropriate error to the client driver.

16.6.12.3.3 Split transaction execution state machine for isochronous

In this section, all references to microframe are in the context of a microframe within an H-Frame.

If the active bit in the status byte is a zero, the host controller ignores the siTD and continues traversing the periodic schedule. Otherwise the host controller processes the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in [Tracking split transaction progress for isochronous transfers](#), plus the variable cMicroFrameBit defined in [Split transaction execution state machine for interrupt](#), to track the progress of an isochronous split transaction. The figure below illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the Active bit in the Status field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

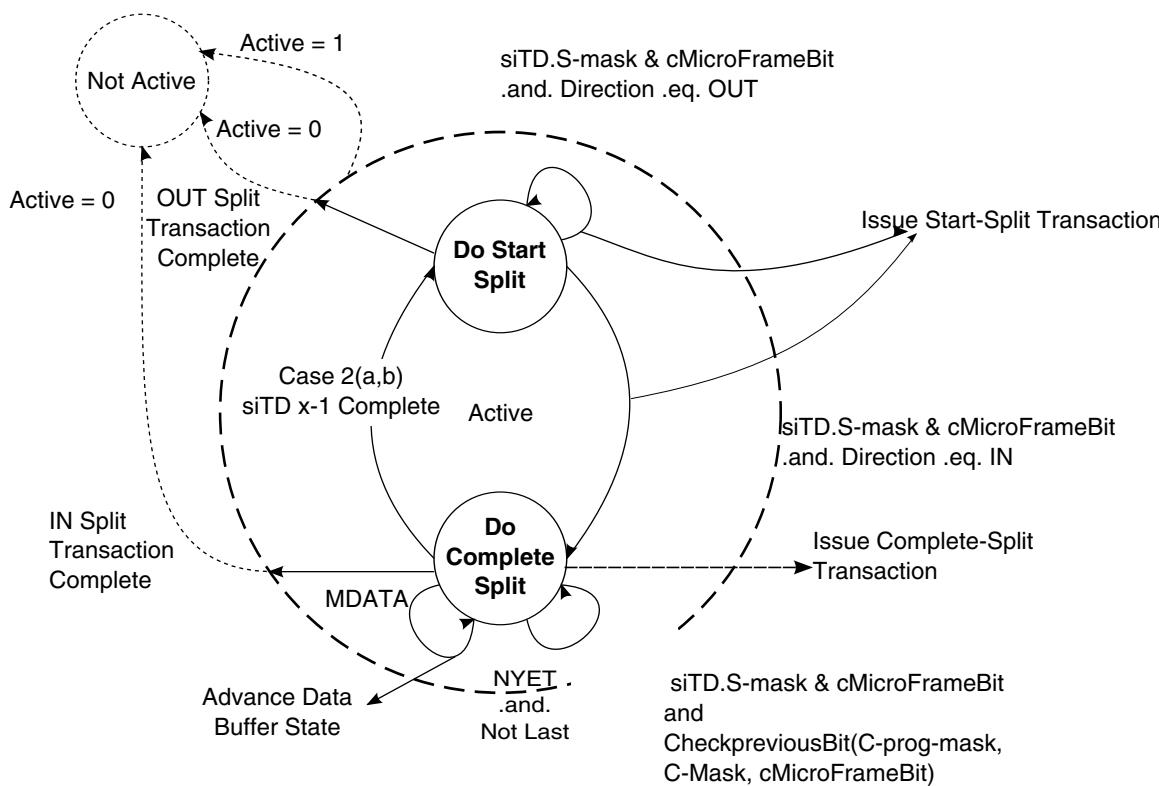


Figure 16-68. Split transaction state machine for isochronous

16.6.12.3.4 Periodic isochronous-do-start-split

Isochronous split transaction OUTs use only this state.

An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the siTD[S-mask] against cMicroFrameBit. If there is a one in the appropriate position, the siTD executes a start-split transaction. By definition, the host controller cannot reach an siTD at the wrong time. If the I/O field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the siTD[Total Bytes To Transfer] field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the I/O field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory buffer address for the data payload is constructed by concatenating siTD[Current Offset]

with the page pointer indicated by the page select field (siTD[P]). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the siTD[P] bit from a zero to a one, and begin using the siTD Page 1 with siTD[Current Offset] as the memory address pointer. The field siTD[TP] is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases, the host controller simply uses the value in siTD[TP] to mark the start-split with the correct transaction position code.

T-count is always initialized to the number of start-splits for the current frame. TP is always initialized to the first required transaction position identifier. The scheduling boundary case (see [Figure 16-67](#)) is used to determine the initial value of TP. The initial cases are summarized in the table below.

Table 16-89. Initial conditions for OUT siTD TP and T-count fields

Case	T-Count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates T-count and TP appropriately so that the next start-split is correctly annotated. The table below illustrates all of the TP and T-count transitions, which must be accomplished by the host controller.

Table 16-90. Transaction position (TP)/transaction count (T-count) transition table

TP	T-Count Next	TP Next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when T-count starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when T-count starts at greater than 2.
MID	!=1	MID	TP stays at MID while T-count is not equal to 1 (for example, greater than 1). This case can occur for any of the scheduling boundary cases where the T-count starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the T-count starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The siTD[Total Bytes To Transfer] and the siTD[Current Offset] fields are adjusted to reflect the number of bytes transferred.
- The siTD[P] (page select) bit is updated appropriately.
- The siTD[TP] and siTD[T-count] fields are updated appropriately as defined in [Table 16-90](#).

These fields are then written back to the memory based siTD. The S-mask is fixed for the life of the current budget. As mentioned above, TP and T-count are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of S-mask, the actual number of start-split transactions depends on T-count (or equivalently, Total Bytes to Transfer). The host controller must clear the Active bit when it detects that all of the schedule data has been sent to the bus. Setting the Active bit to zero depends on siTD.TP being 00 or 11, and siTD.Total Bytes decrements to 0. Software must ensure that TP, T-count and Total Bytes to Transfer are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination yields undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer does not progress appropriately. The transaction translator observes protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation is incorrect as received by the transaction translator).

Example scenarios are described in [Split transaction for isochronous-processing example](#).

The host controller can optionally track the progress of an OUT split transaction by setting appropriate bits in the siTD[C-prog-mask] as it executes each scheduled start-split. The `checkPreviousBit()` algorithm defined in [Periodic isochronous-do complete split](#), can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed microframes. It can then clear the siTD's Active bit and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

16.6.12.3.5 Periodic isochronous-do complete split

This state is only used by a split-transaction isochronous IN endpoint. This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint.

Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The sequence in which they are applied depends on which microframe the host controller is currently executing, which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched. The individual tests are as follows.

- Test A

cMicroFrameBit is bit-wise ANDed with the siTD[C-mask] field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this microframe. This test is always applied to a newly fetched siTD that is in this state.

- Test B

The siTD[C-prog-mask] bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is given below (this is slightly different than the algorithm used in [Periodic interrupt-do-complete-split](#)). The sequence in which this test is applied depends on the current value of FRINDEX[2-0]. If FRINDEX[2-0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
    Boolean rvalue = TRUE;
    previousBit = cMicroFrameBit rotate-right(1)
    -- Bit-wise anding previousBit with C-mask indicates whether there
    -- was an intent to send a complete split in the previous micro-
    -- frame. So, if the 'previous bit' is set in C-mask, check
    -- C-prog-mask to make sure it happened.
    if previousBit bitAND siTD.C-mask then
        if not (previousBit bitAND siTD.C-prog-mask) then
            rvalue = FALSE
        End if
    End if
    Return rvalue
End Algorithm
```

If Test A is true and FRINDEX[2-0] is zero or one, this is a case 2a or 2b scheduling boundary (see [Figure 16-66](#)). See [Complete-split for scheduling boundary cases 2a, 2b](#), for details in handling this condition.

If Test A and Test B evaluate to true, the host controller executes a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates QH[C-prog-mask] by bit-ORing with cMicroFrameBit. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must perform the following actions:

1. Decrement the number of bytes received from siTD[Total Bytes To Transfer]
2. Adjust siTD[Current Offset] by the number of bytes received
3. Adjust the siTD[P] (page select) field if the transfer caused the host controller to use the next page pointer
4. Set any appropriate bits in the siTD[Status] field, depending on the results of the transaction.

Note that if the host controller encounters a condition where siTD[Total Bytes To Transfer] is zero, and it receives more data, the host controller must not write the additional data to memory. The siTD[Status-Active] bit must be cleared and the siTD[Status-Babble Detected] bit must be set. The fields siTD[Total Bytes To Transfer], siTD[Current Offset], and siTD[P] are not required to be updated as a result of this transaction attempt.

The host controller accepts (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of siTD[Total Bytes To Transfer]) MDATA and DATA0/1 data payloads up to and including 192 bytes. The host controller may optionally clear siTD[Status-Active] and set siTD[Status-Babble Detected] when it receives MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- **ERR**

The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the ERR bit in the siTD[Status] field and clears the Active bit.

- **Transaction Error (XactErr)**

The complete-split transaction encounters a Timeout, CRC16 failure, and so on. The siTD[Status] field XactErr field is set and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the microframe occurs, the Active bit is cleared.

- **DATAx (0 or 1)**

This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the Active bit is cleared. If the Bytes To Transfer field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This short packet event does not set the USB interrupt status bit (USBSTS[UI]) to a one. The host controller will not detect this condition.

- **NYET (and Last)**

On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in [Periodic interrupt-do-complete-split](#). If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the Active bit is cleared. No bits are set in the Status field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled complete-splits with NYETs, meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.

- MDATA (and Last)

See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the S-mask and/or C-masks incorrectly. The host controller must set the XactErr bit and clear the Active bit.

- NYET (and not Last)

See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for C-prog-mask) and stay in this state.

- MDATA (and not Last)

The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from microframe X to X+1 and during microframe X, the transaction translator responds with an MDATA and the data accumulated up to the end of microframe X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the Missed Micro-Frame status bit and clears the Active bit.

16.6.12.3.6 Complete-split for scheduling boundary cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see [Figure 16-66](#)) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. The table below enumerates the transaction state fields.

Table 16-91. Summary siTD split transaction state

Buffer state	Status	Execution progress
Total bytes To transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and T-count are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the siTD[Back Pointer] field to reference a valid siTD and have the T bit in the siTD[Back Pointer] field cleared. Otherwise, software must set the T bit in siTD[Back Pointer]. The host controller's rules for interpreting when to use the siTD[Back Pointer] field are listed below. These rules apply only when the siTD's Active bit is a one and the SplitXState is Do Complete Split.

- When cMicroFrameBit is a 0x1 and the siTD_X[Back Pointer] T-bit is zero, or
- If cMicroFrameBit is a 0x2 and siTD_X[S-mask[0]] is zero

When either of these conditions apply, then the host controller must use the transaction state from siTD_{X-1}.

In order to access siTD_{X-1}, the host controller reads on-chip the siTD referenced from siTD_X[Back Pointer].

The host controller must save the entire state from siTD_X while processing siTD_{X-1}. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of siTD[Back Pointers].

If siTD_{X-1} is active (Active bit is set and SplitXStat is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see [Table 16-91](#)) of siTD_{X-1} is appropriately advanced based on the results and written back to memory. If the resultant state of siTD_{X-1}'s Active bit is a one, then the host controller returns to the context of siTD_X, and follows its next pointer to the next schedule item. No updates to siTD_X are necessary.

If siTD_{X-1} is active (Active bit is set and SplitXStat is Do Start Split), then the host controller must clear the Active bit and set the Missed Micro-Frame status bit and the resultant status is written back to memory.

If siTD_{X-1} 's Active bit is cleared, (because it was cleared when the host controller first visited siTD_{X-1} via siTD_X 's back pointer, it transitioned to zero as a result of a detected error, or the results of siTD_{X-1} 's complete-split transaction cleared it), then the host controller returns to the context of siTD_X and transitions its SplitXState to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if cMicroframeBit is 1 and $\text{siTD}_X[\text{S-mask}[0]]$ is 1). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of siTD_X , then follows $\text{siTD}_X[\text{Next Pointer}]$ to the next schedule item. If the criterion is not met, the host controller simply follows $\text{siTD}_X[\text{Next Pointer}]$ to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of siTD_{X-1} will have its Active bit cleared when the host controller returns to the context of siTD_X . Also, note that software should not initialize an siTD with C-mask bits 0 and 1 set and an S-mask with bit 0 set. This scheduling combination is not supported and the behavior of the host controller is undefined.

16.6.12.3.7 Split transaction for isochronous-processing example

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines.

The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced using the Execute Transaction queue head traversal state machine.

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, the table below illustrates a few frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 16-92. Example case 2a-software scheduling siTDs for an IN endpoint

siTD_X		Micro-frames								InitialSplitXState		
		#	Masks	0	1	2	3	4	5	6	7	
X	S-mask						1					Do start split

Table continues on the next page...

Table 16-92. Example case 2a-software scheduling siTDs for an IN endpoint (continued)

siTD _x		Micro-frames								InitialSplitXState
	C-mask	1	1					1	1	
X+1	S-mask					1				Do complete split
	C-mask	1	1					1	1	
X+2	S-mask					1				Do complete split
	C-mask	1	1					1	1	
X+3	S-mask	Repeats previous pattern								Do complete split
	C-mask									

This example shows the first three siTDs for the transaction stream. Since this is the case-2a frame-wrap case, S-masks of all siTDs for this endpoint have a value of 0x10 (a one bit in microframe 4) and C-mask value of 0xC3 (one-bits in microframes 0,1,6 and 7). Additionally, software ensures that the Back Pointer field of each siTD references the appropriate siTD data structure (and the Back Pointer T-bits are cleared).

The initial SplitXState of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in microframes 0 and 1 are ignored because the state is Do Start Split. During microframe 4, the host controller determines that it can run a start-split (and does) and changes SplitXState to Do Complete Split. During microframes 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has its SplitXState initialized to Do Complete Split. As the host controller continues to traverse the schedule during H-Frame X+1, it will visit the second siTD eight times. During microframes 0 and 1 it will detect that it must execute complete-splits.

During H-Frame X+1, microframe 0, the host controller detects that siTD_{X+1}'s Back Pointer[T] bit is a zero, saves the state of siTD_{X+1} and fetches siTD_X. It executes the complete split transaction using the transaction state of siTD_X. If the siTD_X split transaction is complete, siTD's Active bit is cleared and results written back to siTD_X. The host controller retains the fact that siTD_X is retired and transitions the SplitXState in siTD_{X+1} to Do Start Split. At this point, the host controller is prepared to execute the start-split for siTD_{X+1} when it reaches microframe 4. If the split-transaction completes early (transaction-complete is defined in [Periodic isochronous-do complete split](#)), that is, before all the scheduled complete-splits have been executed, the host controller changes siTD_X[SplitXState] to Do Start Split early and naturally skips the remaining scheduled complete-split transactions. For this example, siTD_{X+1} does not receive a DATA0 response until H-Frame X+2, microframe 1.

During H-Frame X+2, microframe 0, the host controller detects that siTD_{X+2}'s Back Pointer[T] bit is zero, saves the state of siTD_{X+2} and fetches siTD_{X+1}. As described above, it executes another split transaction, receives an MDATA response, updates the

transfer state, but does not modify the Active bit. The host controller returns to the context of siTD_{X+2} , and traverses its next pointer without any state change updates to siTD_{X+2} .

During H-Frame $X+2$, microframe 1, the host controller detects siTD_{X+2} 's S-mask[0] bit is zero, saves the state of siTD_{X+2} and fetches siTD_{X+1} . It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and clears the Active bit. It returns to the state of siTD_{X+2} and changes its SplitXState to Do Start Split. At this point, the host controller is prepared to execute start-splits for siTD_{X+2} when it reaches microframe 4.

16.6.13 Port test modes

EHCI host controllers implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the *USB Specification Revision 2.0*.

The required, port test sequence, assuming the CF-bit in the CONFIGFLAG register is set, is as follows:

1. Disable the periodic and asynchronous schedules by clearing the USBCMD[ASE] and USBCMD[PSE].
2. Place all enabled root ports into the suspended state by setting the Suspend bit in the PORTSC register (PORTSC[SUSP]).
3. Clear USBCMD[RS] (run/stop) and wait for USBSTS[HCH] to transition to a one. In Device mode, the Test Mode starts only if Run/Stop bit is set to 1. In Host mode, the Test Mode starts regardless of Run/Stop bit.
4. Set the Port Test Control field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then USBCMD[RS] must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
5. When the test is complete, system software must ensure the host controller is halted (HCH bit is a one) then it terminates and exits test mode by setting USBCMD[RST].

16.6.14 Interrupts

The EHCI host controller hardware provides interrupt capability based on a number of sources.

The following list describes the general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions)
- Host controller events (Port change events, and so on)
- Host controller error events

All transaction-based sources are maskable through the host controller's Interrupt Enable register (USBINTR). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the interrupt threshold control field in the USBCMD register. The value of this register controls when the host controller generates an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight microframes. This means that the host controller will not generate interrupts any more frequently than once every eight microframes.

[Host system error](#) details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to system memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, CPU control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS. It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

NOTE

The only method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register from a one to a zero.

16.6.14.1 Transfer/transaction based interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

16.6.14.1.1 Transaction error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully.

The table below lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the XactErr status bit in the appropriate interface data structure.

Table 16-93. Summary of transaction errors

Event/ Result	Queue Head/qTD/iTD/siTD Side Effects		USBSTS[USBERRINT]
	Cerr	Status Field	
CRC	-1	XactErr set	1 ¹
Timeout	-1	XactErr set	1 ¹
Bad PID ²	-1	XactErr set	1 ¹
Babble	N/A	See Serial bus babble	1
Buffer Error	N/A	See Data buffer error	

¹ If occurs in a queue head, then USBERRINT is asserted only when Cerr counts down from a one to a zero. In addition the queue is halted.

² The host controller received a response from the device, but it could not recognize the PID as a valid PID.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the XactErr status bit in the queue head is set and the Cerr field is decremented. When the PID Code indicates a SETUP, the following responses are protocol errors and result in XactErr bit being set and the Cerr field being decremented.

- EPS field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- EPS field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- EPS field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

16.6.14.1.2 Serial bus babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling.

In general, this is called a packet babbles. When a device sends more data than the maximum length number of bytes, the host controller sets the babbles detected bit to a one and halts the endpoint if it is using a queue head. Maximum length is defined as the minimum of total bytes to transfer and maximum packet size. The Cerr field is not decremented for a packet babbles condition (only applies to queue heads).

A babbles condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babbles. A frame babbles condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babbles is detected.

USBSTS[UEI] (USB error interrupt) is set and if the USBINTR[UEE] (USB error interrupt enable) is set, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that babbles across a microframe EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babbles checking for the duration of the bus transaction or do packet babbles checking based solely on Maximum Packet Size. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence. The EHCI interface allows system software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device re-sends its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babbles check when it observes the data PID mismatch.

16.6.14.1.3 Data buffer error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction.

This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the Data Buffer Error bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This forces the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the transaction translator section of the *USB Specification Revision 2.0*.

NOTE

When the USB controller is in device mode and the host sends two consecutive ISO OUT transactions (for example: OUT - DATA0 - OUT - DATA1) with a short inter-packet delay between DATA0 and the second OUT (less than 200 ns), the device sees the DATA1 packet as a short-packet even if it is correctly formed. In this case, the device terminates the transfer, generating an IOC interrupt (USBSTS[UI]). Note however, that DATA0 is correctly received.

16.6.14.1.4 USB interrupt (interrupt on completion (IOC))

Transfer descriptors (iTDS, siTDS, and queue heads (qTDS)) contain a bit that can be set to cause an interrupt on their completion.

The completion of the transfer associated with that schedule item causes USBSTS[UI] (USB interrupt) to be set. In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set. If USBINTR[UE] (USB interrupt enable) is set, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, USBSTS[UEI] (USB error interrupt) is also set.

16.6.14.1.5 Short packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer.

Whenever a short packet completion occurs during a queue head execution, USBSTS[UI] (USB interrupt bit) is set. If the USB interrupt enable bit is set (USBINTR[UE]), a hardware interrupt is signaled to the system at the next interrupt threshold.

16.6.14.2 Host controller event interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance).

16.6.14.2.1 Port change events

Port registers contain status and status change bits.

When the status change bits are set, the host controller sets the USBSTS[PCI]. If the port change interrupt enable bit (PCE) in the USBINTR register is set, the host controller issues a hardware interrupt. The port status change bits in PORTSC include:

- Connect change status (CSC)
- Port enable/disable change (PEC)
- Over-current change (OCC)
- Force port resume (FPR)

16.6.14.2.2 Frame list rollover

This event indicates that the host controller has wrapped the frame list.

The current programmed size of the frame list effects how often this interrupt occurs. If the frame list size is 1024, then the interrupt occurs every 1024 milliseconds, if it is 512, then it occurs every 512 milliseconds, and so on. When a frame list rollover is detected, the host controller sets the frame list rollover bit, USBSTS[FRI]. If USBINTR[FRE] is set (frame list rollover enable), the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

16.6.14.2.3 Interrupt on async advance

This event is used for deterministic removal of queue heads from the asynchronous schedule.

Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of USBCMD[IAA]. If it is set, it sets USBSTS[AAI]. If USBINTR[AAE] is set, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in [Removing queue heads from asynchronous schedule](#).

16.6.14.2.4 Host system error

The host controller is a bus master and any interaction between the host controller and the system may experience errors.

The type of host error may be catastrophic to the host controller making it impossible for the host controller to continue in a coherent fashion. Behavior for these types of errors is to halt the host controller. Host-based error must result in the following actions:

- USBCMD[RS] is cleared.
- USBSTS[SEI] and USBSTS[HCH] register are set
- If the host system error enable bit, USBINTR[SEE] is set, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

The table below summarizes the required actions taken on the various host errors.

Table 16-94. Summary behavior on host system errors

Cycle type	Master abort	Target abort	Data phase parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal
siTD fetch (read)	Fatal	Fatal	Fatal
siTD status write-back (write)	Fatal	Fatal	Fatal
iTD fetch (read)	Fatal	Fatal	Fatal
iTD status write-back (write)	Fatal	Fatal	Fatal
qTD fetch (read)	Fatal	Fatal	Fatal
qHD status write-back (write)	Fatal	Fatal	Fatal
Data write	Fatal	Fatal	Fatal
Data read	Fatal	Fatal	Fatal

NOTE

After a host system error, software must reset the host controller using USBCMD[RST] before re-initializing and restarting the host controller.

16.7 Device data structures

This section defines the interface data structures used to communicate control, status, and data between device controller driver (DCD) software and the device controller. The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device queue heads and transfer descriptors.

NOTE

Software must ensure that no interface data structure reachable by the device controller spans a 4K-page boundary.

The data structures defined in the section are (from the device controller's perspective) a mix of read-only and read/writable fields. The device controller must preserve the read-only fields on all data structure writes.

The USB DR module includes DCD software called the USB 2.0 Device API. The device API provides an easy to use Application Program Interface for developing device (peripheral) applications. The device API incorporates and abstracts for the application developer all of the elements of the program interface.

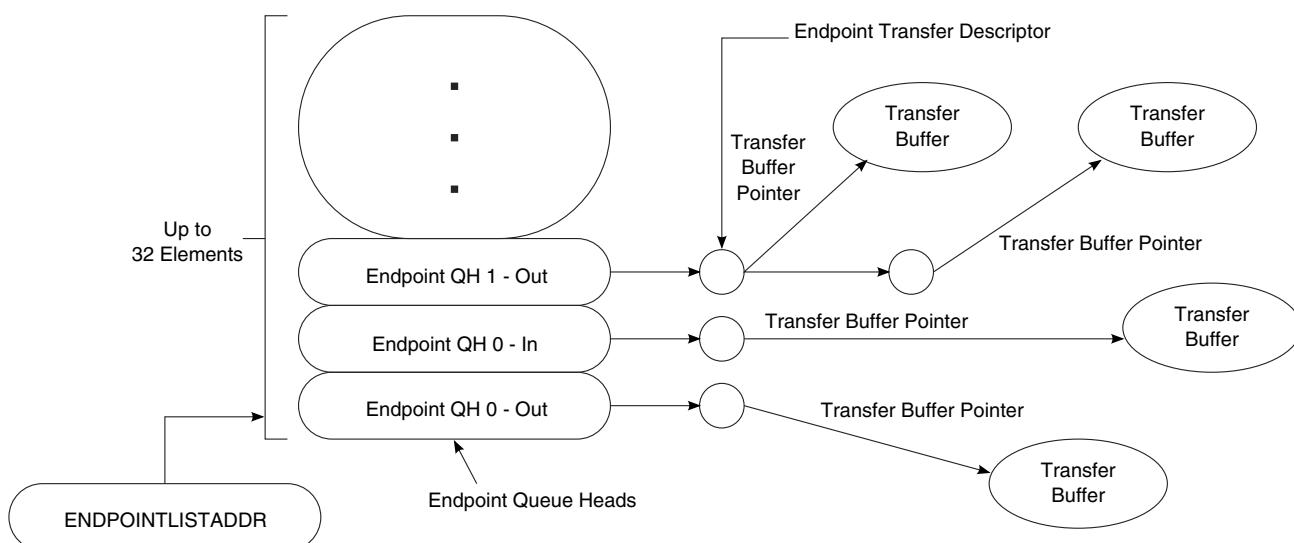


Figure 16-69. Endpoint queue head organization

16.7.1 Endpoint queue head

The device endpoint queue head (dQH) is where all transfers are managed.

The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries. During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

The figure below shows the endpoint queue head structure.

Table 16-95. Endpoint queue head layout

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	15	14	1 3	1 2	11	10	9	8	7	6	5	4	3	2	1	0	offset
Mult	zlt	00	Maximum Packet Length								ios	000_0000_0000_0000														0x00						
Current dTD Pointer ¹																				0_0000		0x04										
Next dTD Pointer ¹																				0000	T ¹	0x08 ²										
0	Total Bytes ¹										ioc ¹	000	MultO ¹	00	Status ¹														0x0C ²			
Buffer Pointer (Page 0) ¹																				Current Offset ¹										0x10 ²		
Buffer Pointer (Page 1) ¹																				Reserved										0x14 ²		
Buffer Pointer (Page 2) ¹																				Reserved										0x18 ²		
Buffer Pointer (Page 3) ¹																				Reserved										0x1C ²		
Buffer Pointer (Page 4) ¹																				Reserved										0x20 ²		
Reserved																														0x24		
Setup Buffer Bytes 3-0 ¹																														0x28		
Setup Buffer Bytes 7-4 ¹																														0x2C		

1. Device controller read/write; all others read-only.
2. Offsets 0x08 through 0x20 contain the transfer overlay.

16.7.1.1 Endpoint capabilities/characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint.

Device controller software should not attempt to modify this information while the corresponding endpoint is enabled.

The table below describes the endpoint capabilities and characteristics fields.

Table 16-96. Endpoint capabilities/characteristics

Bits	Name	Description
31-30	Mult	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTQ) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. NOTE: Non-ISO endpoints must set Mult = 00. NOTE: ISO endpoints must set Mult = 01, 10, or 11 as needed.
29	zlt	Zero length termination select. This bit is used to indicate when a zero length packet is used to terminate transfers where the total transfer length is a multiple. This bit is not relevant for Isochronous transfers.

Table continues on the next page...

Table 16-96. Endpoint capabilities/characteristics (continued)

Bits	Name	Description
		0 Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	-	Reserved, should be cleared. These bit reserved for future use and should be cleared.
26-16	Maximum Packet Length	Maximum packet length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	ios	Interrupt on setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	-	Reserved, should be cleared. Bits reserved for future use and should be cleared.

16.7.1.2 Transfer overlay

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is read, the dTD is copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

16.7.1.3 Current dTD pointer

The current dTD pointer is used by the device controller to locate the transfer in progress.

This word is for USB_DR controller (hardware) use only and should not be modified by DCD software.

The table below describes the current dTD pointer fields.

Table 16-97. Current dTD pointer

Bits	Description
31-5	Current dtd. This field is a pointer to the dTD that is represented in the transfer overlay area. This field is modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved, should be cleared. Bit reserved for future use and should be cleared.

16.7.1.4 Setup buffer

The setup buffer is dedicated storage for the 8-byte data that follows a setup PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The table below describes the multiple mode control fields.

Table 16-98. Multiple mode control

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

16.7.2 Endpoint transfer descriptor (dTD)

The dTD describes the location and quantity of data to be sent/received for given transfer to the device controller.

The DCD should not attempt to modify any field in an active dTD except the Next Link Pointer, which should only be modified as described in [Managing transfers with transfer descriptors](#).

The figure below shows the endpoint transfer descriptor.

Table 16-99. Endpoint transfer descriptor (dTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	offset
Next Link Pointer																										0000	T	0x00				
0	Total Bytes ¹				ioc	000	MultO	00	Status ¹															0x04								

Table continues on the next page...

Table 16-99. Endpoint transfer descriptor (dTD) (continued)

Buffer Pointer (Page 0)	Current Offset ¹	0x08
Buffer Pointer (Page 1)	0 Frame Number ¹	0x0C
Buffer Pointer (Page 2)	0000_0000_0000	0x10
Buffer Pointer (Page 3)	0000_0000_0000	0x14
Buffer Pointer (Page 4)	0000_0000_0000	0x18

1. Device controller read/write; all others read-only.

The table below describes the next dTD pointer fields.

Table 16-100. Next dTD pointer

Bits	Description
31-5	Next transfer element pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved, should be cleared. Bits reserved for future use and should be cleared.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The table below describes the next dTD token fields.

Table 16-101. dTD token

Bits	Description
31	Reserved, should be cleared. Bit reserved for future use and should be cleared.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K(5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K. Therefore, the maximum recommended transfer is 16K(4000H). If the value of the field is zero when the controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for IN transfers that Total Bytes To Transfer be a multiple of Maximum Packet Length. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less than Maximum Packet Length.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved, should be cleared. Bits reserved for future use and should be cleared.
11-10	Multiplier Override (MultiO). This field can be used for transmit ISO's (that is, ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total bytes = 15; MultiO = 2

Table continues on the next page...

Table 16-101. dTD token (continued)

Bits	Description
	Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes/Max. Packet Size) except for the case when Total bytes = 0; then MultO should be 1. NOTE: Non-ISO and non-TX endpoints must set MultO = 00.
9-8	Reserved, should be cleared. Bits reserved for future use and should be cleared.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active 6 Halted 5 Data Buffer Error 3 Transaction Error 4,2,0 Reserved, should be cleared

The following tables describe the buffer pointer page n fields.

Table 16-102. Buffer pointer page 0

Bits	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.

Table 16-103. Buffer pointer page 1

Bits	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
11	Reserved
10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

Table 16-104. Buffer pointer pages 2-4

Bits	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
11-0	Reserved

16.8 Device operational model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

16.8.1 Device controller initialization

After hardware reset, the USB DR module is disabled until the run/stop bit (USBCMD[RS]) is set to a '1'.

In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A queue head must be prepared so that the device controller can store the incoming setup packet.

To configure the external ULPI PHY the following initialization sequence is required.

In order to initialize a device, the software should perform the following steps:

1. Set the controller mode to device mode. Optionally set USBMODE[SDIS] (streaming disable).

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

2. Program PORTSC[PTS] if using a non-ULPI PHY.
3. Set CONTROL[USB_EN]
4. Allocate and initialize device queue heads in system memory Minimum: Initialize device queue heads 0 Tx and 0 Rx.

NOTE

All device queue heads must be initialized for control endpoints before the endpoint is enabled. Device queue heads for non-control endpoints must be initialized before the endpoint can be used.

For information on device queue heads, refer to [Device data structures](#).

5. Configure the ENDPOINTLISTADDR pointer.

For additional information on ENDPOINTLISTADDR, refer to the register table.

6. Enable the microprocessor interrupt associated with the USB DR module and optionally change setting of USBCMD[ITC].

Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.

For a list of available interrupts refer to the USBINTR and the USBSTS register tables.

7. Set USBCMD[RS] to run mode.

After the run bit is set, a device reset will occur. The DCD must monitor the reset event and set the DEVICEADDR register, set the ENDPTCTRLx registers, and adjust the software state as described in [Bus reset](#).

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework command set.

16.8.2 Port state and control

From a chip or system reset, the USB_DR controller enters the powered state.

A transition from the powered state to the attach state occurs when the run/stop bit (USBCMD[RS]) is set to a '1'. After receiving a reset on the bus, the port will enter the defaultFS or defaultHS state in accordance with the protocol reset described in Appendix C.2 of the *USB Specification Rev. 2.0*. The figure below depicts the state of a USB 2.0 device.

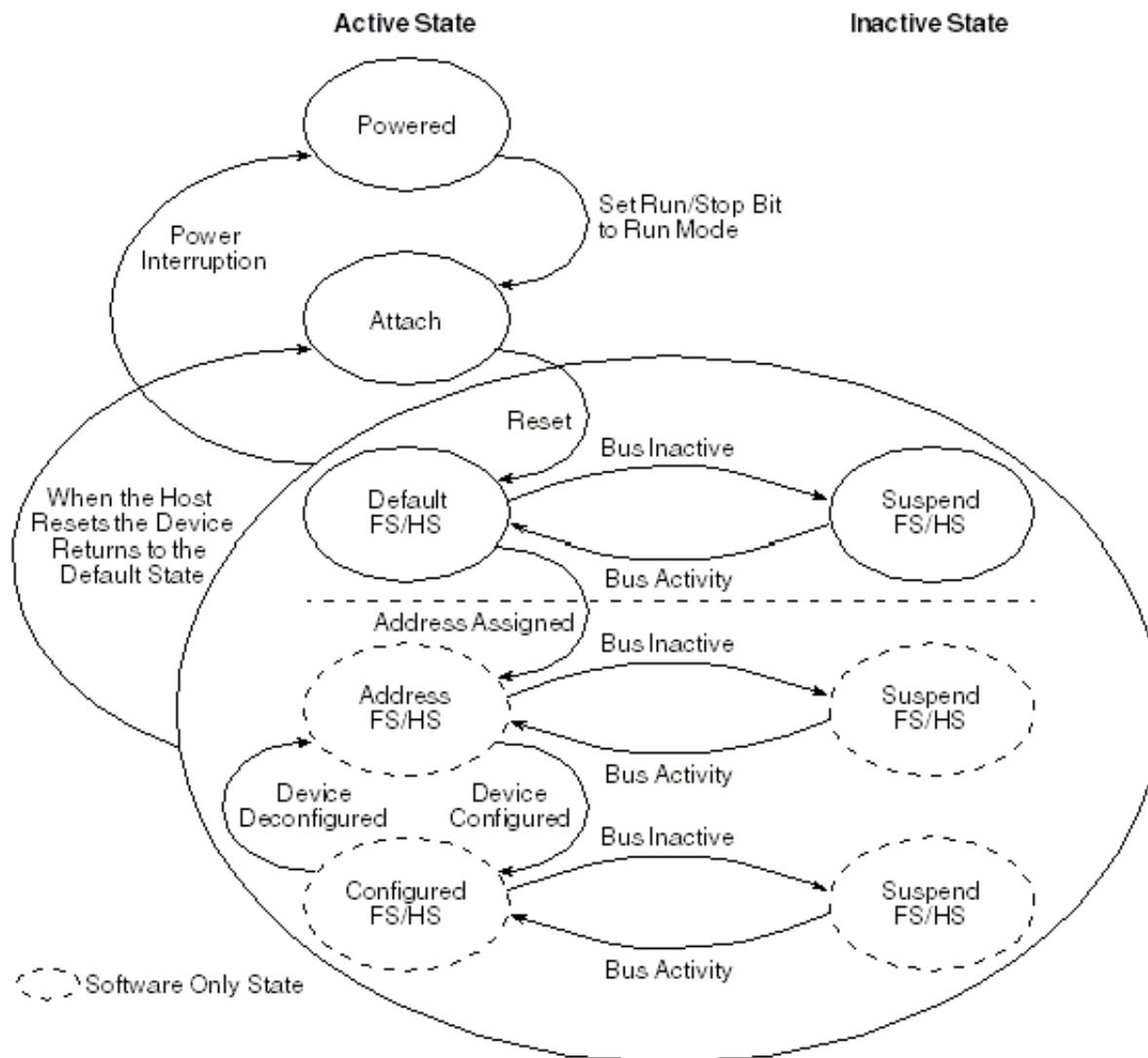


Figure 16-70. USB 2.0 device states

States Powered, Attach, DefaultFS/HS, SuspendFS/HS are implemented in the USB_DR controller and are communicated to the DCD using status bits, as shown in the table below:

Table 16-105. Device controller state information bits

Bits	Register
DCSuspend (SLI)	USBSTS
USB Reset Received (URI)	USBSTS
Port Change Detect (PCI)	USBSTS
High-Speed Port	PORTRSC

It is the responsibility of the DCD to maintain a state variable to differentiate between the defaultFS/HS state and the address/configured states. Change of state from default to address and the configured states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the address state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the configured indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the ENDPTCTRL n registers and initializing the associated queue heads.

16.8.2.1 Bus reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the USB_DR controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB reset interrupt enable bit, USBINTR[URE], is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions are canceled by the device controller. The concept of priming is clarified below, but the DCD must perform the following tasks when a reset is received:

1. Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.
2. Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.
3. Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFF_FFFF to ENDPTFLUSH.
4. Read the reset bit in the PORTSC register (PORTSC[PR]) and make sure that it is still active.
 - A USB reset occurs for a minimum of 3 ms, and the DCD must reach this point in the reset cleanup before end of the reset occurs.
 - If it does not, a hardware reset of the device controller is recommended. A hardware reset can be performed by writing a one to the USB_DR controller reset bit in (USBCMD[RST]). Note that a hardware reset will cause the device to detach from the bus by clearing USBCMD[RS] bit. Thus, the DCD must completely re-initialize the USB_DR controller after a hardware reset.
5. Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSC to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9, Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode. Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.

16.8.2.2 Suspend/resume

This section discusses the suspend and resume functions.

16.8.2.2.1 Suspend description

In order to conserve power, USB_DR controller automatically enters the suspended state when no bus traffic has been observed for a specified period.

When suspended, the USB_DR controller maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

The USB_DR controller exits suspend mode when there is bus activity. It may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wake-up. The ability of a device to signal remote wake-up is optional. The USB_DR controller is capable of remote wake-up signaling. When the USB_DR controller is reset, remote wake-up signaling must be disabled.

16.8.2.2.2 Suspend operational model

The USB_DR controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period.

After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming DC Suspend Interrupt is enabled). When the USBSTS[SLI] (device controller suspend) is set, the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation. Information on the bus power limits in suspend state can be found in USB 2.0 specification.

16.8.2.2.3 Resume

If the USB_DR controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port.

In addition, the USB_DR controller can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the PORTSC[FPR] (resume bit) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one or more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (Chapter 9) of the USB 2.0 Specification.

16.8.3 Managing endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a control type data channel used for device discovery and enumeration. Other types of endpoints supported by USB include bulk, interrupt, and isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB_DR controller supports up to six endpoint specified numbers. The DCD can enable, disable, and configure each endpoint.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a queue head allocated in memory. If the maximum of 6 endpoint numbers, one for each endpoint direction are being used by the device controller, then 12 queue heads are required. The operation of an endpoint and use of queue heads are described later in this document.

16.8.3.1 Endpoint initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled.

The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRL n register. Each 32-bit ENDPTCTRL n is split into an upper and lower half. The lower half of ENDPTCTRL n is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRL n register otherwise the behavior is undefined. The table below shows how to construct a configuration word for endpoint initialization.

Table 16-106. Device controller endpoint initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	0

16.8.3.1.1 Stalling

There are two occasions where the USB_DR controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework (Chapter 9). A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRL n register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints and is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRL n register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the ENDPTCTRL n register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The table below describes the device controller stall response matrix.

Table 16-107. Device controller stall response matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL Bit.	USB Response
SETUP packet received by a non-control endpoint	N/A	None	STALL
IN/OUT/PING packet received by a non-control endpoint	'1	None	STALL
IN/OUT/PING packet received by a non-control endpoint	'0	None	ACK/NAK/NYET
SETUP packet received by a control endpoint	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1	None	STALL
IN/OUT/PING packet received by a control endpoint	'0	None	ACK/NAK/NYET

16.8.3.2 Data toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the *Universal Serial Bus Revision 2.0 Specification*.

16.8.3.2.1 Data toggle reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the ENDPTCTRL n register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

16.8.3.2.2 Data toggle inhibit

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the data toggle Inhibit bit active ('1') causes the USB_DR controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the USB_DR controller checks the DATA0/DATA1 bit against the data toggle state bit to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the USB_DR controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

16.8.3.3 Device operational model for packet transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the *Universal Serial Bus Revision 2.0 Specification*.

A USB host will send requests to the USB_DR controller in an order that cannot be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 2 (transmit direction) is configured as a bulk pipe, then the host sends IN requests to that endpoint. This USB_DR controller prepares packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as 'priming' the endpoint. This term is used throughout the following documentation to describe the

USB_DR controller operation so the DCD can be architected properly use priming. Further, note that the term 'flushing' is used to describe the action of clearing a packet that was queued for execution.

16.8.3.3.1 Priming transmit endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it is stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO has been sized to account for the maximum latency that can be incurred by the system memory bus.

16.8.3.3.2 Priming receive endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD.

At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

16.8.3.4 Interrupt/bulk endpoint operational model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD is retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and the following tables describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

$$N = \text{INT}(\text{number of bytes}/\text{max. packet length}) + 1$$

With Zero Length Termination (ZLT) = 1

$$N = \text{MAXINT}(\text{number of bytes}/\text{max. packet length})$$

Table 16-108. Variable length transfer protocol example (ZLT = 0)

Bytes (dTD)	Max. packet length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	3	256	256	0
512	512	2	512	0	-

Table 16-109. Variable length transfer protocol example (ZLT = 1)

Bytes (dTD)	Max. packet length (dQH)	N	P1	P2	P3
511	256	2	256	255	-
512	256	2	256	256	-
512	512	1	512	-	-

NOTE

The MULT field in the dQH must be set to '00' for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint is flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD is cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the USB_DR controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH is left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly re-initialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

16.8.3.4.1 Interrupt/bulk endpoint bus response matrix

The table below shows the interrupt/bulk endpoint bus response matrix.

Table 16-110. Interrupt/bulk endpoint bus response matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error ¹	N/A
Out	STALL	NAK	Receive + NYET/ACK ²	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. Force Bit Stuff Error.

2. NYET/ACK-NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR-System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

16.8.3.5 Control endpoint operation model

This section discusses the control endpoint operation model.

16.8.3.5.1 Setup phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase.

The USB_DR controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Setup Packet Handling

- Disable Setup Lockout by writing '1' to Setup Lockout Mode (SLOM) in USBMODE (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode as '0' will result in a potential compliance issue.

- After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:
 - Write '1' to clear corresponding bit ENDPTSETUPSTAT.
 - Write '1' to Setup Tripwire (SUTW) in USBCMD register.
 - Duplicate contents of dQH.SetupBuffer into local software byte array.
 - Read Setup TripWire (SUTW) in USBCMD register. (if set-continue; if cleared-goto 2)
 - Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.
 - Process setup packet using local software byte array copy and execute status/handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed and de-allocated before linking a new status and/or handshake dTD for the most recent setup packet.

16.8.3.5.2 Data phase

If the control transfer requires a data stage following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, that is, The ENDPTPRIME bit goes to zero and the ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTATUS) to enforce data coherency with the setup packet.

NOTE

The MULT field in the dQH must be set to '00' for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

16.8.3.5.3 Status phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

NOTE

The MULT field in the dQH must be set to '00' for bulk, interrupt, and control endpoints.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

16.8.3.5.4 Control endpoint bus response matrix

The table below shows the device controller response to packets on a control endpoint, according to the device controller state.

Table 16-111. Control endpoint bus response matrix

Token type	Endpoint state					Setup lockout
	Stall	Not primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR ¹	
In	STALL	NAK	Transmit	BS Error ²	N/A	N/A
Out	STALL	NAK	Receive + NYET/ ACK ³	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

1. SYSERR-System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

2. Force Bit Stuff Error.

3. NYET/ACK-NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

16.8.3.6 Isochronous endpoint operational model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the USB_DR controller is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note that MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD is held ready until executed or canceled by the DCD.

The USB_DR controller in host mode uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit is cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

When the USB controller is in device mode on P4080, Rev 2, and the host sends two consecutive ISO OUT transactions (for example: OUT - DATA0 - OUT - DATA1) with a short inter-packet delay between DATA0 and the second OUT (less than 200 ns), the device sees the DATA1 packet as a short-packet even if it is correctly formed. In this case, the device terminates the transfer, generating an IOC interrupt (USBSTS[UI]). Note however, that DATA0 is correctly received. On P4080, Rev 3, the device correctly handles consecutive ISO OUT transactions.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the Transaction Error bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
 - MULT counter reaches zero.
 - Fulfillment Error [Transaction Error bit is set]
 - #Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:

- MULT counter reaches zero.
- Non-MDATA Data PID is received
- Overflow Error:
 - Packet received is > maximum packet length. [Buffer Error bit is set]
 - Packet received exceeds total bytes allocated in dTD. [Buffer Error bit is set]
 - Fulfillment Error [Transaction Error bit is set]
 - # Packets Occurred > 0 AND # Packets Occurred < MULT
 - CRC Error [Transaction Error bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

16.8.3.6.1 Isochronous pipe synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro)frame number (FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro)frame number [N], the DCD should interrupt on SOF during frame N - 1. When the FRINDEX = N - 1, the DCD must write the prime bit. The USB_DR controller will prime the isochronous endpoint in (micro)frame N - 1 so that the device controller will execute delivery during (micro)frame N.

NOTE

Priming an endpoint towards the end of (micro)frame N - 1 will not guarantee delivery in (micro)frame N. The delivery may actually occur in (micro)frame N + 1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

16.8.3.6.2 Isochronous endpoint bus response matrix

The table below shows the isochronous endpoint bus response matrix.

Table 16-112. Isochronous endpoint bus response matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL ¹ Packet	NULL Packet	Transmit	BS Error ²	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. Zero Length Packet.

2. Force Bit Stuff Error.

16.8.4 Managing queue heads

The figure below shows the endpoint queue head diagram.

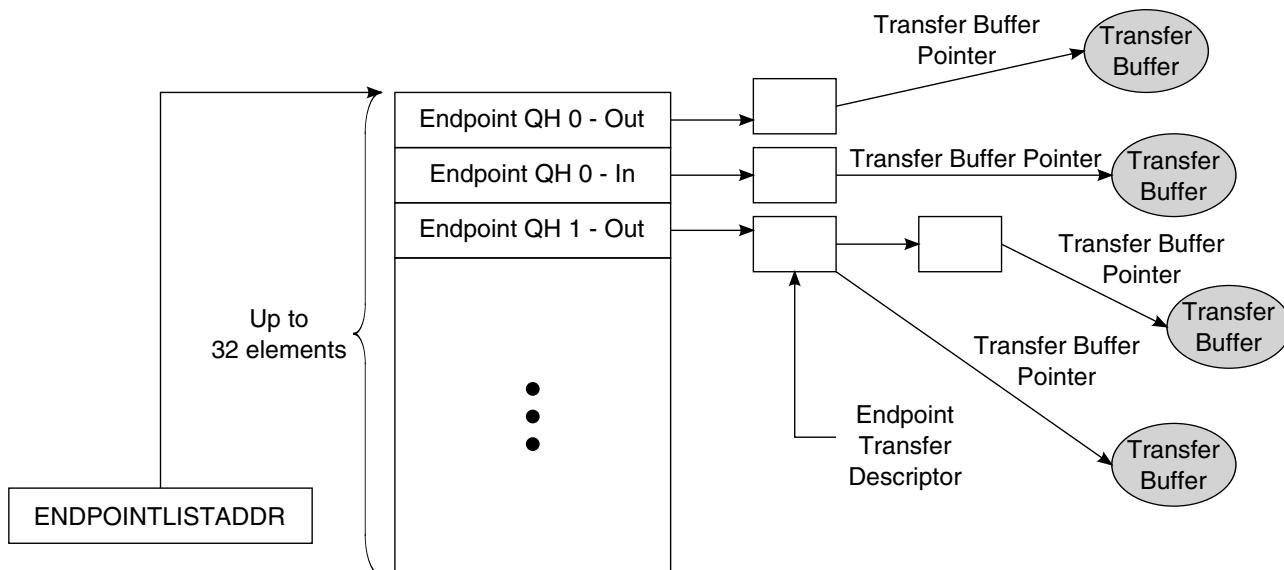


Figure 16-71. Endpoint queue head diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTd). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQHs in a sequential list as shown in [Figure 16-71](#). The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTd has been retired, it will no longer be part of the linked list

from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see [Software link pointers](#)).

In addition to the current and next pointers and the dTD overlay, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

16.8.4.1 Queue head initialization

One pair of device queue heads must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1, 2, or 3 as required bandwidth an in conjunction with the USB Chapter 9 protocol. Note that in FS mode, the multiplier field can only be 1 for ISO endpoints.
- Write the next dTD Terminate bit field to '1.'
- Write the Active bit in the status field to '0.'
- Write the Halt bit in the status field to '0.'

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTDs.

16.8.4.2 Operational model for setup transfers

As discussed in [Control endpoint operation model](#), setup transfer requires special treatment by the DCD.

A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a '1' to the corresponding bit in ENDPTSETUPSTAT.

NOTE

The acknowledge must occur before continuing to process the setup packet.

NOTE

After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in [Flushing/depriming an endpoint](#).

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete.

Existing control packets in progress must be flushed and the new control packet completed.

4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

16.8.5 Managing transfers with transfer descriptors

This section describes software link pointers, transfer descriptors, transfer completion and the device error matrix.

16.8.5.1 Software link pointers

It is necessary for the DCD software to maintain head and tail pointers for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head and Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

The figure below shows the software link pointers.

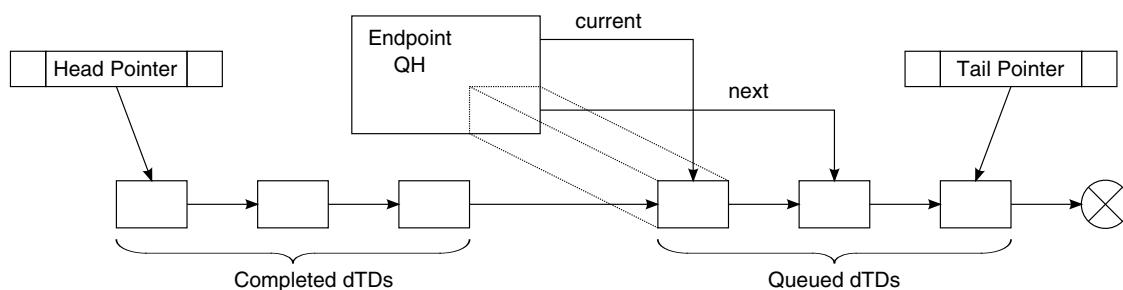


Figure 16-72. Software link pointers

16.8.5.2 Building a transfer descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4-0 would be equal to '00000'.

Write the following fields:

1. Initialize first seven DWords to '0'.
2. Set the terminate bit to '1'.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to '1' and all remaining status bits set to '0'.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

16.8.5.3 Executing a transfer descriptor

To safely add a dTD, the DCD must account for the event in which the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

First, determine whether the link list is empty by checking the DCD driver to see if the pipe is empty (internal representation of linked-list should indicate if any packets are outstanding). Then follow the sequence of actions in the following list as appropriate, depending on whether the link list is empty or not empty.

Case 1: Link list is empty

1. Write dQH next pointer AND dQH terminate bit to '0' as a single DWord operation.
2. Clear active and halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing '1' to correct bit position in ENDPTPRIME.

Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in ENDPTPRIME-if '1' DONE.
3. Set ATDTW bit in USBCMD register to '1'.
4. Read correct status bit in ENDPTSTATUS. (store in tmp. variable for later).
5. Read ATDTW bit in USBCMD register.
If '0' goto 3.
If '1' continue to 6.
6. Write ATDTW bit in USBCMD register to '0'.
7. If status bit read in (4) is '1' DONE.
8. If status bit read in (4) is '0' then Goto Case 1: Step 1.

16.8.5.4 Transfer completion

After a dTD has been initialized and the associated endpoint primed, the device controller will execute the transfer upon the host-initiated request.

The DCD is notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

16.8.5.5 Flushing/depriming an endpoint

It is necessary for the DCD to flush to deprime one or more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are '0'.
3. Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.
4. Read ENDPTSTATUS to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:

Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush by repeating steps 1-3 until each endpoint is successfully flushed.

16.8.5.6 Device error matrix

The table below summarizes packet errors that are not automatically handled by the USB controller.

Table 16-113. Device error matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below provides the error descriptions.

Table 16-114. Error descriptions

Overflow	Number of bytes received exceeded max. packet size or total buffer length. NOTE: This error also sets the Halt bit in the dQH. If there are dTDs remaining in the linked list for the endpoint, they will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the dead (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

16.8.6 Servicing interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

16.8.6.1 High-frequency interrupts

High frequency interrupts in particular should be handled in the order shown in the table below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 16-115. Interrupt handling order

Execution order	Interrupt	Action
1a	USB Interrupt ¹ ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Managing queue heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ENDPTCOMPLETE	Handle completion of dTD as indicated in Managing queue heads .
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts will stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

16.8.6.2 Low-frequency interrupts

The low frequency events include the interrupts shown in the table below.

These interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

Table 16-116. Low frequency interrupt events

Interrupt	Action
Port Change	Change software state information.
Sleep enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

16.8.6.3 Error interrupts

Error interrupts are the least frequently occurring events. They should be placed last in the interrupt service routine.

The table below shows the error interrupt events.

Table 16-117. Error interrupt events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

16.9 Deviations from the EHCI specifications

The host mode operation of the USB DR module is nearly EHCI-compatible with few minor differences. For the most part, the module conforms to the data structures and operations described in Section 3, "Data Structures," and Section 4, "Operational Model," in the EHCI specification. The particulars of the deviations occur in the following areas:

- Embedded transaction translator-Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation-In host mode, the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface-The module does not have a PCI interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

16.9.1 Embedded transaction translator function

The DR module supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate transaction translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

16.9.1.1 Capability registers

The following additions have been added to the capability registers to support the embedded transaction translator function:

- N_TT added to HSCPARAMS-Host Controller Structural Parameters
- N_PTT added to HSCPARAMS-Host Controller Structural Parameters

See [Host Controller Structural Parameters \(USB_HSCPARAMS\)](#), for usage information.

16.9.1.2 Operational registers

The following additions have been added to the operational registers to support the embedded TT:

- ASYNCTTSTS is a new register.
- Addition of two-bit Port Speed (PSPD) to the PORTSC register.

16.9.1.3 Discovery differences

In a standard EHCI controller design, the EHCI host controller driver detects a full speed (FS) or low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

The module always sets the port enable after the port reset operation regardless of the result of the host device chirp result. The resulting port speed is indicated by the PSPD field in PORTSC. Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected full- and low-speed devices or hubs. The table below summarizes the functional differences between EHCI and EHCI with embedded TT.

Table 16-118. Functional differences between EHCI and EHCI with embedded TT

Standard EHCI	EHCI with embedded transaction translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from PORTSC.

Table continues on the next page...

Table 16-118. Functional differences between EHCI and EHCl with embedded TT (continued)

Standard EHCI	EHCl with embedded transaction translator
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using PORTSC.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (that is, Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (that is, Split target hub is the root hub)]

16.9.1.4 Data structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the root hub.

The following list demonstrates how the hub address and endpoint speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS)-Async. (Bulk/Control Endpoints) Periodic (Interrupt)
 - Hub Address = 0
 - Transactions to direct attached device/hub.
 - QH.EPS = Port Speed
 - Transactions to a device downstream from direct attached FS hub.
 - QH.EPS = Downstream Device Speed

NOTE

When QH.EPS = 01 (LS) and PORTSC[PSPD] = 00 (FS), a LS-pre-pid is sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behavior may result.

2. siTD (for direct attach FS)-Periodic (ISO Endpoint)
 - All FS ISO transactions:
 - Hub Address = 0
 - siTD.EPS = 00 (full speed)

Maximum packet size must less than or equal to 1023 or undefined behavior may result.

16.9.1.5 Operational model

The operational models are well defined for the behavior of the transaction translator (see *Universal Serial Bus Revision 2.0 Specification*) and for the EHCI controller moving packets between system memory and a USB-HS hub.

Since the embedded transaction translator exists within the DR module there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and transaction translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 transaction translator operational models.

16.9.1.5.1 Microframe pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the host (H) and the bus (B).

The embedded transaction translator shall use the same pipeline algorithms specified in the *Universal Serial Bus Revision 2.0 Specification* for a Hub-based transaction translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the microframe pipeline implemented in the embedded transaction translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

NOTE

When programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded transaction translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based transaction translators.

Once periodic transfers are exhausted, any stored asynchronous transfer are moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer cannot babble through the SOF (start of B-frame 0).

16.9.1.5.2 Split state machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded transaction translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded transaction translator. The table below summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 16-119. Emulated handshakes

Condition	Emulate TT response
Start-Split: All asynchronous buffers full	NAK
Start-Split: All periodic buffers full	ERR
Start-Split: Success for start of Async. Transaction	ACK
Start-Split: Start Periodic Transaction	No handshake (Ok)
Complete-Split: Failed to find transaction in queue	Bus Time Out
Complete-Split: Transaction in Queue is Busy	NYET
Complete-Split: Transaction in Queue is Complete	[Actual handshake from FS/LS device]

16.9.1.5.3 Asynchronous transaction scheduling and buffer management

The following *Universal Serial Bus Revision 2.0 Specification* items are implemented in the embedded transaction translator:

- USB 2.0-11.17.3
 - Sequencing is provided and a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.
- USB 2.0-11.17.4
 - Transaction tracking for 2 data pipes.
- USB 2.0-11.17.5
 - Clear_TT_Buffer capability provided

16.9.1.5.4 Periodic transaction scheduling and buffer management

The following *Universal Serial Bus Revision 2.0 Specification* items are implemented in the embedded transaction translator:

- USB 2.0-11.18.6.[1-2]
 - Abort of pending start-splits
 - EOF (and not started in microframes 6)
 - Idle for more than 4 microframes
 - Abort of pending complete-splits

- EOF
- Idle for more than 4 microframes

NOTE

There is no data schedule mechanism for these transactions other than the microframe pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 msec) or else undefined behavior may result.

16.9.1.5.5 Multiple transaction translators

The maximum number of embedded transaction translators that is currently supported is one as indicated by the N_TT field in the HCSPARAMS register.

See [Host Controller Structural Parameters \(USB_HCSPARAMS\)](#), for more information.

16.9.2 Device operation

The co-existence of a device operational controller within the DR module has little effect on EHCI compatibility for host operation except as noted in this section.

16.9.3 Non-zero fields the register file

Some of the reserved fields and reserved addresses in the capability registers and operational registers have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields in the DR module) in the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the module must properly mask EHCI reserved fields (some of which are device fields in the DR module registers).

16.9.4 SOF interrupt

The SOF interrupt is a free running 125 µsec interrupt for host mode.

EHCI does not specify this interrupt, but it has been added for convenience and as a potential software time base. The free running interrupt is shared with the device-mode start-of-frame interrupt. See [USB Status \(USB_USBSTS\)](#), and [USB Interrupt Enable \(USB_USBINTR\)](#), for more information.

16.9.5 Embedded design

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

16.9.5.1 Frame adjust register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like those provided by the Frame Adjust register in the PCI configuration registers.

Starts of microframes are timed precisely to 125 μ sec using the transceiver clock as a reference clock. That is, 60-MHz transceiver clock for 8-bit physical interfaces and full-speed serial interfaces or 30-MHz transceiver clock.

16.9.6 Miscellaneous variations from EHCI

The modules support multiple physical interfaces which can operate in different modes when the module is configured with the software programmable Physical Interface Modes.

The control bits for selecting the PHY operating mode have been added to the PORTSC register providing a capability that is not defined by the EHCI specification.

16.9.6.1 Discovery

This section discusses port reset and port speed detection.

16.9.6.1.1 Port reset

The port connect methods specified by EHCI require setting the port reset bit in the register for a duration of 10 msec.

Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10 msec reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to the reset the device.
- Software shall write a '0' to the reset the device after 10 msec.
 - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device is now operational and at this point the port speed has been determined.

16.9.6.1.2 Port speed detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed.

Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non-HS devices. Therefore, the following differences are important regarding port speed detection:

- Port owner is read-only and always reads 0.
- A 2-bit port speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
 - A 1-bit high-speed indicator has been added to PORTSC to signify that the port is in HS vs. FS/LS

Chapter 17

DUART

17.1 DUART Overview

This chapter describes the dual universal asynchronous receiver/transmitters (DUART). It describes the functional operation, the initialization sequence, and the programming details for the DUART registers and features.

The DUART consists of two universal asynchronous receiver/transmitters (UARTs). Note that this device implements two DUART modules. DUART1 contains UART1 and UART2; DUART2 contains UART3 and UART4. The UARTs act independently; all references to UART refer to one of these receiver/transmitters. The DUART programming model is compatible with the PC16552D.

The UART interface is point to point, meaning that only two UART devices are attached to the connecting signals. As shown in [Figure 17-1](#), each UART module consists of the following:

- Receive and transmit buffers
- Clear to send (CTS_B) input port and request to send (RTS_B) output port for data flow control
- 16-bit counter for baud rate generation
- Interrupt control logic

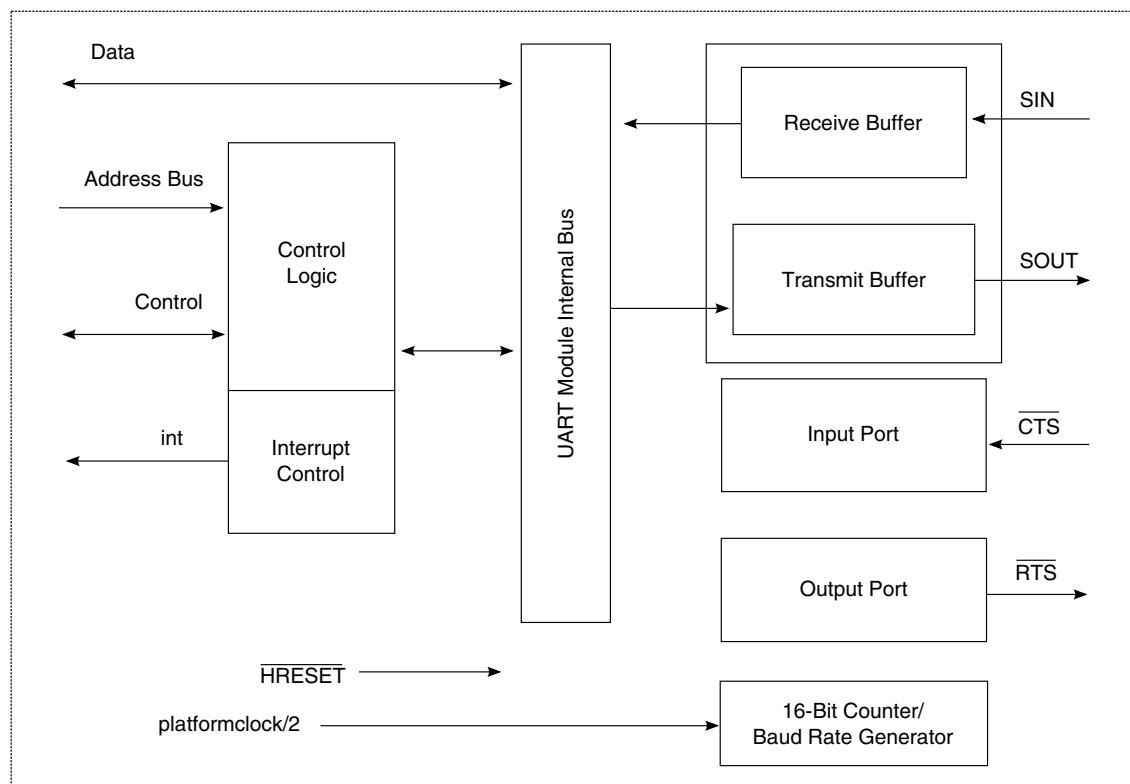


Figure 17-1. UART Block Diagram

17.2 DUART Features Summary

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with original PC16450 UART and PC16550D (improved version of PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the CCB clock (platform clock/2) by 1 to ($2^{16} - 1$) and generate a 16x clock for the transmitter and receiver engines
- Clear to send (CTS_B) and request to send (RTS_B) modem control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation

- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

17.3 DUART Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the platform clock/2.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a start bit, parity (if any), stop bits, and transfers the assembled character (with start, stop, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

17.3.1 DUART Signal Mode Selection

The UART signals are multiplexed with other functions on the device. The following relationships apply:

- UART1 shares signals with GPIO and UART3
- UART2 shares signals with GPIO and UART4

See [UART and GPIO Signal Multiplexing](#), for more information. The functionality of these signals is determined by the UART field in the reset configuration word (RCW[UART]). Note that RCW[UART] = 0x0 defaults to all GPIO functionality on the UART signal pins; the RCW must be initialized to the desired UART signaling for proper UART operation.

[Figure 17-2](#) shows the signal multiplexing for UART1 CTS_B/RTS with UART3 SIN/SOUT.

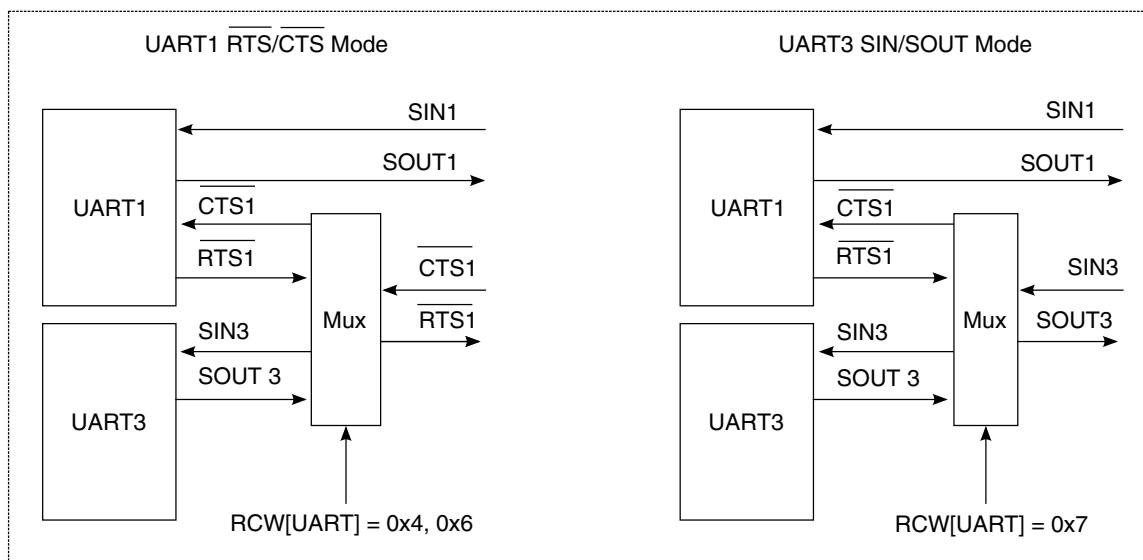


Figure 17-2. UART1/UART3 Signal Multiplexing

[Figure 17-3](#) shows the signal multiplexing for UART2 CTS_B/RTS_B and UART4 SIN/SOUT.

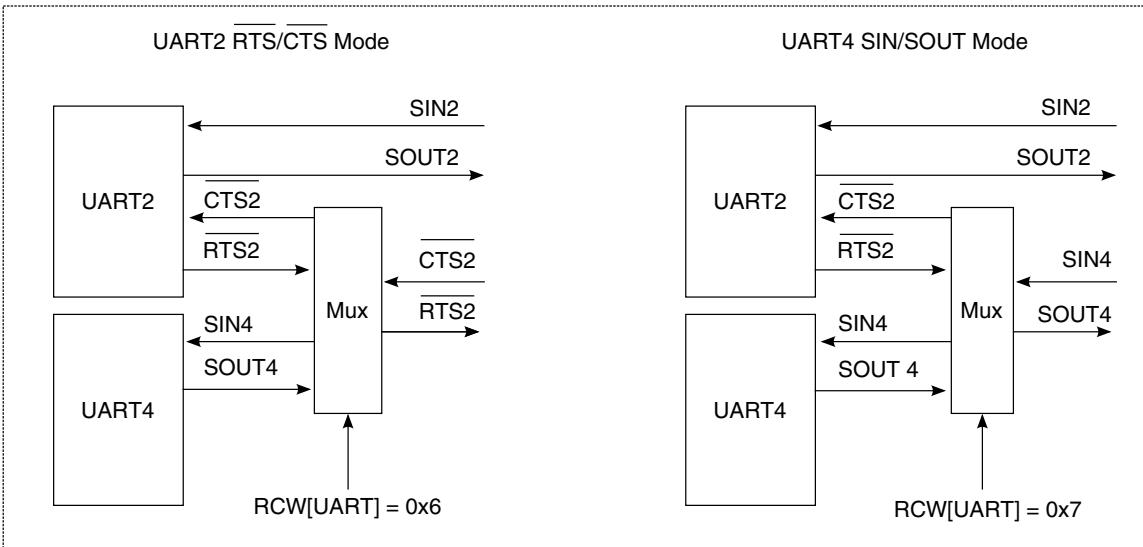


Figure 17-3. UART2/UART4 Signal Multiplexing

17.4 DUART External Signal Descriptions

The DUART signals are described in [Table 17-1](#). Note that although the actual device signal names are prepended with the `UART_` prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

Table 17-1. DUART Signals-Detailed Signal Descriptions

Signal	I/O	Description	
UART _n _SIN	I	Serial data in. Data is received on the receivers of UART1 , UART2, UART3, and UART4 through the respective serial data input signal, with the least-significant bit received first. Note that UART3_SIN is not available when UART1 is configured for RTS_B/CTS_B mode and UART4_SIN is not available when UART2 is configured for RTS_B/CTS_B mode.	
		State Meaning	Asserted/Negated-Represents the data being received on the UART interface.
		Timing	Assertion/Negation-An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to sample the data on SIN.
UART _n _SOUT	O	Serial data out. The serial data output signals for the UART1 , UART2, UART3, and UART4 are set ('mark' condition) when the transmitter is disabled, idle, or operating in the local loopback mode. Data is shifted out on these signals, with the least significant bit transmitted first. Note that UART3_SOUT is not available when UART1 is configured for RTS/CTS_B mode and UART4_SOUT is not available when UART2 is configured for RTS_B/CTS_B mode.	
		State Meaning	Asserted/Negated-Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation-An internal logic sample signal, <i>rxcnt</i> , uses the frequency of the baud-rate generator to update and drive the data on SOUT.
UART _n _CTS_B	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective RTS_B outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal. Note that UART1_CTS is not available when UART3 is configured for SIN/SOUT mode and UART2_CTS is not available when UART4 is configured for SIN/SOUT mode.	
		State Meaning	Asserted/Negated-Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation-Sampled at the rising edge of every input clock (platform clock/2).
UART _n _RTS_B	O	Request to send. These active-low output signals can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send (CTS_B) input of a transmitter, this signal can be used to control serial data flow. Note that UART1_RTS is not available when UART3 is configured for SIN/SOUT mode and UART2_RTS is not available when UART4 is configured for SIN/SOUT mode.	
		State Meaning	Asserted/Negated-Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation-Updated and driven at the rising edge of every input clock (platform clock/2).

17.5 DUART Memory Map/Register Definition

The table below lists the DUART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the block base address and offset.

There are four complete sets of registers (one for each UART). The four UARTs on the device are identical, except that the registers for each UART are located at different offsets. Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART1, UART2, UART3, or UART4.

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to [UART line control register \(DUART_ULCRn\)](#) for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Reads and writes to these registers must be byte-wide operations. The table below provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000-0xFFFF are reserved.

NOTE

UART2 has the same memory-mapped registers that are described for UART1 from 0x500 to 0x510 except the offsets range from 0x600 to 0x610. Similarly, the registers for UART3 are located at offsets 0x500--0X510 from the DUART2 block base address and the registers for UART4 are located at offsets 0x600--0x610 from the DUART2 block base address.

DUART memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_C500	UART receiver buffer register (DUART1_URBR1)	8	R	00h	17.5.1/1094
11_C500	UART transmitter holding register (DUART1_UTHR1)	8	W	00h	17.5.2/1095
11_C500	UART divisor least significant byte register (DUART1_UDLB256)	8	R/W	00h	17.5.3/1095
11_C501	UART divisor most significant byte register (DUART1_UDMB1)	8	R/W	00h	17.5.4/1097
11_C501	UART interrupt enable register (DUART1_UIER1)	8	R/W	00h	17.5.5/1097
11_C502	UART interrupt ID register (DUART1_UIIR1)	8	R	01h	17.5.6/1098
11_C502	UART FIFO control register (DUART1_UFCR1)	8	W	00h	17.5.7/1100
11_C502	UART alternate function register (DUART1_UAFR1)	8	R/W	00h	17.5.8/1101
11_C503	UART line control register (DUART1_ULCR1)	8	R/W	00h	17.5.9/1101

Table continues on the next page...

DUART memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
11_C504	UART modem control register (DUART1_UMCR1)	8	R/W	00h	17.5.10/1103
11_C505	UART line status register (DUART1_ULSR1)	8	R	60h	17.5.11/1104
11_C506	UART modem status register (DUART1_UMSR1)	8	R	00h	17.5.12/1106
11_C507	UART scratch register (DUART1_USCR1)	8	R/W	00h	17.5.13/1106
11_C510	UART DMA status register (DUART1_UDSR1)	8	R	01h	17.5.14/1107
11_C600	UART receiver buffer register (DUART1_URBR2)	8	R	00h	17.5.1/1094
11_C600	UART transmitter holding register (DUART1_UTHR2)	8	W	00h	17.5.2/1095
11_C600	UART divisor least significant byte register (DUART1_UDLB)	8	R/W	00h	17.5.3/1095
11_C601	UART divisor most significant byte register (DUART1_UDMB2)	8	R/W	00h	17.5.4/1097
11_C601	UART interrupt enable register (DUART1_UIER2)	8	R/W	00h	17.5.5/1097
11_C602	UART interrupt ID register (DUART1_UIIR2)	8	R	01h	17.5.6/1098
11_C602	UART FIFO control register (DUART1_UFCR2)	8	W	00h	17.5.7/1100
11_C602	UART alternate function register (DUART1_UAFR2)	8	R/W	00h	17.5.8/1101
11_C603	UART line control register (DUART1_ULCR2)	8	R/W	00h	17.5.9/1101
11_C604	UART modem control register (DUART1_UMCR2)	8	R/W	00h	17.5.10/1103
11_C605	UART line status register (DUART1_ULSR2)	8	R	60h	17.5.11/1104
11_C606	UART modem status register (DUART1_UMSR2)	8	R	00h	17.5.12/1106
11_C607	UART scratch register (DUART1_USCR2)	8	R/W	00h	17.5.13/1106
11_C610	UART DMA status register (DUART1_UDSR2)	8	R	01h	17.5.14/1107
11_D500	UART receiver buffer register (DUART2_URBR1)	8	R	00h	17.5.1/1094
11_D500	UART transmitter holding register (DUART2_UTHR1)	8	W	00h	17.5.2/1095
11_D500	UART divisor least significant byte register (DUART2_UDLB256)	8	R/W	00h	17.5.3/1095
11_D501	UART divisor most significant byte register (DUART2_UDMB1)	8	R/W	00h	17.5.4/1097
11_D501	UART interrupt enable register (DUART2_UIER1)	8	R/W	00h	17.5.5/1097
11_D502	UART interrupt ID register (DUART2_UIIR1)	8	R	01h	17.5.6/1098
11_D502	UART FIFO control register (DUART2_UFCR1)	8	W	00h	17.5.7/1100
11_D502	UART alternate function register (DUART2_UAFR1)	8	R/W	00h	17.5.8/1101
11_D503	UART line control register (DUART2_ULCR1)	8	R/W	00h	17.5.9/1101

Table continues on the next page...

DUART memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
11_D504	UART modem control register (DUART2_UMCR1)	8	R/W	00h	17.5.10/1103
11_D505	UART line status register (DUART2_ULSR1)	8	R	60h	17.5.11/1104
11_D506	UART modem status register (DUART2_UMSR1)	8	R	00h	17.5.12/1106
11_D507	UART scratch register (DUART2_USCR1)	8	R/W	00h	17.5.13/1106
11_D510	UART DMA status register (DUART2_UDSR1)	8	R	01h	17.5.14/1107
11_D600	UART receiver buffer register (DUART2_URBR2)	8	R	00h	17.5.1/1094
11_D600	UART transmitter holding register (DUART2_UTHR2)	8	W	00h	17.5.2/1095
11_D600	UART divisor least significant byte register (DUART2_UDLB)	8	R/W	00h	17.5.3/1095
11_D601	UART divisor most significant byte register (DUART2_UDMB2)	8	R/W	00h	17.5.4/1097
11_D601	UART interrupt enable register (DUART2_UIER2)	8	R/W	00h	17.5.5/1097
11_D602	UART interrupt ID register (DUART2_UIIR2)	8	R	01h	17.5.6/1098
11_D602	UART FIFO control register (DUART2_UFCR2)	8	W	00h	17.5.7/1100
11_D602	UART alternate function register (DUART2_UAFR2)	8	R/W	00h	17.5.8/1101
11_D603	UART line control register (DUART2_ULCR2)	8	R/W	00h	17.5.9/1101
11_D604	UART modem control register (DUART2_UMCR2)	8	R/W	00h	17.5.10/1103
11_D605	UART line status register (DUART2_ULSR2)	8	R	60h	17.5.11/1104
11_D606	UART modem status register (DUART2_UMSR2)	8	R	00h	17.5.12/1106
11_D607	UART scratch register (DUART2_USCR2)	8	R/W	00h	17.5.13/1106
11_D610	UART DMA status register (DUART2_UDSR2)	8	R	01h	17.5.14/1107

17.5.1 UART receiver buffer register (DUARTx_URBRn)

(ULCR[DLAB] = 0)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, [UART line status register \(DUART_ULSRn\)](#). Note that these registers have same offset as the UTHRs.

Address: Base address + 500h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	DATA							
Write								
Reset	0	0	0	0	0	0	0	0

DUARTx_URBRn field descriptions

Field	Description
0–7 DATA	Data received from the transmitter on the UART bus (read only)

17.5.2 UART transmitter holding register (DUARTx_UTHRn)

(ULCR[DLAB] = 0)

A write to these 8-bit registers causes the UART devices to transfer 5-8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR is the first byte onto the bus.

UDSR[TXRDY] indicates when the FIFO is full. Refer to [UART DMA status register \(DUART_UDSRn\)](#) for more details.

Address: Base address + 500h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read								
Write	DATA							
Reset	0	0	0	0	0	0	0	0

DUARTx_UTHRn field descriptions

Field	Description
0–7 DATA	Data that is written to UTHR (write only)

17.5.3 UART divisor least significant byte register (DUARTx_UDLBn)

(ULCR[DLAB] = 1)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = input clock frequency/(16 x [UDMB||UDLB]).

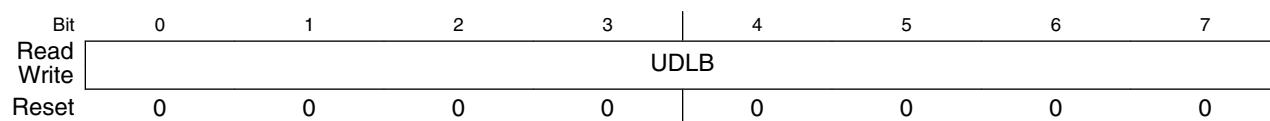
Equivalently, [UDMB||UDLB:0b0000] = input clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in the table below.

The table below shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible. Note that because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 17-72. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		DUART Input Clock (Platform Clock/2) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1953	07A1	300	9600.61444	0.0064
19,200	977	03D1	300	19,191.40225	0.0448
38,400	488	01E8	300	38,422.13115	0.0576
57,600	326	0146	300	57,515.33742	0.1470
115,200	163	00A3	300	115,030.67485	0.1470
230,400	81	0051	300	231,481.48148	0.4694
9,600	2168	0878	333	9599.86162	0.0014
19,200	1084	043C	333	19,199.72325	0.0014
38,400	542	021E	333	38,399.44649	0.0014
57,600	361	0169	333	57,652.35457	0.0909
115,200	181	00B5	333	114,986.18785	0.1856
230,400	90	005A	333	231,250.00000	0.3689
9,600	2604	0A2C	400	9600.61444	0.0064
19,200	1302	0516	400	19,201.22888	0.0064
38,400	651	028B	400	38,402.45776	0.0064
57,600	434	01B2	400	57,603.68664	0.0064
115,200	217	00D9	400	115,207.37327	0.0064
230,400	109	006D	400	229,357.79817	0.4523

Address: Base address + 500h offset + (256d x i), where i=0d to 1d



DUART_x_UDLB_n field descriptions

Field	Description
0–7 UDLB	Divisor least significant byte. This is concatenated with UDMB.

17.5.4 UART divisor most significant byte register (DUART_x_UDMB_n)

(ULCR[DLAB] = 1)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = input clock frequency/(16 x [UDMB||UDLB]).

Equivalently, [UDMB||UDLB:0b0000] = input clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in [Table 17-7](#).

Address: Base address + 501h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read					UDMB			
Write								

Reset 0 0 0 0 0 0 0 0 0

DUART_x_UDMB_n field descriptions

Field	Description
0–7 UDMB	Divisor most significant byte

17.5.5 UART interrupt enable register (DUART_x_UIER_n)

(ULCR[DLAB] = 0)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Address: Base address + 501h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read			Reserved		EMSI	ERLSI	ETHREI	ERDAI
Write								

Reset 0 0 0 0 0 0 0 0 0

DUARTx_UIERn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 EMSI	Enable modem status interrupt. 0 Mask interrupts caused by UMSR[DCTS] being set 1 Enable and assert interrupts when the clear-to-send bit in the UART modem status register (UMSR) changes state
5 ERLSI	Enable receiver line status interrupt. 0 Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1 Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set
6 ETHREI	Enable transmitter holding register empty interrupt. 0 Mask interrupt when ULSR[THRE] is set 1 Enable and assert interrupts when ULSR[THRE] is set
7 ERDAI	Enable received data available interrupt. 0 Mask interrupt when new receive data is available or receive data time out has occurred 1 Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in the FIFO mode

17.5.6 UART interrupt ID register (DUARTx_UIIRn)

(ULCR[DLAB] = 0)

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. Modem status

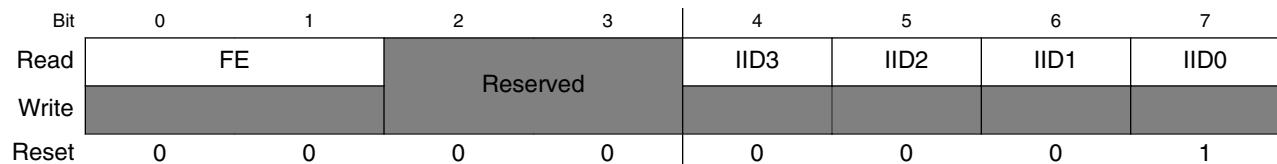
See the table below for more details.

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Table 17-80. UIIR IID Bits Summary

IID Bits IID[3-0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	-	-	-	-
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	CTS_B input value changed since last read of UMSR	Read the UMSR.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d



DUARTx_UIIRn field descriptions

Field	Description
0–1 FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3 -	This field is reserved. Reserved
4 IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5 IID2	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
6 IID1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
7 IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

17.5.7 UART FIFO control register (DUARTx_UFCRn)

(ULCR[DLAB] = 0)

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read				Reserved				
Write	RTL				DMS	TFR	RFR	FEN
Reset	0	0	0	0	0	0	0	0

DUARTx_UFCRn field descriptions

Field	Description
0–1 RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3 -	This field is reserved. Reserved
4 DMS	DMA mode select. See DMA Mode Select for more information. 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5 TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6 RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0

Table continues on the next page...

DUARTx_UFCR n field descriptions (continued)

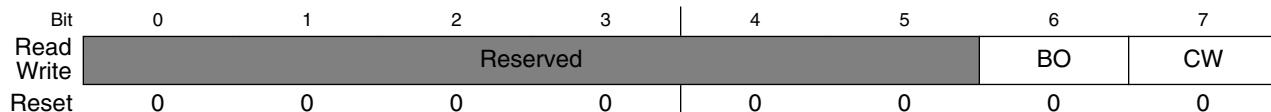
Field	Description
7 FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

17.5.8 UART alternate function register (DUARTx_UAFR n)

(ULCR[DLAB] = 1)

The UAFRs give software the ability to gate off the baud clock and write to both UART1/UART2 registers or both UART3/UART4 registers simultaneously with the same write operation.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d

**DUARTx_UAFR n field descriptions**

Field	Description
0–5 -	This field is reserved. Reserved
6 BO	Baud clock select. 0 The baud clock is not gated off. 1 The baud clock is gated off.
7 CW	Concurrent write enable. 0 Disables concurrent writing to both UART1 and UART2 (DUART1) or UART3 and UART4 (DUART2) 1 Enables concurrent writes to corresponding UART registers. For DUART1, a write to a register in UART1 is also a write to the corresponding register in UART2 and vice versa; for DUART2, a write to a register in UART3 is also a write to the corresponding register in UART4 and vice versa. The user needs to ensure that the LCR[DLAB] of both UARTs are in the same state before executing a concurrent write to register addresses 0xn00, 0xn01 and 0xn02, where n is the offset of the corresponding UART.

17.5.9 UART line control register (DUARTx_ULCR n)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See the table below for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Table 17-88. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

Address: Base address + 503h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	DLAB	SB	SP	EPS	PEN	NSTB		
Write	0	0	0	0	0	0	0	0
Reset								

DUARTx_ULCRn field descriptions

Field	Description
0 DLAB	Divisor latch access bit. 0 Access to all registers except UDLB, UAFLR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFLR)
1 SB	Set break. 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2 SP	Stick parity.

Table continues on the next page...

DUARTx_ULCRn field descriptions (continued)

Field	Description
	0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3 EPS	Even parity select. See the table above for more information. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4 PEN	Parity enable. 0 No parity generation and checking 1 Generate parity bit as a transmitter, and check parity as a receiver
5 NSTB	Number of STOP bits. 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1½ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7 WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

17.5.10 UART modem control register (DUARTx_UMCRn)

The UMCRs control the interface with the external peripheral device on the UART bus.

Address: Base address + 504h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write		Reserved		LOOP	Reserved		RTS	Reserved
Reset	0	0	0	0	0	0	0	0

DUARTx_UMCRn field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 LOOP	Local loopback mode. 0 Normal operation 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].

Table continues on the next page...

DUARTx_UMCRx field descriptions (continued)

Field	Description
4–5 -	This field is reserved. Reserved
6 RTS	Ready to send. 0 Negates corresponding UART_RTS_B output 1 Assert corresponding UART_RTS_B output. Informs external modem or peripheral that the UART is ready for sending/receiving data
7 -	This field is reserved. Reserved

17.5.11 UART line status register (DUARTx_ULSRn)

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

Address: Base address + 505h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	RFE	TEMT	THRE	BI	FE	PE	OE	DR
Write								
Reset	0	1	1	0	0	0	0	0

DUARTx_ULSRn field descriptions

Field	Description
0 RFE	Receiver FIFO error. 0 This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1 TEMT	Transmitter empty. 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2 THRE	Transmitter holding register empty. 0 The UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.

Table continues on the next page...

DUARTx_ULSRn field descriptions (continued)

Field	Description
3 BI	<p>Break interrupt.</p> <p>NOTE: For a single break signal, BI and DR are set multiple times, approximately once every character period. The BI and DR bits continue to be set each character period after they are cleared. This continues for the entire duration of the break signal. To accommodate this behavior, read URBR, which returns zeros and clears DR. Then delay one character period and read URBR again. Note that at the end of the break signal, a random character may be falsely detected and received in the URBR, with ULSR[DR] being set.</p> <ul style="list-style-type: none"> 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4 FE	<p>Framing error.</p> <ul style="list-style-type: none"> 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then receives the following new data.
5 PE	<p>Parity error.</p> <ul style="list-style-type: none"> 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.
6 OE	<p>Overrun error.</p> <ul style="list-style-type: none"> 0 This bit is cleared when ULSR is read. 1 Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7 DR	<p>Data ready.</p> <ul style="list-style-type: none"> 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

17.5.12 UART modem status register (DUARTx_UMSRn)

The UMSRs track the status of the modem (or external peripheral device) clear to send (CTS_B) signal for the corresponding UART.

Address: Base address + 506h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read				CTS				
Write		Reserved				Reserved		
Reset	0	0	0	0	0	0	0	0

DUARTx_UMSRn field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CTS	Clear to send. Represents the inverted value of the CTS_B input pin from the external peripheral device 0 Corresponding CTS_B is negated 1 Corresponding CTS_B is asserted. The modem or peripheral device is ready for data transfers.
4–6 -	This field is reserved. Reserved
7 DCTS	Clear to send. 0 No change on the corresponding CTS_B signal since the last read of UMSR[CTS] 1 The CTS_B value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition

17.5.13 UART scratch register (DUARTx_USCRn)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Address: Base address + 507h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read					DATA			
Write								
Reset	0	0	0	0	0	0	0	0

DUART_x_USCR_n field descriptions

Field	Description
0–7 DATA	Data

17.5.14 UART DMA status register (DUART_x_UDSR_n)

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Table 17-103. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 17-104. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear when the transmitter FIFO is not yet full.

Table 17-105. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 17-106. UDSR[RXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	

Table continues on the next page...

Table 17-106. UDSR[RXRDY] Cleared Conditions (continued)

DMS	FEN	DMA Mode	Meaning
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

Address: Base address + 510h offset + (256d × i), where i=0d to 1d



DUARTx_UDSRn field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6 TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown above in Table 17-36 . 1 This bit is set, as shown above in Table 17-35 .
7 RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown above in Table 17-38 . 1 This bit is set, as shown above in Table 17-37 .

17.6 DUART Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from one-half of the platform clock.

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see [FIFO Mode](#). The transmitting registers convert the data to a serial bit stream, by inserting the appropriate START, STOP, and optional parity bits. Finally, the registers output a composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt-driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the

assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

17.6.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in [Figure 17-130](#). Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

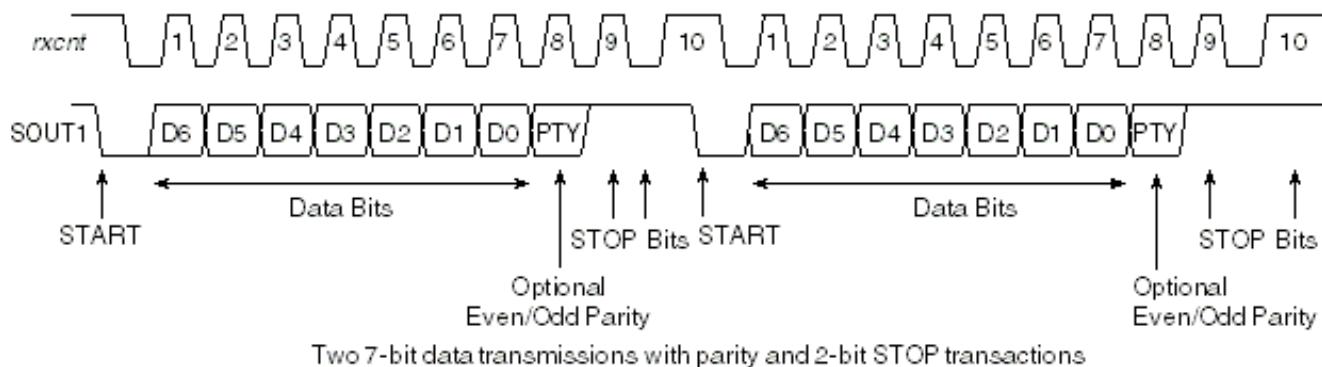


Figure 17-130. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, rxcnt, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

17.6.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. [Figure 17-130](#) shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in the UART line control register (ULCR). When the bus is idle, SOUT is high.

17.6.1.2 Data Transfer

Each data transfer contains 5-8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5-8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

17.6.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see [UART line control register \(DUART_ULCRn\)](#)). Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data a parity error can occur if an unexpected parity value is detected. (See [UART line status register \(DUART_ULSRn\)](#).)

17.6.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

17.6.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the DUART module input clock (platform clock/2) and dividing it by any divisor from 1 to $2^{16} - 1$.

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

$$\text{Baud rate} = (1/16) \times (\text{DUART module input clock frequency}/\text{divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

17.6.3 Local Loopback Mode

Local loopback mode is provided for diagnostic testing. The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the modem control register UMCR[RTS] is internally tied to the modem status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The CTS (input signal) is disconnected, RTS is internally connected to CTS, and the RTS (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loopback mode the transmit and receive data paths of the DUART can be verified. Note that in local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

17.6.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

17.6.4.1 Framing Error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0

being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

17.6.4.2 Parity Error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO . ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

17.6.4.3 Overrun Error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

17.6.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs and set the FIFO receiver trigger level UFCR[RTL] to control the received data available interrupt UIER[ERDAI].

The UFCR also selects the type of DMA signaling. The UDSR[RXRDY] indicates the status of the receiver FIFO. The DMA status registers (UDSR[TXRDY]) indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

17.6.5.1 FIFO Interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through UIER[ERDAI]). See [UART interrupt enable register \(DUART_UIERn\)](#) for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. Interrupt ID3 UIIR[IID3] bit is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IIDn]) is cleared when the URBR is read. See [UART interrupt ID register \(DUART_UIIRn\)](#) for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

17.6.5.2 DMA Mode Select

The UDSR[RXRDY] bit reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] is cleared), UDSR[RXRDY] is cleared when there is at least one character in the receiver FIFO or URBR and it is set when there are no more characters in the receiver FIFO or URBR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[RXRDY] is cleared when the trigger level or a time-out has been reached and it is set when there are no more characters in the receiver FIFO.

The UDSR[TXRDY] bit reflects the status of the transmitter FIFO or UTHR. In mode 0 (UFCR[DMS] is cleared), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set after the first character is loaded into the transmitter FIFO or UTHR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set when the transmitter FIFO is full.

See [UART DMA status register \(DUART_UDSRn\)](#) for a complete description of the UDSR[RXRDY] and UDSR[TXRDY] bits.

17.6.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 7 (UIIR[IID0]), is cleared. The interrupt enable register (UIER) is used to mask specific interrupt types. For more details refer to the description of UIER in [UART interrupt enable register \(DUART_UIERn\)](#).

When the interrupts are disabled in UIER, polling software cannot use UIIR[IID0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the modem status (UMSR) registers. UIIR[IID0] can be used for polling if the interrupts are enabled in UIER.

17.7 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01n1.)
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Update the DUART configuration register (DCR).
3. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
4. Set the data attributes and control bits of the external modem or peripheral device.
5. Set the interrupt enable register (UIER).
6. To start a write transfer, write to the UTHR.
7. Poll UIIR if the interrupts generated by the DUART are masked.

Chapter 18

PCI Express Interface Controller

18.1 Introduction

The PCI Express controller provides the mechanism to communicate with PCI Express devices.

The PCI Express interface is compatible with the *PCI Express™ Base Specification, Revision 2.0, with backward compatibility with 1.1*(available at <http://www.pcisig.org>). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol. The specific emphasis is directed at how the device implements the PCI Express specification. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express.

NOTE

Much of the available PCI Express literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Note that this is inconsistent with the terminology in the rest of this manual where the terms 'word' and 'double word' refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

The PCI Express controller connects the internal platform to a 5.0-GHz serial interface. The P4080 offers three instantiations of this controller yielding up to three PCI Express links simultaneously.

- PCI Express controller 1 (PEX1) is a x8 controller.
- PCI Express controller 2 (PEX2) is a x4 controller.
- PCI Express controller 3 (PEX3) is a x4 controller.

NOTE

The actual link width depends on the reset configuration and link training.

The remainder of this chapter refers to a single PCI Express controller offering up to a x8 link interface. Notes are included to indicate variations for multiple instantiations.

As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. Upon coming out of reset, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner. Once link autonegotiation is successful, the controller is in operation.

Internally, the design contains queues to keep track of inbound and outbound transactions. There is control logic that handles buffer management, bus protocol, transaction spawning and tag generation. In addition, there are memory blocks used to store inbound and outbound data.

The PCI Express controller can be configured to operate as either a PCI Express root complex (RC) or an endpoint (EP) device. An RC device connects the host CPU/memory subsystem to I/O devices while an EP device typically denotes a peripheral or I/O device. In RC mode, a PCI Express type 1 configuration header is used; in EP mode, a PCI Express type 0 configuration header is used.

This figure shows a high-level block diagram of the PCI Express controller.

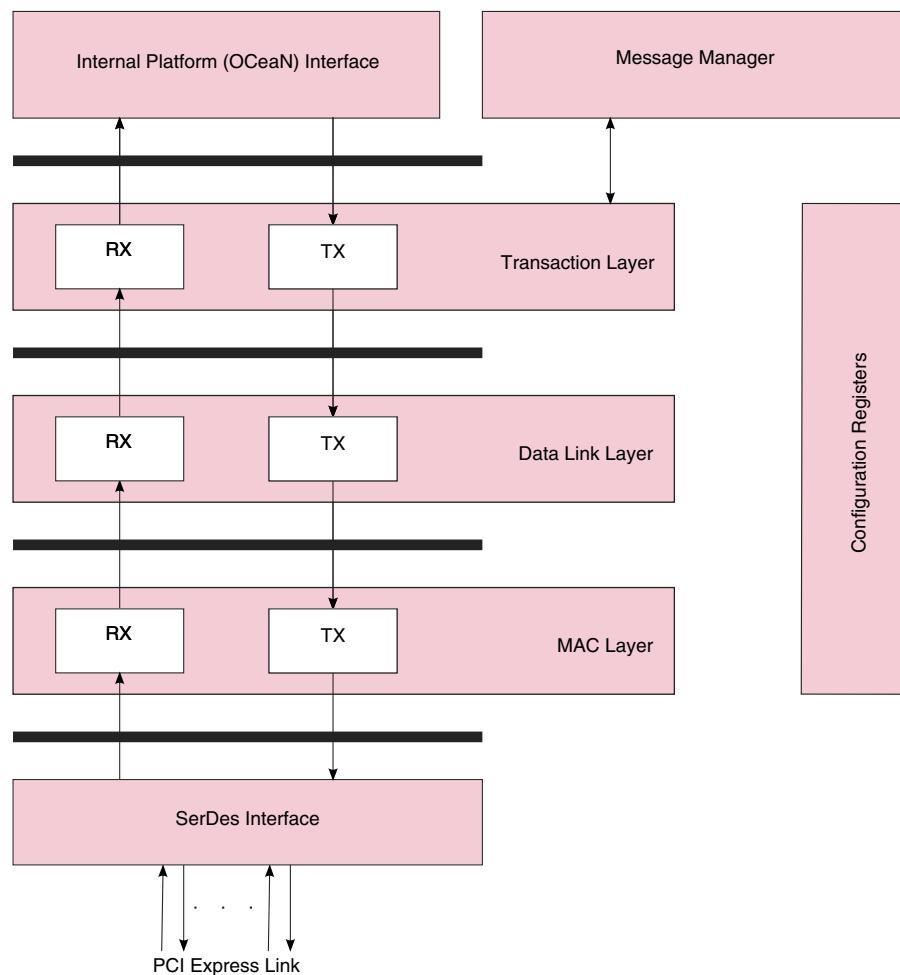


Figure 18-1. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. In addition, configuration and I/O transactions are supported if the PCI Express controller is in RC mode. As a target interface, the PCI Express controller accepts read and write operations to local memory space. When configured as an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers. Message generation and acceptance are supported in both RC and EP modes. Locked transactions and inbound I/O transactions are not supported.

18.1.1 Outbound transactions

Outbound internal platform transactions to PCI Express are first mapped to a translation window to determine what PCI Express transactions are to be issued.

A transaction from the internal platform can become a PCI Express Memory, I/O, Message, or Configuration transaction depending on the window attributes.

A transaction may be broken up into smaller sized transactions depending on the original request size, transaction type, and either the PCI Express device control register [MAX_PAYLOAD_SIZE] field for write requests or the PCI Express device control register [MAX_READ_SIZE] field for read requests. The controller performs PCI Express ordering rule checking to determine which transaction is to be sent on the PCI Express link.

In general, transactions are serviced in the order that they are received from the internal platform . Only when there is a stalled condition does the controller apply PCI Express ordering rules to outstanding transactions. For posted write transactions, once all data has been received from the internal platform , the data is forwarded to the PCI Express link and the transaction is considered as done. For non-posted write transactions, the controller waits for the completion packets to return before considering the transaction finished. For non-posted read transactions, the controller waits for all completion packets to return and then forwards all data back to the internal platform before terminating the transaction.

Note that after reset or when recovering from a link down condition, external transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external requests.

18.1.2 Inbound transactions

Inbound PCI Express transactions to internal platform are first mapped to a translation window to determine what internal platform transactions are to be issued.

A transaction may be broken up into smaller sized transactions when sending to the internal platform depending on the original request size, byte enables and starting/ending addresses. The controller performs PCI Express ordering rule checking to determine what transaction is to be sent next to the internal platform .

In general, transactions are serviced in the order that they are received from the PCI Express link. Only when there is a stalled condition does the controller apply PCI Express ordering to outstanding transactions. For posted write transactions, once all data has been received from the PCI Express link, the data is forwarded to the internal platform and the transaction is considered as done. For non-posted read transactions, the controller forwards internal platform data back to the PCI Express link.

Note that the controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

18.2 PCI Express features summary

The following is a list of features supported by the PCI Express controller:

- Compatible with the *PCI Express Base Specification, Revision 2.0, with backward compatibility with 1.1*
- Supports root complex (RC) and endpoint (EP) configurations
- 32- and 64-bit PCI Express address support
- 36-bit internal platform address support
- x8, x4, x2, and x1 link support
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes
- Supports strong and relaxed transaction ordering rules
- Enforces outbound PCI Express ordering rules and inbound internal platform priority
- PCI Express configuration registers (type 0 in EP mode, type 1 in RC mode)
- Baseline and advanced error reporting support
- One virtual channel (VC0)
- 256-byte maximum payload size (MAX_PAYLOAD_SIZE)
- Supports three inbound general-purpose translation windows, one 64-bit MSI window (RC only), and one configuration window
- Supports 32- or 64-bit MSI interrupts
- Supports four outbound translation windows and one default window
- Supports eight non-posted and four (with SRIO) or six (without SRIO) posted PCI Express inbound transactions
- Supports up to six internal platform reads and eight internal platform writes for outbound transaction. (The maximum number of outstanding transactions at any given time is eight.)
- Credit-based flow control management
- Supports PCI Express messages and interrupts
- Accepts up to 256-byte transactions from the internal platform

18.3 PCI Express modes of operation

Several parameters that affect the PCI Express controller modes of operation are determined at power-on reset (POR) by reset configuration word (RCW) fields as described in the following table.

Table 18-1. POR parameters for PCI Express controller

RCW parameter	Description
Host/Agent Configuration HOST_AGT_Bn	Selects between root complex (RC) and endpoint (EP) modes
SerDes Protocol SRDS_PRTCL	Determines the link width
SerDes clock ratio and SerDes clock divider SRDS_RATIO_Bn, SRDS_DIV_Bn	Determine the link speed

18.3.1 Root complex/endpoint modes

The PCI Express controller can function as either a root complex (RC) or an endpoint (EP) on the PCI Express link.

The host/agent configuration field, RCW[HOST_AGT_Bn] determines the RC/EP mode for the corresponding SerDes bank. See [Reset Configuration Word \(RCW\)](#), for more information.

18.3.2 Link width

The initial link width is determined by the SerDes protocol configuration field (RCW[SRDS_PRTCL]). See [Reset Configuration Word \(RCW\)](#) for more information. The specific configurations are detailed in [SerDes Lane Assignments and Multiplexing](#).

18.3.3 Link speed

The initial link speed is determined by the SerDes clock ratio field (RCW[SRDS_RATIO_Bn]) and the SerDes clock divider field (RCW[SRDS_DIV_Bn]).

See [RCW Field Definitions](#) for more information. The specific configurations are detailed in [Reference Clocks for SerDes Protocols](#).

18.4 PCI Express signal descriptions

This topic describes the signals of the PCI Express controller.

PCI Express defines the connection between two devices as a link, which can be composed of a single or multiple lanes. Each lane consists of a differential pair for transmitting (SD_n_TX and SD_n_TX_B) and a differential pair for receiving (SD_n_RX and SD_n_RX_B) with an embedded data clock.

This table contains detailed descriptions of the external PCI Express interface signals.

Table 18-2. PCI Express interface signals-Detailed signal descriptions

Signal	I/O	Description	
SD_RX[n]	I	Receive data. The receive data signals carry PCI Express packet information.	
		Refer to SerDes Lane Assignments and Multiplexing , for specific lane assignments. Note that lane reversal may affect the logical lane assignment. Refer to Lane reversal for more information.	
		State Meaning	Asserted/Negated-PCI Express interface.
		Timing	Assertion/Negation-As described in the <i>PCI Express Base Specification, Revision 2.0</i> .
SD_RX[n]_B	I	Receive data, inverted. SD_RX[n]_B are the inverted forms of the receive data signals (SD_RX[n]).	
		State Meaning	Asserted/Negated-Represents the inverse of data being received from the PCI Express interface.
		Timing	Assertion/Negation-As described in the <i>PCI Express Base Specification, Revision 2.0</i> .
SD_TX[n]	O	Transmit data. The transmit data signals carry PCI Express packet information.	
		Refer to SerDes Lane Assignments and Multiplexing , for specific lane assignments. Note that lane reversal may affect the logical lane assignment. Refer to Lane reversal , for more information.	
		State Meaning	Asserted/Negated-Represents data being transmitted to the PCI Express interface.
SD_TX[n]_B	O	Transmit data, inverted. SD_TX[n]_B are the inverted form of the transmit data signals SD_TX[n].	
		State Meaning	Asserted/Negated-Represents the inverse of data being transmitted to the PCI Express interface.

Table continues on the next page...

Table 18-2. PCI Express interface signals-Detailed signal descriptions (continued)

Signal	I/O	Timing	Description
		Timing	Assertion/Negation-As described in the <i>PCI Express Base Specification, Revision 2.0</i> .

18.5 Memory map/register overview

The PCI Express interface supports the following register types:

- Memory-mapped registers-these registers control PCI Express address translation, PCI error management, and PCI Express configuration register access. These registers are described in the sections below.
- PCI Express configuration registers contained within the PCI Express configuration space-these registers are specified by the PCI Express specification for every PCI Express device. These registers are described in [PCI Express configuration space access](#) and its subsections.

18.6 PCI Express memory-mapped registers

The PCI Express memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PEXCSRBAR on the PCI Express side) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI Express configuration data register, PEX_CONFIG_DATA) must only be accessed as 32-bit quantities.

Also note that although the table explicitly lists only the registers for the PCI Express controller 1, the register maps for PCI Express controllers 2 and 3 are the same except for the block base address. Memory-mapped registers for PCI Express controller 1 begin at block base address 0x20_0000, the registers for PCI Express controller 2 begin at 0x20_1000, and the registers for PCI Express controller 3 begin at 0x20_2000.

PEX memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_0000	PCI Express configuration address register (PEX1_PEX_CONFIG_ADDR)	32	R/W	0000_0000h	18.6.1/1130
20_0004	PCI Express configuration data register (PEX1_PEX_CONFIG_DATA)	32	R/W	0000_0000h	18.6.2/1131
20_000C	PCI Express outbound completion timeout register (PEX1_PEX_OTB_CPL_TOR)	32	R/W	0013_FFFFh	18.6.3/1132
20_0010	PCI Express configuration retry timeout register (PEX1_PEX_CONF_RTY_TOR)	32	R/W	0400_FFFFh	18.6.4/1133
20_0014	PCI Express configuration register (PEX1_PEX_CONFIG)	32	R/W	0004_0028h	18.6.5/1134
20_0020	PCI Express PME & message detect register (PEX1_PEX_PME_MES_DR)	32	w1c	0000_0000h	18.6.6/1136
20_0024	PCI Express PME & message disable register (PEX1_PEX_PME_MES_DISR)	32	R/W	0000_0000h	18.6.7/1138
20_0028	PCI Express PME & message interrupt enable register (PEX1_PEX_PME_MES_IER)	32	R/W	0000_0000h	18.6.8/1140
20_002C	PCI Express power management command register (PEX1_PEX_PMCR)	32	R/W	0000_0000h	18.6.9/1142
20_0BF8	IP block revision register 1 (PEX1_PEX_IP_BLK_REV1)	32	R	0208_0201h	18.6.10/1142
20_0BFC	IP block revision register 2 (PEX1_PEX_IP_BLK_REV2)	32	R	0000_0000h	18.6.11/1143
20_0C00	PCI Express outbound translation address register (PEX1_PEXOTAR0)	32	R/W	0000_0000h	18.6.12/1143
20_0C04	PCI Express outbound translation extended address register n (PEX1_PEXOTEAR0)	32	R/W	0000_0000h	18.6.13/1144
20_0C10	PCI Express outbound window attributes register n (PEX1_PEXOWAR0)	32	R/W	8004_4023h	18.6.14/1145
20_0C20	PCI Express outbound translation address register (PEX1_PEXOTAR1)	32	R/W	0000_0000h	18.6.12/1143
20_0C24	PCI Express outbound translation extended address register n (PEX1_PEXOTEAR1)	32	R/W	0000_0000h	18.6.13/1144
20_0C28	PCI Express outbound window base address register n (PEX1_PEXOWBAR1)	32	R/W	0000_0000h	18.6.15/1147
20_0C30	PCI Express outbound window attributes register n (PEX1_PEXOWAR1)	32	R/W	0004_4023h	18.6.16/1148
20_0C40	PCI Express outbound translation address register (PEX1_PEXOTAR2)	32	R/W	0000_0000h	18.6.12/1143
20_0C44	PCI Express outbound translation extended address register n (PEX1_PEXOTEAR2)	32	R/W	0000_0000h	18.6.13/1144
20_0C48	PCI Express outbound window base address register n (PEX1_PEXOWBAR2)	32	R/W	0000_0000h	18.6.15/1147
20_0C50	PCI Express outbound window attributes register n (PEX1_PEXOWAR2)	32	R/W	0004_4023h	18.6.16/1148

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_0C60	PCI Express outbound translation address register (PEX1_PEXOTAR3)	32	R/W	0000_0000h	18.6.12/ 1143
20_0C64	PCI Express outbound translation extended address register n (PEX1_PEXOTEAR3)	32	R/W	0000_0000h	18.6.13/ 1144
20_0C68	PCI Express outbound window base address register n (PEX1_PEXOWBAR3)	32	R/W	0000_0000h	18.6.15/ 1147
20_0C70	PCI Express outbound window attributes register 3 (PEX1_PEXOWAR3)	32	R/W	0000_0000h	18.6.17/ 1151
20_0C80	PCI Express outbound translation address register (PEX1_PEXOTAR4)	32	R/W	0000_0000h	18.6.12/ 1143
20_0C84	PCI Express outbound translation extended address register n (PEX1_PEXOTEAR4)	32	R/W	0000_0000h	18.6.13/ 1144
20_0C88	PCI Express outbound window base address register n (PEX1_PEXOWBAR4)	32	R/W	0000_0000h	18.6.15/ 1147
20_0C90	PCI Express outbound window attributes register 4 (PEX1_PEXOWAR4)	32	R/W	0004_4023h	18.6.18/ 1154
20_0D00	PCI Express MSI inbound translation address register (PEX1_PEXMSIITAR)	32	R	0000_0000h	18.6.19/ 1156
20_0D08	PCI Express MSI inbound window base address register (PEX1_PEXMSIIBAR)	32	R/W	0000_0000h	18.6.20/ 1157
20_0D0C	PCI Express MSI inbound window base extended address register (PEX1_PEXMSIIBEAR)	32	R/W	0000_0000h	18.6.21/ 1157
20_0D10	PCI Express MSI inbound window attributes register (PEX1_PEXMSIIBAR)	32	R/W	See section	18.6.22/ 1158
20_0DA0	PCI Express inbound translation address register n (PEX1_PEXITAR3)	32	R/W	0000_0000h	18.6.23/ 1160
20_0DA8	PCI Express inbound window base address register n (PEX1_PEXIWBAR3)	32	R/W	0000_0000h	18.6.24/ 1161
20_0DAC	PCI Express inbound window base extended address register n (PEX1_PEXIWBEAR3)	32	R/W	0000_0000h	18.6.25/ 1161
20_0DB0	PCI Express inbound window attributes register n (PEX1_PEXIWAR3)	32	R/W	20F4_4023h	18.6.26/ 1162
20_0DC0	PCI Express inbound translation address register n (PEX1_PEXITAR2)	32	R/W	0000_0000h	18.6.23/ 1160
20_0DC8	PCI Express inbound window base address register n (PEX1_PEXIWBAR2)	32	R/W	0000_0000h	18.6.24/ 1161
20_0DCC	PCI Express inbound window base extended address register n (PEX1_PEXIWBEAR2)	32	R/W	0000_0000h	18.6.25/ 1161
20_0DD0	PCI Express inbound window attributes register n (PEX1_PEXIWAR2)	32	R/W	20F4_4023h	18.6.26/ 1162
20_0DE0	PCI Express inbound translation address register n (PEX1_PEXITAR1)	32	R/W	0000_0000h	18.6.23/ 1160
20_0DE8	PCI Express inbound window base address register n (PEX1_PEXIWBAR1)	32	R/W	0000_0000h	18.6.24/ 1161

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_0DF0	PCI Express inbound window attributes register n (PEX1_PEXIWAR1)	32	R/W	20F4_4023h	18.6.26/ 1162
20_0E00	PCI Express error detect register (PEX1_PEX_ERR_DR)	32	w1c	0000_0000h	18.6.27/ 1165
20_0E08	PCI Express error interrupt enable register (PEX1_PEX_ERR_EN)	32	R/W	0000_0000h	18.6.28/ 1168
20_0E10	PCI Express error disable register (PEX1_PEX_ERR_DISR)	32	R/W	0000_0000h	18.6.29/ 1171
20_0E20	PCI Express error capture status register (PEX1_PEX_ERR_CAP_STAT)	32	R/W	0000_0000h	18.6.30/ 1173
20_0E28	PCI Express error capture register n (PEX1_PEX_ERR_CAP_R0)	32	R/W	0000_0000h	18.6.31/ 1174
20_0E2C	PCI Express error capture register n (PEX1_PEX_ERR_CAP_R1)	32	R/W	0000_0000h	18.6.31/ 1174
20_0E30	PCI Express error capture register n (PEX1_PEX_ERR_CAP_R2)	32	R/W	0000_0000h	18.6.31/ 1174
20_0E34	PCI Express error capture register n (PEX1_PEX_ERR_CAP_R3)	32	R/W	0000_0000h	18.6.31/ 1174
20_1000	PCI Express configuration address register (PEX2_PEX_CONFIG_ADDR)	32	R/W	0000_0000h	18.6.1/1130
20_1004	PCI Express configuration data register (PEX2_PEX_CONFIG_DATA)	32	R/W	0000_0000h	18.6.2/1131
20_100C	PCI Express outbound completion timeout register (PEX2_PEX_OTB_CPL_TOR)	32	R/W	0013_FFFFh	18.6.3/1132
20_1010	PCI Express configuration retry timeout register (PEX2_PEX_CONF_RTY_TOR)	32	R/W	0400_FFFFh	18.6.4/1133
20_1014	PCI Express configuration register (PEX2_PEX_CONFIG)	32	R/W	0004_0028h	18.6.5/1134
20_1020	PCI Express PME & message detect register (PEX2_PEX_PME_MES_DR)	32	w1c	0000_0000h	18.6.6/1136
20_1024	PCI Express PME & message disable register (PEX2_PEX_PME_MES_DISR)	32	R/W	0000_0000h	18.6.7/1138
20_1028	PCI Express PME & message interrupt enable register (PEX2_PEX_PME_MES_IER)	32	R/W	0000_0000h	18.6.8/1140
20_102C	PCI Express power management command register (PEX2_PEX_PMCR)	32	R/W	0000_0000h	18.6.9/1142
20_1BF8	IP block revision register 1 (PEX2_PEX_IP_BLK_REV1)	32	R	0208_0201h	18.6.10/ 1142
20_1BFC	IP block revision register 2 (PEX2_PEX_IP_BLK_REV2)	32	R	0000_0000h	18.6.11/ 1143
20_1C00	PCI Express outbound translation address register (PEX2_PEXOTAR0)	32	R/W	0000_0000h	18.6.12/ 1143
20_1C04	PCI Express outbound translation extended address register n (PEX2_PEXOTEAR0)	32	R/W	0000_0000h	18.6.13/ 1144

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_1C10	PCI Express outbound window attributes register n (PEX2_PEXOWAR0)	32	R/W	8004_4023h	18.6.14/ 1145
20_1C20	PCI Express outbound translation address register (PEX2_PEXOTAR1)	32	R/W	0000_0000h	18.6.12/ 1143
20_1C24	PCI Express outbound translation extended address register n (PEX2_PEXOTEAR1)	32	R/W	0000_0000h	18.6.13/ 1144
20_1C28	PCI Express outbound window base address register n (PEX2_PEXOWBAR1)	32	R/W	0000_0000h	18.6.15/ 1147
20_1C30	PCI Express outbound window attributes register n (PEX2_PEXOWAR1)	32	R/W	0004_4023h	18.6.16/ 1148
20_1C40	PCI Express outbound translation address register (PEX2_PEXOTAR2)	32	R/W	0000_0000h	18.6.12/ 1143
20_1C44	PCI Express outbound translation extended address register n (PEX2_PEXOTEAR2)	32	R/W	0000_0000h	18.6.13/ 1144
20_1C48	PCI Express outbound window base address register n (PEX2_PEXOWBAR2)	32	R/W	0000_0000h	18.6.15/ 1147
20_1C50	PCI Express outbound window attributes register n (PEX2_PEXOWAR2)	32	R/W	0004_4023h	18.6.16/ 1148
20_1C60	PCI Express outbound translation address register (PEX2_PEXOTAR3)	32	R/W	0000_0000h	18.6.12/ 1143
20_1C64	PCI Express outbound translation extended address register n (PEX2_PEXOTEAR3)	32	R/W	0000_0000h	18.6.13/ 1144
20_1C68	PCI Express outbound window base address register n (PEX2_PEXOWBAR3)	32	R/W	0000_0000h	18.6.15/ 1147
20_1C70	PCI Express outbound window attributes register 3 (PEX2_PEXOWAR3)	32	R/W	0000_0000h	18.6.17/ 1151
20_1C80	PCI Express outbound translation address register (PEX2_PEXOTAR4)	32	R/W	0000_0000h	18.6.12/ 1143
20_1C84	PCI Express outbound translation extended address register n (PEX2_PEXOTEAR4)	32	R/W	0000_0000h	18.6.13/ 1144
20_1C88	PCI Express outbound window base address register n (PEX2_PEXOWBAR4)	32	R/W	0000_0000h	18.6.15/ 1147
20_1C90	PCI Express outbound window attributes register 4 (PEX2_PEXOWAR4)	32	R/W	0004_4023h	18.6.18/ 1154
20_1D00	PCI Express MSI inbound translation address register (PEX2_PEXMSIITAR)	32	R	0000_0000h	18.6.19/ 1156
20_1D08	PCI Express MSI inbound window base address register (PEX2_PEXMSIIBAR)	32	R/W	0000_0000h	18.6.20/ 1157
20_1D0C	PCI Express MSI inbound window base extended address register (PEX2_PEXMSIIBEAR)	32	R/W	0000_0000h	18.6.21/ 1157
20_1D10	PCI Express MSI inbound window attributes register (PEX2_PEXMSIIBAR)	32	R/W	See section	18.6.22/ 1158
20_1DA0	PCI Express inbound translation address register n (PEX2_PEXITAR3)	32	R/W	0000_0000h	18.6.23/ 1160

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_1DA8	PCI Express inbound window base address register n (PEX2_PEXIWBAR3)	32	R/W	0000_0000h	18.6.24/ 1161
20_1DAC	PCI Express inbound window base extended address register n (PEX2_PEXIWBEAR3)	32	R/W	0000_0000h	18.6.25/ 1161
20_1DB0	PCI Express inbound window attributes register n (PEX2_PEXIWAR3)	32	R/W	20F4_4023h	18.6.26/ 1162
20_1DC0	PCI Express inbound translation address register n (PEX2_PEXITAR2)	32	R/W	0000_0000h	18.6.23/ 1160
20_1DC8	PCI Express inbound window base address register n (PEX2_PEXIWBAR2)	32	R/W	0000_0000h	18.6.24/ 1161
20_1DCC	PCI Express inbound window base extended address register n (PEX2_PEXIWBEAR2)	32	R/W	0000_0000h	18.6.25/ 1161
20_1DD0	PCI Express inbound window attributes register n (PEX2_PEXIWAR2)	32	R/W	20F4_4023h	18.6.26/ 1162
20_1DE0	PCI Express inbound translation address register n (PEX2_PEXITAR1)	32	R/W	0000_0000h	18.6.23/ 1160
20_1DE8	PCI Express inbound window base address register n (PEX2_PEXIWBAR1)	32	R/W	0000_0000h	18.6.24/ 1161
20_1DF0	PCI Express inbound window attributes register n (PEX2_PEXIWAR1)	32	R/W	20F4_4023h	18.6.26/ 1162
20_1E00	PCI Express error detect register (PEX2_PEX_ERR_DR)	32	w1c	0000_0000h	18.6.27/ 1165
20_1E08	PCI Express error interrupt enable register (PEX2_PEX_ERR_EN)	32	R/W	0000_0000h	18.6.28/ 1168
20_1E10	PCI Express error disable register (PEX2_PEX_ERR_DISR)	32	R/W	0000_0000h	18.6.29/ 1171
20_1E20	PCI Express error capture status register (PEX2_PEX_ERR_CAP_STAT)	32	R/W	0000_0000h	18.6.30/ 1173
20_1E28	PCI Express error capture register n (PEX2_PEX_ERR_CAP_R0)	32	R/W	0000_0000h	18.6.31/ 1174
20_1E2C	PCI Express error capture register n (PEX2_PEX_ERR_CAP_R1)	32	R/W	0000_0000h	18.6.31/ 1174
20_1E30	PCI Express error capture register n (PEX2_PEX_ERR_CAP_R2)	32	R/W	0000_0000h	18.6.31/ 1174
20_1E34	PCI Express error capture register n (PEX2_PEX_ERR_CAP_R3)	32	R/W	0000_0000h	18.6.31/ 1174
20_2000	PCI Express configuration address register (PEX3_PEX_CONFIG_ADDR)	32	R/W	0000_0000h	18.6.1/1130
20_2004	PCI Express configuration data register (PEX3_PEX_CONFIG_DATA)	32	R/W	0000_0000h	18.6.2/1131
20_200C	PCI Express outbound completion timeout register (PEX3_PEX_OTB_CPL_TOR)	32	R/W	0013_FFFFh	18.6.3/1132
20_2010	PCI Express configuration retry timeout register (PEX3_PEX_CONF_RTY_TOR)	32	R/W	0400_FFFFh	18.6.4/1133

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_2014	PCI Express configuration register (PEX3_PEX_CONFIG)	32	R/W	0004_0028h	18.6.5/1134
20_2020	PCI Express PME & message detect register (PEX3_PEX_PME_MES_DR)	32	w1c	0000_0000h	18.6.6/1136
20_2024	PCI Express PME & message disable register (PEX3_PEX_PME_MES_DISR)	32	R/W	0000_0000h	18.6.7/1138
20_2028	PCI Express PME & message interrupt enable register (PEX3_PEX_PME_MES_IER)	32	R/W	0000_0000h	18.6.8/1140
20_202C	PCI Express power management command register (PEX3_PEX_PMCR)	32	R/W	0000_0000h	18.6.9/1142
20_2BF8	IP block revision register 1 (PEX3_PEX_IP_BLK_REV1)	32	R	0208_0201h	18.6.10/1142
20_2BFC	IP block revision register 2 (PEX3_PEX_IP_BLK_REV2)	32	R	0000_0000h	18.6.11/1143
20_2C00	PCI Express outbound translation address register (PEX3_PEXOTAR0)	32	R/W	0000_0000h	18.6.12/1143
20_2C04	PCI Express outbound translation extended address register n (PEX3_PEXOTEAR0)	32	R/W	0000_0000h	18.6.13/1144
20_2C10	PCI Express outbound window attributes register n (PEX3_PEXOWAR0)	32	R/W	8004_4023h	18.6.14/1145
20_2C20	PCI Express outbound translation address register (PEX3_PEXOTAR1)	32	R/W	0000_0000h	18.6.12/1143
20_2C24	PCI Express outbound translation extended address register n (PEX3_PEXOTEAR1)	32	R/W	0000_0000h	18.6.13/1144
20_2C28	PCI Express outbound window base address register n (PEX3_PEXOWBAR1)	32	R/W	0000_0000h	18.6.15/1147
20_2C30	PCI Express outbound window attributes register n (PEX3_PEXOWAR1)	32	R/W	0004_4023h	18.6.16/1148
20_2C40	PCI Express outbound translation address register (PEX3_PEXOTAR2)	32	R/W	0000_0000h	18.6.12/1143
20_2C44	PCI Express outbound translation extended address register n (PEX3_PEXOTEAR2)	32	R/W	0000_0000h	18.6.13/1144
20_2C48	PCI Express outbound window base address register n (PEX3_PEXOWBAR2)	32	R/W	0000_0000h	18.6.15/1147
20_2C50	PCI Express outbound window attributes register n (PEX3_PEXOWAR2)	32	R/W	0004_4023h	18.6.16/1148
20_2C60	PCI Express outbound translation address register (PEX3_PEXOTAR3)	32	R/W	0000_0000h	18.6.12/1143
20_2C64	PCI Express outbound translation extended address register n (PEX3_PEXOTEAR3)	32	R/W	0000_0000h	18.6.13/1144
20_2C68	PCI Express outbound window base address register n (PEX3_PEXOWBAR3)	32	R/W	0000_0000h	18.6.15/1147
20_2C70	PCI Express outbound window attributes register 3 (PEX3_PEXOWAR3)	32	R/W	0000_0000h	18.6.17/1151

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_2C80	PCI Express outbound translation address register (PEX3_PEXOTAR4)	32	R/W	0000_0000h	18.6.12/ 1143
20_2C84	PCI Express outbound translation extended address register n (PEX3_PEXOTEAR4)	32	R/W	0000_0000h	18.6.13/ 1144
20_2C88	PCI Express outbound window base address register n (PEX3_PEXOWBAR4)	32	R/W	0000_0000h	18.6.15/ 1147
20_2C90	PCI Express outbound window attributes register 4 (PEX3_PEXOWAR4)	32	R/W	0004_4023h	18.6.18/ 1154
20_2D00	PCI Express MSI inbound translation address register (PEX3_PEXMSIITAR)	32	R	0000_0000h	18.6.19/ 1156
20_2D08	PCI Express MSI inbound window base address register (PEX3_PEXMSIIWBAR)	32	R/W	0000_0000h	18.6.20/ 1157
20_2D0C	PCI Express MSI inbound window base extended address register (PEX3_PEXMSIIWBAR)	32	R/W	0000_0000h	18.6.21/ 1157
20_2D10	PCI Express MSI inbound window attributes register (PEX3_PEXMSIIWAR)	32	R/W	See section	18.6.22/ 1158
20_2DA0	PCI Express inbound translation address register n (PEX3_PEXITAR3)	32	R/W	0000_0000h	18.6.23/ 1160
20_2DA8	PCI Express inbound window base address register n (PEX3_PEXIWBAR3)	32	R/W	0000_0000h	18.6.24/ 1161
20_2DAC	PCI Express inbound window base extended address register n (PEX3_PEXIWBAR3)	32	R/W	0000_0000h	18.6.25/ 1161
20_2DB0	PCI Express inbound window attributes register n (PEX3_PEXIWAR3)	32	R/W	20F4_4023h	18.6.26/ 1162
20_2DC0	PCI Express inbound translation address register n (PEX3_PEXITAR2)	32	R/W	0000_0000h	18.6.23/ 1160
20_2DC8	PCI Express inbound window base address register n (PEX3_PEXIWBAR2)	32	R/W	0000_0000h	18.6.24/ 1161
20_2DCC	PCI Express inbound window base extended address register n (PEX3_PEXIWBAR2)	32	R/W	0000_0000h	18.6.25/ 1161
20_2DD0	PCI Express inbound window attributes register n (PEX3_PEXIWAR2)	32	R/W	20F4_4023h	18.6.26/ 1162
20_2DE0	PCI Express inbound translation address register n (PEX3_PEXITAR1)	32	R/W	0000_0000h	18.6.23/ 1160
20_2DE8	PCI Express inbound window base address register n (PEX3_PEXIWBAR1)	32	R/W	0000_0000h	18.6.24/ 1161
20_2DF0	PCI Express inbound window attributes register n (PEX3_PEXIWAR1)	32	R/W	20F4_4023h	18.6.26/ 1162
20_2E00	PCI Express error detect register (PEX3_PEX_ERR_DR)	32	w1c	0000_0000h	18.6.27/ 1165
20_2E08	PCI Express error interrupt enable register (PEX3_PEX_ERR_EN)	32	R/W	0000_0000h	18.6.28/ 1168
20_2E10	PCI Express error disable register (PEX3_PEX_ERR_DISR)	32	R/W	0000_0000h	18.6.29/ 1171

Table continues on the next page...

PEX memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20_2E20	PCI Express error capture status register (PEX3_PEX_ERR_CAP_STAT)	32	R/W	0000_0000h	18.6.30/ 1173
20_2E28	PCI Express error capture register n (PEX3_PEX_ERR_CAP_R0)	32	R/W	0000_0000h	18.6.31/ 1174
20_2E2C	PCI Express error capture register n (PEX3_PEX_ERR_CAP_R1)	32	R/W	0000_0000h	18.6.31/ 1174
20_2E30	PCI Express error capture register n (PEX3_PEX_ERR_CAP_R2)	32	R/W	0000_0000h	18.6.31/ 1174
20_2E34	PCI Express error capture register n (PEX3_PEX_ERR_CAP_R3)	32	R/W	0000_0000h	18.6.31/ 1174

18.6.1 PCI Express configuration address register (PEXx_PEX_CONFIG_ADDR)

The PCI Express configuration address register contains address information for accesses to PCI Express internal and external configuration registers.

Both root complex (RC) and endpoint (EP) configuration headers contain 4096 bytes of address space. To access a register within the header, both the extended register number and the register number fields are concatenated to form the 4-byte aligned address of the register. That is, the register address is: Extended register number || Register number || 00b.

Address: Base address + 0h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved			EXTREGN				BUSN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DEVN				FUNCN				REGN						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEX_CONFIG_ADDR field descriptions

Field	Description
0 EN	Enable. This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed. If this bit is cleared, writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data.
1–3 -	This field is reserved. Reserved
4–7 EXTREGN	Extended register number. This field allows access to extended PCI Express configuration space (that is, the registers in the offset range from 0x100 to 0xFFFF).

Table continues on the next page...

PEXx_PEX_CONFIG_ADDR field descriptions (continued)

Field	Description
8–15 BUSN	Bus number. PCI bus number to access
16–20 DEVN	Device number. Device number to access on specified bus
21–23 FUNCN	Function number. Function to access within specified device
24–29 REGN	Register number. 32-bit register to access within specified device
30–31 -	This field is reserved. Reserved

18.6.2 PCI Express configuration data register (PEXx_PEX_CONFIG_DATA)

The PCI Express configuration data register is a 32-bit port for internal and external configuration access. Note that accesses of 1, 2, or 4 bytes to the PCI Express configuration data register are allowed. Also note that accesses to the little-endian PCI Express configuration space must be properly formatted. See [Byte order for configuration transactions](#) for more information.

Address: Base address + 4h offset

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

PEXx_PEX_CONFIG_DATA field descriptions

Field	Description
0–31 Data	A read or write to this register starts a PCI Express configuration cycle if the PEX_CONFIG enable bit is set (PEX_CONFIG_ADDR[EN] = 1).

18.6.3 PCI Express outbound completion timeout register (PEXx_PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register contains the maximum wait time for a response to come back as a result of an outbound non-posted request before a timeout condition occurs. This timeout register is also used to timeout requests that were not acknowledged (PEX_ERR_DET[PAT]) which is a fatal error.

Address: Base address + Ch offset

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

PEXx_PEX_OTB_CPL_TOR field descriptions

Field	Description
0 TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
1–5 -	This field is reserved. Reserved
6–31 TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. The completion timeout is determined by TC and the setting of the completion timeout disable bit (CPL_TOD) in Device Control 2 Register. NOTE: The maximum computed timeout value based on TC cannot exceed 1.34s at 400 MHz and 2.01s at 266.66 MHz. When Device Control 2[CPL_TOD] is cleared, the timeout value is based on TC and Device Control 2[CPL_TO_VAL]. When completion timeout value (CPL_TO_VAL) is: 0000 the timeout value is TC 0001 the timeout value is TC/64 0010 the timeout value is TC/4 0101 the timeout value is TC 0110 the timeout value is TCx4 1001 the timeout value is TCx16 1010 the timeout value is TCx64 For all other values of completion timeout the timeout value is TC.

Table continues on the next page...

PEXx_PEX_OTB_CPL_TOR field descriptions (continued)

Field	Description
	<p>When Device Control 2[CPL_TOD] is set, one TC unit is 8x the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz, and 30 ns at 266.66 MHz.</p> <p>The following are examples of timeout periods based on different TC settings:</p> <ul style="list-style-type: none"> 0x00_0000 Reserved 0x10_FFFF 22.28 ms at 400 MHz controller clock; 33.34 ms at 266.66 MHz controller clock 0xFF_FFFF 335.54 ms at 400 MHz controller clock; 503.31 ms at 266.66 MHz controller clock 0x3FF_FFFF 1.34 s at 400 MHz controller clock; 2.01 s at 266.66 MHz controller clock

18.6.4 PCI Express configuration retry timeout register (PEXx_PEX_CONF_RTY_TOR)

The PCI Express configuration retry timeout register contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.

Address: Base address + 10h offset

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

PEXx_PEX_CONF_RTY_TOR field descriptions

Field	Description
0 RD	<p>Retry disable. This bit disables the retry of a configuration transaction that receives a CRS status response packet.</p> <p>0 Enable retry of a configuration transaction in response to receiving a CRS status response until the timeout counter (defined by the PEX_CONF_RTY_TOR[TC] field) has expired.</p> <p>1 Disable retry of a configuration transaction regardless of receiving a CRS status response.</p>
1–3 -	This field is reserved. Reserved
4–31 TC	<p>Timeout counter. This is the value that is used to load the CRS response counter.</p> <p>One TC unit is 8x the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz and 30 ns at 266.66 MHz.</p> <p>Timeout period based on different TC settings:</p> <ul style="list-style-type: none"> 0x000_0000 Reserved

Table continues on the next page...

PEXx_PEX_CONF_RTY_TOR field descriptions (continued)

Field	Description
	0x400_FFFF 1.34 s at 400 MHz controller clock, 2.02 s at 266.66 MHz controller clock 0xFFF_FFFF 5.37 s at 400 MHz controller clock, 8.05 s at 266.66 MHz controller clock

18.6.5 PCI Express configuration register (PEXx_PEX_CONFIG)

The PCI Express configuration register contains various control switches for the controller.

Address: Base address + 14h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved														PC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved		OB_CK		Reserved						SAC		Reserved		SP	
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0

PEXx_PEX_CONFIG field descriptions

Field	Description
0–12 -	This field is reserved. Reserved
13–15 PC	Posted credits. Number of posted credits (PH, PD) supported. PC defaults to 4 PH and 64 PD. If SRIO is disabled or bridging is guaranteed not to occur, then PC can be changed to 6 PH/96 PD for better performance. Whether bridging is permitted is a system behavior not determined or checked by the hardware. System software may determine that bridging is not permitted (for example, by confirming there are no enabled inbound ATMU windows in PCI Express targeting SRIO, and no enabled inbound ATMU windows in SRIO targeting PCI Express), and change the value of PC to advertise more credits. NOTE: If PEX_CONFIG[PC] is altered by software, then the link needs to be retrained so that credits are re-advertised on the bus. All other encodings are reserved 100 4 PH and 64 PD 110 6 PH and 96 PD (only supported if SRIO is disabled or bridging is guaranteed not to occur)
16–17 -	This field is reserved. Reserved
18 OB_CK	Outbound transaction address checking enable.

Table continues on the next page...

PEXx_PEX_CONFIG field descriptions (continued)

Field	Description
	<p>0 Disable checking for all outbound transaction addresses (memory and I/O) against the base/limit registers. There is no checking of outbound addresses. This setting is compatible with the previous generation PCI Express controller behavior.</p> <p>1 Enable checking for all outbound addresses (memory and I/O) against the base/limit registers. If an outbound transaction address does not hit into the base/limit registers, it will cause an error.</p>
19–26 -	This field is reserved. Reserved
27 SAC	Sense ASPM Control. This bit controls the default value of ASPM of PEX Link Control Register's bit 0. See PCI Express Link Control Register (Link_Control_Register) for more information.
28–29 -	This field is reserved. Reserved
30 SP	Slot Present. This bit controls the default value of the PCI Express capabilities register [slot] bit. See PCI Express Capabilities Register (Capabilities_Register) for more information.
31 SCC	Slot Clock Configuration. This bit controls the default value of the PCI Express link status register [SCC] bit. See PCI Express Link Status Register (Link_Status_Register) for more information.

18.6.6 PCI Express PME & message detect register (PEXx_PEX_PME_MES_DR)

The PCI Express PME and message detect register logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.

Address: Base address + 20h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PTO		Reserved	ENL23	EXL23		HRD		LDD		LUD		-			
W	w1c			w1c	w1c				w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

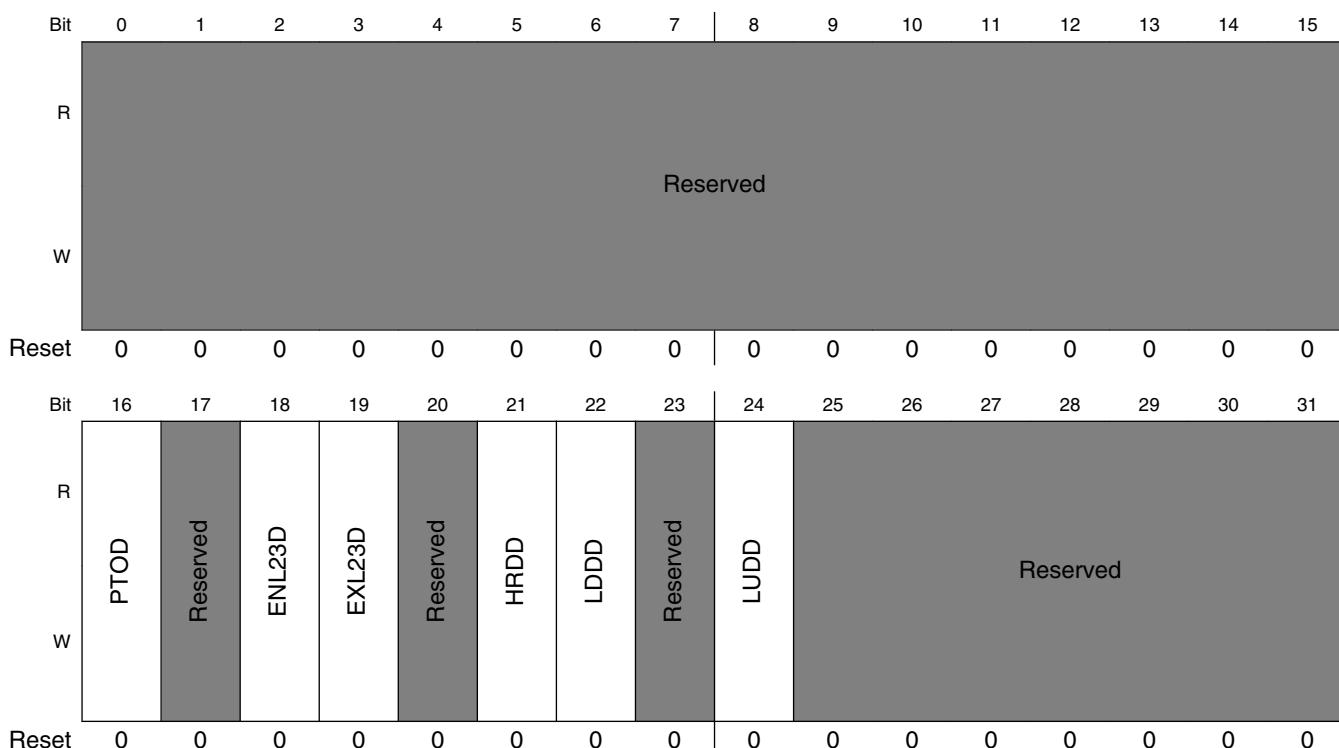
PEXx_PEX_PME_MES_DR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 PTO	PME turn off. This bit indicates the detection of a PME_Turn_Off message. This bit is only valid in EP mode. 1 A PME_Turn_Off_message is detected 0 No PME_Turn_Off message detected
17 -	This field is reserved. Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
18 ENL23	Entered L2/L3 ready state. This bit indicates that the PCI Express controller has entered L2/L3 state. This is only valid in RC mode. 1 L2/L3 ready state has been entered 0 L2/L3 ready state has not been entered
19 EXL23	Exit L2/L3 ready state. This bit indicates that the PCI Express controller has exited the L2/L3 state. This is only valid in RC mode. 1 Exit L2/L3 state has been detected 0 Exit L2/L3 state not detected
20 -	This field is reserved. Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
21 HRD	Hot reset detected. This bit indicates that the PCI Express controller has detected a hot reset condition on the link. The controller is reset and cleans up all outstanding transactions. Link retraining takes place once hot reset state is exited. This is valid only in EP mode. 1 Hot reset request has been detected 0 Hot reset request not detected
22 LDD	Link down detected. This bit indicates that a link down condition has been detected. The controller is reset and then cleans up all outstanding transactions. Link retraining takes place once the controller has cleaned itself up. Note that for EP, this bit and HRD are typically set when a hot reset event is detected. 1 Link down has been detected 0 Link down not detected
23 -	This field is reserved. Reserved
24 LUD	Link up detected. This bit indicates that a link up condition has been detected. It means that link training has been successful. 1 Link up detected 0 Link up not detected
25–31 -	Reserved. Note that during normal operation, these bits may be set (falsely). The bits may be ignored and cleared (w1c) without consequence.

18.6.7 PCI Express PME & message disable register (PEXx_PEX_PME_MES_DISR)

The PCI Express PME and message disable register when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.

Address: Base address + 24h offset



PEXx_PEX_PME_MES_DISR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 PTOD	PME turn off disable. When set, this bit disables the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
17 -	This field is reserved. Reserved
18 ENL23D	Entered_L2/L3 ready disable. When set, this bit disables the setting of PEX_PME_MES_DR[ENL23] bit. 1 Disable Entered_L2/L3 ready state detection 0 Enable Entered_L2/L3 ready state detection

Table continues on the next page...

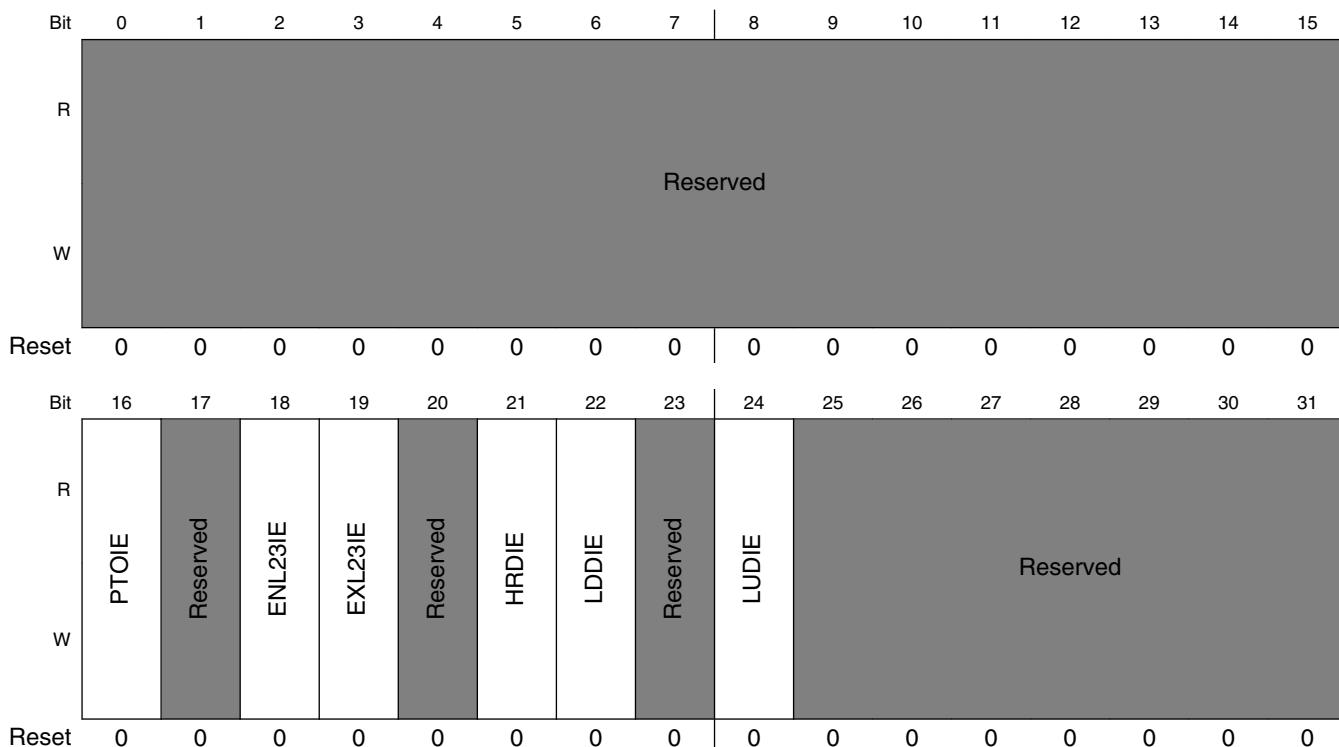
PEXx_PEX_PME_MES_DISR field descriptions (continued)

Field	Description
19 EXL23D	Exited_L2/L3 ready disable. When set, this bit disables the setting of PEX_PME_MES_DR[EXL23] bit. 1 Disable Exited_L2/L3 ready state detection 0 Enable Exited_L2/L3 ready state detection
20 -	This field is reserved. Reserved
21 HRDD	Hot reset detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[HRD] bit. 1 Disable hot reset state detection 0 Enable hot reset state detection
22 LDDD	Link down detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
23 -	This field is reserved. Reserved
24 LUDD	Link up detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[LUD] bit. 1 Disable link up detection 0 Enable link up detection
25–31 -	This field is reserved. Reserved

18.6.8 PCI Express PME & message interrupt enable register (PEXx_PEX_PME_MES_IER)

The PCI Express PME and message interrupt enable register allows for the detection of a message or a PME event to generate an interrupt, provided that the corresponding bit in the PCI Express PME and message detect register is set.

Address: Base address + 28h offset



PEXx_PEX_PME_MES_IER field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 generates an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
17 -	This field is reserved. Reserved
18 ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 generates an interrupt.

Table continues on the next page...

PEXx_PEX_PME_MES_IER field descriptions (continued)

Field	Description
	1 Enable Entered_L2/L3 ready state interrupt generation 0 Disable Entered_L2/L3 ready state interrupt generation
19 EXL23IE	Exited L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[EXL23]=1 generates an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
20 -	This field is reserved. Reserved
21 HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 generates an interrupt. 1 Enable hot reset state interrupt generation 0 Disable hot reset state interrupt generation
22 LDDIE	Link down detected interrupt enable. When set and PEX_PME_MES_DR[LDD]=1 generates an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation
23 -	This field is reserved. Reserved
24 LUDIE	Link up detected interrupt enable. When set and PEX_PME_MES_DR[LUD]=1 generates an interrupt. 1 Enable link up detected interrupt generation 0 Disable link up detected interrupt generation
25–31 -	This field is reserved. Reserved

18.6.9 PCI Express power management command register (PEXx_PEX_PMCR)

The PCI Express power management command register provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

Address: Base address + 2Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved												SPMES	EXL2S	PTOMR	
W													0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEX_PMCR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29 SPMES	Set PME status. This sets the PME status bit and if PME is enabled (see PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register) , for more information) it transmits a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
30 EXL2S	Exit L2 state. When set exits the link state out of L2/L3 ready state in order to send new requests. The request is only made when entered_L2/L3 ready state is active. This bit is self-clearing. When the link has exited L2/L3 ready state, the status bit Exit_L2/L3 ready state is set. This bit should not be used when in EP mode.
31 PTOMR	PME_Turn_Off message request. When set broadcasts a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

18.6.10 IP block revision register 1 (PEXx_PEX_IP_BLK_REV1)

Address: Base address + BF8h offset

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

PEXx_PEX_IP_BLK_REV1 field descriptions

Field	Description
0–15 IP_ID	Block ID
16–23 IP_MJ	Block Major Revision
24–31 IP_MN	Block Minor Revision

18.6.11 IP block revision register 2 (PEXx_PEX_IP_BLK_REV2)

Address: Base address + BFCh offset

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

PEXx_PEX_IP_BLK_REV2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IP_INT	Block integration option
16–23 -	This field is reserved. Reserved
24–31 IP_CFG	Block configuration option

18.6.12 PCI Express outbound translation address register (PEXx_PEXOTARn)

The PCI Express outbound translation address registers select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: Base address + C00h offset + (32d × i), where i=0d to 4d

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

PEXx_PEXOTAR*n* field descriptions

Field	Description
0-11 TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [43:32] (bit 32 is the lsb).
12-31 TA	Translation address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to PCI Express address bits [31:12].

18.6.13 PCI Express outbound translation extended address register n (PEXx_PEXOTEARn)

The PCI Express outbound translation extended address registers contain the most-significant bits of a 64 bit translation address.

Address: Base address + C04h offset + (32d × i), where i=0d to 4d

PEXx PEXOTEARn field descriptions

Field	Description
0–11 - Reserved	This field is reserved.
12–31 TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [63:44].

18.6.14 PCI Express outbound window attributes register n (PEXx_PEXOWAR0)

The PCI Express outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Address: Base address + C10h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN															
	DIEN	Reserved		ROE	NS	Reserved			TC			Reserved		RTT		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
				WTT					Reserved				OWS			
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1

PEXx_PEXOWAR0 field descriptions

Field	Description
0 EN	<p>Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1.</p> <p>0 Disable outbound translation window 1 Enable outbound translation window</p>
1 DIEN	<p>Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Byte ordering for more information.</p> <p>NOTE: The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.</p> <p>0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering.</p>
2 -	This field is reserved. Reserved
3 ROE	<p>Relaxed ordering enable. When this bit and the RO bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the relaxed ordering bit for the packet is enabled. This bit only applies to memory transactions.</p> <p>0 Default ordering 1 Relaxed ordering</p>
4 NS	<p>No snoop enable. When this bit and the NSE bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the no snoop bit for the packet is enabled. This bit only applies to memory transactions.</p> <p>0 Snoopable 1 No snoop</p>
5–7 -	This field is reserved. Reserved
8–10 TC	<p>Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0.</p> <p>NOTE: Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class.</p> <p>000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7</p>
11 -	This field is reserved. Reserved
12–15 RTT	<p>Read transaction type. Read transaction type to run on the PCI Express link.</p> <p>Settings not shown are reserved.</p> <p>0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.</p>

Table continues on the next page...

PEXx_PEXOWAR0 field descriptions (continued)

Field	Description
	0100 Memory read 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
16–19 WTT	Write transaction type. Write transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
20–25 -	This field is reserved. Reserved
26–31 OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$ -byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. Patterns not shown are reserved. 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size

18.6.15 PCI Express outbound window base address register n (PEXx_PEXOWBARn)

The PCI Express outbound window base address registers select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set.

Address: Base address + C28h offset + (32d × i), where i=0d to 3d

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

PEXx_PEXOWBARn field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–11 WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. Correspond to internal platform address bits [0:3]. (where 0 is the msb of the internal platform address)
12–31 WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to internal platform address bits [4:23].

18.6.16 PCI Express outbound window attributes register n (PEXx_PEXOWARn)

The PCI Express outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Address: Base address + C30h offset + (32d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	DIEN	Reserved	ROE	NS	Reserved			TC			Reserved		RTT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R			WTT		Reserved								OWS			
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1

PEXx_PEXOWARn field descriptions

Field	Description
0 EN	Enable. This bit enables this address translation window. 0 Disable outbound translation window 1 Enable outbound translation window
1 DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Byte ordering for more information. NOTE: The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering.
2 -	This field is reserved. Reserved
3 ROE	Relaxed ordering enable. When this bit and the RO bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the relaxed ordering bit for the packet is enabled. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4 NS	No snoop enable. When this bit and the NSE bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the no snoop bit for the packet is enabled. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7 -	This field is reserved. Reserved
8–10 TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. NOTE: Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7
11 -	This field is reserved. Reserved
12–15 RTT	Read transaction type. Read transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.

Table continues on the next page...

PEXx_PEXOWARn field descriptions (continued)

Field	Description
	0100 Memory read 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
16–19 WTT	Write transaction type. Write transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
20–25 -	This field is reserved. Reserved
26–31 OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$ -byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. 000000 Reserved 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved 111111 Reserved

18.6.17 PCI Express outbound window attributes register 3 (PEXx_PEXOWAR3)

The PCI Express outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Address: Base address + C70h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
EN	DIEN	Reserved	ROE	NS	Reserved				TC			Reserved			RTT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
WTT									Reserved				OWS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEXOWAR3 field descriptions

Field	Description
0 EN	Enable. This bit enables this address translation window. 0 Disable outbound translation window 1 Enable outbound translation window
1 DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Byte ordering for more information. NOTE: The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering.

Table continues on the next page...

PEXx_PEXOWAR3 field descriptions (continued)

Field	Description
2 -	This field is reserved. Reserved
3 ROE	Relaxed ordering enable. When this bit and the RO bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the relaxed ordering bit for the packet is enabled. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4 NS	No snoop enable. When this bit and the NSE bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the no snoop bit for the packet is enabled. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7 -	This field is reserved. Reserved
8–10 TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. NOTE: Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7
11 -	This field is reserved. Reserved
12–15 RTT	Read transaction type. Read transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory read 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
16–19 WTT	Write transaction type. Write transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.

Table continues on the next page...

PEXx_PEXOWAR3 field descriptions (continued)

Field	Description
20–25 -	This field is reserved. Reserved
26–31 OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$ -byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. 000000 Reserved 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved 111111 Reserved

18.6.18 PCI Express outbound window attributes register 4 (PEXx_PEXOWAR4)

The PCI Express outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Address: Base address + C90h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
EN	DIEN	Reserved	ROE	NS	Reserved				TC			Reserved			RTT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
WTT									Reserved				OWS			
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

PEXx_PEXOWAR4 field descriptions

Field	Description
0 EN	Enable. This bit enables this address translation window. 0 Disable outbound translation window 1 Enable outbound translation window
1 DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Byte ordering for more information. NOTE: The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering.

Table continues on the next page...

PEXx_PEXOWAR4 field descriptions (continued)

Field	Description
2 -	This field is reserved. Reserved
3 ROE	Relaxed ordering enable. When this bit and the RO bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the relaxed ordering bit for the packet is enabled. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4 NS	No snoop enable. When this bit and the NSE bit of the PCI Express device control register (described in PCI Express Device Control Register (Device_Control_Register)) is set, the no snoop bit for the packet is enabled. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7 -	This field is reserved. Reserved
8–10 TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. NOTE: Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7
11 -	This field is reserved. Reserved
12–15 RTT	Read transaction type. Read transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory read 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.
16–19 WTT	Write transaction type. Write transaction type to run on the PCI Express link. Settings not shown are reserved. 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary.

Table continues on the next page...

PEXx_PEXOWAR4 field descriptions (continued)

Field	Description
20–25 -	This field is reserved. Reserved
26–31 OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$ -byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 64-Gbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. 000000 Reserved 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved 111111 Reserved

18.6.19 PCI Express MSI inbound translation address register (PEXx_PEXMSIITAR)

The PCI Express MSI inbound translation address register contains the translated internal platform address to be used. The translated address is always mapped to CCSRBAR space similarly to PEXITAR0. This register only applies in RC mode.

Address: Base address + D00h offset

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

PEXx_PEXMSIITAR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–11 TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:3] where bit 0 is the msb of the internal platform address. Mapped to CCSRBAR.
12–31 TA	Translation address. Target address which indicates the starting point of the inbound translated address. Mapped to CCSRBAR.

18.6.20 PCI Express MSI inbound window base address register (PEXx_PEXMSIWIWBAR)

The PCI Express MSI inbound window base address register selects the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for the 64-bit MSI inbound transactions are compared to this window. This register only applies in RC mode.

Address: Base address + D08h offset

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

Reset 0

PEXx_PEXMSIWIWBAR field descriptions

Field	Description
0–11 WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32].
12–31 WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

18.6.21 PCI Express MSI inbound window base extended address register (PEXx_PEXMSIWIWBLEAR)

The PCI Express MSI inbound window base extended address register contains the most-significant bits of a 64-bit MSI base address. This register only applies in RC mode.

Address: Base address + D0Ch offset

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

Reset 0

PEXx_PEXMSIWIWBLEAR field descriptions

Field	Description
0–11 -Reserved	This field is reserved. Reserved
12–31 WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44].

18.6.22 PCI Express MSI inbound window attributes register (PEXx_PEXMSIWAR)

The PCI Express MSI inbound window attributes register defines the window sizes to translate and other attributes for the translations. The size is fixed at 16-Mbyte window. This register only applies in RC mode.

All other settings are reserved.

Address: Base address + D10h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	DIEN	PF	Reserved				TRGT				RTT				
W																
Reset	0	0	0	0	0	0	0	0	n	n	n	n	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WTT			Reserved				IWS								
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1

PEXx_PEXMSIWAR field descriptions

Field	Description
0 EN	Enable. This bit controls the enabling/disabling of the MSI translation window. 0 Disable inbound window translation 1 Enable inbound window translation
1 DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Byte ordering for more information. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering.
2 PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
3–7 -	This field is reserved. Reserved
8–11 TRGT	Target interface. NOTE: PEXMSIWAR n [TRGT] must not be set to target the PCI Express interface itself. That is, PEXMSIWAR n for PCI Express controller 1 should never target PCI Express 1. NOTE: If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.

Table continues on the next page...

PEXx_PEXMSIWAR field descriptions (continued)

Field	Description																										
	<p>NOTE: Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.</p> <table> <tr><td>0000</td><td>PCI Express 1-PCI Express controller 1 should not use this encoding</td></tr> <tr><td>0001</td><td>PCI Express 2-PCI Express controller 2 should not use this encoding</td></tr> <tr><td>0010</td><td>PCI Express 3-PCI Express controller 3 should not use this encoding</td></tr> <tr><td>0011-0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO 1</td></tr> <tr><td>1001</td><td>Serial RapidIO 2</td></tr> <tr><td>1010-1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1-PCI Express controller 1 should not use this encoding	0001	PCI Express 2-PCI Express controller 2 should not use this encoding	0010	PCI Express 3-PCI Express controller 3 should not use this encoding	0011-0111	Reserved	1000	Serial RapidIO 1	1001	Serial RapidIO 2	1010-1110	Reserved	1111	Local memory space										
0000	PCI Express 1-PCI Express controller 1 should not use this encoding																										
0001	PCI Express 2-PCI Express controller 2 should not use this encoding																										
0010	PCI Express 3-PCI Express controller 3 should not use this encoding																										
0011-0111	Reserved																										
1000	Serial RapidIO 1																										
1001	Serial RapidIO 2																										
1010-1110	Reserved																										
1111	Local memory space																										
12–15 RTT	<p>Read transaction type. Read transaction type to send to target interface.</p> <p>If the transaction is not going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>0100</td><td>Read</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>If the transaction is going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>0100</td><td>Read, do not snoop local processor</td></tr> <tr><td>0101</td><td>Read, snoop local processor</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table>	0000	Reserved	0100	Read	0101	Reserved	0110	Reserved	0111	Reserved	1000	Reserved	1111	Reserved	0000	Reserved	0100	Read, do not snoop local processor	0101	Read, snoop local processor	0110	Reserved	1000	Reserved	1111	Reserved
0000	Reserved																										
0100	Read																										
0101	Reserved																										
0110	Reserved																										
0111	Reserved																										
1000	Reserved																										
1111	Reserved																										
0000	Reserved																										
0100	Read, do not snoop local processor																										
0101	Read, snoop local processor																										
0110	Reserved																										
1000	Reserved																										
1111	Reserved																										
16–19 WTT	<p>Write transaction type. Write transaction type to send to target interface.</p> <p>If the transaction is not going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>0100</td><td>Write</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>If the transaction is going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>0100</td><td>Write, do not snoop local processor</td></tr> <tr><td>0101</td><td>Write, snoop local processor</td></tr> </table>	0000	Reserved	0100	Write	0101	Reserved	0110	Reserved	0111	Reserved	1000	Reserved	1111	Reserved	0000	Reserved	0100	Write, do not snoop local processor	0101	Write, snoop local processor						
0000	Reserved																										
0100	Write																										
0101	Reserved																										
0110	Reserved																										
0111	Reserved																										
1000	Reserved																										
1111	Reserved																										
0000	Reserved																										
0100	Write, do not snoop local processor																										
0101	Write, snoop local processor																										

Table continues on the next page...

PEXx_PEXMSIWAR field descriptions (continued)

Field	Description
	1000 Reserved 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 IWS	Inbound window size. All other settings are reserved. 010111 16-Mbyte window size

18.6.23 PCI Express inbound translation address register n (PEXx_PEXITARn)

The PCI Express inbound translation address registers contain the translated internal platform address to be used. Note that PEXITAR0 does not exist in the memory-mapped space; it is a fixed translation to the internal configuration (CCSR) space.

Address: Base address + DA0h offset + (32d × i), where i=0d to 2d

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

Reset 0

PEXx_PEXITARn field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–11 TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:3] where bit 0 is the msb of the internal platform address.
12–31 TA	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to internal platform address bits [4:23].

18.6.24 PCI Express inbound window base address register n (PEXx_PEXIWBARn)

The PCI Express inbound window base address registers select the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for inbound transactions are compared to these windows. In RC mode, the BAR for inbound window 0 is located in the PCI Express type 1 configuration header space and the BARs for inbound windows 1-3 (PEXIWBAR[1-3]) are implemented as described in this section. In endpoint (EP) mode, these registers are not implemented in the memory-mapped space. Reading these registers in EP mode returns all zeros and writing to these offsets has no consequences. All base address registers in EP are located in the PCI Express type 0 configuration header space. Note that PEXIWBAR1 only supports 32-bit PCI Express address space.

Address: Base address + DA8h offset + (32d × i), where i=0d to 2d

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

PEXx_PEXIWBARn field descriptions

Field	Description
0–11 WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32]. Note that the extended address is supported for windows 2 and 3 only; for PEXIWBAR1, these bits are reserved and must be 0.
12–31 WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

18.6.25 PCI Express inbound window base extended address register n (PEXx_PEXIWBEARn)

The PCI Express inbound window base extended address registers contain the most-significant bits of a 64 bit base address.

Address: Base address + DACH offset + (32d × i), where i=0d to 1d

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

PEXx_PEXIWB n field descriptions

Field	Description
0–11 -	This field is reserved. Reserved
12–31 WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

**18.6.26 PCI Express inbound window attributes register n
(PEXx_PEXIWAR n)**

The PCI Express inbound window attributes registers define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Address: Base address + DB0h offset + (32d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	DIEN	PF	Reserved					TRGT			RTT				
Reset	0	0	1	0	0	0	0	0	1	1	1	1	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTT				Reserved				IWS							
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

PEXx_PEXIWAR n field descriptions

Field	Description
0 EN	Enable. This bit controls the enabling/disabling of the translation window. 0 Disable inbound window translation 1 Enable inbound window translation
1 DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Byte ordering for more information. NOTE: The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering.
2 PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
3–7 -	This field is reserved. Reserved

Table continues on the next page...

PEXx_PEXIWARn field descriptions (continued)

Field	Description																																		
8–11 TRGT	<p>Target interface.</p> <p>Target interface. If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.</p> <p>NOTE: PEXIWAR n [TRGT] must not be set to target the initiating interface itself. For example, PEXIWAR n for PCI Express controller 1 should never target PCI Express 1.</p> <p>NOTE: If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.</p> <p>NOTE: Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.</p> <table> <tr><td>0000</td><td>PCI Express 1-PCI Express controller 1 should not use this encoding</td></tr> <tr><td>0010</td><td>PCI Express 3-PCI Express controller 3 should not use this encoding</td></tr> <tr><td>0011-0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO 1</td></tr> <tr><td>1001</td><td>Serial RapidIO 2</td></tr> <tr><td>1010-1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1-PCI Express controller 1 should not use this encoding	0010	PCI Express 3-PCI Express controller 3 should not use this encoding	0011-0111	Reserved	1000	Serial RapidIO 1	1001	Serial RapidIO 2	1010-1110	Reserved	1111	Local memory space																				
0000	PCI Express 1-PCI Express controller 1 should not use this encoding																																		
0010	PCI Express 3-PCI Express controller 3 should not use this encoding																																		
0011-0111	Reserved																																		
1000	Serial RapidIO 1																																		
1001	Serial RapidIO 2																																		
1010-1110	Reserved																																		
1111	Local memory space																																		
12–15 RTT	<p>Read transaction type. Read transaction type to send to target interface.</p> <p>If the transaction is not going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>If the transaction is going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read, do not snoop local processor</td></tr> <tr><td>0101</td><td>Read, snoop local processor</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table>	0000	Reserved	...		0100	Read	0101	Reserved	0110	Reserved	0111	Reserved	1000	Reserved	...		1111	Reserved	0000	Reserved	...		0100	Read, do not snoop local processor	0101	Read, snoop local processor	0110	Reserved	1000	Reserved	...		1111	Reserved
0000	Reserved																																		
...																																			
0100	Read																																		
0101	Reserved																																		
0110	Reserved																																		
0111	Reserved																																		
1000	Reserved																																		
...																																			
1111	Reserved																																		
0000	Reserved																																		
...																																			
0100	Read, do not snoop local processor																																		
0101	Read, snoop local processor																																		
0110	Reserved																																		
1000	Reserved																																		
...																																			
1111	Reserved																																		
16–19 WTT	<p>Write transaction type. Write transaction type to send to target interface.</p> <p>If the transaction is not going to local memory space</p> <table> <tr><td>0000</td><td>Reserved</td></tr> </table>	0000	Reserved																																
0000	Reserved																																		

Table continues on the next page...

PEXx_PEXIWARn field descriptions (continued)

Field	Description
	<p>...</p> <p>0100 Write</p> <p>0101 Reserved</p> <p>0110 Reserved</p> <p>0111 Reserved</p> <p>1000 Reserved</p> <p>...</p> <p>1111 Reserved</p> <p>If the transaction is going to local memory space</p> <p>0000 Reserved</p> <p>...</p> <p>0100 Write, do not snoop local processor</p> <p>0101 Write, snoop local processor</p> <p>1000 Reserved</p> <p>...</p> <p>1111 Reserved</p>
20–25 -	This field is reserved. Reserved
26–31 IWS	<p>Inbound window size. Inbound translation window size N which is the encoded $2^{(N+1)}$-bytes window size. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs.</p> <p>000000 Reserved</p> <p>...</p> <p>001010 Reserved</p> <p>001011 4-Kbyte window size</p> <p>001100 8-Kbyte window size</p> <p>...</p> <p>011111 4-Gbyte window size</p> <p>100000 8-Gbyte window size</p> <p>100001 16-Gbyte window size</p> <p>100010 32-Gbyte window size</p> <p>100011 64-Gbyte window size</p> <p>100100 Reserved</p> <p>...</p> <p>111111 Reserved</p>

18.6.27 PCI Express error detect register (PEXx_PEX_ERR_DR)

The PCI Express error detect register contains error status bits that are detected by hardware. The detected error bits are write-1-to-clear type registers. Reading from these registers occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0b0200_0000 is written to the register. When an error is detected the appropriate error bit is set. Subsequent errors sets the appropriate error bits in the error detection registers, but only the first error for a particular unit have any relevant information captured. The interrupt enable bits are used to allow or block the error reporting to the interrupt mechanism while the disable bits are used to prevent or allow the setting of the status bits.

Address: Base address + E00h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ME	Reserved							PCT	PAT	PCAC	PNM	CDNSC	CRSNC	ICCA	IACA
	w1c								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRST	MIS	IOIS	CIS	CIEP	IOEP	OAC	IOIA	IMBA	IOBA	LDDE	Reserved				
	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEX_ERR_DR field descriptions

Field	Description
0 ME	Multiple errors. Detecting multiple errors of the same type. An error is considered as multiple error when its detect bit is set and the same error is occurring again. Note that this bit does not track the ordering of when the error occurs. 1 Multiple errors were detected. 0 Multiple errors were not detected.

Table continues on the next page...

PEXx_PEX_ERR_DR field descriptions (continued)

Field	Description
1–7 -	This field is reserved. Reserved
8 PCT	PCI Express completion time-out. A completion time-out condition was detected for a non-posted, outbound PCI Express transaction. An error response is sent back to the requestor. Note that a completion timeout counter only starts when the non-posted request was able to send to the link partner. <ul style="list-style-type: none"> 1 A completion time-out on the PCI Express link was detected. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system. 0 No completion time-out on the PCI Express link detected.
9 PAT	PCI Express Ack time-out. An internal request timeout was seen. This is considered as a internal fatal error. Once this bit is set, all outstanding transactions will be canceled. No new transactions will be accepted until this bit is clear. It is expected that software needs to perform the proper clean-up operation (that is, hot reset) to get the link to function again before clearing this bit. <ul style="list-style-type: none"> 1 An internal ACK time-out inside the PCI Express controller was detected. 0 No internal ACK time-out inside the PCI Express controller was detected.
10 PCAC	PCI Express completer abort (CA) completion. A completion with CA status was received. <ul style="list-style-type: none"> 1 A completion with CA status was detected. 0 No completion with CA status detected.
11 PNM	PCI Express no map. An inbound transaction that is not mapped to any inbound windows was detected. In RC mode, a posted transaction will be dropped silently and this bit will be set. A nonposted transaction will return a completion without data (Cpl) packet with a UR completion status to the requester and this bit is set. For EP mode, a Cpl packet with a UR completion status is sent back to the requester but does not set this bit. This bit is also set when an IO Write or IO Read transaction is received in RC mode. <ul style="list-style-type: none"> 1 A no-map transaction was detected in RC mode. 0 No no-map transaction detected.
12 CDNSC	Completion with data not successful. A completion with data packet was received with a non successful status (that is, UR, CA or CRS status). <ul style="list-style-type: none"> 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
13 CRSNC	CRS non configuration. A completion was detected for a non configuration cycle and with CRS status. See PCI Express configuration space access for more information. <ul style="list-style-type: none"> 1 CRS non configuration packet was detected. 0 No CRS non configuration packet detected.
14 ICCA	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access. Access to an illegal configuration space from PEX_CONFIG_ADDR/PEX_CONFIG_DATA was detected. <ul style="list-style-type: none"> 1 Invalid CONFIG_ADDR/PEX_CONFIG_DATA access detected 0 No invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detected
15 IACA	Invalid ATMU configuration access. Access to an illegal configuration space from an ATMU window was detected. <ul style="list-style-type: none"> 1 Invalid ATMU configuration access was detected 0 No invalid ATMU configuration access detected

Table continues on the next page...

PEXx_PEX_ERR_DR field descriptions (continued)

Field	Description
16 CRST	CRS thresholded. An outbound configuration transaction was retried and thresholded due to a CRS completion status. An error response is sent back to the requestor. See PCI Express configuration retry timeout register (PEX_PEX_CONF_RTY_TOR) , for more information. 1 A CRS threshold condition was detected for an outbound configuration transaction 0 No CRS threshold condition detected
17 MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See Outbound ATMU message generation , for more information. 1 An invalid size outbound message transaction was detected 0 No invalid size outbound message transaction detected
18 IOIS	I/O invalid size. An outbound I/O transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound I/O transaction was detected 0 No invalid size outbound I/O transaction detected
19 CIS	Configuration invalid size. An outbound configuration transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound configuration transaction was detected 0 No invalid size outbound configuration transaction detected
20 CIEP	Configuration invalid EP. An outbound ATMU configuration transaction request was seen when in EP mode. 1 An outbound configuration transaction while in EP was detected 0 No outbound configuration transaction in EP detected
21 IOIEP	I/O invalid EP. An outbound I/O transaction request was seen when in EP mode. 1 An outbound I/O transaction while in EP was detected 0 No outbound I/O transaction in EP detected
22 OAC	Outbound ATMU crossing. An outbound crossing ATMU transaction was detected. 1 An outbound transaction that hits in one window and crosses overing it was detected 0 No outbound ATMU crossing condition detected
23 IOIA	I/O invalid address. An outbound I/O transaction with a translated address of greater than 4 Gbytes was detected. 1 A greater than 4-Gbyte I/O address was detected 0 No greater than 4-Gbyte I/O address detected
24 IMBA	Invalid memory base address. An outbound memory transaction with a translated address not hitting into the Memory Base/Limit register or Prefetchable Memory Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid memory base address was detected 0 No invalid memory base address detected
25 IIOBA	Invalid I/O base address. An outbound I/O transaction with a translated address not hitting into the IO Base/Limit register or IO Upper Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid I/O base address was detected 0 No invalid I/O base address detected

Table continues on the next page...

PEXx_PEX_ERR_DR field descriptions (continued)

Field	Description
26 LDDE	Link down data error. An outbound read transaction was issued through an ATMU window when the link was down. 1 Link down data error detected 0 No Link down data error detected
27–31 -	This field is reserved. Reserved

18.6.28 PCI Express error interrupt enable register (PEXx_PEX_ERR_EN)

The PCI Express error interrupt enable register allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.

Address: Base address + E08h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								PCTIE	PATIE	PCACIE	PNMIE	CDSNCIE	CRSNCIE	ICCAIE	IACALIE
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRSTIE	MISIE	IOISIE	CISIE	CIEPIE	IOIEPIE	OACIE	IOIAIE	IMBAIE	IIIOBAIE	LDDEIE	Reserved				
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEX_ERR_EN field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8 PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 generates an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
9 PATIE	PCI Express Ack time-out interrupt enable. When set and PEX_ERR_DR[PAT]=1 will generate an interrupt.

Table continues on the next page...

PEXx_PEX_ERR_EN field descriptions (continued)

Field	Description
	1 Enable PCI Express ack time-out interrupt generation 0 Disable PCI Express ack time-out interrupt generation
10 PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 generates an interrupt. 1 Enable completion with CA status interrupt generation 0 Disable completion with CA status interrupt generation
11 PNMIE	PCI Express no map interrupt enable. When set and PEX_ERR_DR[PNM]=1 generates an interrupt. 1 Enable no map PCI Express packet interrupt generation 0 Disable no map PCI Express packet interrupt generation
12 CDNSCIE	Completion with data not successful interrupt enable. When this bit is set and PEX_ERR_DR[CDNSC] = 1 generates an interrupt. 1 Enable completion with data non successful interrupt generation 0 Disable completion with data non successful interrupt generation
13 CRSNCIE	CRS non configuration interrupt enable. When this bit is set and PEX_ERR_DR[CRSNC] = 1 generates an interrupt. 1 Enable CRS non configuration interrupt generation 0 Disable CRS non configuration interrupt generation
14 ICCAIE	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access interrupt enable. When set and PEX_ERR_DR[ICCA]=1 generates an interrupt. 1 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation 0 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation
15 IACAIE	Invalid ATMU configuration access. When set and PEX_ERR_DR[IACA]=1 generates an interrupt. 1 Enable invalid ATMU configuration access interrupt generation 0 Disable invalid ATMU configuration access interrupt generation
16 CRSTIE	CRS thresholded interrupt enable. When set and PEX_ERR_DR[CRST]=1 generates an interrupt. 1 Enable CRS threshold interrupt generation 0 Disable CRS threshold interrupt generation
17 MISIE	Message invalid size interrupt enable. When set and PEX_ERR_DR[MIS]=1 generates an interrupt. 1 Enable invalid outbound message size interrupt generation 0 Disable invalid outbound message size interrupt generation
18 IOISIE	I/O invalid size interrupt enable. When set and PEX_ERR_DR[IOIS]=1 generates an interrupt. 1 Enable invalid outbound I/O size interrupt generation 0 Disable invalid outbound I/O size interrupt generation
19 CISIE	Configuration invalid size interrupt enable. When set and PEX_ERR_DR[CIS]=1 generates an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation
20 CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 generates an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation

Table continues on the next page...

PEXx_PEX_ERR_EN field descriptions (continued)

Field	Description
21 IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 generates an interrupt. 1 Enable outbound I/O transaction EP mode interrupt generation 0 Disable outbound I/O transaction EP mode interrupt generation
22 OACIE	Outbound ATMU crossing interrupt enable. When set and PEX_ERR_DR[OAC]=1 generates an interrupt. 1 Enable outbound crossing ATMU interrupt generation 0 Disable outbound crossing ATMU interrupt generation
23 IOIAIE	I/O address invalid enable. When set and PEX_ERR_DR[IOIA]=1 generates an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
24 IMBAIE	Invalid memory base address interrupt enable. When set and PEX_ERR_DR[IMBA]=1 will generate an interrupt. 1 Enable memory base address interrupt generation 0 Disable memory base address interrupt generation
25 IIOBAIE	Invalid I/O base address interrupt enable. When set and PEX_ERR_DR[IIOBA]=1 will generate an interrupt. 1 Enable I/O base address interrupt generation 0 Disable I/O base address interrupt generation
26 LDDEIE	Link down data error interrupt enable. When set and PEX_ERR_DR[LDDEIE]=1 will generate an interrupt. 1 Enable link down data error interrupt generation 0 Disable link down data error interrupt generation
27–31 -	This field is reserved. Reserved

18.6.29 PCI Express error disable register (PEXx_PEX_ERR_DISR)

The PCI Express error disable register controls the setting of the PCI Express error detect register's bits.

Address: Base address + E10h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	MED	Reserved						PCTD	PATD	PCACD	PNMD	CNSCD	CRSNCD	ICCAD	IACAD		
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	CRSTD	MISD	IOISD	CISD	CIEPD	IOIEPD	OACD	IOIAD	IMBAIE	IIOBAIE	LDDED	Reserved					
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

PEXx_PEX_ERR_DISR field descriptions

Field	Description
0 MED	Multiple errors disable. When set disables the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
1–7 -	This field is reserved. Reserved
8 PCTD	PCI Express completion time-out disable. When set disables the setting of PEX_ERR_DR[PCT] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection
9 PATD	PCI Express Ack time-out disable. When set will disable the setting of PEX_ERR_DR[PAT] bit. 1 Disable PCI Express ack time-out detection 0 Enable PCI Express ack time-out detection
10 PCACD	PCI Express CA completion disable. When set disables the setting of PEX_ERR_DR[PCAC] bit. 1 Disable completion with CA status detection 0 Enable completion with CA status detection
11 PNMD	PCI Express no map disable. When set disables the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection

Table continues on the next page...

PEXx_PEX_ERR_DISR field descriptions (continued)

Field	Description
12 CDNSCD	Completion with data not successful disable. When set disables the setting of PEX_ERR_DR[CDNSC] bit. 1 Disable completion with data not successful detection 0 Enable completion with data not successful detection
13 CRSNCD	CRS non configuration disable. When set disables the setting of PEX_ERR_DR[CRSNC] bit. 1 Disable CRS non configuration detection 0 Enable CRS non configuration detection
14 ICCAD	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access disable. When set disables the setting of PEX_ERR_DR[ICCA] bit. 0 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection 1 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection
15 IACAD	Invalid ATMU configuration access. When set disables the setting of PEX_ERR_DR[IACA] bit. 0 Enable invalid ATMU configuration access detection 1 Disable invalid ATMU configuration access detection
16 CRSTD	CRS thresholded disable. When set disables the setting of PEX_ERR_DR[CRST] bit. 0 Enable CRS threshold detection 1 Disable CRS threshold detection
17 MISD	Message invalid size disable. When set disables the setting of PEX_ERR_DR[MIS] bit. 0 Enable invalid outbound message size detection 1 Disable invalid outbound message size detection
18 IOISD	I/O invalid size disable. When set disables the setting of PEX_ERR_DR[IOIS] bit. 0 Enable invalid outbound I/O size detection 1 Disable invalid outbound I/O size detection
19 CISD	Configuration invalid size disable. When set disables the setting of PEX_ERR_DR[CIS] bit. 0 Enable invalid outbound configuration size detection 1 Disable invalid outbound configuration size detection
20 CIEPD	Configuration invalid EP disable. When set disables the setting of PEX_ERR_DR[CIEP] bit. 0 Enable outbound configuration transaction EP mode detection 1 Disable outbound configuration transaction EP mode detection
21 IOIEPD	I/O invalid EP disable. When set disables the setting of PEX_ERR_DR[IOEP] bit. 0 Enable outbound I/O transaction EP mode detection 1 Disable outbound I/O transaction EP mode detection
22 OACD	Outbound ATMU crossing disable. When set disables the setting of PEX_ERR_DR[OAC] bit. 0 Enable outbound crossing ATMU detection 1 Disable outbound crossing ATMU detection
23 IOIAD	I/O invalid address disable. When set disables the setting of PEX_ERR_DR[IOIA] bit. 0 Enable greater than 4G I/O address detection 1 Disable greater than 4G I/O address detection

Table continues on the next page...

PEXx_PEX_ERR_DISR field descriptions (continued)

Field	Description
24 IMBAIE	Invalid memory base address disable. When set will disable the setting of PEX_ERR_DR[IMBA]. 0 Enable memory base address detection 1 Disable memory base address detection
25 IIOBAIE	Invalid I/O base address disable. When set will disable the setting of PEX_ERR_DR[IIOBA]. 0 Enable I/O base address detection 1 Disable I/O base address detection
26 LDDED	Link down data error disable. When set will disable the setting of PEX_ERR_DR[LDDE]. 0 Enable link down data error detection 1 Disable link down data error detection
27–31 -	This field is reserved. Reserved

18.6.30 PCI Express error capture status register (PEXx_PEX_ERR_CAP_STAT)

The PCI Express error capture status register allows vital error information to be captured when an error occurs. Note that no further error capturing is performed until the ECV bit is cleared.

Address: Base address + E20h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TO								Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									GSID					ECV		
W															w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PEXx_PEX_ERR_CAP_STAT field descriptions

Field	Description
0 TO	Transaction originator. This field indicates whether the originator of the transaction is from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 1 Transaction originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA 0 Transaction not originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA

Table continues on the next page...

PEXx_PEX_ERR_CAP_STAT field descriptions (continued)

Field	Description
1-22 -	This field is reserved. Reserved
23-30 GSID	Global source ID. This field indicates the internal platform global source ID that the error transaction originates. This field only applies to non PEX_CONFIG_ADDR/PEX_CONFIG_DATA transactions. All settings not shown are reserved. 0x00 PCI Express 1 0x01 PCI Express 2 0x02 PCI Express 3 0x08 SRIO1 0x09 SRIO2 0x5D SRIO Message Unit (RMU) 0x70 DMA1 0x71 DMA2 0x81 Core 0 data 0x83 Core 1 data 0x85 Core 2 data 0x87 Core 3 data 0x89 Core 4 data 0x8B Core 5 data 0x8D Core 6 data 0x8F This GSID value can indicate any of the following sources: <ul style="list-style-type: none">• Core 7 Data• eSDHC• NPC• PBL• USB1, USB2• PME• SEC• FMan1, FMan2• QMan, BMan
31 ECV	Error capture valid. This bit indicates that the capture registers 0-3 contain valid info. This bit when set indicates that the captured registers contain valid capturing information. No new capturing is done unless this bit is cleared by writing a 1 to it.

18.6.31 PCI Express error capture register n (PEXx_PEX_ERR_CAP_Rn)

PEX_ERR_CAP_Rn allow vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

See [Error capture registers](#) for further details.

Address: Base address + E28h offset + (4d × i), where i=0d to 3d

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

PEXx_PEX_ERR_CAP_Rn field descriptions

Field	Description
0–31 DATA	Captured Data

18.7 PCI Express configuration-space registers

This table lists the PCI Express configuration-space registers.

Table 18-251. PCI Express configuration-space registers

Address offset (Hex)	
000-03F	PCI Compatible configuration header (See PCI compatible configuration headers for more information.)
040-0FF	PCI-compatible device-specific configuration space (See PCI compatible device-specific configuration space for more information.)
100-134	PCI Express Extended Configuration Registers (See PCI Express extended configuration space for more information.)
138-3FF	Reserved
400-6FF	PCI Express Controller Internal CSRs ¹ (See PCI Express controller internal CSRs for more information.)
700-FFF	Reserved

1. Note that the PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers returns all 0s.

18.7.1 PCI compatible configuration headers

The first 64 bytes of the 256-byte, PCI-compatible configuration space consists of a predefined header that every PCI device must support.

Many registers in the predefined header are defined the same for all PCI Express devices.

These registers are shown in this figure.

Reserved		Address Offset (Hex)
	Device ID	00
	Status	04
	Class Code	08
BIST	Header Type	0C
	Latency Timer	10
	Cache Line Size	14
Header Type (0/1) Specific		18
Header Type (0/1) Specific		1C
Header Type (0/1) Specific		20
Header Type (0/1) Specific		24
Header Type (0/1) Specific		28
Header Type (0/1) Specific		2C
Header Type (0/1) Specific		30
Capabilities Pointer		34
Interrupt Pin		38
Interrupt Line		3C

Figure 18-250. PCI Express PCI-compatible configuration header common registers

The remaining registers in the header may have differing layouts depending on the function of the device. There are two header types applicable to PCI Express. Type 0 headers are typically used by endpoints; Type 1 headers are used by root complexes and switches/bridges.

18.8 Type 0 configuration header registers

The figure below shows the type 0 header.

NOTE

The BIST register (0x0F) is optional and reserved on the PCI Express controller.

Address Offset (Hex)
00
04
08
0C
10
14
18
1C
20
24
28
2C
30
34
38
3C

Reserved

Device ID Vendor ID

Status Command

Class Code Revision ID

BIST Header Type Latency Timer Cache Line Size

Base Address Registers

Subsystem ID Subsystem Vendor

Expansion ROM Base Address

Capabilities Pointer

MAX_LAT MIN_GNT Interrupt Pin Interrupt Line

Figure 18-251. PCI Express PCI-Compatible Configuration Header—Type 0

18.8.1 PCI Express Vendor ID Register (Vendor_ID_Register)

The vendor ID register is used to identify the manufacturer of the device.

Address: 0h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write																
Reset	0	0	0	1	1	0	0	1	0	1	0	1	0	1	1	1

Vendor_ID_Register field descriptions

Field	Description
Vendor_ID	0x1957 (Freescale)

18.8.2 PCI Express Device ID Register (Device_ID_Register)

The device ID register is used to identify the device.

NOTE

See bitfield description table for reset value.

Address: 2h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Device_ID_Register field descriptions

Field	Description
Device_ID	Device_ID 0x0400 P4080 with security 0x0401 P4080 without security 0x0408 P4040 with security 0x0409 P4040 without security

18.8.3 PCI Express Command Register (Command_Register)

The command register provides control over the ability to generate and respond to PCI Express cycles.

Address: 4h

Bit	15	14	13	12	11	10	9	8
Read	Reserved					Interrupt_Disable	Reserved	SERR
Write	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read	Reserved	Parity_error_response	Reserved			Bus_master	Memory_space	I_O_space
Write	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

Command_Register field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10 Interrupt_Disable	Controls the ability to generate INTx interrupt messages. Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set. 0 Enables INTx interrupt messages 1 Disables INTx interrupt messages
9 -	This field is reserved. Reserved
8 SERR	Controls the reporting of fatal and non-fatal errors detected by the device to the root complex. NOTE: The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in PCI Express Device Control Register (Device_Control_Register) , and the advance error reporting capability structure described in sections PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reportin g Capability ID Register) through PCI Express Error Source ID Register (Error_Source_ID_Register) .
	0 Disables reporting 1 Enables reporting
7 -	This field is reserved. Reserved
6 Parity_error_response	Controls whether this PCI Express controller responds to parity errors. NOTE: The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in PCI Express Device Control Register (Device_Control_Register) , and the advance error reporting capability structure described in sections PCI Express Advanced

Table continues on the next page...

Command_Register field descriptions (continued)

Field	Description
	<p style="color: blue; font-size: small;">Error Reporting Capability ID Register (Advanced_Error_Reportin g_Capability_ID_Register)</p> <p>through PCI Express Error Source ID Register (Error_Source_ID_Register) .</p> <p>0 Parity errors are ignored and normal operation continues.</p> <p>1 Parity errors cause the appropriate bit in the PCI Express status register to be set. However, note that errors are reported based on the values set in the PCI Express error enable and detection registers.</p>
5–3 -	This field is reserved. Reserved
2 Bus_master	<p>Indicates whether this PCI Express device is configured as a master.</p> <p>EP mode: Clearing this bit prevents the device from issuing any memory or I/O transactions. Because MSI interrupts are effectively memory writes, clearing this bit also disables the ability of the device to issue MSI interrupts.</p> <p>RC mode: Clearing this bit disables the ability of the device to forward memory transactions upstream. This causes any inbound memory transaction (except those targeting PEXCSRBAR) to be treated as an unsupported request. Note that this description does not apply for inbound memory transactions targeting PEXCSRBAR which are forwarded if the Memory_space bit is set.</p> <p>0 Disables the ability to generate PCI Express accesses</p> <p>1 Enables this PCI Express controller to behave as a PCI Express bus master</p>
1 Memory_space	<p>Controls whether this PCI Express device (as a target) responds to memory accesses.</p> <p>EP mode: Clearing this bit prevents the device from accepting any memory transaction.</p> <p>RC mode: This bit is ignored for inbound memory transactions except those targeting PEXCSRBAR. Clearing this bit prevents the chip from accepting inbound memory transactions to PEXCSRBAR. This bit does not affect outbound memory transactions.</p> <p>0 This PCI Express device does not respond to PCI Express memory space accesses.</p> <p>1 This PCI Express device responds to PCI Express memory space accesses.</p>
0 I_O_space	<p>I/O space.</p> <p>EP mode: Clearing this bit prevents the device from accepting any IO transaction. Note that this bit is a don't care in EP mode since the device does not support IO transaction.</p> <p>RC mode: This bit is ignored. It does not affect outbound IO transaction.</p> <p>0 This PCI Express device (as a target) does not respond to PCI Express I/O space accesses.</p> <p>1 This PCI Express device (as a target) does respond to PCI Express I/O space accesses.</p>

18.8.4 PCI Express Status Register (Status_Register)

The status register is used to record status information for PCI Express related events.

Address: 6h

Bit	15	14	13	12	11	10	9	8
Read	Detected_parity_error	Signaled_system_error	Received_master_abort	Received_target_abort	Signaled_target_abort	Reserved	Master_data_parity_error_detected	
Write	w1c	w1c	w1c	w1c	w1c			w1c
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read	Reserved			Capabilities_List	Interrupt_Status	Reserved		
Write	0	0	0	1	0	0	0	0
Reset	0	0	0	1	0	0	0	0

Status_Register field descriptions

Field	Description
15 Detected_parity_error	Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register. ⁸
14 Signaled_system_error	Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set. ⁷
13 Received_master_abort	Set whenever a requestor receives a completion with unsupported request completion status. ⁷
12 Received_target_abort	Set whenever a device receives a completion with completer abort completion status. ⁷
11 Signaled_target_abort	Set whenever a device completes a request using completer abort completion status. ⁷
10–9 -	This field is reserved. Reserved
8 Master_data_parity_error_detected	Set by the requestor (primary side for Type1 headers) when either the requestor receives a completion marked poisoned or the requestor poisons a write request. Note that the parity error enable bit (bit 6) in the command register must be set for this bit to be set. ⁷

Table continues on the next page...

Status_Register field descriptions (continued)

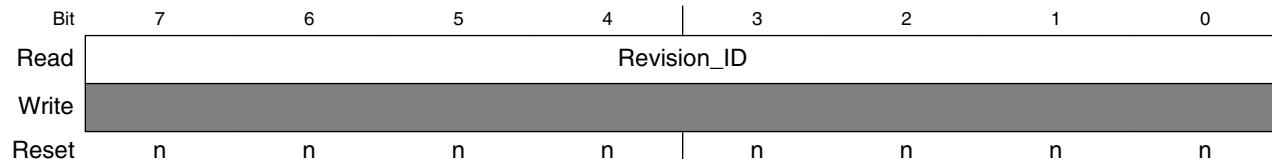
Field	Description
7–5 -	This field is reserved. Reserved
4 Capabilities_List	All PCI Express devices are required to implement the PCI Express capability structure.
3 Interrupt_Status	Set when an INTx interrupt message is pending internally to the device. Note that this bit is associated with INTx messages and not Message Signaled Interrupts.
-	This field is reserved. Reserved

1. The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in [PCI Express Device Control Register \(Device_Control_Register\)](#), and the advance error reporting capability structure described in sections [PCI Express Advanced Error Reporting Capability ID Register \(Advanced_Error_Reportin](#)g_Capability_ID_Register) through [PCI Express Error Source ID Register \(Error_Source_ID_Register\)](#).

18.8.5 PCI Express Revision ID Register (Revision_ID_Register)

The revision ID register is used to identify the revision of the device.

Address: 8h



Revision_ID_Register field descriptions

Field	Description
Revision_ID	Revision specific.

18.8.6 PCI Express Class Code Register (Class_Code_Register)

The class code register is comprised of three single-byte fields-base class (offset 0x0B), sub-class (offset 0x0A), and programming interface (offset 0x09)-that indicate the basic functionality of the function.

NOTE

See bitfield description table for Programming_Interface reset value.

Address: 9h

Bit	23	22	21	20	19	18	17	16	15	14	13	12		11	10	9	8	7	6	5	4	3	2	1	0
R	Base_Class												Sub_Class	Programming_Interface											
W																									
Reset	0	0	0	0	1	0	1	1	0	0	1	0		0	0	0	0	0	0	0	0	0	0	0	n

Class_Code_Register field descriptions

Field	Description
23–16 Base_Class	Base_Class 0x0B Processor
15–8 Sub_Class	Sub_Class 0x20 PowerPC
Programming_Interface	Programming_Interface 0x00 RC mode 0x01 EP mode

18.8.7 PCI Express Cache Line Size Register (Cache_Line_Size_Register)

The cache line size register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

Address: Ch

Bit	7	6	5	4		3	2	1	0
Read Write	Cache_Line_Size								
Reset	0	0	0	0		0	0	0	0

Cache_Line_Size_Register field descriptions

Field	Description
Cache_Line_Size	Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). Note that for PCI Express operation this register is ignored.

18.8.8 PCI Express Latency Timer Register (Latency_Timer_Register)

The latency timer register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

Address: Dh

Bit	7	6	5	4		3	2	1	0
Read					Latency_Timer				
Write									
Reset	0	0	0	0		0	0	0	0

Latency_Timer_Register field descriptions

Field	Description
Latency_Timer	Note that for PCI Express operation this register is ignored.

18.8.9 PCI Express Header Type Register (Header_Type_Register)

The PCI Express header type register is used to identify the layout of the PCI compatible header.

NOTE

See bitfield description table for reset value.

Address: Eh

Bit	7	6	5	4		3	2	1	0
Read	Multifunction				Header_Layout				
Write									
Reset	0	0	0	0		0	0	0	n

Header_Type_Register field descriptions

Field	Description
7 Multifunction	Identifies whether a device supports multiple functions 0 Single function device 1 Multiple function device
Header_Layout	All other encodings reserved. 0x00 Endpoint. See Type 0 configuration header registers for type 0 layout. 0x01 Root Complex. See Type 1 configuration header registers for type 1 layout.

18.8.10 PCI Express Base Address Register 0 (PEXCSRBAR)

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to [PCI Express inbound ATMUs](#) for more information.

Base address register 0, also known as the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10 is a special fixed 16 -Mbyte window that is used for inbound configuration accesses. Note that PEXCSRBAR cannot be updated through the inbound ATMU registers.

Address: 10h

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

PEXCSRBAR field descriptions

Field	Description
31–24 ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 16 Mbyte.
23–4 -	This field is reserved. Reserved
3 PREF	Prefetchable
2–1 TYPE	Type. 00 Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator

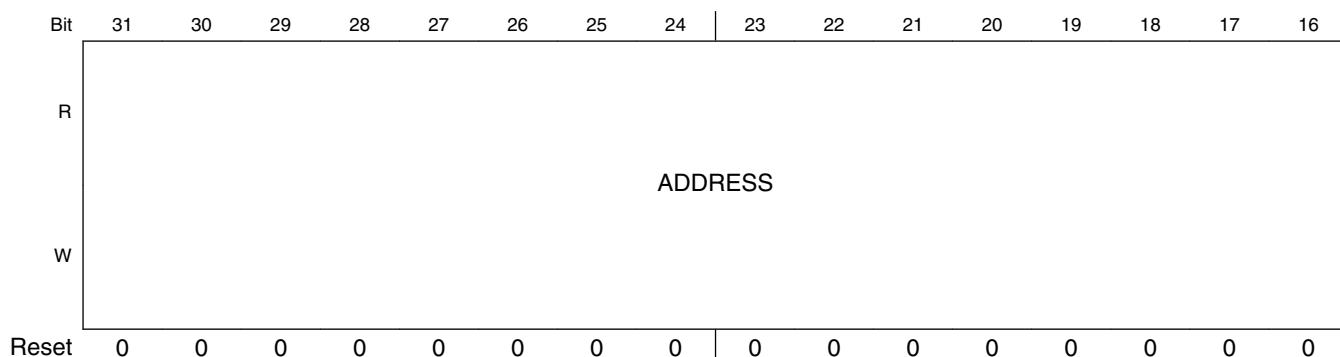
18.8.11 PCI Express Base Address Register 1 (BAR1)

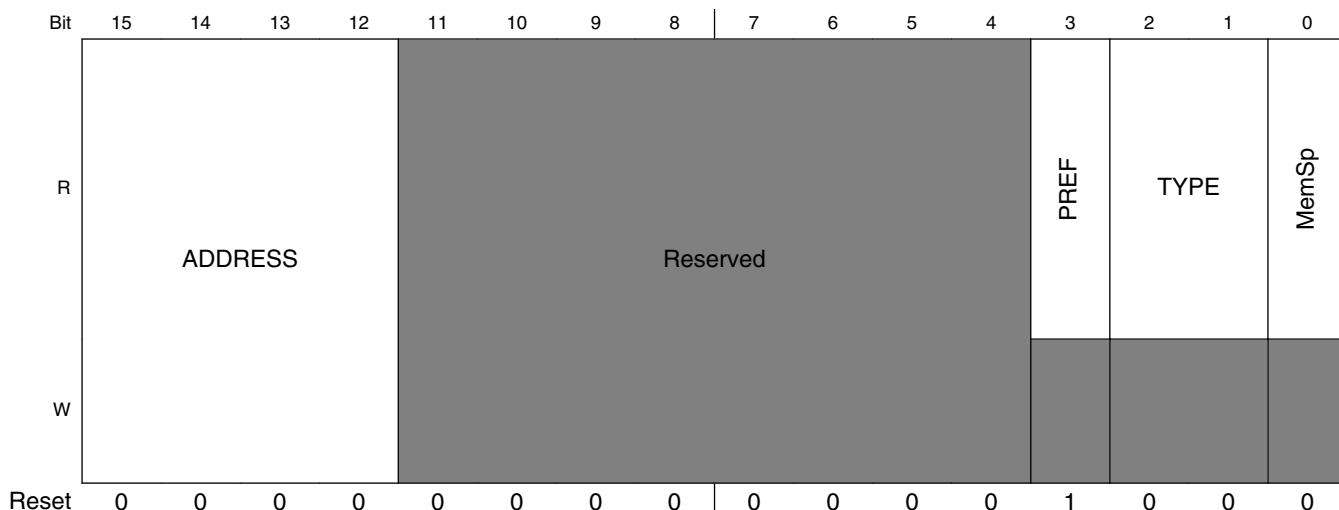
This register is supported only for EP mode.

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space BAR, and two 64-bit memory space BARs. Refer to [PCI Express inbound ATMs](#) for more information.

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space.

Address: 14h





BAR1 field descriptions

Field	Description
31–12 ADDRESS	Indicates the base address where the inbound memory window begins. The number of upper bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes register (PEXIWAR1).
11–4 -	This field is reserved. Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable. This bit is determined by PEXIWAR1[PF].
2–1 TYPE	Type. 00 Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator.

18.8.12 PCI Express Base Address Register 2,4 (BARn)

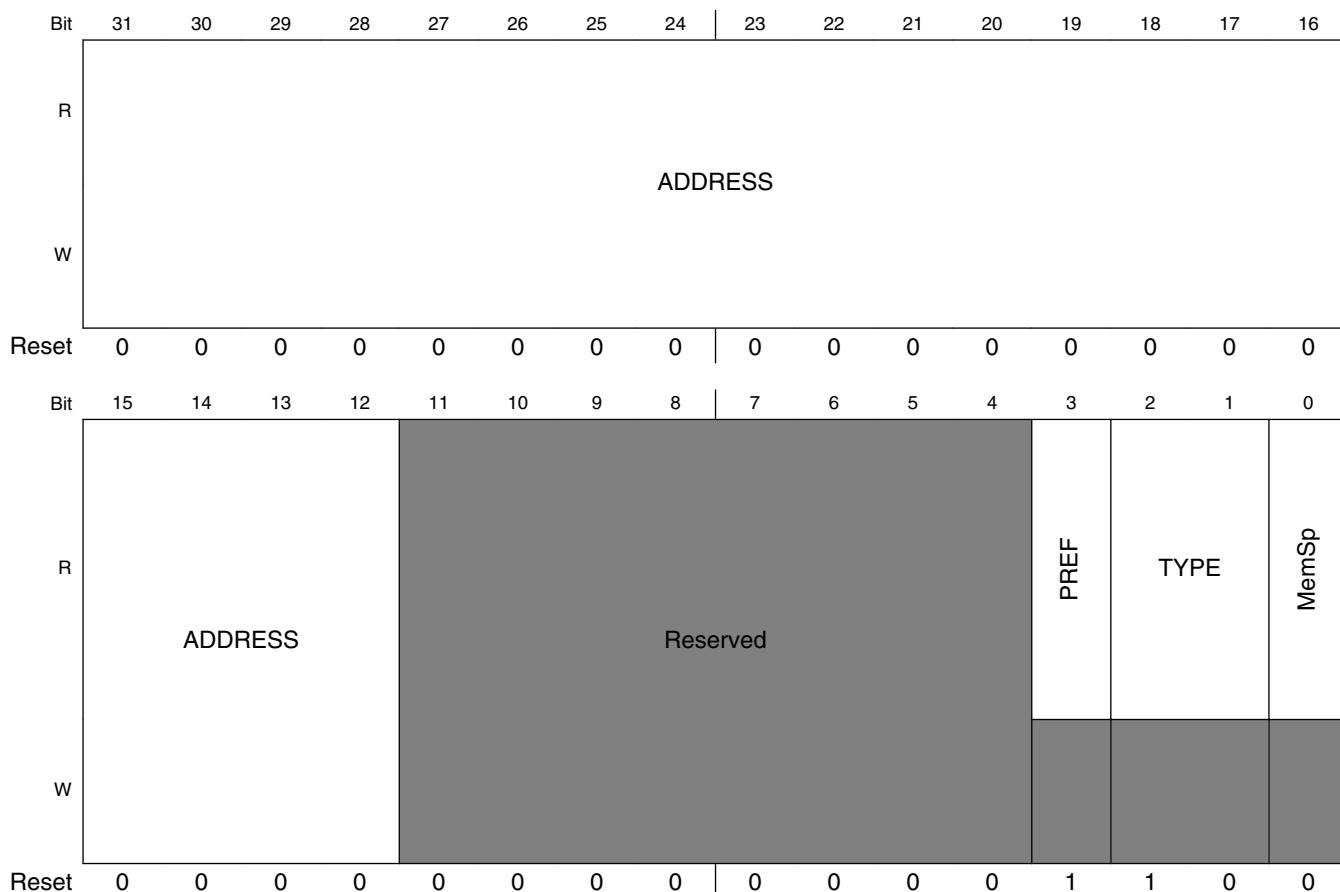
This register is supported only for EP mode.

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to [PCI Express inbound ATMUs](#), for more information.

Base address register 2 at offset 0x18 and base address register 4 at offset 0x20 are used to define the lower portion of the 64-bit inbound memory windows.

Type 0 configuration header registers

Address: base + offset + (8d × i), where i=0d to 1d

**BARn field descriptions**

Field	Description
31–12 ADDRESS	Indicates the lower portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x18 and PEXIWAR3 for offset 0x20).
11–4 -	This field is reserved. Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable. This bit is determined by PEXIWAR n [2].
2–1 TYPE	Type. 0b10 Locate anywhere in 64-bit address space.
0 MemSp	Memory space indicator

18.8.13 PCI Express Base Address Register 3,5 (BARn)

This register is supported only for EP mode.

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to [PCI Express inbound ATMUs](#), for more information.

Base address register 3 at offset 0x1C and base address register 5 at offset 0x24 are used to define the upper portion of the 64-bit inbound memory windows.

Address: base + offset + (8d × i), where i=0d to 1d

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

Reset 0

BARn field descriptions

Field	Description
ADDRESS	Indicates the upper portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x1C and PEXIWAR3 for offset 0x24). If no access to local memory is to be permitted by external requestors, then all bits are programmed.

18.8.14 PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register)

This register is supported only for EP mode.

The PCI Express subsystem vendor ID register is used to identify the subsystem.

NOTE

Reset value taken from PEX_SSVID_UPDATE[SS_ID].

Address: 2Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read																
Write																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Subsystem_Vendor_ID_Register field descriptions

Field	Description
Subsystem_Vendor_ID	The value for subsystem vendor ID is determined by the PCI Express subsystem vendor ID update register. See PCI Express Subsystem Vendor ID Update Register (Subsystem_Vendor_ID_Update_Register) , for more information.

18.8.15 PCI Express Subsystem ID Register (Subsystem_ID_Register)

This register is supported only for EP mode.

The PCI Express subsystem ID register is used to identify the subsystem.

NOTE

Reset value taken from PEX_SSVID_UPDATE[SSV_ID].

Address: 2Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read									Subsystem_ID							
Write																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

Subsystem_ID_Register field descriptions

Field	Description
Subsystem_ID	The value for subsystem ID is determined by the PCI Express subsystem vendor ID update register. See PCI Express Subsystem Vendor ID Update Register (Subsystem_Vendor_ID_Update_Register) , for more information.

18.8.16 Capabilities Pointer Register (Capabilities_Pointer_Register)

The capabilities pointer identifies additional functionality supported by the device.

Address: 34h

Bit	7	6	5	4	3	2	1	0
Read				Capabilities_Pointer				
Write								
Reset	0	1	0	0	0	1	0	0

Capabilities_Pointer_Register field descriptions

Field	Description
Capabilities_Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to PCI compatible device-specific configuration space for more information.

18.8.17 PCI Express Interrupt Line Register (Interrupt_Line_Register)

This register is supported only for EP mode.

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

Address: 3Ch

Bit	7	6	5	4		3	2	1	0
Read	Interrupt_Line								
Write	0 0 0 0 0 0 0 0								
Reset	0 0 0 0 0 0 0 0								

Interrupt_Line_Register field descriptions

Field	Description
Interrupt_Line	Used to communicate interrupt line routing information. N/A for RC mode.

18.8.18 PCI Express Interrupt Pin Register (Interrupt_Pin_Register)

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

This register only applies to EP mode. Reset value is 0x00 for RC mode and 0x01 for EP mode.

Address: 3Dh

Bit	7	6	5	4		3	2	1	0
Read	Interrupt_pin								
Write									
Reset	0 0 0 0 0 0 0 0								

Interrupt_Pin_Register field descriptions

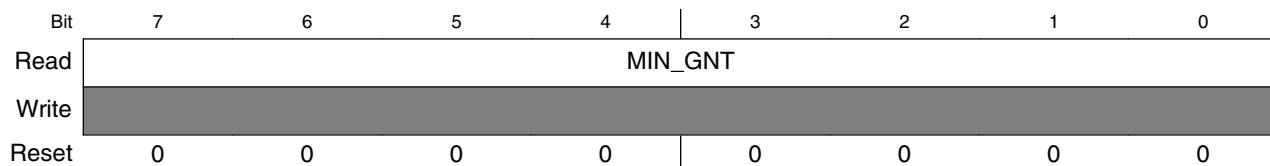
Field	Description
Interrupt_pin	Legacy INTx message used by this device. All other settings Reserved. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD

18.8.19 PCI Express Minimum Grant Register (Minimum_Grant_Register)

This register is supported only for EP mode.

This register does not apply to PCI Express. It is present for legacy purposes.

Address: 3Eh



Minimum_Grant_Register field descriptions

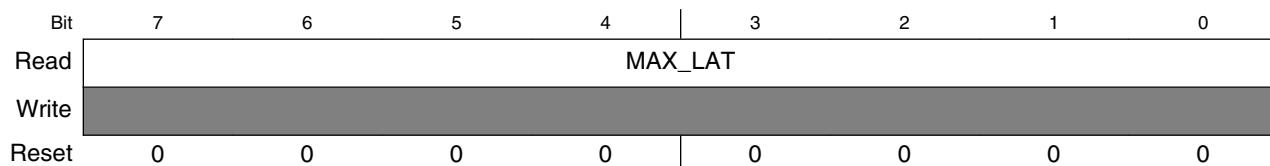
Field	Description
MIN_GNT	Does not apply for PCI Express.

18.8.20 PCI Express Maximum Latency Register (Maximum_Latency_Register)

This register is supported only for EP mode.

This register does not apply to PCI Express. It is present for legacy purposes.

Address: 3Fh



Maximum_Latency_Register field descriptions

Field	Description
MAX_LAT	Does not apply for PCI Express.

18.9 Type 1 configuration header registers

The figure below shows the type 1 header.

NOTE

The first 16 bytes of the type 1 header are identical to the type 0 header. See [Type 0 configuration header registers](#) for descriptions of those registers. This section describes the registers that are unique to the type 1 header beginning at offset 0x10.

NOTE

The BIST register (0x0F) is optional and reserved on the PCI Express controller.

Reserved				Address Offset (Hex)		
	Device ID			00		
	Status			04		
	Class Code			08		
BIST	Header Type	Latency Timer	Cache Line Size	0C		
Base Address Register 0				10		
				14		
Second Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18		
Secondary Status		I/O Limit	I/O Base	1C		
Memory Limit		Memory Base		20		
Prefetchable Memory Limit		Prefetchable Memory Base		24		
Prefetchable Base Upper 32 Bits				28		
Prefetchable Limit Upper 32 Bits				2C		
I/O Limit upper 16 Bits		I/O Base Upper 16 Bits		30		
				34		
Expansion ROM Base Address				38		
Bridge Control		Interrupt Pin	Interrupt Line	3C		

Figure 18-276. PCI Express PCI-Compatible Configuration Header—Type 1

NOTE

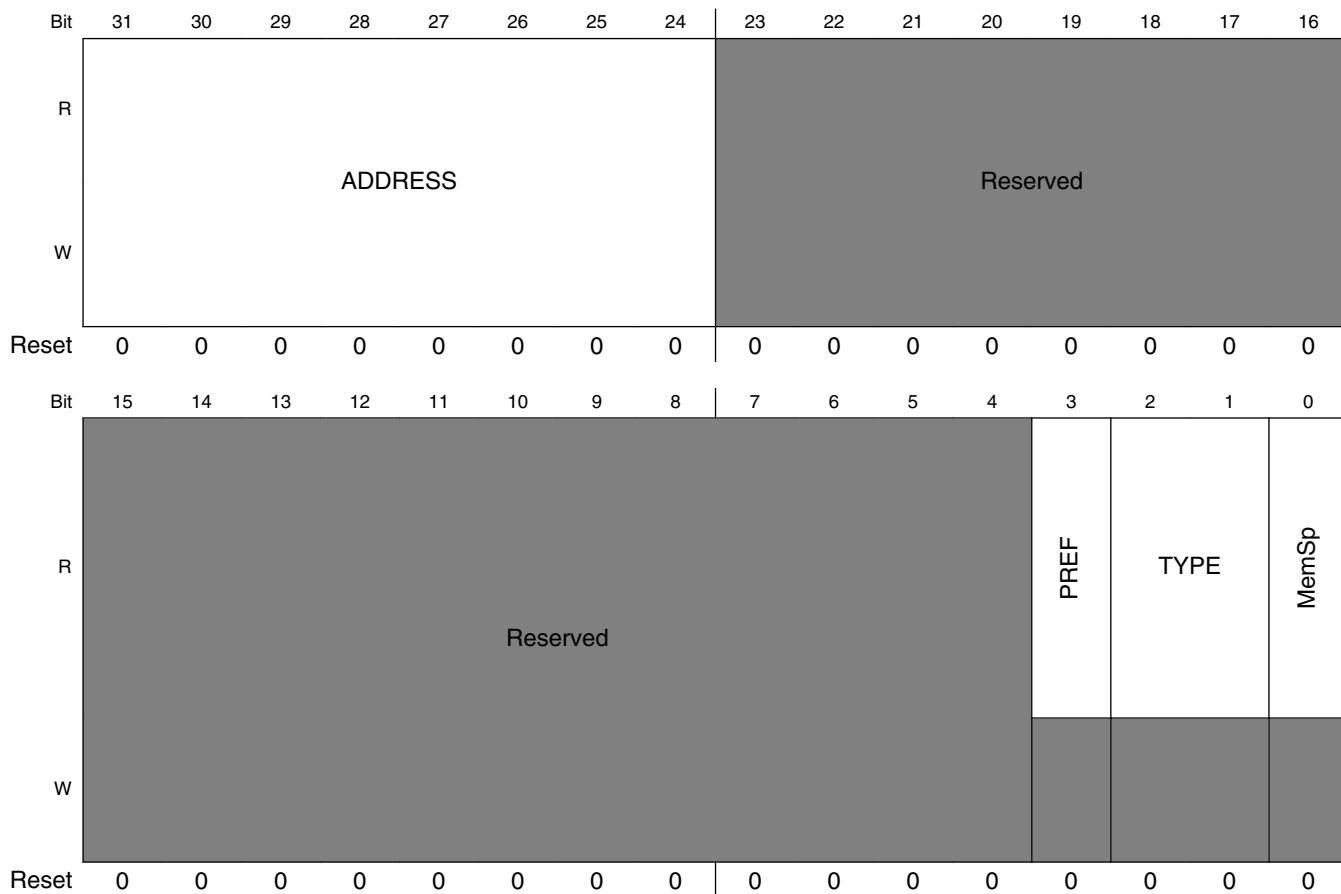
The secondary latency timer register (0x1B) does not apply to PCI Express. It must be read-only and return all zeros when read.

18.9.1 PCI Express Base Address Register 0 (PEXCSRBAR)

Base address register 0, also known as the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10 is a special fixed 16 -Mbyte window that is used for inbound configuration accesses.

Note that PEXCSRBAR cannot be updated through the inbound ATMU registers.

Address: 10h



PEXCSRBAR field descriptions

Field	Description
31–24 ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 16 Mbyte.
23–4 -	This field is reserved. Reserved
3 PREF	Prefetchable
2–1 TYPE	Type. 00 Locate anywhere in 32-bit address space.

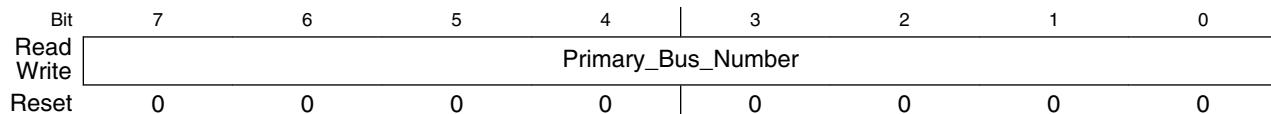
Table continues on the next page...

PEXCSRBAR field descriptions (continued)

Field	Description
0 MemSp	Memory space indicator

18.9.2 PCI Express Primary Bus Number Register (Primary_Bus_Number_Register)

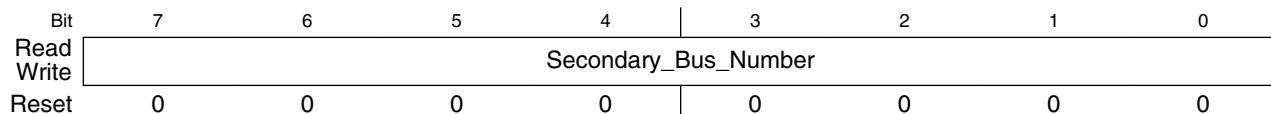
Address: 18h

**Primary_Bus_Number_Register field descriptions**

Field	Description
Primary_Bus_Number	Bus that is connected to the upstream interface. Note that this register is programmed during system enumeration; in RC mode this register should remain 0x00.

18.9.3 PCI Express Secondary Bus Number Register (Secondary_Bus_Number_Register)

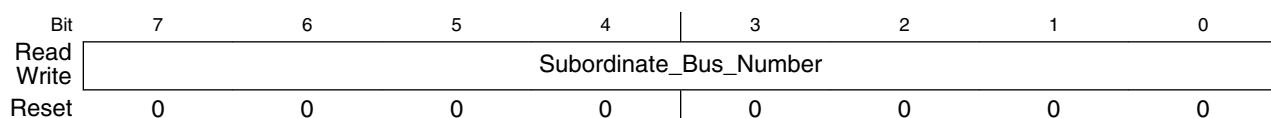
Address: 19h

**Secondary_Bus_Number_Register field descriptions**

Field	Description
Secondary_Bus_Number	Bus that is directly connected to the downstream interface. Note that this register is programmed during system enumeration; in RC mode, this register is typically programmed to 0x01.

18.9.4 PCI Express Subordinate Bus Number Register (Subordinate_Bus_Number_Register)

Address: 1Ah



Subordinate_Bus_Number_Register field descriptions

Field	Description
Subordinate_Bus_Number	Highest bus number that is on the downstream interface.

18.9.5 PCI Express I/O Base Register (IO_Base_Register)

Note that this device does not support inbound I/O transactions.

Address: 1Ch

Bit	7	6	5	4	3	2	1	0
Read Write								
Reset	0	0	0	0	0	0	0	1

IO_Base_Register field descriptions

Field	Description
7–4 IO_Start_Address	Specifies bits 15:12 of the I/O space start address NOTE: I/O Start Address is writeable in root complex, read-only in end point
Address_Decode_Type	Specifies the number of I/O address bits. All other settings reserved. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode

18.9.6 PCI Express I/O Limit Register (IO_Limit_Register)

Note that this device does not support inbound I/O transactions.

Address: 1Dh

Bit	7	6	5	4	3	2	1	0
Read Write								
Reset	0	0	0	0	0	0	0	1

IO_Limit_Register field descriptions

Field	Description
7–4 IO_Limit_Address	Specifies bits 15:12 of the I/O space ending address
Address_Decode_Type	Specifies the number of I/O address bits. All other settings reserved. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode

**18.9.7 PCI Express Secondary Status Register
(Secondary_Status_Register)**

Note that the errors in this register may be masked by corresponding bits in the secondary status interrupt mask register (PEX_SS_INTR_MASK) and that by default all of the errors are masked. See [Secondary Status Interrupt Mask Register \(Secondary_Status_Interrupt_Mask_Register\)](#) for more information.

Address: 1Eh

Bit	15	14	13	12	11	10	9	8
Read	DPE	SSE	RMA	RTA	STA			MDPE
Write	w1c	w1c	w1c	w1c	w1c	Reserved		w1c
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read	Reserved							
Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Secondary_Status_Register field descriptions

Field	Description
15 DPE	Detected parity error. This bit is set whenever the secondary side receives a poisoned TLP regardless of the state of the parity error response bit.
14 SSE	Signaled system error. This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL message, provided the SERR enable bit in the command register is set to enable reporting.
13 RMA	Received master abort. This bit is set when the secondary side receives an unsupported request (UR) completion.
12 RTA	Received target abort. This bit is set when the secondary side receives a completer abort (CA) completion.

Table continues on the next page...

Secondary_Status_Register field descriptions (continued)

Field	Description
11 STA	Signaled target abort. This bit is set when the secondary side issues a CA completion.
10–9 - Reserved	This field is reserved.
8 MDPE	Master data parity error. This bit is set when the parity error response bit is set and the secondary side requestor receives a poisoned completion or poisons a write request. If the parity error response bit is cleared, this bit is never set.
- Reserved	This field is reserved. Reserved

18.9.8 PCI Express Memory Base Register (Memory_Base_Register)

Address: 20h

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Read																	
Write																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Memory_Base_Register field descriptions

Field	Description
15–4 Memory_Base	Specifies bits 31:20 of the non-prefetchable memory space start address. Typically used for specifying memory-mapped I/O space. NOTE: Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in an unsupported request response.
- Reserved	This field is reserved. Reserved

18.9.9 PCI Express Memory Limit Register (Memory_Limit_Register)

Address: 22h

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Read																	
Write																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Memory_Limit_Register field descriptions

Field	Description
15–4 Memory_Limit	Specifies bits 31:20 of the non-prefetchable memory space ending address. Typically used for specifying memory-mapped I/O space.

Table continues on the next page...

Memory_Limit_Register field descriptions (continued)

Field	Description
	NOTE: Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in unsupported request response.
-	This field is reserved. Reserved

18.9.10 PCI Express Prefetchable Memory Base Register (Prefetchable_Memory_Base_Register)

Address: 24h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PF_Memory_Base												Address_Decode_Type			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Prefetchable_Memory_Base_Register field descriptions

Field	Description
15–4 PF_Memory_Base	Specifies bits 31:20 of the prefetchable memory space start address. NOTE: Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in unsupported request response.
Address_Decode_Type	Specifies the number of prefetchable memory address bits. All other settings reserved. 0x00 32-bit memory address decode 0x01 64-bit memory address decode

18.9.11 PCI Express Prefetchable Memory Limit Register (Prefetchable_Memory_Limit_Register)

Address: 26h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	PF_Memory_Limit												Address_Decode_Type			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Prefetchable Memory Limit Register field descriptions

Field	Description				
15–4 PF_Memory_Limit	<p>Specifies bits 31:20 of the prefetchable memory space ending address.</p> <p>NOTE: Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in unsupported request response.</p>				
Address_Decode_Type	<p>Specifies the number of prefetchable memory address bits.</p> <p>All other settings reserved.</p> <table border="0" data-bbox="362 412 1344 437"> <tr> <td data-bbox="362 412 417 421">0x00</td> <td data-bbox="417 412 753 421">32-bit memory address decode</td> </tr> <tr> <td data-bbox="362 421 417 431">0x01</td> <td data-bbox="417 421 753 431">64-bit memory address decode</td> </tr> </table>	0x00	32-bit memory address decode	0x01	64-bit memory address decode
0x00	32-bit memory address decode				
0x01	64-bit memory address decode				

18.9.12 PCI Express Prefetchable Base Upper 32 Bits Register (Prefetchable_Base_Upper_32_Bits_Register)

Address: 28h

Prefetchable_Base_Upper_32_Bits_Register field descriptions

Field	Description
PF_Base_Upper_32_Bits	Specifies bits 64:32 of the prefetchable memory space start address when the address decode type field in the prefetchable memory base register is 0x01.

18.9.13 PCI Express Prefetchable Limit Upper 32 Bits Register (Prefetchable_Limit_Upper_32_Bits_Register)

Address: 2Ch

Prefetchable_Limit_Upper_32_Bits_Register field descriptions

Field	Description
PF_Limit_Upper_32_Bits	Specifies bits 64-32 of the prefetchable memory space ending address when the address decode type field in the prefetchable memory limit register is 0x01.

18.9.14 PCI Express I/O Base Upper 16 Bits Register (IO_Base_Upper_16_Bits_Register)

Note that this device does not support inbound I/O transactions.

Address: 30h

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
Write	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	

IO_Base_Upper_16_Bits_Register field descriptions

Field	Description
IO_Base_Upper_16_Bits	Specifies bits 31-16 of the I/O space start address when the address decode type field in the I/O base register is 0x01.

18.9.15 PCI Express I/O Limit Upper 16 Bits Register (IO_Limit_Upper_16_Bits_Register)

Note that this device does not support inbound I/O transactions.

Address: 32h

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Read	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
Write	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	

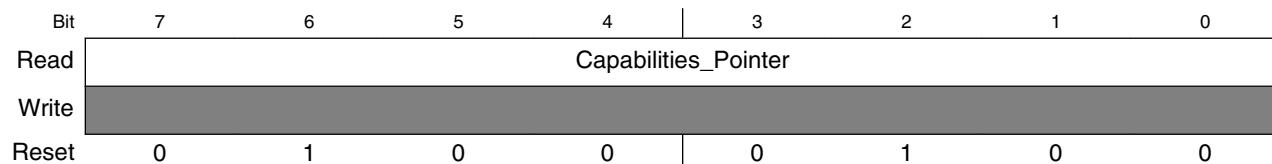
IO_Limit_Upper_16_Bits_Register field descriptions

Field	Description
IO_Limit_Upper_16_Bits	Specifies bits 31-16 of the I/O space ending address when the address decode type field in the I/O limit register is 0x01.

18.9.16 Capabilities Pointer Register (Capabilities_Pointer_Register)

The capabilities pointer identifies additional functionality supported by the device.

Address: 34h



Capabilities_Pointer_Register field descriptions

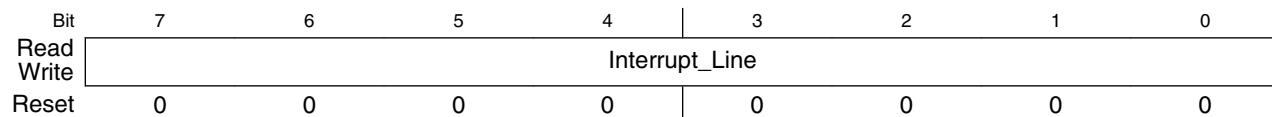
Field	Description
Capabilities_Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to PCI compatible device-specific configuration space for more information.

18.9.17 PCI Express Interrupt Line Register (Interrupt_Line_Register)

This register only applies to endpoint mode.

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

Address: 3Ch



Interrupt_Line_Register field descriptions

Field	Description
Interrupt_Line	Used to communicate interrupt line routing information.

18.9.18 PCI Express Interrupt Pin Register (Interrupt_Pin_Register)

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses when configured in EP mode.

This register only applies to EP mode. Reset value is 0x00 for RC mode and 0x01 for EP mode.

Address: 3Dh

Bit	7	6	5	4		3	2	1	0
Read					Interrupt_pin				
Write									
Reset	0	0	0	0		0	0	0	0

Interrupt_Pin_Register field descriptions

Field	Description
Interrupt_pin	<p>Legacy INTx message used by this device. All other settings reserved.</p> <p>0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD</p>

18.9.19 PCI Express Bridge Control Register (Bridge_Control_Register)

Address: 3Eh

Bit	15	14	13	12		11	10	9	8
Read					Reserved				
Write									
Reset	0	0	0	0		0	0	0	0

Bit	7	6	5	4		3	2	1	0
Read	Reserved	Scnd_RST	Reserved		VGA_EN	ISA_EN	SERR_EN	PER	
Write					0	0	0	0	
Reset	0	0	0	0		0	0	0	0

Bridge_Control_Register field descriptions

Field	Description
15–7 -	This field is reserved. Reserved
6 Scnd_RST	Secondary bus reset
5–4 -	This field is reserved. Reserved
3 VGA_EN	VGA enable

Table continues on the next page...

Bridge_Control_Register field descriptions (continued)

Field	Description
2 ISA_EN	ISA enable
1 SERR_EN	SERR enable. This bit controls the propagation of ERR_COR, ERR_NONFATAL, and ERR_FATAL responses received on the secondary side.
0 PER	Parity error response.

18.10 PCI compatible device-specific configuration space

The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just after the 64-byte PCI-compatible configuration header).

		Address Offset (Hex)
Reserved		00
	PCI-Compatible Configuration Header	3F
		40
Power Mgmt Capabilities	Next Pointer (0x4C)	44
Data	Power Management Status & Control	48
PCI Express Capabilities	Next Pointer (EP: 0x88/RC: NULL)	4C
	PCI Express Capability ID	
	Device Capabilities	50
Device Status	Device Control	54
	Link Capabilities	58
Link Status	Link Control	5C
	Slot Capabilities	60
Slot Status	Slot Control	64
	Root Control (RC mode only)	68
	Root Status	6C
	Device Capabilities 2	70
Device Status 2	Device Control 2	74
	Link Capabilities 2	78
Link Status 2	Link Control 2	7C
	Slot Capabilities 2	80
Slot Status 2	Slot Control 2	84
MSI Message Control	Next Pointer (NULL)	88
	MSI Message Capability ID	
	MSI Message Address	8C
	MSI Upper Message Address	90
	MSI Message Data	94
		98
		FF

Figure 18-296. PCI Compatible Device-Specific Configuration Space

18.10.1 PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register)

Address: 44h

Bit	7	6	5	4		3	2	1	0
Read	Power_Mgmt_Capability_ID								
Write									
Reset	0	0	0	0		0	0	0	1

Power_Management_Capability_ID_Register field descriptions

Field	Description
Power_Mgmt_Capability_ID	Power Management = 0x01

18.10.2 PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)

Address: 46h

Bit	15	14	13	12		11	10	9	8	
Read	PME_Support									
Write										
Reset	n*	1*	1*	1*		1*	1	1	0	
Bit	7	6	5	4		3	2	1	0	
Read	AUX_Curr		DSI		Reserved		PME_CLK		Version	
Write										
Reset	0	0	0	0		0	0	1	1	

* Notes:

- PME_Support field: Reset value is 11111 in RC mode and 01111 in EP mode.

Power_Management_Capabilities_Register field descriptions

Field	Description
15–11 PME_Support	Indicates the power states that this device supports
10 D2	D2 Support
9 D1	D1 Support

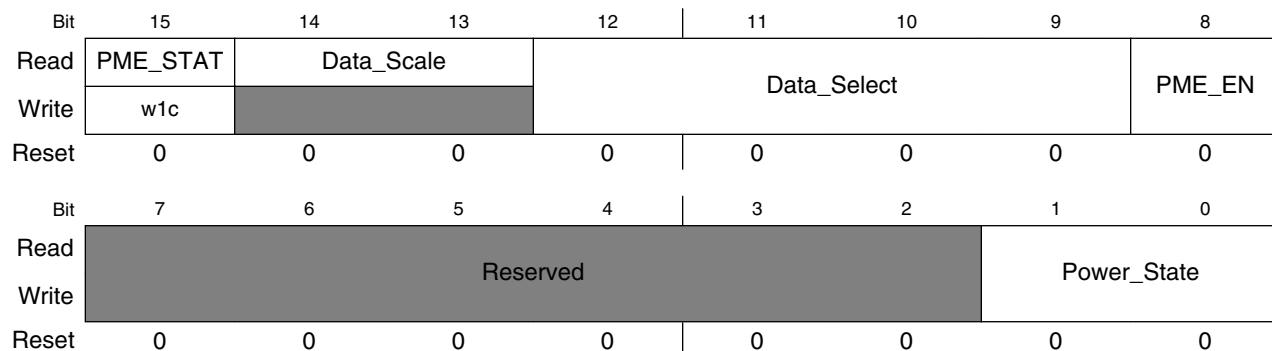
Table continues on the next page...

Power_Management_Capabilities_Register field descriptions (continued)

Field	Description
8–6 AUX_Curr	AUX Current
5 DSI	Device Specific Initialization
4 -	This field is reserved. Reserved
3 PME_CLK	Does not apply to PCI Express.
Version	Setting depends on version of the specification.

18.10.3 PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register)

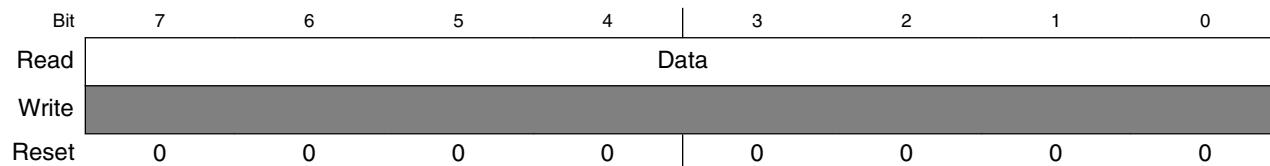
Address: 48h

**Power_Management_Status_and_Control_Register field descriptions**

Field	Description
15 PME_STAT	PME Status
14–13 Data_Scale	Obtained directly from the <i>PCI Express Base Specification</i> .
12–9 Data_Select	Obtained directly from the <i>PCI Express Base Specification</i> .
8 PME_EN	PME Enable NOTE: Bitfield access is sticky.
7–2 -	This field is reserved. Reserved
Power_State	Power state. Indicates the current power state of the function. 0x00 D0 0x01 D1 0x02 D2 0x03 D3

18.10.4 PCI Express Power Management Data Register (Power_Management_Data_Register)

Address: 4Bh

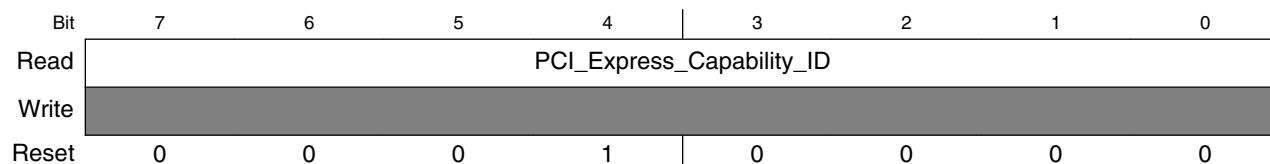


Power_Management_Data_Register field descriptions

Field	Description
Data	Obtained directly from the <i>PCI Express Base Specification</i> .

18.10.5 PCI Express Capability ID Register (Capability_ID_Register)

Address: 4Ch

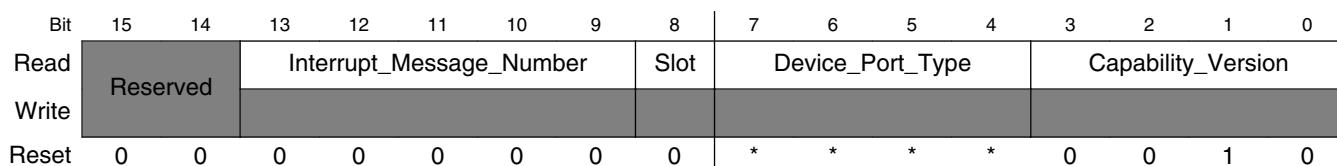


Capability_ID_Register field descriptions

Field	Description
PCI_Express_Capability_ID	PCI Express = 0x10

18.10.6 PCI Express Capabilities Register (Capabilities_Register)

Address: 4Eh



* Notes:

- Device_Port_Type field: Reset value depends on mode: 0100 (RC mode), 0000 (EP mode)

Capabilities_Register field descriptions

Field	Description
15–14 -	This field is reserved. Reserved
13–9 Interrupt_ Message_ Number	If this function is allocated more than one MSI interrupt number, then this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register, of this capability structure, are set.
8 Slot	Slot Implemented (RC mode only)
7–4 Device_Port_ Type	Device_Port_Type 0100 (RC mode) 0000 (EP mode)
Capability_ Version	Indicates the defined PCI Express capability structure version number. Must be 2h for 2.0 specification.

18.10.7 PCI Express Device Capabilities Register (Device_Capabilities_Register)

Refer to PCI Express Base Specification for register details.

Address: 50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				CSPLS		CSPLV						Reserved			
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserv ed	PIP	AIP	ABP	EP_L1_LAT			EP_L0s_LAT			ET	PHAN_FCT	MAX_PL_SIZE_SUP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Device_Capabilities_Register field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–26 CSPLS	Captured Slot Power Limit Scale
25–18 CSPLV	Captured Slot Power Limit Value
17–15 -	This field is reserved. Reserved

Table continues on the next page...

Device_Capabilities_Register field descriptions (continued)

Field	Description
14 PIP	Power Indicator Present
13 AIP	Attention Indicator Present
12 ABP	Attention Button Present
11–9 EP_L1_LAT	Endpoint L1 Acceptable Latency
8–6 EP_L0s_LAT	Endpoint L0s Acceptable Latency
5 ET	Extended Tag Field Supported
4–3 PHAN_FCT	Phantom Functions Supported
MAX_PL_SIZE_SUP	Maximum payload size supported.

18.10.8 PCI Express Device Control Register (Device_Control_Register)

Address: 54h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved	MAX_READ_SIZE			NSE	APE	PFE	ETE
Reset	0	0	1	0	1	0	0	0
Bit	7	6	5	4	3	2	1	0
Read Write	MAX_PAYLOAD_SIZE			RO	URR	FER	NFER	CER
Reset	0	0	0	1	0	0	0	0

Device_Control_Register field descriptions

Field	Description
15 -	This field is reserved. Reserved
14–12 MAX_READ_SIZE	Maximum read request size
11 NSE	No snoop enable
10 APE	AUX power PM enable
9 PFE	Phantom functions enable

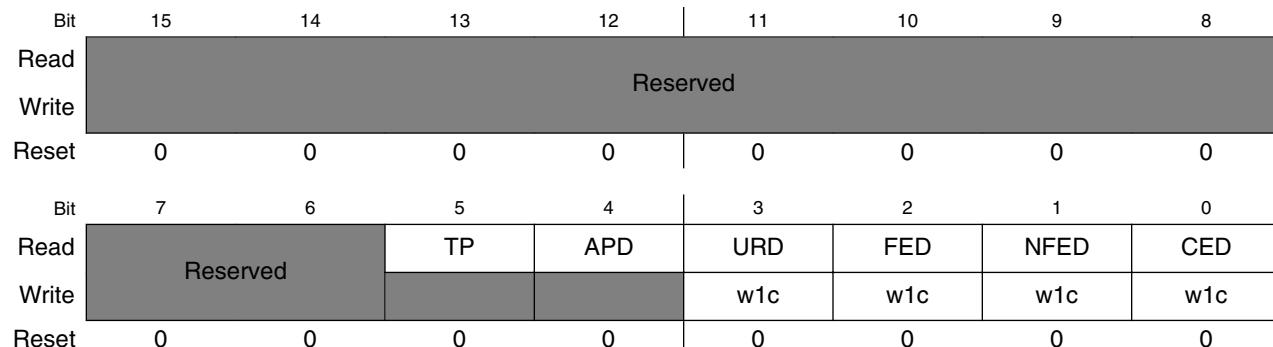
Table continues on the next page...

Device_Control_Register field descriptions (continued)

Field	Description
8 ETE	Extended tag field enable
7–5 MAX_PAYLOAD_SIZE	Maximum payload size 000 For 128 bytes MAX_PAYLOAD_SIZE 001 For 256 bytes MAX_PAYLOAD_SIZE
4 RO	Relaxed ordering NOTE: Configuration software must clear this bit if relaxed ordering is not desired. 1 Relaxed ordering is enabled.
3 URR	Unsupported request reporting
2 FER	Fatal error reporting
1 NFER	Non-fatal error reporting
0 CER	Correctable error reporting

18.10.9 PCI Express Device Status Register (Device_Status_Register)

Address: 56h

**Device_Status_Register field descriptions**

Field	Description
15–6 -	This field is reserved. Reserved
5 TP	Transactions pending
4 APD	AUX power detected
3 URD	Unsupported request detected

Table continues on the next page...

Device_Status_Register field descriptions (continued)

Field	Description
2 FED	Fatal error detected
1 NFED	Non-fatal error detected
0 CED	Correctable error detected

18.10.10 PCI Express Link Capabilities Register (Link_Capabilities_Register)

NOTE

See bitfield descriptions for reset values.

Address: 58h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Port_Number								Reserved		LBWN	Reserved		L1_EX_LAT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	n	0	0	0	1	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L1_EX_LAT	L0s_EX_LAT		ASPM		MAX_LINK_W					MAX_LINK_SP					
W																
Reset	1	1	0	1	0	1	0	0	n	n	n	n	0	0	1	0

Link_Capabilities_Register field descriptions

Field	Description
31–24 Port_Number	Port number for PCI Express link

Table continues on the next page...

Link_Capabilities_Register field descriptions (continued)

Field	Description
23–22 -	This field is reserved. Reserved
21 LBWN	Link bandwidth notification capable. Reset value depends on RC/EP configuration: In EP-mode, this bit is 0; in RC-mode it is 1. 0 EP 1 RC
20–18 -	This field is reserved. Reserved
17–15 L1_EX_LAT	L1 exit latency Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock. 000 Less than 1 µs 001 1 µs to less than 2 µs 010 2 µs to less than 4µs 011 4 µs to less than 8µs 100 8 µs to less than 16 µs 101 16µs to less than 32µs 110 32µs - 64 µs 111 More than 64 µs
14–12 L0s_EX_LAT	L0s exit latency Note that exit latencies may be influenced by PCI Express reference clock configuration depending upon whether a component uses a common or separate reference clock. 000 Less than 64 ns 001 64 ns to less than 128 ns 010 128 ns to less than 256 ns 011 256 ns to less than 512 ns 100 512 ns to less than 1µs 101 1 µs to less than 2 µs 110 2 µs – 4 µs 111 More than 4 µs
11–10 ASPM	Active state power management (ASPM) Support 01 L0s entry supported 11 L0s and L1entry supported
9–4 MAX_LINK_W	Maximum link width. Reset value depends upon the maximum link width of the PCI Express controller is capable of supporting. 000001 x1 000010 x2 000100 x4 001000 x8
MAX_LINK_SP	Maximum link speed 0010 5.0 GT/s and 2.5 GT/s link

18.10.11 PCI Express Link Control Register (Link_Control_Register)

Address: 5Ch

Bit	15	14	13	12	11	10	9	8
Read	Reserved				LABIE	LBMIE	HWAWD	Reserved
Write					0	0	0	0
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read	EXT_SYNC	CCC	RL		RCB	Reserved	ASPM_CTL	
Write	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0

Link_Control_Register field descriptions

Field	Description
15–12 -	This field is reserved. Reserved
11 LABIE	Link autonomous bandwidth interrupt enable
10 LBMIE	Link bandwidth management interrupt enable
9 HWAWD	Hardware autonomous width disable
8 -	This field is reserved. Reserved
7 EXT_SYNC	Extended sync
6 CCC	Common clock configuration
5 RL	Retrain link (Reserved for EP devices). In RC mode, setting this bit initiates link retraining by directing the Physical Layer LTSSM to the Recovery state, if the link has already been up; reads of this bit always return 0.
4 LD	Link disable (Reserved for EP devices)
3 RCB	Read completion boundary
2 -	This field is reserved. Reserved
ASPM_CTL	Active state power management (ASPM) control

18.10.12 PCI Express Link Status Register (Link_Status_Register)

Address: 5Eh

Bit	15	14	13	12	11	10	9	8
Read	LABS	LBMS	Reserved	SCC	LT	Reserved	NEG_LINK_W	
Write	w1c	w1c						
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read	NEG_LINK_W				LINK_SP			
Write								
Reset	0	0	0	1	0	0	0	1

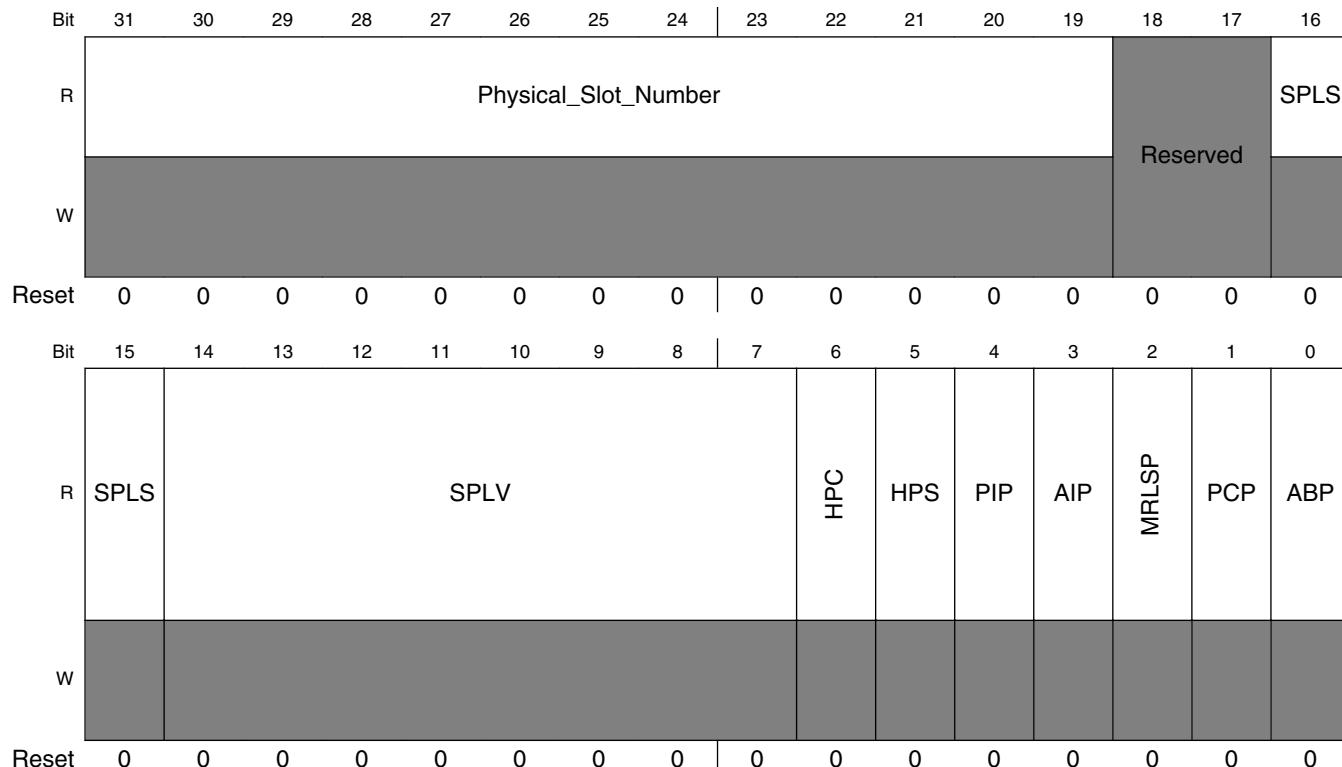
Link_Status_Register field descriptions

Field	Description
15 LABS	Link autonomous bandwidth status NOTE: This bifield is write-1-to-clear in RC mode; it is read-only in EP mode.
14 LBMS	Link bandwidth management status NOTE: This bifield is write-1-to-clear in RC mode; it is read-only in EP mode.
13 -	This field is reserved. Reserved
12 SCC	Slot clock configuration
11 LT	Link training
10 -	This field is reserved. Reserved.
9–4 NEG_LINK_W	Negotiated link width
LINK_SP	Negotiated link speed. 0001 2.5 GT/s 0010 5.0 GT/s

18.10.13 PCI Express Slot Capabilities Register (Slot_Capabilities_Register)

This register is supported only for RC mode.

Address: 60h



Slot_Capabilities_Register field descriptions

Field	Description
31–19 Physical_Slot_Number	This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18–17 -	This field is reserved. Reserved
16–15 SPLS	Slot power limit scale.
14–7 SPLV	Slot power limit value.
6 HPC	Hot plug capable.

Table continues on the next page...

Slot_Capabilities_Register field descriptions (continued)

Field	Description
5 HPS	Hot plug surprise.
4 PIP	Power indicator present.
3 AIP	Attention indicator present.
2 MRLSP	MRL sensor present.
1 PCP	Power controller present.
0 ABP	Attention button present.

18.10.14 PCI Express Slot Control Register (Slot_Control_Register)

This register is supported only for RC mode.

Address: 64h

Bit	15	14	13	12	11	10	9	8
Read Write	Reserved					PCC	PIC	
Reset	0	0	0	0	0	0	1	1
Bit	7	6	5	4	3	2	1	0
Read Write	AIC		HPIE	CCIE	PDCE	MRLSCE	PFDE	ABPE
Reset	1	1	0	0	0	0	0	0

Slot_Control_Register field descriptions

Field	Description
15–11 -	This field is reserved. Reserved
10 PCC	Power controller control.
9–8 PIC	Power indicator control. If a power indicator is implemented, set the power indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting for the previous command to complete in which case the read value is undefined. Defined encodings are shown below. NOTE: The default value of this field must be one of the non-reserved values. If the power indicator present bit in the slot capabilities register is 0, this bit is permitted to be read-only with a value of 00.

Table continues on the next page...

Slot_Control_Register field descriptions (continued)

Field	Description
	00 Reserved 01 On 10 Blink 11 Off
7–6 AIC	<p>Attention indicator control.</p> <p>If an attention indicator is implemented, writes to this field set the attention indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting for the previous command to complete in which case the read value is undefined.</p> <p>NOTE: The default value of this field must be one of the non-reserved values. If the attention indicator present bit in the slot capabilities register is 0, this bit is permitted to be readonly with a value of 00.</p> 00 Reserved 01 On 10 Blink 11 Off
5 HPIE	Hot plug interrupt enable.
4 CCIE	Command completed interrupt enable.
3 PDCE	Presence detect changed enable.
2 MRLSCE	MRL sensor changed enable.
1 PFDE	Power fault detected enable.
0 ABPE	Attention button pressed enable.

18.10.15 PCI Express Slot Status Register (Slot_Status_Register)

This register is supported only for RC mode.

Address: 66h

Bit	15	14	13	12		11	10	9	8
Read	Reserved								
Write									
Reset	0	0	0	0		0	0	0	0
Bit	7	6	5	4		3	2	1	0
Read	Reserved	PDS	MRLSS	CC	PDC	MRLSC	PFD	ABP	
Write									
Reset	0	1	0	0		1	0	0	0

Slot_Status_Register field descriptions

Field	Description
15–7 -	This field is reserved. Reserved
6 PDS	Presence detect state. In the PCI Express specification, this bit indicates the presence of a card in the slot; however, on this device, this bit is set regardless of whether there is a card present or not. 0 Slot empty 1 Card present in slot
5 MRLSS	MRL sensor state. 0 MRL closed 1 MRL open
4 CC	Command completed.
3 PDC	Presence detect changed.
2 MRLSC	MRL sensor changed.
1 PFD	Power fault detected.
0 ABP	Attention button pressed.

18.10.16 PCI Express Root Control Register (Root_Control_Register)

This register is supported only for RC mode.

Address: 68h

Bit	15	14	13	12		11	10	9	8
Read Write	Reserved								
Reset	0	0	0	0		0	0	0	0
Bit	7	6	5	4		3	2	1	0
Read Write	Reserved				PMEIE	SEFEE	SENFEE	SECEE	
Reset	0	0	0	0	0	0	0	0	0

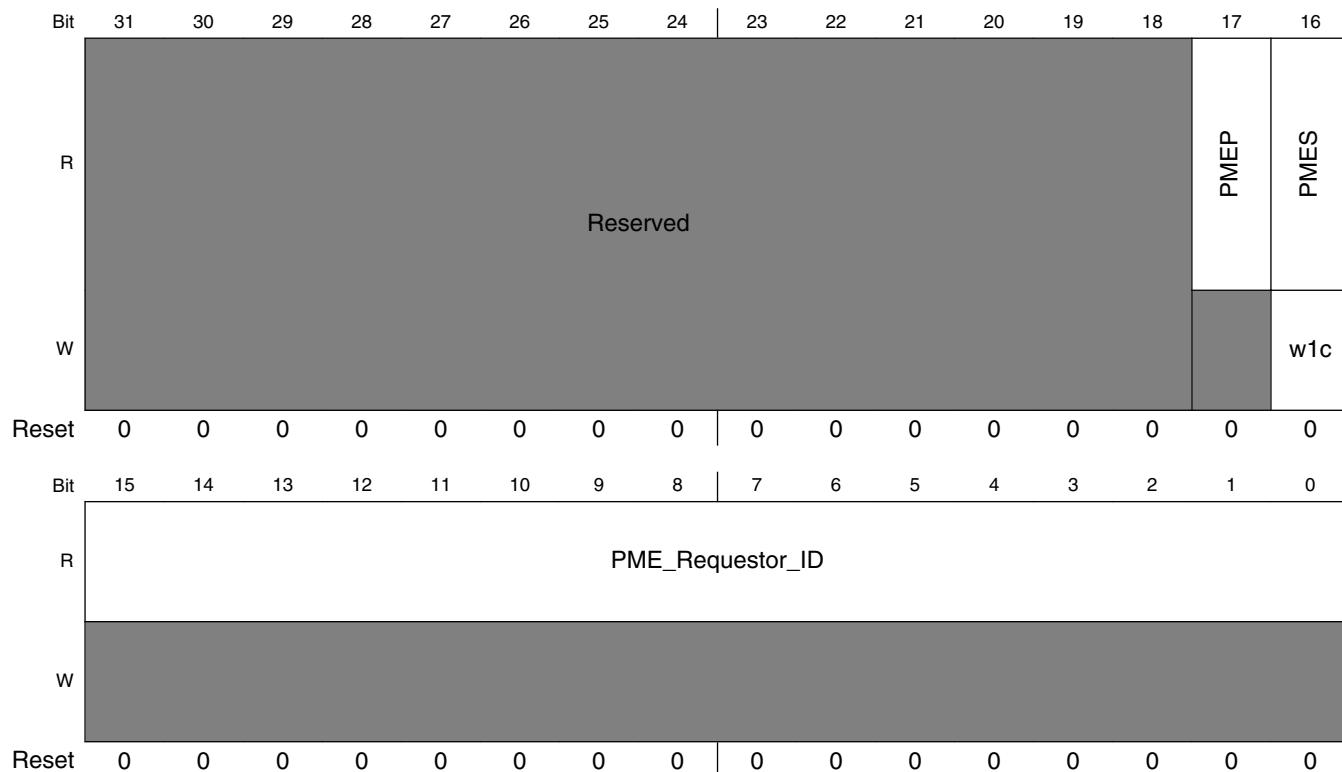
Root_Control_Register field descriptions

Field	Description
15–4 -	This field is reserved. Reserved
3 PMEIE	PME interrupt enable.
2 SEFEE	System error on fatal error enable.
1 SENFEE	System error on non-fatal error enable.
0 SECEE	System error on correctable error enable.

18.10.17 PCI Express Root Status Register (Root_Status_Register)

This register is supported only for RC mode.

Address: 6Ch



Root_Status_Register field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 PMEP	PME pending.
16 PMES	PME status.
PME_Requestor_ID	PME requestor ID.

18.10.18 PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register)

Address: 70h

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

Device_Capabilities_2_Register field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 CPL_TO_DS	Completion timeout disable supported
CPL_TO_RS	Completion timeout ranges supported

18.10.19 PCI Express Device Control 2 Register (Device_Control_2_Register)

Address: 74h

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
Read	Reserved																
Write																	
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Device_Control_2_Register field descriptions

Field	Description
15–5 -	This field is reserved. Reserved
4 CPL_TOD	Completion timeout disable
CPL_TO_VAL	Completion timeout value

18.10.20 PCI Express Link Control 2 Register (Link_Control_2_Register)

Address: 7Ch

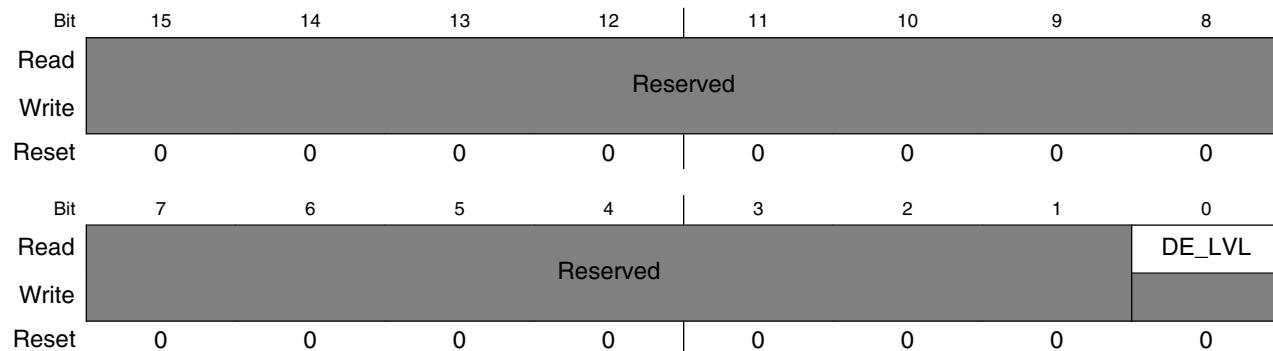
Bit	15	14	13	12	11	10	9	8
Read Write	Reserved			CDE	CSOS	EMC	TxM	
Reset	0	0	0	0	0	0	0	0
Bit	7	6	5	4	3	2	1	0
Read Write	TxM	SDE	HWASD	EC	T_LS			
Reset	0	0	0	0	0	0	1	0

Link_Control_2_Register field descriptions

Field	Description
15–13 -	This field is reserved. Reserved
12 CDE	Compliance de-emphasis
11 CSOS	Compliance SOS
10 EMC	Enter modified compliance
9–7 TxM	Transmit margin
6 SDE	Selectable de-emphasis
5 HWASD	Hardware autonomous speed disable
4 EC	Enter compliance
T_LS	Target link speed

18.10.21 PCI Express Link Status 2 Register (Link_Status_2_Register)

Address: 7Eh

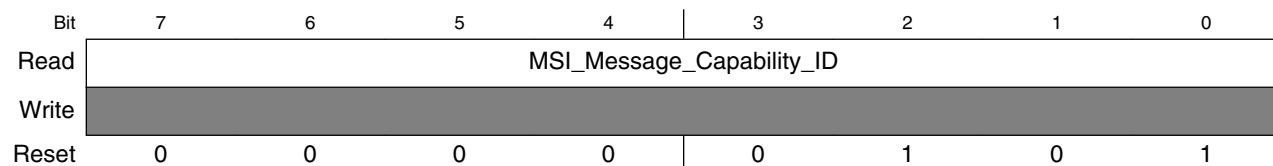


Link_Status_2_Register field descriptions

Field	Description
15–1 -	This field is reserved. Reserved
0 DE_LVL	Current de-emphasis level

18.10.22 PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register)

Address: 88h

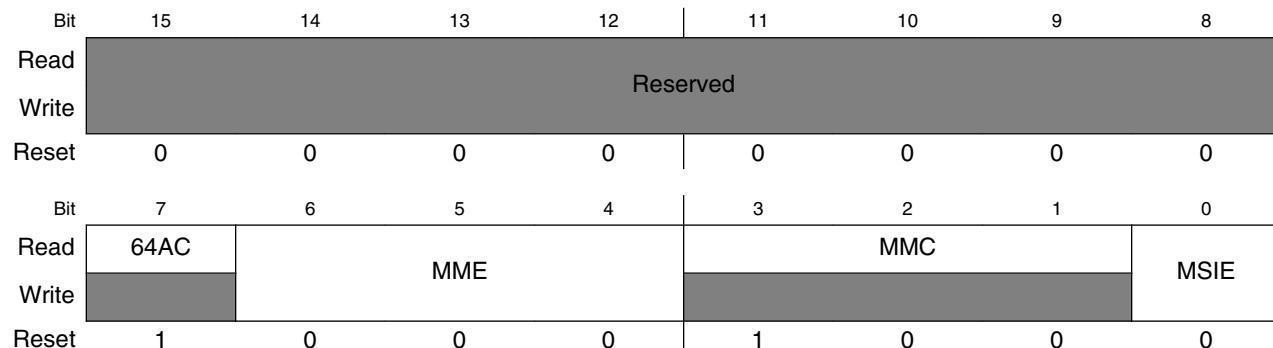


MSI_Message_Capability_ID_Register field descriptions

Field	Description
MSI_Message_Capability_ID	MSI Message = 0x05

18.10.23 PCI Express MSI Message Control Register (MSI_Message_Control_Register)

Address: 8Ah



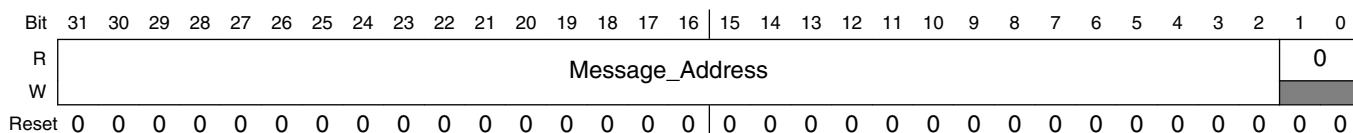
MSI_Message_Control_Register field descriptions

Field	Description
15–8 -	This field is reserved. Reserved
7 64AC	64-bit address capable.
6–4 MME	Multiple message enable.
3–1 MMC	Multiple message capable.
0 MSIE	MSI enable.

18.10.24 PCI Express MSI Message Address Register (MSI_Message_Address_Register)

This register only applies to endpoint mode.

Address: 8Ch



MSI_Message_Address_Register field descriptions

Field	Description
31–2 Message_Address	System-specified message address
Reserved	This read-only field is reserved and always has the value 0.

18.10.25 PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register)

This register only applies to endpoint mode.

Address: 90h

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

Reset 0

MSI_Message_Upper_Address_Register field descriptions

Field	Description
Message_Upper_Address	System-specified message upper address

18.10.26 PCI Express MSI Message Data Register (MSI_Message_Data_Register)

This register is supported only for EP mode.

Address: 94h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Read																	
Write																	

Reset 0

MSI_Message_Data_Register field descriptions

Field	Description
Message_Data	System-specified message.

18.11 PCI Express extended configuration space

Address Offset (Hex)	
000	PCI-Compatible Configuration Header
03F	
040	PCI-Compatible Device-Specific Configuration Space
0FF	
100	Next Capability Offset (NULL)/Capability Version
104	Advanced Error Reporting Capability ID
108	Uncorrectable Error Status
10C	Uncorrectable Error Mask
110	Uncorrectable Error Severity
114	Correctable Error Status
118	Correctable Error Mask
11C	Advanced Error Capabilities and Control
120	
124	Header Log
128	
12C	Root Error Command (RC mode only)
130	Root Error Status (RC mode only)
134	Error Source ID
138	Correctable Error Source ID
3FF	
400	PCI Express Controller Internal CSRs*
6FF	
700	
FFF	

*Note that the PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers returns all 0s.

Figure 18-323. PCI Express extended configuration space

18.11.1 PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reportng_Capability_ID_Register)

Address: 100h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Capability_ID															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Advanced_Error_Reportng_Capability_ID_Register field descriptions

Field	Description														
Capability_ID	Advanced error reporting capability = 0x0001														

18.11.2 PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)

Address: 104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R													URE	ECRCE	MTLP	RXO	UC
W													w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	CA	CTO	FCPE	PTLP									DLPE				TE
W	w1c	w1c	w1c	w1c									w1c				w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Uncorrectable_Error_Status_Register field descriptions

Field	Description														
31–21 -	This field is reserved. Reserved														

Table continues on the next page...

Uncorrectable_Error_Status_Register field descriptions (continued)

Field	Description
20 URE	Unsupported request error status.
19 ECRCE	ECRC error status.
18 MTLP	Malformed TLP status.
17 RXO	Receiver overflow status.
16 UC	Unexpected completion status.
15 CA	Completer abort status.
14 CTO	Completion timeout status. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system.
13 FCPE	Flow control protocol error status.
12 PTLP	Poisoned TLP status.
11–5 -	This field is reserved. Reserved
4 DLPE	Data link protocol error status.
3–1 -	This field is reserved. Reserved
0 TE	Training error status.

18.11.3 PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)

Address: 108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
R	Reserved												UREM	ECRCEM	MTLPM	RXOM	UCM
W													UREM	ECRCEM	MTLPM	RXOM	UCM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
R	CAM	CTOM	FCPEM	PTLPM	Reserved								DLPEM	Reserved			TEM
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Uncorrectable_Error_Mask_Register field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 UREM	Unsupported request error mask
19 ECRCEM	ECRC error mask
18 MTLPM	Malformed TLP mask
17 RXOM	Receiver overflow mask
16 UCM	Unexpected completion mask
15 CAM	Completer abort mask
14 CTOM	Completion timeout mask
13 FCPEM	Flow control protocol error mask
12 PTLPM	Poisoned TLP mask
11–5 -	This field is reserved. Reserved
4 DLPEM	Data link protocol error mask

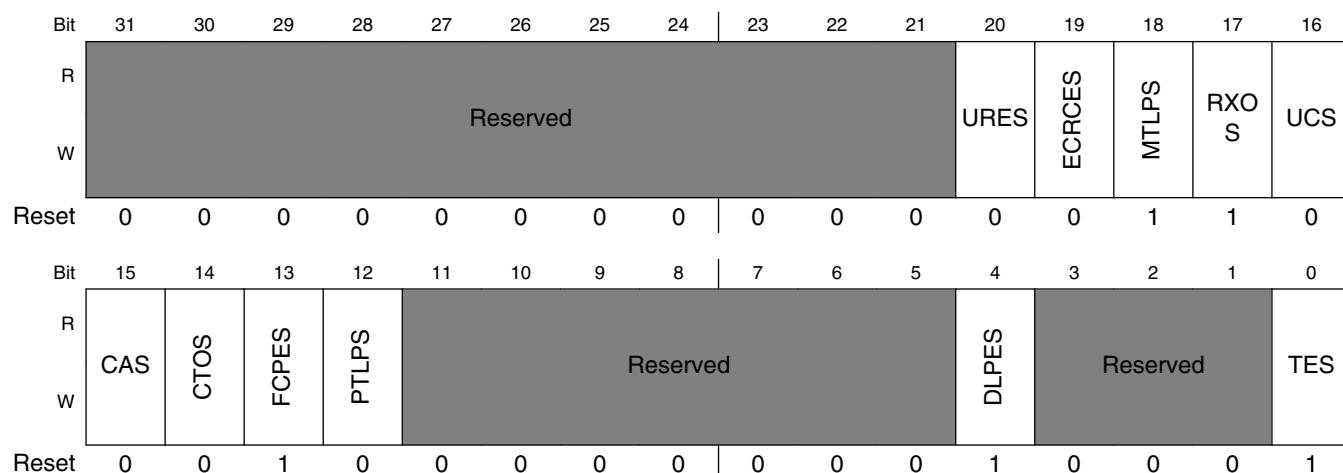
Table continues on the next page...

Uncorrectable_Error_Mask_Register field descriptions (continued)

Field	Description
3–1 -	This field is reserved. Reserved
0 TEM	Training error mask

18.11.4 PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register)

Address: 10Ch

**Uncorrectable_Error_Severity_Register field descriptions**

Field	Description
31–21 -	This field is reserved. Reserved
20 URES	Unsupported request error severity
19 ECRCES	ECRC error severity
18 MTLPS	Malformed TLP severity
17 RXOS	Receiver overflow severity
16 UCS	Unexpected completion severity
15 CAS	Completer abort severity
14 CTOS	Completion timeout severity

Table continues on the next page...

Uncorrectable_Error_Severity_Register field descriptions (continued)

Field	Description
13 FCPES	Flow control protocol error severity
12 PTLPS	Poisoned TLP severity
11–5 -	This field is reserved. Reserved
4 DLPES	Data link protocol error severity
3–1 -	This field is reserved. Reserved
0 TES	Training error severity

18.11.5 PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)

Address: 110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R			ADVNF	RTTO				RNR	BDLLP	BTLP						RXE
			Reserved													
W			w1c	w1c				w1c	w1c	w1c						w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Correctable_Error_Status_Register field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 ADVNF	Advisory Non-Fatal Error Status indicates the occurrence of the advisory error, and the Advisory Non-Fatal Error Mask bit in the Correctable Error Mask register is checked to determine whether to proceed further with logging and signaling
12 RTTO	Replay timer timeout status
11–9 -	This field is reserved. Reserved
8 RNR	REPLAY_NUM Rollover status
7 BDLLP	Bad DLLP status
6 BTLP	Bad TLP status
5–1 -	This field is reserved. Reserved
0 RXE	Receiver error status

18.11.6 PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)

Address: 114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		ADVNFEM	RTTOM	Reserved			RNRM	BDLLPM	BTLP	Reserved					RXEM
W																
Reset	0	0			1	0	0				0	0	0	0	0	

Correctable_Error_Mask_Register field descriptions

Field	Description
31–14 -	This field is reserved. Reserved

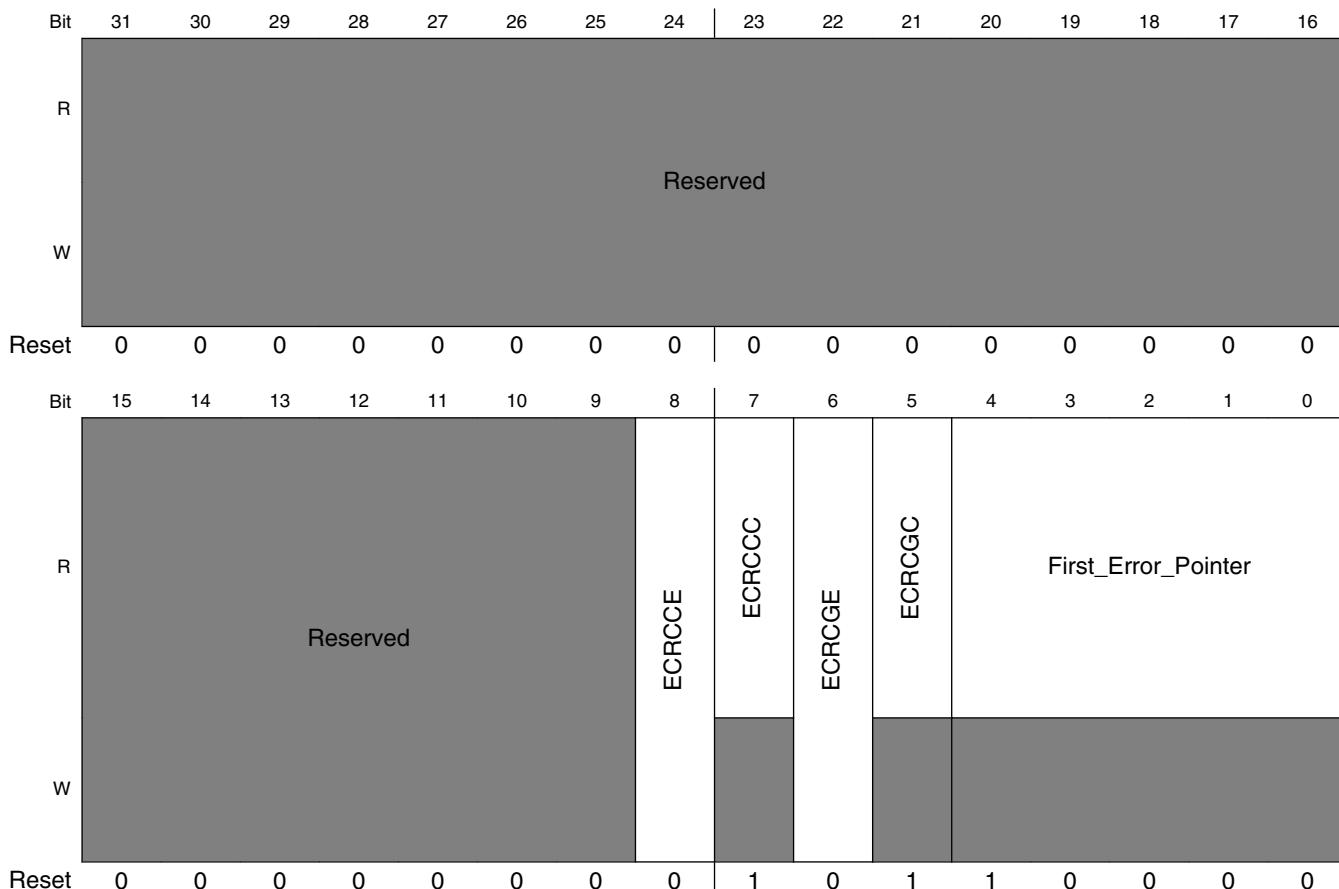
Table continues on the next page...

Correctable_Error_Mask_Register field descriptions (continued)

Field	Description
13 ADVNFEM	Advisory Non-Fatal Error Mask – This bit is Set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
12 RTTOM	Replay timer timeout mask
11–9 -	This field is reserved. Reserved
8 RNRM	REPLAY_NUM Rollover mask
7 BDLLPM	Bad DLLP mask
6 BTLPBM	Bad TLP mask
5–1 -	This field is reserved. Reserved
0 RXEM	Receiver error mask

18.11.7 PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)

Address: 118h



Advanced_Error_Capabilities_and_Control_Register field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 ECRCCE	ECRC checking enable On P4080 Rev 2, end-to-end CRC (ECRC) errors on packets may not be detected even if ECRC checking is enabled. Therefore, disable ECRC checking on P4080, Rev 2, by clearing the ECRC checking enable bit ([ECRCCE] = 0). Note that this does not affect the ability of the device to properly detect and respond to link-level CRC (LCRC) errors. ECRC checking is fully functional on P4080, Rev 3.
7 ECRCCE	ECRC checking capable
6 ECRCGE	ECRC generation enable

Table continues on the next page...

Advanced_Error_Capabilities_and_Control_Register field descriptions (continued)

Field	Description
5 ECRCGC	ECRC generation capable
First_Error_Pointer	The First Error Pointer is a read-only field that identifies the bit position of the first error reported in the uncorrectable error status register (see PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)).

18.11.8 PCI Express Header Log Register 1 (Header_Log_Register_DWORD1)

The first DWORD of the PCI Express header log register is shown in the figure below.

Address: 11Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_0								Byte_1								Byte_2								Byte_3							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Header_Log_Register_DWORD1 field descriptions

Field	Description
31–24 Byte_0	Byte n of the TLP header associated with the error.
23–16 Byte_1	Byte n of the TLP header associated with the error.
15–8 Byte_2	Byte n of the TLP header associated with the error.
Byte_3	Byte n of the TLP header associated with the error.

18.11.9 PCI Express Header Log Register 2 (Header_Log_Register_DWORD2)

The second DWORD of the PCI Express header log register is shown in the figure below.

Address: 120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_4				Byte_5				Byte_6				Byte_7																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Header_Log_Register_DWORD2 field descriptions

Field	Description																													
31–24 Byte_4	Byte n of the TLP header associated with the error.																													
23–16 Byte_5	Byte n of the TLP header associated with the error.																													
15–8 Byte_6	Byte n of the TLP header associated with the error.																													
Byte_7	Byte n of the TLP header associated with the error.																													

18.11.10 PCI Express Header Log Register 3 (Header_Log_Register_DWORD3)

The third DWORD of the PCI Express header log register is shown in the figure below.

Address: 124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_8				Byte_9				Byte_A				Byte_B																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Header_Log_Register_DWORD3 field descriptions

Field	Description																													
31–24 Byte_8	Byte n of the TLP header associated with the error.																													

Table continues on the next page...

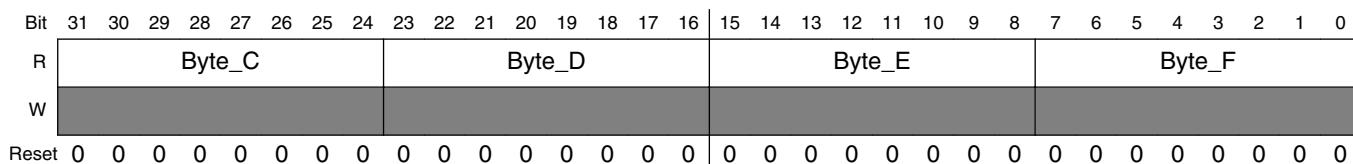
Header_Log_Register_DWORD3 field descriptions (continued)

Field	Description
23–16 Byte_9	Byte n of the TLP header associated with the error.
15–8 Byte_A	Byte n of the TLP header associated with the error.
Byte_B	Byte n of the TLP header associated with the error.

**18.11.11 PCI Express Header Log Register 4
(Header_Log_Register_DWORD4)**

The fourth DWORD of the PCI Express header log register is shown in the figure below.

Address: 128h

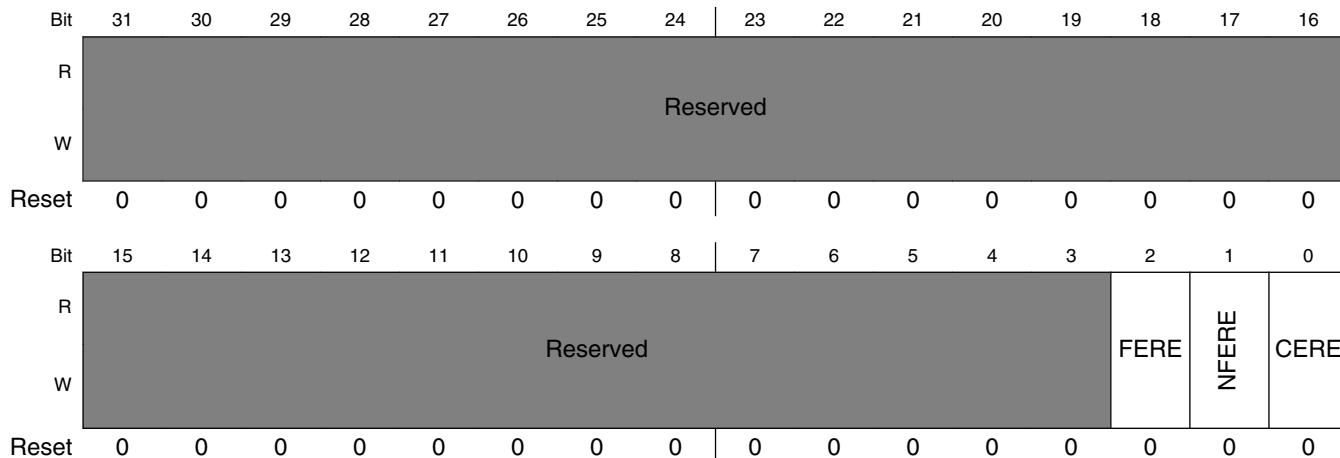
**Header_Log_Register_DWORD4 field descriptions**

Field	Description
31–24 Byte_C	Byte n of the TLP header associated with the error.
23–16 Byte_D	Byte n of the TLP header associated with the error.
15–8 Byte_E	Byte n of the TLP header associated with the error.
Byte_F	Byte n of the TLP header associated with the error.

18.11.12 PCI Express Root Error Command Register (Root_Error_Command_Register)

This register is supported only for RC mode.

Address: 12Ch



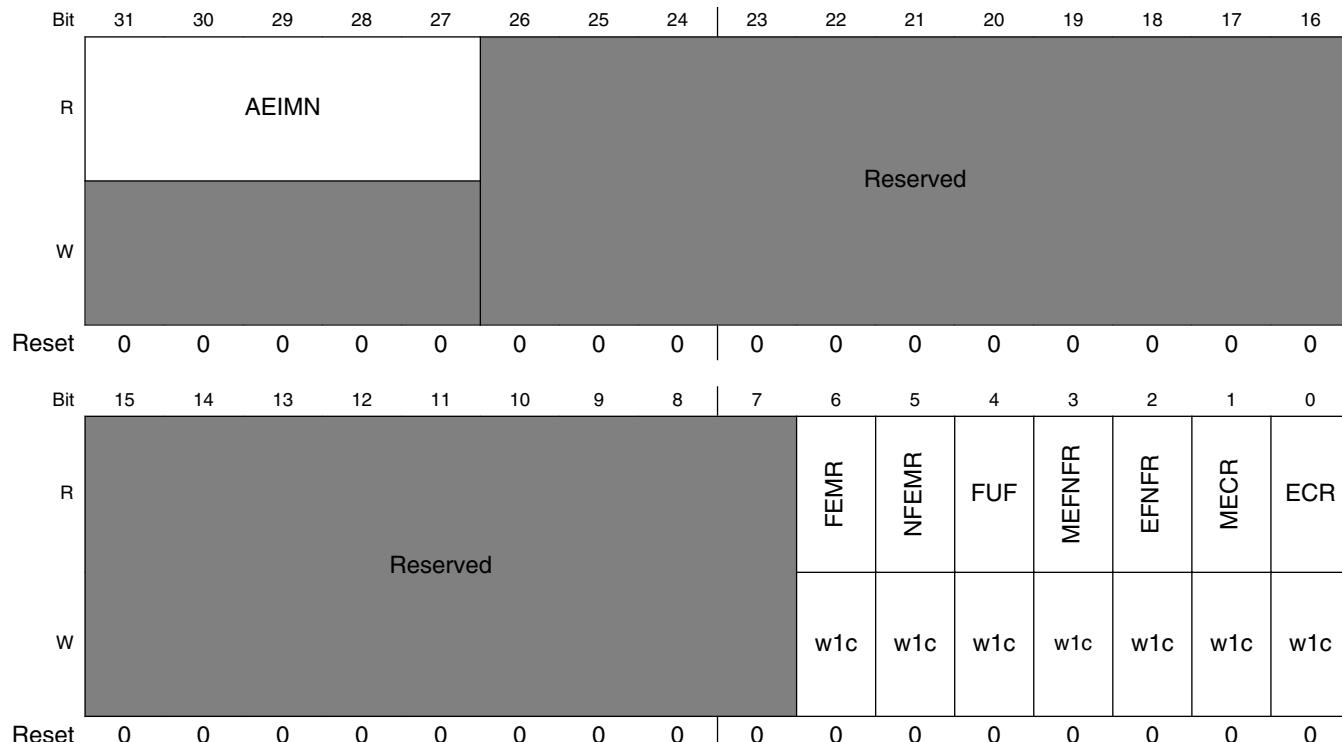
Root_Error_Command_Register field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
2 FERE	Fatal error reporting enable.
1 NFERE	Non-fatal error reporting enable
0 CERE	Correctable error reporting enable

18.11.13 PCI Express Root Error Status Register (Root_Error_Status_Register)

This register is supported only for RC mode.

Address: 130h



Root_Error_Status_Register field descriptions

Field	Description
31–27 AEIMN	Advanced error interrupt message number.
26–7 -	This field is reserved. Reserved
6 FEMR	Fatal error messages received.
5 NFEMR	Non-fatal error messages received.
4 FUF	First uncorrectable fatal.
3 MEFNFR	Multiple ERR_FATAL/NONFATAL received.
2 EFNFR	ERR_FATAL/NONFATAL received.

Table continues on the next page...

Root_Error_Status_Register field descriptions (continued)

Field	Description
1 MECR	Multiple ERR_COR received.
0 ECR	ERR_COR received.

18.11.14 PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register)

Address: 134h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	ERR_COR_Source_ID															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Correctable_Error_Source_ID_Register field descriptions

Field	Description
ERR_COR_Source_ID	Loaded with the Requestor ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.

18.11.15 PCI Express Error Source ID Register (Error_Source_ID_Register)

Address: 136h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	Error_Source_ID															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Error_Source_ID_Register field descriptions

Field	Description
Error_Source_ID	ERR_FATAL/NONFATAL source ID

18.11.16 LTSSM State Status Register (LTSSM_State_Status_Register)

The PCI Express link training and status state machine (LTSSM) state status register provides detailed information about link training status. This register is useful for debugging link training failures.

NOTE

The Status Code in PEX_LTSSM_STAT changes while reacting to the link status. Therefore, software may need to read PEX_LTSSM_STAT multiple times to ensure the value has stabilized.

The following table provides the encodings for the status code field of the PEX_LTSSM_STAT register.

Table 18-337. PEX_LTSSM_STAT Status Codes

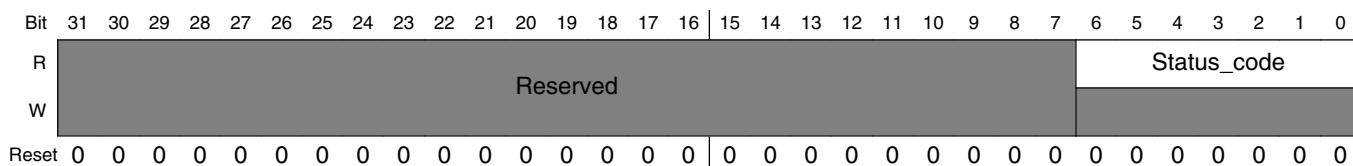
Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
00	Detect quiet	27	TX L0s FTS; RX L0s FTS
01	Detect active (0)	28	L0 to L1 (0)
02	Detect active (1)	29	L0 to L1 (1)
03	Detect active (2)	2A	L1 entry
04	Polling active (0)	2B	L1 idle (0)
05	Polling active (1)	2C	L1 idle (1)
06	Polling config (0)	2D	L0 to L2 (0)
07	Polling config (1)	2E	L0 to L2 (1)
08	Polling compliance	2F	L2 entry
09	Configuration link width start (0)	30	L2 idle (0)
0A	Configuration link width start (1)	31	L2 idle (1)
0B	Configuration link width accept (0)	32	Recovery lock (0)
0C	Configuration link width accept (1)	33	Recovery lock (1)
0D	Configuration lane number wait (0)	34	Recovery lock (2)
0E	Configuration lane number wait (1)	35	Recovery cfg (0)
0F	Configuration lane number wait (2)	36	Recovery cfg (1)
10	Configuration lane number wait (3)	37	Recovery idle (0)
11	Configuration lane number accept	38	Recovery idle (1)
12	Configuration complete (0)	39	Recovery to configuration
13	Configuration complete (1)	3A	Recovery cfg to configuration
14	Configuration idle (0)	3F	L0 no training

Table continues on the next page...

**Table 18-337. PEX_LTSSM_STAT Status Codes
(continued)**

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
15	Configuration idle (1)	7F	Detect quiet EI
16	L0	49	Configuration link width start-RC
17	TX L0; RX L0s entry	4A	Configuration link width accept-RC
18	TX L0; RX L0s idle	4B	Configuration lane number wait-RC
19	TX L0; RX L0s fast training sequence (FTS)	4C	Configuration lane number accept-RC
1A	TX L0s entry (0); RX L0	60	Loopback slave active (0)
1B	TX L0s entry (0); RX L0s idle	61	Loopback slave active (1)
1C	TX L0s entry (0); RX L0s FTS	62	Loopback slave exit
1D	TX L0s entry (1); RX L0	68	Hot reset (0)
1E	TX L0s entry (1); RX L0s idle	69	Hot reset (1)
1F	TX L0s entry (1); RX L0s FTS	6A	Hot reset (0)-RC
20	TX L0s idle; RX L0	6B	Hot reset (1)-RC
21	TX L0s idle; RX L0s entry	75	Disabled (0)
22	TX L0s idle; RX L0s idle	71	Disabled (1)
23	TX L0s idle; RX L0s FTS	72	Disabled (2)
24	TX L0s FTS; RX L0	73	Disabled (3)
25	TX L0s FTS; RX L0s entry	74	Disabled (4)
26	TX L0s FTS; RX L0s idle	78	L0 to L1/L2-RC

Address: 404h



LTSSM_State_Status_Register field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
Status_code	Status code. See Table 18-337 for encodings.

18.11.17 N_FTS Control Register (PEX_N_FTS_CTL)

The PCI Express N_FTS control register is used to set the N_FTS value that is advertised by the PCI Express controller during link training. If this value is changed after the link is up, the new value will take effect during the next link training. The N_FTS value is decided by the L0s exit latency of the RX link of the PHY.

Address: 41Ch

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

Reset 0

PEX_N_FTS_CTL field descriptions

Field	Description
31–24 N_FTS_CM_GEN2	N_FTS common mode Gen2. This is the number of Fast Training Sequence (FTS) ordered sets that the PHY requires to enable its RX circuits to achieve bit and symbol lock and come out of ASPM L0s link power state when devices on either side of the link use a common reference clock in 5Gbps (Gen2) data rate. This N_FTS value is advertised by the PCI Express controller to the remote device during link training if the common clock configuration (CCC) bit in the Link Control Register is set. At a given time, either N_FTS_GEN2 or N_FTS_CM_GEN2 mode value is used based on the setting of common clock configuration bit (CCC) in the Link Control Register.
23–16 N_FTS_GEN2	Number of FTS Gen2. This is the number of Fast Training Sequence (FTS) ordered sets that the PHY requires to enable its RX circuits to achieve bit and symbol lock and come out of ASPM L0s link power state in 5Gbps (Gen2) data rate. This N_FTS value is advertised by the PCI Express controller to the remote device during link training.
15–8 N_FTS_CM	N_FTS common mode. This is the number of Fast Training Sequence (FTS) ordered sets that the PHY requires to enable its RX circuits to achieve bit and symbol lock and come out of ASPM L0s link power state when devices on either side of the link use a common reference clock. This N_FTS value is advertised by the PCI Express controller to the remote device during link training if the common clock configuration (CCC) bit in the Link Control Register is set. At a given time, either N_FTS or N_FTS_CM value is used based on the setting of common clock configuration (CCC) bit in the Link Control Register.
N_FTS	Number of FTS This is the number of Fast Training Sequence (FTS) ordered sets that the PHY requires to enable its RX circuits to achieve bit and symbol lock and come out of ASPM L0s link power state. This N_FTS value is advertised by the PCI Express controller to the remote device during link training.

18.11.18 ACK Replay Time-Out Register (PEX_ACK_REPLY_TIMEOUT)

The PCI Express ACK replay time-out register is used to program time-out values for ACK DLLP transmission and reception in the data link layer. The ACK receive time-out is called the replay time-out since TLPs are re-transmitted after this timeout occurs. Both values should be in terms of PCI Express controller clock cycles.

NOTE

When ASPM is enabled, the value of the REPLAY_TIMEOUT field must be adjusted to avoid replay time-out errors. See [Configuring ACK replay time-out when ASPM is enabled](#) for more information.

Address: 438h

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

PEX_ACK_REPLY_TIMEOUT field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–13 REPLAY_ TIMEOUT	Time-out value to wait for reception of ACK DLLP from link side by DLL before re-transmitting TLPs.
ACK_LATENCY_ TIMEOUT	Time-out value to force transmission of ACK DLLP by DLL after a TLP is received.

18.11.19 PCI Express Controller Core Clock Ratio Register (Controller_Core_Clock_Ratio_Register)

The PCI Express controller clock frequency is one-half the platform clock frequency.

The PCI Express controller core clock ratio register is used to program the ratio of the actual PCI Express controller clock frequency to the default controller core frequency (333 MHz). This is required only when a PCI Express controller clock frequency other than the default 333 MHz has to be used.

As an example of programming PEX_GCLK_RATIO, consider the case where the actual PCI Express controller clock is 250 MHz, the ratio of the actual clock to the default clock (333 MHz) is 3:4. that is, the default core clock has to be multiplied by the ratio (3/4, which is equivalent to 12/16). So the register has to be programmed with the decimal numerator value 12 or 0x0000_000C.

Address: 440h

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

Controller_Core_Clock_Ratio_Register field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
Clock_Ratio_Numerator	The numerator of the ratio of the actual PCI Express controller clock frequency used to the default core clock frequency of 333 MHz . The denominator of the ratio is fixed at 16. The default value of this register is 0x10 (16 decimal), which corresponds to a ratio of 1:1 (or 16/16)

18.11.20 PCI Express Power Management Timer Register (Power_Management_Timer_Register)

The PCI Express power management timer register is used to program the time-in values for entering L0s and L1 power management states.

Address: 450h

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

Power_Management_Timer_Register field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–12 L1_WAIT_PERIOD	Wait period (in PCI Express controller core clock cycles) before entering L1 power state after all functions are in a non-D0 power state. The value is calculated as: Time (in μ sec) x PCI Express controller core clock frequency (in MHz) The time value must be less than 2 μ sec; the default value (0x190) is 1 μ sec for the default clock frequency of 400 MHz . If the system is not configured to run at 400 MHz, then the default value should be changed throughout the software as appropriate.

Table continues on the next page...

Power_Management_Timer_Register field descriptions (continued)

Field	Description
L0s_TIME_IN	<p>Time in value (in PCI Express controller core clock cycles) for entering L0s power state. The value is calculated as:</p> <p>Time (in μsec) x PCI Express controller core clock frequency (in MHz)</p> <p>The maximum time value is 7 μsec; the default value (0x960) is 6 μsec for the default clock frequency of 400 MHz. If the system is not configured to run at 400 MHz, then the default value should be changed throughout the software as appropriate.</p>

18.11.21 PCI Express PME Time-Out Register (PME_Time_Out_Register)

This register is supported only for EP mode.

The PCI Express PME time-out register is used to program the time-out value that the controller uses before re-sending a PME message to the host. If PME is requested by a function and the host does not clear the associated PME_STAT bit even after this time-out has expired, the PME message is sent again to the host by the PCI Express controller.

Address: 454h

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

PME_Time_Out_Register field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
PME_TIMEOUT	<p>The PME time-out value specifies the interval before PME messages are resent by the controller, provided the PME_STAT bit in the PCI Express power management status and control register (offset 0x48) is not cleared by the host. The value for PME_TIMEOUT is specified in terms of PCI Express controller core clock cycles. The value is calculated as:</p> <p>Time (in μsec) x PCI Express controller core clock frequency (in MHz)</p> <p>The minimum time value is 100 msec; the default value (0x1FC1E20) is 100 msec for the default clock frequency of 333 MHz.</p>

18.11.22 PCI Express Subsystem Vendor ID Update Register (Subsystem_Vendor_ID_Update_Register)

This register is supported only for EP mode.

The PCI Express subsystem vendor ID update register is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.

When used as an endpoint, the controller's initialization software programs the desired subsystem ID and subsystem vendor ID values in PEX_SSVID_UPDATE before setting the CFG_READY bit in the PEX_CFG_READY register (see [Configuration Ready Register \(Configuration_Ready_Register\)](#)). That way, when the host begins system enumeration, the correct values are present in the Type 0 configuration header.

Address: 478h

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

Reset 0

Subsystem_Vendor_ID_Update_Register field descriptions

Field	Description
31–16 SS_ID	Subsystem ID [15-0] value
SSV_ID	Subsystem vendor ID [15-0] value

18.11.23 Configuration Ready Register (Configuration_Ready_Register)

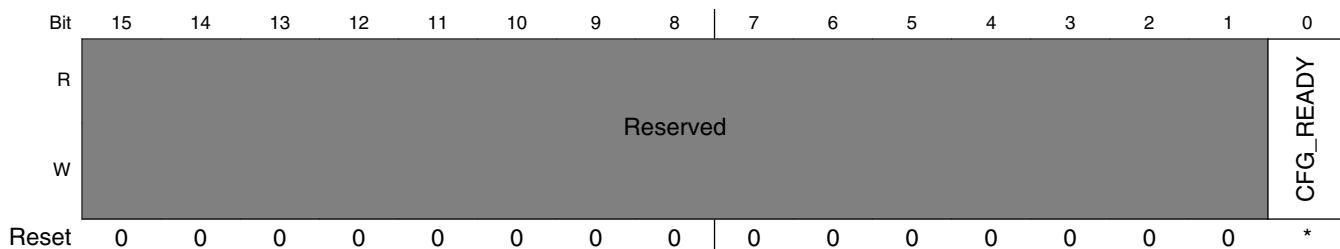
The PCI Express configuration ready register is used to indicate configuration complete status to the transaction layer. The transaction layer handles configuration requests from external hosts only after the CFG_READY bit is set. All the configuration requests received from external hosts before the CFG_READY bit is set are completed with configuration request retry status (CRS). The CFG_READY bit in this register should be set after all relevant configuration registers have been programmed. This makes sure the external host reads the correct capability advertisements during enumeration.

Address: 4B0h

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

Reserved

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



* Notes:

- CFG_READY field: Defined during POR. See description for details.

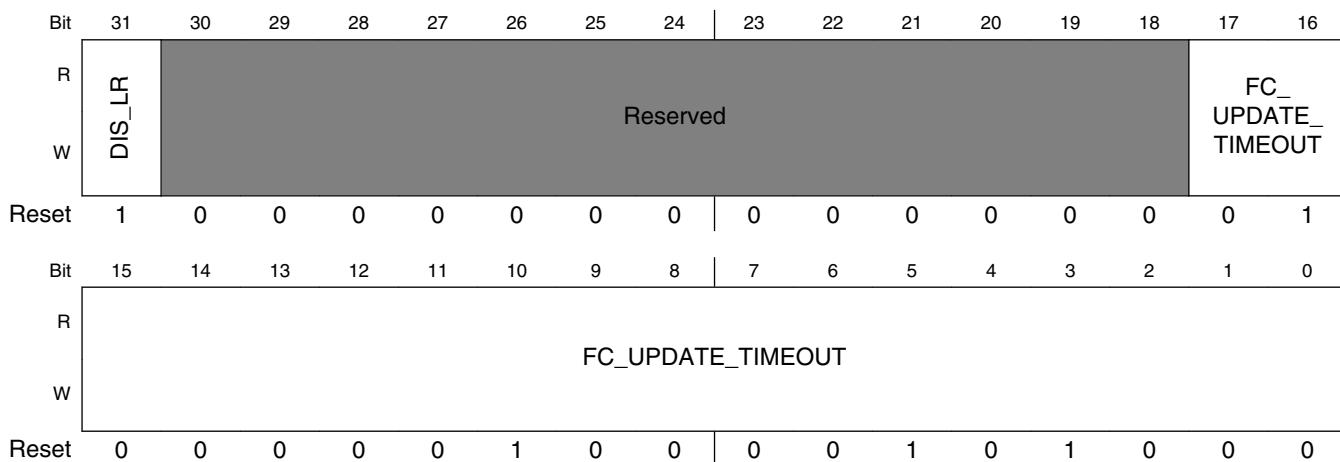
Configuration_Ready_Register field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CFG_READY	Configuration ready Note that the reset state of this bit is determined during POR. 1 The transaction layer accepts inbound configuration requests. 0 The transaction layer responds to all inbound configuration requests with retry (CRS)

18.11.24 Flow Control Update Timeout Register (Flow_Control_Update_Timeout_Register)

The PCI Express flow control update timeout register is used to program the timeout value for update-FC DLLP reception in terms of PCI Express controller core clock cycles.

Address: 4B8h



Flow_Control_Update_Timeout_Register field descriptions

Field	Description
31 DIS_LR	<p>Disable link retraining on FC update timeout error.</p> <p>The PCI Express protocol allows for optional link retraining by a device if FC updates are not received from the remote device within the timeout period programmed in bits [17-0] of this register. If the FC updates were lost due to link error issues, this retraining will help to restore normal operation. By default, this retraining is disabled. Since this is an optional feature, the user is allowed to enable or disable such automatic retraining, if desired.</p> <p>1 Disable link retraining on FC update timeout error. 0 Enable link retraining on FC update timeout error.</p>
30–18 -	This field is reserved. Reserved
FC_UPDATE_TIMEOUT	<p>FC update timeout value specifies the interval (in PCI Express controller core clock cycles) to wait for the reception of consecutive FC-update DLLPs before indicating a flow-control protocol error. The value is calculated as:</p> <p>Time (in μsec) \times PCI Express controller core clock frequency (in MHz)</p> <p>The maximum time value is 200 μsec; the default value (0x10428) is 200 μsec for the default clock frequency of 333 MHz .</p>

18.11.25 Secondary Status Interrupt Mask Register (Secondary_Status_Interrupt_Mask_Register)

This register is supported only for RC mode.

The PCI Express secondary status interrupt mask register can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See [PCI Express Secondary Status Register \(Secondary_Status_Register\)](#), for more information. By default, interrupt generation due to secondary status errors is disabled.

Address: 5A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R											M_DPE	M_SSE	M_RMA	M_RTA	M_STA	M_MDPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Secondary_Status_Interrupt_Mask_Register field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5 M_DPE	Mask detected parity error. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled
4 M_SSE	Mask signaled system error. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled
3 M_RMA	Mask received master abort. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled
2 M_RTA	Mask received target abort. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled
1 M_STA	Mask signaled target abort. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled
0 M_MDPE	Mask master data parity error. Default value is 1 1 Mask enabled (interrupt cannot be generated) 0 Mask disabled

18.12 Functional description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds.

Usually the requests and responses occur through a network of links, but to the requestor and to the completer, the intermediate components are transparent.



Figure 18-349. Requestor/completer relationship

Each PCI Express device is divided into two halves-transmit (TX) and receive (RX), and each of these halves is further divided into three layers-transaction, data link, and physical-as shown in [Figure 18-350](#).

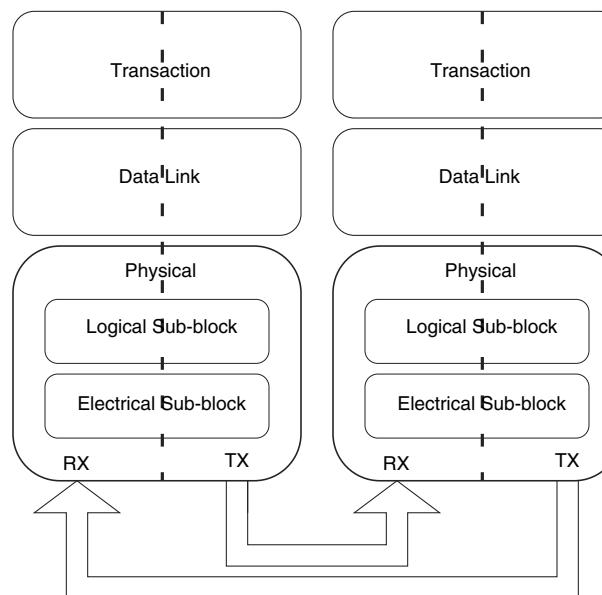


Figure 18-350. PCI Express high-level layering

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs), and each subsequent layer adds the necessary encodings and framing-as shown in [Figure 18-351](#). As packets are received, they are decoded and processed by the same layers but in reverse order, so they may be processed by the layer or by the device application software.

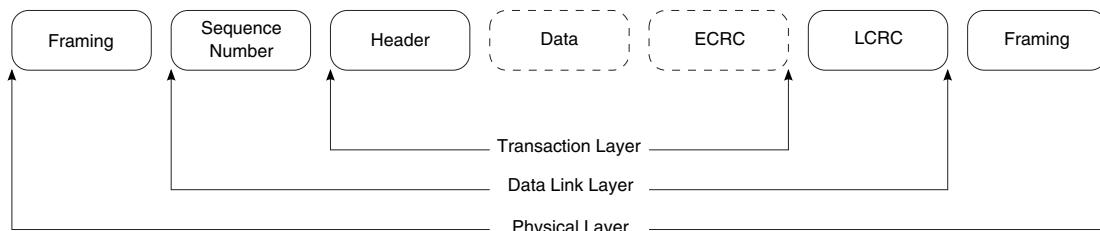


Figure 18-351. PCI Express packet flow

18.12.1 Architecture

This section describes implementation details of the PCI Express controller.

18.12.1.1 PCI Express transactions

[Table 18-347](#) contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 18-347. PCI Express transactions

PCI Express transaction	Supported as an Initiator	Supported as a target	Definition
Mrd	Yes	Yes	Memory read request
MRdLk	No	No	Memory read lock. As a target, CplLk with UR status is returned.
MWr	Yes	Yes	Memory write request to memory-mapped PCI-Express space
IOrd	Yes (RC only)	No	I/O Read request. As a target, Cpl with UR status is returned.
IOWr	Yes (RC only)	No	I/O write request. As a target, Cpl with UR status is returned.
CfgRd0	Yes (RC only)	Yes	Configuration read type 0
CfgWr0	Yes (RC only)	Yes	Configuration write type 0
CfgRd1	Yes (RC only)	No	Configuration read type 1. As a target, Cpl with UR status is returned.
CfgWr1	Yes (RC only)	No	Configuration write type 1. As a target, Cpl with UR status is returned.
Msg	Yes	Yes	Message request
MsgD	Yes (RC only)	Yes (EP only)	Message request with data payload. Note that Set_Slot_Power_Limit is the only message with data that is supported and then only when the controller is an initiator and in RC mode or a target and in EP mode.
Cpl	Yes	Yes	Completion without data
CpID	Yes	Yes	Completion with data
CplLk	No	Yes	Completion for locked memory read without data. The only time that CplLk is returned with UR status is when the controller receives a MRdLk command.
CpIDLk	No	No	Completion for locked memory read with data

18.12.1.2 Byte ordering

Whenever data cross a bridge between two busses, the byte ordering of data on the source and destination buses must be considered.

The internal platform bus of this device is inherently big endian and the PCI Express bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge-address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its default byte ordering policy, but it also may be configured to use a data invariance policy. The byte ordering policy is controlled by the ATMUs using the data invariance enable parameter in the inbound and outbound window attributes registers (PEXIWAR n [DIEN] and PEXOWAR n [DIEN]). Note that by using the ATMUs to control the byte ordering policy, it is possible to allow data invariance for certain devices or regions, while mapping other devices or regions using address invariance. However, in such cases, it is important that software keep track of which devices and regions use which byte ordering policy and access them accordingly.

18.12.1.2.1 Address invariance

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register.

This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

The following figure shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across an address invariant bridge to a little endian destination.

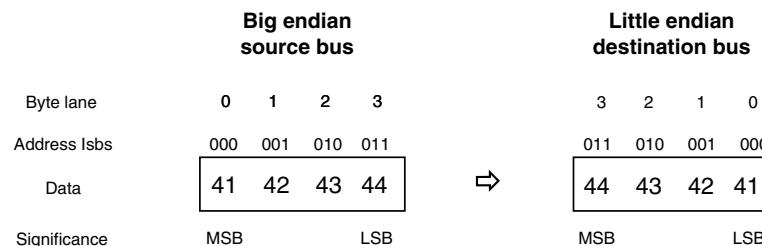


Figure 18-352. Address invariant byte ordering—4 bytes outbound

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

The following figure shows data flowing the other way, from a little endian source to a big endian destination.

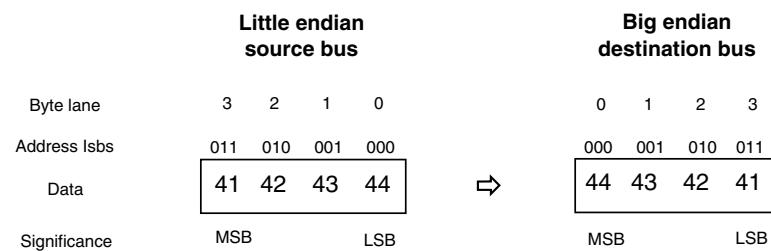


Figure 18-353. Address invariant byte ordering—4 bytes inbound

The following figure shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

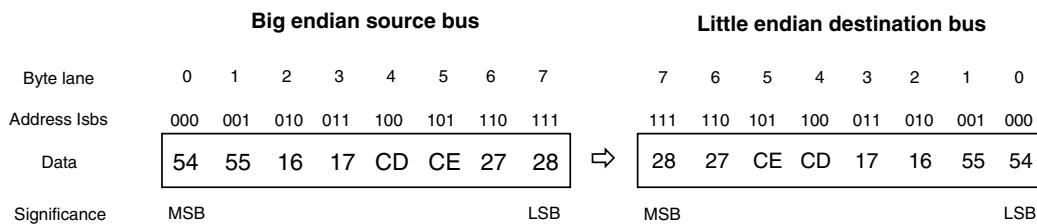


Figure 18-354. Address invariant byte ordering—8 bytes outbound

The following figure shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

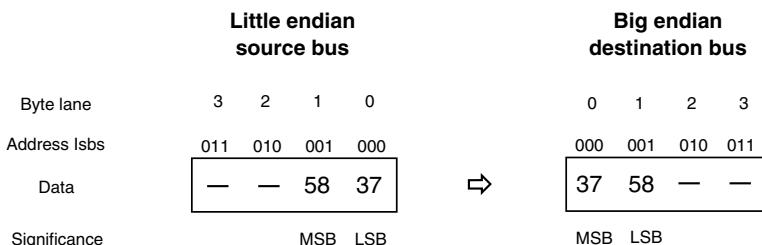


Figure 18-355. Address invariant byte ordering—2 bytes inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

18.12.1.2.2 Data invariance

The alternate policy of data invariance preserves the relative significance of data by translating the byte addressing within a given boundary.

The problem with data invariance is it can only preserve the significance of one size of data-in this device's case, 4-bytes. For scalars of different sizes or for data structures employing multiple sized scalars, it has the very undesirable effect of scrambling their addresses and formats. This puts a much greater burden on software to reconstruct non-homogeneous data structures that have passed through a data invariant bridge. It also requires knowledge of all potential paths that the data may have traversed and all the data invariance translations in each of those paths.

Freescale provides the option of using data invariance on some of its devices to allow them to be used in systems employing external devices that require this policy. Note that the data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.

The following figure shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across a bridge to a little endian destination using a 4-byte data invariance policy.

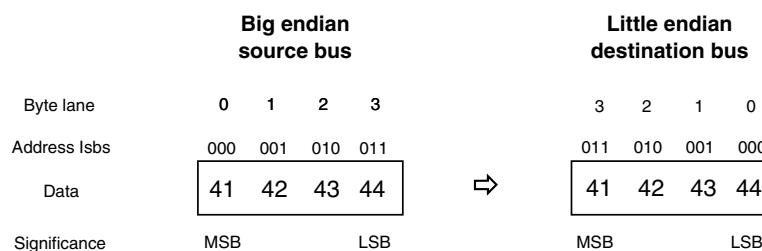


Figure 18-356. Data invariant byte ordering—4 bytes outbound

Note that the byte significance of the scalar has been preserved, even though the addresses of the individual bytes within the scalar have changed. This has effectively converted an aligned 4-byte big endian scalar to a 4-byte little endian scalar.

The following figure shows data flowing the other way, from a little endian source to a big endian destination.

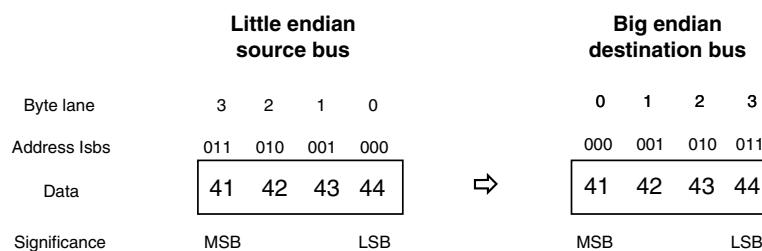


Figure 18-357. Data invariant byte ordering—4 bytes inbound

As before, the significance of the individual bytes has been preserved. It appears that the aligned 4-byte little endian scalar has been converted to a 4-byte big endian scalar. However, note what happens when a different size scalar is transferred across the bridge. The following figure shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using a 4-byte data invariance policy.

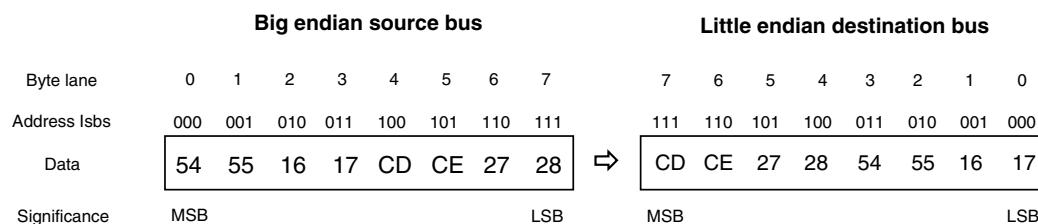


Figure 18-358. Data invariant byte ordering—8 bytes outbound

Here, the significance of bytes within the scalar has not been preserved. The data invariance has broken the scalar into halves at the 4-byte boundary. If a little endian device needs to access the 8-byte scalar, it will first need to reconstruct the data by rearranging the two halves.

The following figure shows an inbound transfer of a 2-byte scalar, 0x5837, using data invariance.

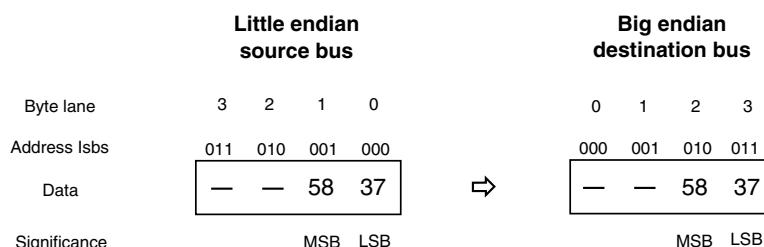


Figure 18-359. Data invariant byte ordering—2 bytes inbound

Again, the significance of the data has been preserved, but where the little endian device accesses the 2-byte scalar at an address offset 0, a big endian device will need to access it at an address offset by 2.

18.12.1.2.3 Byte order for configuration transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI Express specification defines PCI Express configuration registers as little endian.

All accesses to the PCI Express configuration port, PEX_CONFIG_DATA, including those targeting the internal PCI Express configuration registers, use the address invariance policy as shown in [Figure 18-360](#). Therefore, software must access PEX_CONFIG_DATA with little-endian formatted data-either using the **lwbrx/stwbrx** instructions or by manipulating the data before writing to and after reading from PEX_CONFIG_DATA.

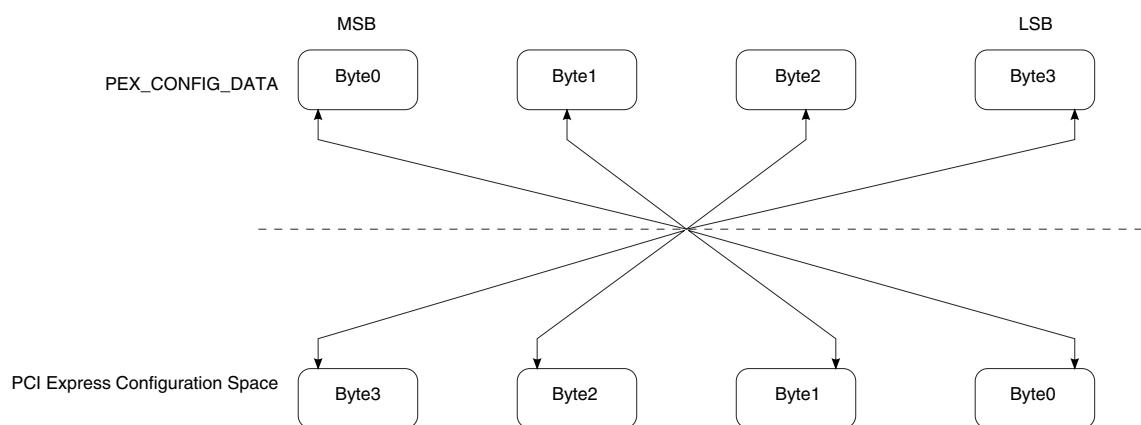


Figure 18-360. PEX_CONFIG_DATA byte ordering

18.12.1.3 Lane reversal

The x8 PCI Express controller supports lane reversal. [Table 18-348](#) describes the supported configurations.

Table 18-348. Lane assignment with and without lane reversal (x8 controller)

Link configuration	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7
x8 link without lane reversal	0	1	2	3	4	5	6	7
x4 link without lane reversal	0	1	2	3	-	-	-	-
x2 link without lane reversal	0	1	-	-	-	-	-	-
x1 link without lane reversal	0	-	-	-	-	-	-	-
x8 link with lane reversal	7	6	5	4	3	2	1	0
x4 link with lane reversal	-	-	-	-	3	2	1	0
x2 link with lane reversal	-	-	-	-	-	-	1	0

Table continues on the next page...

Table 18-348. Lane assignment with and without lane reversal (x8 controller) (continued)

Link configuration	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7
x1 link with lane reversal	-	-	-	-	-	-	-	0
NOTE: The numbers shown in this table (0-7) are the lane numbers assigned to each lane as a result of link initialization and configuration. - indicates that the lane is not part of the configured link.								

Note that lane reversal is only effective for devices that use the full link width supported by the controller (x8 lanes). That is, if a x4 device is connected to lanes 0-3 and the link training fails without lane reversal, the lane reversal causes the link to attempt connection on lanes 7-4 which would be impossible.

The x4 PCI Express controller supports lane reversal. [Table 18-349](#) describes the supported configurations.

Table 18-349. Lane assignment with and without lane reversal (x4 controller)

Link configuration	Lane 0	Lane 1	Lane 2	Lane 3
x4 link without lane reversal	0	1	2	3
x2 link without lane reversal	0	1	-	-
x1 link without lane reversal	0	-	-	-
x4 link with lane reversal	3	2	1	0
x2 link with lane reversal	-	-	1	0
x1 link with lane reversal	-	-	-	0
NOTE: The numbers shown in this table (0-3) are the lane numbers assigned to each lane as a result of link initialization and configuration. - indicates that the lane is not part of the configured link.				

Note that lane reversal is only effective for devices that use the full link width supported by the controller (x4 lanes). That is, if a x1 device is connected to lane 0 and the link training fails without lane reversal, the lane reversal causes the link to attempt connection on lane 3 which would be impossible.

18.12.1.4 Transaction ordering rules

In general, transactions are serviced in the order that they are received.

However, transactions can be reordered as they are sent due to a stalled condition such as a full internal buffer. [Table 18-350](#) describes the transaction ordering rules for this device.

Table 18-350. PCI Express controller TLP transaction ordering rules

Row pass column?		Posted request	Non-posted request		Completion	
			Memory write or message request (col 2)	Read request (Col 3)	I/O or Configuration write request (col 4)	Read completion (Col 5)
Posted request	Memory write or message request (row A)	a) No ¹ b) No ¹	Yes	Yes	a) Yes ² b) N/A ³	a) Yes ² b) N/A ³
Non-posted request	Read request (row B)	No	No	No	a) No ⁴ b) Yes ⁵	a) No ⁴ b) Yes ⁵
	I/O or configuration write request (Row C)	No	No	No	a) No ⁴ b) Yes ⁵	a) No ⁴ b) Yes ⁵
Completion	Read completion (row D)	a) No ⁶ b) Yes ⁷	Yes	Yes	a) No ⁸ b) No ⁸	No
	I/O or configuration write completion (row E)	a) No ⁶ b) Yes ⁷	Yes	Yes	No	No

1. Regardless of the setting of the relaxed ordering (RO) bit, a posted request cannot bypass another posted request.
2. Regardless of the setting of the relaxed ordering bit, a posted request can always bypass a completion.
3. N/A indicates that the original rules at these entries defined by the *PCI Express Base Specification* do not apply to RC or EP.
4. A non-posted request cannot bypass a completion if the relaxed ordering bit is cleared (that is, RO = 0).
5. A non-posted request can bypass a completion if the relaxed ordering bit is set (that is, RO = 1).
6. A read completion, I/O write completion, or configuration write completion cannot bypass a posted request if the relaxed ordering bit is cleared (that is, RO = 0).
7. A read completion, I/O write completion, or configuration write completion can bypass a posted request if the relaxed ordering bit is set (that is, RO = 1).
8. Regardless of the setting of the relaxed ordering bit, a read completion cannot bypass another read completion.

In general, the following points summarize the ordering rules for sending the next outstanding request:

- A posted request can bypass all other transactions except another posted request.
- A non-posted request cannot bypass posted or other non-posted requests, but it can bypass a completion if the relaxed ordering (RO) bit is set.(See [Table 18-350](#)).
- A completion can bypass posted requests if the relaxed ordering (RO) bit is set and can bypass non-posted transactions. However, a completion cannot bypass other completions.

18.12.1.5 PCI Express outbound ATMUs

Outbound transactions should hit into one of the outbound ranges defined by the outbound ATMUs. The outbound address translation windows must be aligned based on the granularity selected by the size fields. Outbound window misses use the default outbound register set (outbound ATMU window 0).

Overlapping outbound windows (1-4) are not supported and will cause undefined behavior. Note that for RC mode, all outbound transactions post ATMU must hit either into the memory base/limit range or the prefetchable memory base/limit range defined in the PCI Express type 1 header. For EP mode, there is no such requirement.

Note that in RC mode, when PEX_CONFIG[OB_CK] is set, the translated address of an outbound transaction must fall into either the memory base/limit range, the prefetchable memory base/limit range, or I/O base/limit range. The transaction will be terminated if the translated address does not fall within these ranges.

Figure 18-361 shows the outbound transaction flow.

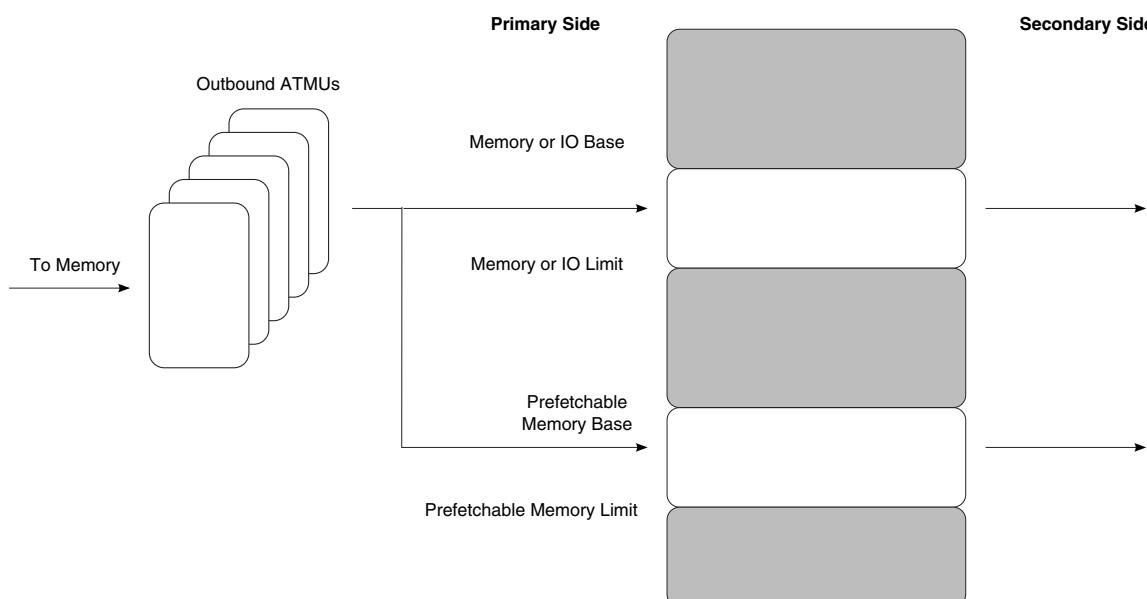


Figure 18-361. RC outbound transaction flow

18.12.1.6 PCI Express inbound ATMUs

There are differences between RC and EP implementations of the inbound ATMU registers as described in the following sections.

18.12.1.6.1 EP inbound ATMU implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible through the PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism.

Note that host software must program these BARs using configuration type 0 cycles. There are 4 inbound BARs.

- Inbound window BAR0, also known as PEXCSRBAR, at configuration address 0x10 (32-bit). This is a fixed 16-Mbyte window used for inbound memory transactions that access memory-mapped registers.
- Inbound window BAR1 at configuration address 0x14 (32-bit)
- Inbound window BAR2 at configuration address 0x18-0x1c (64-bit)
- Inbound window BAR3 at configuration address 0x20-0x24 (64-bit)

The PCI Express controller does not implement a shadow of the inbound BARs in the memory-mapped register set. However, when there is a hit to the BAR(s), the PCI Express controller uses the corresponding translation and attribute registers in the memory-mapped register set (PEXITAR n and PEXIWAR n) for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR is used.

18.12.1.6.2 RC inbound ATMU implementation

In RC mode, inbound window BAR0/PEXCSRBAR is the only inbound BAR that resides in the PCI Express type 1 configuration header (at offset 0x10); the remaining inbound window BARs (PEXIWBAR[1-3]) reside outside of the type 1 header in the memory-mapped register space.

In addition, in RC mode, there is a 64-bit MSI BAR that decodes MSI writes in 64-bit PCI Express memory space.

If the transaction hits any window, the translation is performed and then the transaction is sent to memory. If there is no hit to any one of the BARs, then an UR completion is returned for non-posted transactions. All posted transactions with no BAR hit are ignored.

[Figure 18-362](#) shows the inbound transaction flow in RC mode.

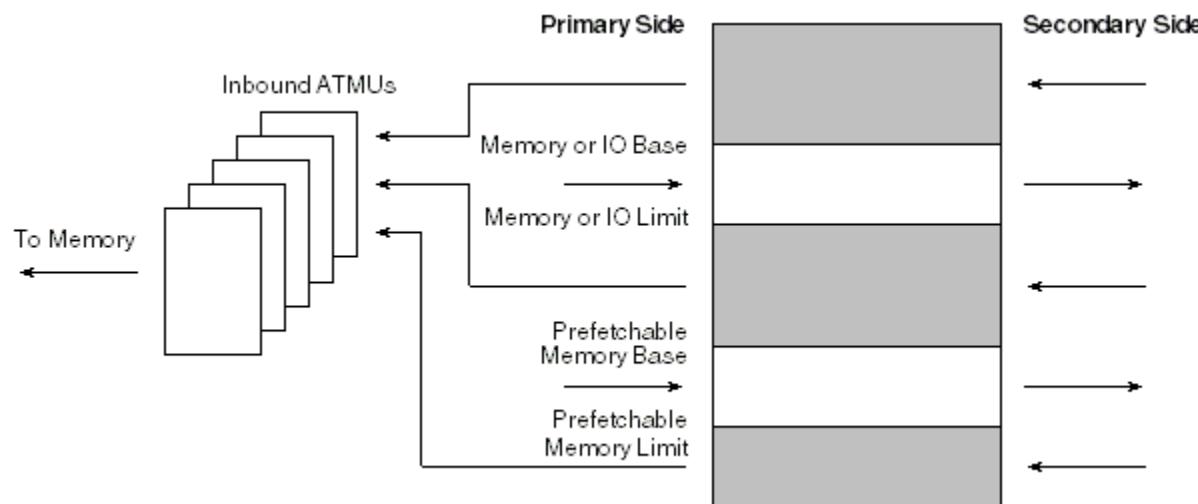


Figure 18-362. RC inbound transaction flow

18.12.1.7 Memory space addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space.

The FMT[0] field in the PCI Express TLP header for a 32-bit address packet is 0; a 64-bit address packet has a FMT[0] = 1. The PCI Express TLP header for a memory read transaction has TYPE[4:0] = 00000 and FMT[1] = 0. A memory write transaction has TYPE[4:0] = 00000 and FMT[1] = 1. As an initiator, the controller is capable of sending 32- or 64-bit memory packets. Any transaction from the internal platform that (after passing through the translation mechanism) has a translated address greater than 4G is sent as a 64-bit memory packet. Otherwise, a 32-bit memory packet is sent. As a target device, the controller is capable of decoding 32- or 64-bit memory packets. This is done through two 32-bit inbound windows and two 64-bit inbound windows. All inbound addresses are translated to 36-bit internal platform addresses.

18.12.1.8 I/O space addressing

The controller does not support I/O transactions as a target. As an initiator, the controller can send I/O transactions in RC mode only.

This can be done by programming one of the outbound translation window's attribute to send I/O transactions. All I/O transactions only access 32-bit address I/O space. The PCI Express TLP header for an I/O read transaction has TYPE[4:0] = 00010 and FMT[1] = 0. The PCI Express TLP header for an I/O write transaction has TYPE[4:0] = 00010 and FMT[1] = 1.

18.12.1.9 Configuration space addressing

As an initiator, the controller supports both type 0 and type 1 configuration cycles when configured in RC mode. There are two methods of generating a configuration transaction; refer to [PCI Express configuration space access](#), for more information.

A configuration transaction can hit into the controller's internal configuration space, it can be sent out on the PCI Express link, or it can be internally terminated.

All configuration transactions sent on PCI Express require a response regardless whether they are read or a write configuration transactions. [Table 18-351](#) shows PCI Express TLP header configuration.

Table 18-351. PCI Express TLP header configuration

TLP header type	Configuration read transaction	Configuration write transaction
Type 0	TYPE[4:0] = 00100 and FMT[1] = 0	TYPE[4:0] = 00100 and FMT[1] = 1
Type 1	TYPE[4:0] = 00101 and FMT[1] = 0	TYPE[4:0] = 00101 and FMT[1] = 1

The controller does not generate configuration transactions in EP mode. Only inbound configuration transactions are supported in EP mode.

18.12.1.10 PCI Express configuration space access

PCI Express configuration space access depends on whether the device is in RC or EP mode, as described in the following sections.

18.12.1.10.1 RC configuration register access

In RC mode, there are two methods for accessing the external PCI Express configuration space:

- PCI Express configuration access registers (PEX_CONFIG_ADDR/ PEX_CONFIG_DATA)
- PCI Express outbound ATMU window

To access internal configuration space, software must rely on the PCI Express configuration access register (PEX_CONFIG_ADDR/ PEX_CONFIG_DATA) mechanism. To access external configuration space, software can either use configuration access registers or the outbound ATMU mechanism. For the configuration access register method, a value must be written to the PEX_CONFIG_ADDR register that specifies the

targeted PCI Express bus, the targeted device on that bus, the targeted function within the device, and the configuration register in that device that should be accessed. The PCI Express controller's bus number is obtained from the PCI Express configuration header (type 1). Then either a write or a read to the PEX_CONFIG_DATA register triggers the actual write or read cycle to the configuration space.

NOTE

Accesses to the little-endian PCI Express configuration space must be properly formatted. See [Byte order for configuration transactions](#), for more information.

External configuration transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_LTSSM_STAT) to check the status of link training before issuing external configuration requests.

Refer to [Configuration accesses and inbound writes to CCSR space](#), for special considerations regarding outbound configuration accesses and inbound writes to CCSR space.

18.12.1.10.1.1 PCI Express configuration access register mechanism

There are two types of configuration transactions (Type 0 and Type 1) needed to support hierarchical bridges.

- If the targeted bus number, and targeted device number equal to the PCI Express controller's bus number and device number, and the targeted function number is zero, then an internal PCI Express configuration cycle access is performed.
- If the targeted bus number does not equal the PCI Express controller's bus number, but does equal the secondary bus number (from the type 1 header) and the targeted device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the targeted bus number does not equal the PCI Express controller's bus number, and does not equal the secondary bus number (from the type 1 header), and the targeted bus number is less than or equal to the subordinate bus number (from the type 1 header), then a Type 1 configuration transaction is sent to the PCI Express link.
- If none of the above conditions occur, then the PCI Express controller returns all 1s for reads and ignores writes. In this case, PEX_ERR_DR[ICCA] is set.

18.12.1.10.1.2 Outbound ATMU configuration mechanism (RC-only)

Software can also program one of the outbound ATMU windows to perform a configuration access. This is accomplished by programming the ReadTType or WriteTType field of the desired PEXOWAR to 0x2. Software must only issue 4-byte or less access to the ATMU configuration window and the access cannot cross a 4-byte boundary. The targeted bus number, targeted device number, targeted function number, register, and targeted extended register number sent are decoded from the outbound translated PCI Express address.

- targeted bus number[7:0] = PCI Express address[27:20]
- targeted device number[4:0] = PCI Express address[19:15]
- targeted function number[2:0] = PCI Express address[14:12]
- targeted extended register number[3:0] = PCI Express address[11:8]
- targeted register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the targeted bus number equals the secondary bus number (from the type 1 header) and targeted device number is 0. A Type 1 configuration cycle is sent to the link if targeted bus number does not equal primary bus and secondary bus numbers and it is less than or equal to the subordinate bus number (from the type 1 header). For all other cases, the PCI Express controller squashes the write and read results in a response with error returned. In this case, PEX_ERR_DR[IACA] is set.

Note that the PCI Express controller does not support access to its internal configuration registers using the outbound ATMU mechanism. That is, the outbound ATMU mechanism must not be used to program the internal registers.

18.12.1.10.2 EP configuration register access

When the PCI Express controller is configured as an EP device it responds to remote host generated configuration cycles.

This is indicated by decoding the configuration command along with type 0 access in the packet. A remote host can access all of the PCI Express configuration area except the PCI Express controller internal CSR registers in the extended PCI Express configuration space at offsets 0x400-0x6FF. The PCI Express controller internal CSR registers are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers return all zeros.

While in EP mode, the PCI Express controller does not support generating configuration accesses as a master. All accesses to PEX_CONFIG_ADDR/PEX_CONFIG_DATA cause the device to access the internal configuration registers regardless of the targeted bus number or targeted device number programmed in the PEX_CONFIG_ADDR

register. There is no configuration mechanism supported in EP mode using the ATMU window. If the outbound ATMU window is configured to issue a configuration transaction, all posted transactions hitting this window are ignored and all non-posted transactions get a response with an error.

18.12.1.11 Serialization of configuration and I/O writes

Configuration and I/O writes are serialized by the controller.

The logic after issuing a configuration write or IO write does not issue any new transactions until the outstanding configuration or I/O write is finished. This means that an acknowledgement packet from the link partner in the form of a CpL TLP packet must be seen or the transaction has timed out. If the CpL packet contains a CRS status, then the logic re-issues the configuration write transaction. It keeps retrying the request until either a status other than CRS is returned or the transaction times out.

Note that it is possible for outbound configuration read request to be requeued and be placed at the end of the request queue due to CRS condition.

18.12.1.12 Messages

Software message generation is supported in both RC and EP modes.

18.12.1.12.1 Outbound ATMU message generation

Software can choose to send a message by programming PEXOWAR n [WTT] = 0x5.

A message is sent by writing a 4-byte transaction in big-endian format that hits in an outbound window configured to send messages.

Part of the 4-byte data is used to store information such as message code and routing information. The data format is slightly different depending on whether data invariance mode is enabled. [Table 18-352](#) describes the message data format in address invariance mode (PEXOWAR n [DIEN] = 0).

Table 18-352. Internal platform Message data format-Address invariance mode

Bits	Name	Reset value	Description
0-15	-	-	Reserved
16-18	Routing	x	Routing mechanism. Contains the message's routing information
19-23	-	-	Reserved
24-31	Code	x	Message code. Contains the actual message type to be sent.

[Table 18-353](#) describes the message data format in data invariance mode (PEXOWARn[DIEN] = 1).

Table 18-353. Internal Platform Message data format-data Invariance mode

Bits	Name	Reset value	Description
0-7	Code	x	Message code. Contains the actual message type to be sent.
8-11	Routing	x	Routing mechanism. Contains the message's routing information
12-31	-	-	Reserved

In addition to the outbound ATMU, the PEX PM Command register also provides the capability to send PME_Turn_Off message or PM_PME message by setting bits 31 or 29. See [PCI Express power management command register \(PEX_PEX_PMCR\)](#) for more information.

[Table 18-354](#) provides a complete list of supported outbound messages depending on whether RC or EP is configured.

Table 18-354. PCI Express ATMU outbound messages

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Assert_INTA	0010 0000	100	N/A	Yes	Sent upstream by endpoint
Assert_INTB	0010 0001	100	N/A	Yes	Sent upstream by endpoint
Assert_INTC	0010 0010	100	N/A	Yes	Sent upstream by endpoint
Assert_INTD	0010 0011	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTA	0010 0100	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTB	0010 0101	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTC	0010 0110	100	N/A	Yes	Sent upstream by endpoint
Deassert_INTD	0010 0111	100	N/A	Yes	Sent upstream by endpoint
PM_Active_State_Nak	0001 0100	100	Yes	N/A	Terminate at receiver
PM_PME	0001 1000	000	N/A	Yes	Sent upstream by PME-requesting component
PME_Turn_Off	0001 1001	011	Yes	N/A	Broadcast downstream
PM_TO_Ack	0001 1011	101	N/A	Yes	Sent upstream by endpoint
ERR_COR	0011 0000	000	N/A	Yes	Sent by component when it detects a correctable error
ERR_NONFATAL	0011 0001	000	N/A	Yes	Sent by component when it detects a Non-fatal, uncorrectable error
ERR_FATAL	0011 0011	000	N/A	Yes	Sent by component when it detects a Fatal, uncorrectable error
Unlock	0000 0000	000	No	N/A	Not supported
Set_Slot_Power_Limit	0101 0000	100	Yes	N/A	Set slot power limit in upstream port
Vendor Defined Type 0	0111 1110		No	No	Not supported
Vendor Defined Type 1	0111 1111		No	No	Not supported

18.12.1.12.2 Inbound messages

[Table 18-355](#) provides a complete list of supported inbound messages in RC mode.

Table 18-355. PCI Express RC inbound message handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	Send to PIC
Assert_INTB	0010 0001	100	Send to PIC
Assert_INTC	0010 0010	100	Send to PIC
Assert_INTD	0010 0011	100	Send to PIC
De-assert_INTA	0010 0100	100	Send to PIC
De-assert_INTB	0010 0101	100	Send to PIC
De-assert_INTC	0010 0110	100	Send to PIC
De-assert_INTD	0010 0111	100	Send to PIC
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	Generate interrupt to PIC if enabled
PME_Turn_Off	0001 1001	011	No action taken
PME_TO_Ack	0001 1011	101	Set PEX_PME_MES_DR[ENL23] bit and generate interrupt to PIC if enabled
ERR_COR	0011 0000	000	Generate interrupt to PIC if enabled
ERR_NONFATAL	0011 0001	000	Generate interrupt to PIC if enabled
ERR_FATAL	0011 0011	000	Generate interrupt to PIC if enabled
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	No action taken
Vendor_Defined Type 0	0111 1110		No action taken
Vendor_Defined Type 1	0111 1111		No action taken

[Table 18-356](#) provides a complete list of supported inbound messages in EP mode.

Table 18-356. PCI Express EP inbound message handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	No action taken
Assert_INTB	0010 0001	100	No action taken
Assert_INTC	0010 0010	100	No action taken
Assert_INTD	0010 0011	100	No action taken
Deassert_INTA	0010 0100	100	No action taken
Deassert_INTB	0010 0101	100	No action taken
Deassert_INTC	0010 0110	100	No action taken
Deassert_INTD	0010 0111	100	No action taken
PM_Active_State_Nak	0001 0100	100	No action taken

Table continues on the next page...

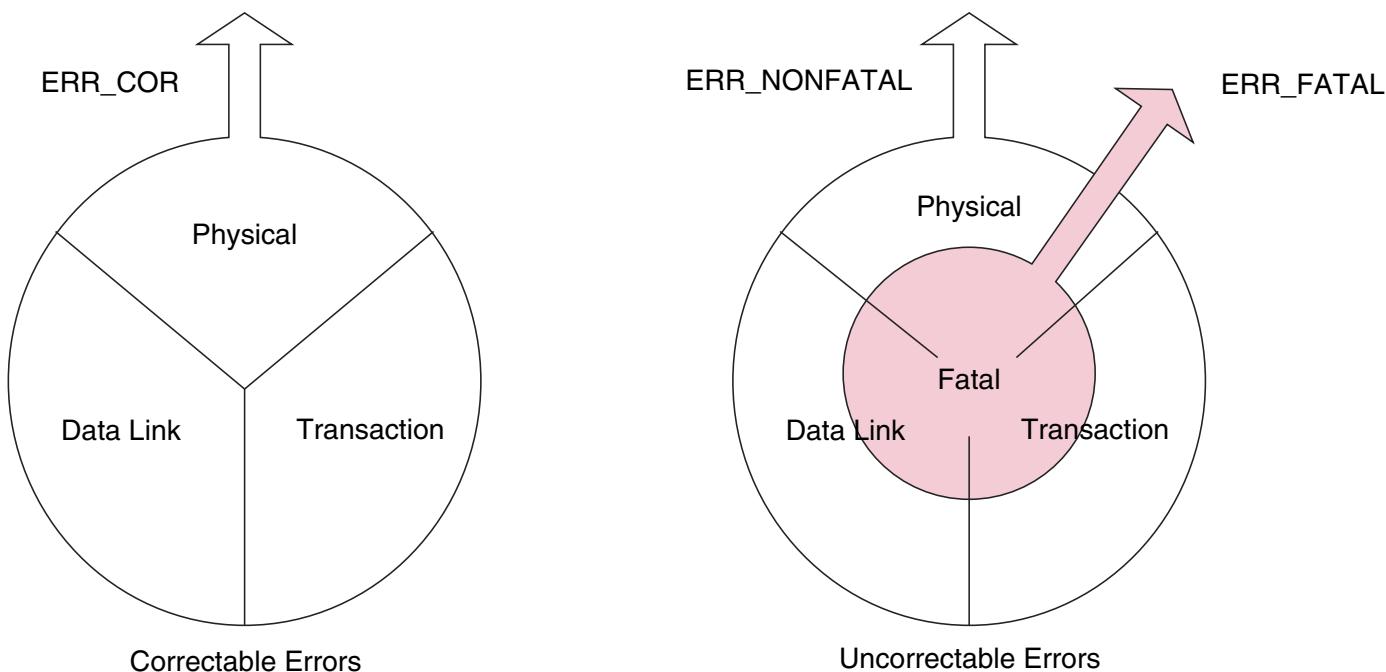
Table 18-356. PCI Express EP inbound message handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
PM_PME	0001 1000	000	No action taken
PME_Turn_Off	0001 1001	011	Set PEX_PME_MES_DR[PTO] bit. Send interrupt if enabled.
PM_TO_Ack	0001 1011	101	No action taken
ERR_COR	0011 0000	000	No action taken
ERR_NONFATAL	0011 0001	000	No action taken
ERR_FATAL	0011 0011	000	No action taken
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.
Vendor_Defined Type 0	0111 1110		No action taken
Vendor_Defined Type 1	0111 1111		No action taken

18.12.1.13 Error handling

The PCI Express specification classifies errors as correctable and uncorrectable.

Correctable errors result in degraded performance, but uncorrectable errors generally result in functional failures. As shown in [Figure 18-363](#) uncorrectable errors can further be classified as fatal or non-fatal.

**Figure 18-363. PCI Express error classification**

18.12.1.13.1 PCI Express error logging and signaling

Figure 18-364 shows the PCI Express-defined sequence of operations related to signaling and logging of errors detected by a device.

Note that the PCI Express controller on this device supports the advanced error handling capabilities shown within the dotted lines.

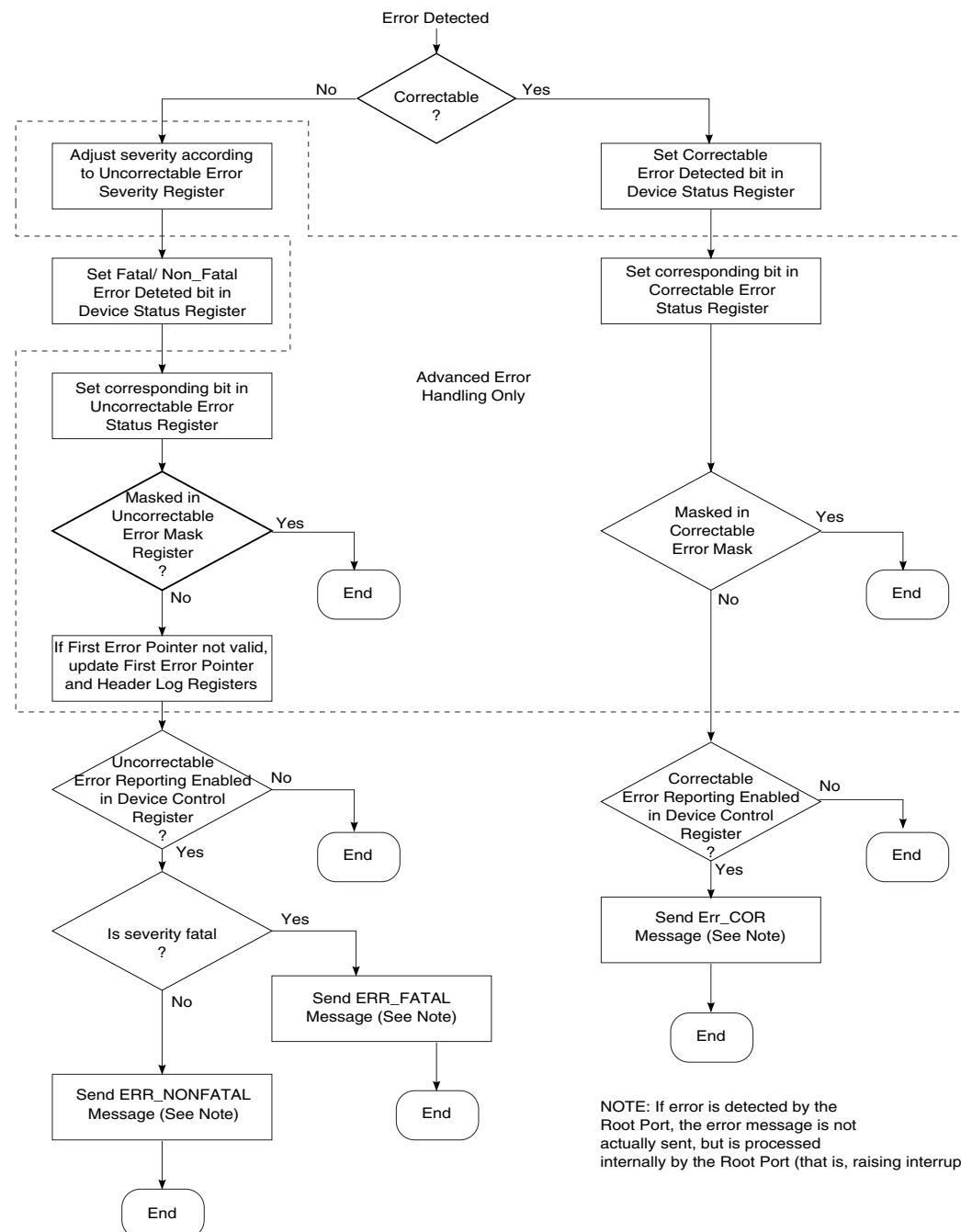


Figure 18-364. PCI Express device error signaling flowchart

18.12.1.13.2 PCI Express controller internal interrupt sources

Table 18-357 describes the sources of the PCI Express controller internal interrupt to the PIC and the preconditions for signaling the interrupt.

Table 18-357. PCI Express internal controller interrupt sources

Status register bit	Preconditions
Any bit in PEX_PME_MES_DR set	The corresponding interrupt enable bits must be set in PEX_PME_MES_IER
Any bit in PEX_ERR_DR set	The corresponding interrupt enable bits must be set in PEX_ERR_EN.
PCI Express Root Status Register[16] (PME status) is set	PCI Express Root Control Register [3] (PME interrupt enable) is set
PCI Express Root Error Status Register[6] (fatal error messages received) is set	PCI Express Root Error Command Register [2] (fatal error reporting enable) is set or PCI Express Root Control Register [2] (system error on fatal error enable) is set
PCI Express Root Error Status Register [5] (non-fatal error messages received) is set	PCI Express Root Error Command Register [1] (non-fatal error reporting enable) is set or PCI Express Root Control Register [1] (system error on non-fatal error enable) is set
PCI Express Root Error Status Register[0] (correctable error messages received) is set	PCI Express Root Error Command Register[0] (correctable error reporting enable) is set or PCI Express Root Control Register[0] (system error on correctable error enable) is set.
Any correctable error status bit in PCI Express Correctable Error Status Register is set	The corresponding error mask bit in PCI Express Correctable Error Mask Register is clear and PCI Express Root Error Command Register[0] (correctable error reporting enable) is set
Any fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[2] (fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
Any non-fatal uncorrectable error status bit in PCI Express Uncorrectable Error Status Register is set. (The corresponding error is classified as non-fatal based on the severity setting in PCI Express Uncorrectable Error Severity Register.)	The corresponding error mask bit in PCI Express Uncorrectable Error Mask Register is clear and either PCI Express Device Control Register[1] (non-fatal error reporting) is set or PCI Express Command Register[8] (SERR) is set.
PCI Express Secondary Status Register[8] (master data parity error) is set.	PCI Express Secondary Status Interrupt Mask Register[0] (mask master data parity error) is cleared and PCI Express Command Register[6] (parity error response) is set.
PCI Express Secondary Status Register[11] (signaled target abort) is set	PCI Express Secondary Status Interrupt Mask Register[1] (mask signaled target abort) is cleared.

Table continues on the next page...

Table 18-357. PCI Express internal controller interrupt sources (continued)

Status register bit	Preconditions
PCI Express Secondary Status Register[12] (received target abort) is set	PCI Express Secondary Status Interrupt Mask Register[2] (mask received target abort) is cleared.
PCI Express Secondary Status Register[13] (received master abort) is set	PCI Express Secondary Status Interrupt Mask Register[3] (mask received master abort) is cleared.
PCI Express Secondary Status Register[14] (signaled system error) is set.	PCI Express Secondary Status Interrupt Mask Register[4] (mask signaled system error) is cleared.
PCI Express Secondary Status Register[15] (detected parity error) is set	PCI Express Secondary Status Interrupt Mask Register[5] (mask detected parity error) is cleared.
PCI Express Slot Status Register[0] (attention button pressed) is set	PCI Express Slot Control Register[0] (attention button pressed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1-0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[1] (power fault detected) is set	PCI Express Slot Control Register[1] (power fault detected enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1-0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[2] (MRL sensor changed) is set	PCI Express Slot Control Register[2] (MRL sensor changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1-0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[3] (presence detect changed) is set	PCI Express Slot Control Register[3] (presence detect changed enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set and either PCI Express PM Control Register[1-0] = 00 (the function power state is D0) or PCI Express PM Control Register[8] (PME enable) is set.
PCI Express Slot Status Register[4] (command completed) is set	PCI Express Slot Control Register[4] (command completed interrupt enable) is set and PCI Express Slot Control Register[5] (hot plug interrupt enable) is set.

18.12.1.13.3 Error conditions

Table 18-358 describes specific error types and the action taken for various transaction types.

Table 18-358. Error conditions

Transaction Type	Error Type	Action
Inbound response	PEX response time out. This case happens when the internal platform sends a non-posted request that did not get a response back after a specific amount of time specified in the outbound completion timeout register (PEX_OTB_CPL_TOR)	Log error (PEX_ERR_DR[PCT]) and send interrupt to PIC, if enabled.
Inbound response	Unexpected PEX response. This can happen if, after the response times out and the internal queue entry is deallocated, the response comes back.	Log unexpected completion error (PCI Express Uncorrectable Status Register[16]) and send interrupt to PIC, if enabled.
Inbound response	Unsupported request (UR) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	Completer abort (CA) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Poisoned TLP (EP=1)	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) timeout for a configuration transaction that originates from PEX_CONFIG_ADDR/PEX_CONFIG_DATA	1.The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout

Table continues on the next page...

Table 18-358. Error conditions (continued)

Transaction Type	Error Type	Action
		(PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction and sends all 1s (0xFFFF_FFFF) data back to requester. 2. Log the error (PEX_ERR_DR[PCT]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	CA response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Poisoned TLP (EP=1) response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) response for Configuration transaction that originates from ATMU	1. The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout (PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction. 2. Log the error (PEX_ERR_DR[CRST]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	CA response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Malformed TLP response	PCI Express controller does not pass the response back to the core. Therefore, a completion timeout error eventually occurs.
Inbound request	Poisoned TLP (EP=1)	1. If it is a posted transaction, the controller drops it. 2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	ECRC error	1. If it is a posted transaction, the controller drops it.

Table continues on the next page...

Table 18-358. Error conditions (continued)

Transaction Type	Error Type	Action
		2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	PCI Express nullified request	The packet is dropped.
Outbound request	Outbound ATMU crossing	Log the error (PEX_ERR_DR[OAC]). The transaction is not sent out on the link.
Outbound request	Illegal message size	Log the error (PEX_ERR_DR[MIS]). The transaction is not sent out on the link.
Outbound request	Illegal I/O size	Log the error (PEX_ERR_DR[IOIS]). The transaction is not sent out on the link.
Outbound request	Illegal I/O address	Log the error (PEX_ERR_DR[IOIA]). The transaction is not sent out on the link.
Outbound request	Illegal configuration size	Log the error (PEX_ERR_DR[CIS]). The transaction is not sent out on the link.
Outbound request	Memory out of range	Log the error (PEX_ERR_DR[IMBA]). The transaction is not sent out on the link.
Outbound request	IO out of range	Log the error (PEX_ERR_DR[IIOBA]). The transaction is not sent out on the link.
Outbound request	Link down	Log the error (PEX_ERR_DR[LDDE]). The transaction is not sent out on the link.
Outbound response	Internal platform response with error (for example, an ECC error on a DDR read or the transaction maps to unknown address space).	Send poisoned TLP (EP=1) completion(s) for data that are known bad. If the poison data happens in the middle of the packet, the rest of the response packet(s) is also poisoned.
Link speed change request	1. Auto request with Auto speed disable bit set or 2. Link partner incapable of 5.0 Gb/s and request speed is 5.0 Gb/s	Clear link speed request and set interrupt in PME Message Detect register. Additionally, log error in link speed status register.
Link width change request	1. Auto request with Auto width disable bit set or 2. Upconfiguration not capable and request was to size-up the width or 3. Not enough detected lanes for a given request	Clear link width request and set interrupt in PME Message Detect register. Additionally, log error in link width status register.

18.12.1.13.4 Error capture registers

The PCI Express error capture registers, PEX_ERR_CAP_R0 through PEX_ERR_CAP_R3, allow vital error information to be captured when an error occurs.

Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

18.12.1.13.4.1 Error capture registers (outbound error)

PEX_ERR_CAP_Rn for the case when the error is caused by an outbound transaction from an internal source and the error is due to timeout condition or PEX_CONFIG_ADDR/PEX_CONFIG_DATA access are shown in the following figure and tables.

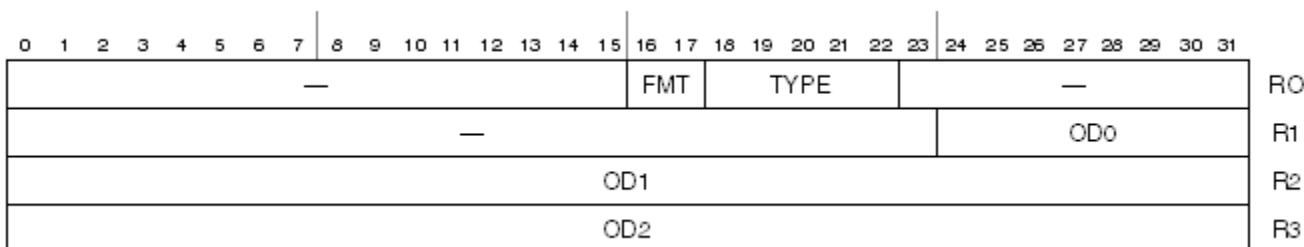


Figure 18-365. PCI Express error capture registers, outbound case

Table 18-359. PCI Express error capture register n field descriptions
internal source, outbound transaction

ERR_CA_P_n	Bits	Name	Description
R0	0-15	Reserved	-
	16-17	FMT	Header format
	18-22	TYPE	Transaction type
	23-31	Reserved	-
R1	0-23	Reserved	-
	24-31	OD0	Internal platform transaction information. Reserved for factory debug.
R2	0-31	OD1	Internal platform transaction information. Reserved for factory debug.
R3	0-31	OD2	Internal platform transaction information. Reserved for factory debug.

18.12.1.13.4.2 Error capture registers (inbound error)

PEX_ERR_CAP_Rn for the case when the error is caused by an inbound transaction from an external source are shown in the following figure and tables.

Inbound transactions that result in any af the following PEX_ERR_DR bits being set will result in PEX_ERR_CAP_STAT[GSID] indicating the controller itself was the source of the error:

- PEX_ERR_DR[PNM]
- PEX_ERR_DR[PCAC]
- PEX_ERR_DR[CDNSC]
- PEX_ERR_DR[CRSNC]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LENGTH[7:0]								TD	EP	ATTR		—	LENGTH[9:8]		—	TC		—	FMT		TYPE		R0								
Byte Count[7:0]								Comp Status		BCM	Byte Count[11:8]		Comp ID[7:0]				Comp ID[15:8]				R1										
—	Lower Address[6:0]							Tag[7:0]					RequesterID[7:0]				RequesterID[15:8]				R2										
Reserved																												R3			

Figure 18-366. PCI Express error capture registers, inbound case, completion transaction

Table 18-360. PCI Express error capture register 1 field descriptions external source, inbound completion transaction

ERR_CAP_n	Bits	Name	Description
R0	24	Reserved	-
	25-26	FMT	Header format
	27-31	TYPE	Transaction type
	16	Reserved	-
	17-19	TC	Traffic class
	20-23	Reserved	-
	8	TD	TLP digest field present
	9	EP	Poisoned packet
	10-11	ATTR	Attributes
	12-13	Reserved	-
R1	14-15, 0-7	LENGTH[9:0]	Packet length
	24-31, 16-23	Comp ID[15:0]	Completion ID, bits 15:0
	8-10	Comp Status	Completion status
	11	BCM	Byte count modified
R2	12-15, 0-7	Byte Count[11:0]	Byte count, bits 11:0
	24-31, 16-23	Requester ID[15:0]	Requester ID, bits 15:0
	8-15	Tag[7:0]	Tag, bits 7:0
	0	Reserved	-
R3	1-7	Lower Address[6:0]	Lower address, bits 6:0
	0-31	Reserved	Not used

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LENGTH[7:0]		TD	EP	ATTR	—	LENGTH[9:8]		—	TC		—	FMT		TYPE		R0															
Last DW BE[3:0]	First DW BE[3:0]	Tag[7:0]				Requester ID[7:0]				Requester ID[15:8]				R1																	
Address[7:2]		Address[15:8]				Address[23:16]				Address[31:24]				R2																	
Reserved																															

Figure 18-367. PCI Express error capture registers, inbound case, memory request transaction (3 DW)

Table 18-361. PCI Express error capture register 1 field descriptions external source, inbound memory request transaction (3 DW)

ERR_CAP_n	Bits	Name	Description
R0	24	Reserved	-
	25-26	FMT	Header format
	27-31	TYPE	Transaction type
	16	Reserved	-
	17-19	TC	Traffic class
	20-23	Reserved	-
	8	TD	TLP digest field present
	9	EP	Poisoned packet
	10-11	ATTR	Attributes
	12-13	Reserved	-
R1	14-15, 0-7	LENGTH[9:0]	Packet length
	24-31, 16-23	Requester ID[15:0]	Requester ID, bits 15:0
	8-15	Tag[7:0]	Tag, bits 7:0
	0-3	Last DW BE[3:0]	Last doubleword byte enables, bits 3:0
R2	4-7	First DW BE[3:0]	First doubleword byte enables, bits 3:0
	24-31, 16-23	Address[31:2]	Address, bits 31:2
	8-15, 0-5	Reserved	-
R3	6-7	Reserved	Not used

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						
LENGTH[7:0]		TD	EP	ATTR	—	LENGTH[9:8]		—	TC		—	FMT		TYPE		RO																					
Last DW BE[3:0]	First DW BE[3:0]	Tag[7:0]		Requester ID[7:0]		Requester ID[15:8]		R1												R2																	
Address[39:32]		Address[47:40]		Address[55:48]		Address[63:56]		R2												R3																	
Address[7:2]	—	Address[15:8]		Address[23:16]		Address[31:24]		R3																													

Figure 18-368. PCI Express error capture registers, inbound case, memory request transaction (4 DW)

Table 18-362. PCI Express error capture register 1 field descriptions external source, inbound memory request transaction (4 DW)

ERR_CAP_n	Bits	Name	Description
R0	24	Reserved	-
	25-26	FMT	Header format
	27-31	TYPE	Transaction type
	16	Reserved	-
	17-19	TC	Traffic class
	20-23	Reserved	-
	8	TD	TLP digest field present
	9	EP	Poisoned packet
	10-11	ATTR	Attributes
	12-13	Reserved	-
R1	14-15, 0-7	LENGTH[9:0]	Packet length
	24-31, 16-23	Requester ID[15:0]	Requester ID, bits 15:0
	8-15	Tag[7:0]	Tag, bits 7:0
	0-3	Last DW BE[3:0]	Last doubleword byte enables, bits 3:0
R2	4-7	First DW BE[3:0]	First doubleword byte enables, bits 3:0
	24-31, 16-23	Address[63:32]	Address, bits 63:32
	8-15, 0-7	Address[63:32]	Address, bits 63:32
R3	24-31, 16-23	Address[31:2]	Address, bits 31:2
	8-15, 0-5	Address[31:2]	Address, bits 31:2
	6-7	Reserved	-

18.12.2 Interrupts

Both INTx and message signaled interrupts (MSI) are supported; however there are differences depending on whether the PCI Express controller is configured as an RC or EP device.

18.12.2.1 EP interrupt generation

18.12.2.1.1 Hardware INTx message generation

Hardware INTx message generation is not supported in EP mode.

18.12.2.1.2 Hardware MSI generation

In EP mode, the PCI Express controller can be configured to automatically generate MSI transactions to the root complex when an interrupt event occurs.

The PCI Express controller uses *sie1* (an internal signal) to trigger the generation of the MSI. To trigger the MSI, interrupt events must be routed to *sie1* by setting the INTTGT field to 0xF1 in the associated Interrupt Level register in the PIC. Note that *sie1* is sent to all PCI Express controllers.

The remote root complex is expected to set up the MSI capability structure of all endpoints at system initialization by filling the Message Address and Message Data registers with appropriate values and setting the MSIE bit in the MSI Message Control register.

With the PCI Express controller properly configured, when it detects the leading edge of *sie1* going active, it generates a PCI Express memory write transaction to the address specified in the MSI Message Address register (and MSI Message Upper Address register) with a data payload as specified in the MSI Message Data register (with leading zeros appended).

18.12.2.1.3 Software INTx message generation

Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in [Outbound ATMU message generation](#).

18.12.2.1.4 Software MSI generation

Host software has to set up the MSI capability registers to enable MSI mode, and have the correct values for the MSI address and data register.

Then local software has to read the MSI address in the MSI capability register and configure the outbound ATMU window to map the translated address to the MSI address. Software has to determine the number of allocated messages in the MSI capability register and allocates the appropriate data values to use. A write to the ATMU window containing the MSI address with the appropriate data value generates the desired MSI transaction to the remote RC.

18.12.2 RC handling of INTx message and MSI interrupts

18.12.2.1 INTx message handling

MSIs are the preferred interrupt signaling mechanism for PCI Express.

However, in RC mode, the PCI Express controller supports the INTx virtual-wire interrupt signaling mechanism (as described in the PCI Express specification). Whenever the controller receives an inbound INTx (INTA, INTB, INTC, or INTD) asserted or negated message, it asserts or negates an equivalent internal INTx signal (*inta*, *intb*, *intc*, or *intd*) to the interrupt controller.

The internal INTx signals from the PCI Express controller may be logically combined with the interrupt request (IRQ n) input signals so that they share the same interrupt controlled by the associated EIVPR n and EIDR n registers in the interrupt controller. Refer to [PCI Express INTx/External Interrupt IRQ \$n\$ Sharing](#) for more information about sharing of PCI Express INTx interrupts and the external interrupt request (IRQ n) signals.

If a PCI Express INTx interrupt is being used, then the interrupt controller must be configured so that external interrupts are level-sensitive (EIVPR n [S] = 1).

18.12.2.2 MSI handling

An inbound MSI cycle must hit into the PEXCSRBAR window with the address offset that points to the MSIIR register in the PIC.

Note that it is the responsibility of the host software to configure each EP's MSI capability registers such that an MSI cycle generated from the EP device is routed to the MSIIR register in the PIC and for the appropriate interrupt to be generated to the core.

18.12.3 Initial credit advertisement

To prevent overflowing of the receiver's buffers and for ordering compliance purposes, the transmitter cannot send transactions unless it has enough flow control (FC) credits to send. Each device maintains an FC credit pool.

The FC information is conveyed between the two link partners by DLLPs during link training (initial credit advertisement). The transaction layer performs the FC accounting functions. One FC unit is four DWs (16-bytes) of data.

Table 18-363. Initial credit advertisement

Credit type	Initial credit advertisement
PH (Memory Write, Message Write)	See description of posted credits (PC) field in PCI Express configuration register (PEX_PEX_CONFIG) for more information.
PD (Memory Write, Message Write)	See description of posted credits (PC) field in PCI Express configuration register (PEX_PEX_CONFIG) for more information.
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	8
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

18.12.4 Power management

All device power states are supported with the exception of D3cold with Vaux. In addition, all link power states are supported with the exception of L2 states.

Only L0s ASPM mode is supported if enabled by configuring the link control register's bits 1-0 in configuration space. Note that there is no power saving in the controller when the device is put into a non-D0 state. The only power saving is the I/O drivers when the controller is put into a non-L0 link state.

Table 18-364. Power management state supported

Component D-state	Permissible interconnect state	Action
D0	L0, L0s	In full operation.

Table continues on the next page...

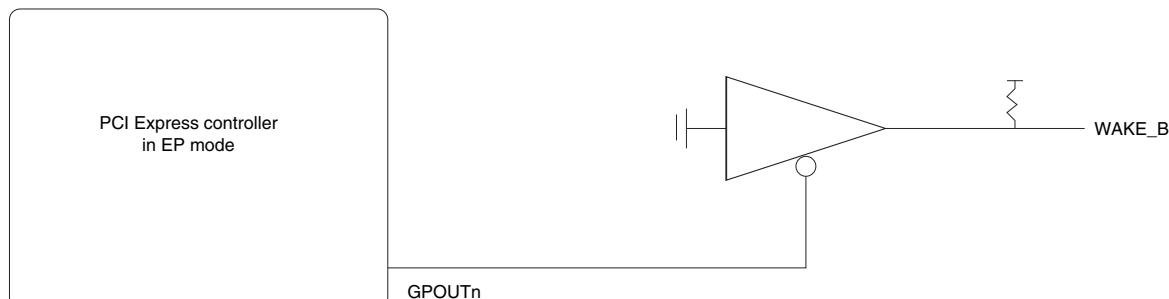
Table 18-364. Power management state supported (continued)

Component D-state	Permissible interconnect state	Action
D1	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX power management command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX power management command register.
D3hot	L0, L0s, L1, L2/ L3 ready	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX power management command register. Note that if a transition of D3hot->D0 occurs, a reset is performed to the controller's configuration space. In addition, link training restarts.
D3cold	L3	Completely off.

18.12.4.1 L2/L3 ready link state

The L2/L3 ready link state is entered after the EP device is put into a D3hot state followed by a PME_Turn_Off/PME_TO_Ack message handshake protocol.

Exiting this state requires a POR reset or a WAKE_B signal from the EP device. The PCI Express controller (in EP mode) does not support the generation of beacon; therefore, as an alternative, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a WAKE_B signal, as shown in [Figure 18-369](#).

**Figure 18-369. WAKE_B Generation example**

In RC mode, the WAKE_B signal from the EP device can be connected to one of the external interrupt inputs to service the WAKE_B request.

18.12.5 Hot reset

When a hot reset condition occurs, the controller (in both RC and EP mode) initiates a clean-up of all outstanding transactions and returns to an idle state.

All configuration register bits that are non-sticky are reset. Link training takes place subsequently. The device is permitted to generate a hot reset condition on the bus when it is configured as an RC device by setting the "Secondary Bus Reset" bit in the Bridge Control Register in the configuration space. As an EP device, it is not permitted to generate a hot reset condition; it can only detect a hot reset condition and initiate the clean-up procedure appropriately.

18.12.6 Link down

Typically, a link down condition occurs after a hot reset event; however, it is possible for the link to go down unexpectedly without a hot reset event.

When this occurs, a link down condition is detected (PEX_PME_MSG_DR[LDD]=1). Link down is treated similarly to a hot reset condition.

Subsequently, while the link is down, all new posted outbound transactions are discarded. All new non-posted ATMU transactions are errored out. Non-posted configuration transactions issued using PEX_CONFIG_ADDR/PEX_CONFIG_DATA toward the link returns 0xFFFF_FFFF (all 1s). As soon as the link is up again, the sending of transaction resumes.

Note that in EP mode, a link down condition causes the controller to reset all non-sticky bits in its PCI Express configuration registers as if it had been hot reset.

18.13 Initialization/application information

The following topics describe the initialization/application information for PCI Express.

18.13.1 EP Boot mode and inbound configuration transactions

In normal boot mode (RCW[BOOT_HO] = 0), the core is allowed to boot and configure the device.

During this time, the PCI Express interface retries all inbound PCI Express configuration transactions. When the core has configured the device to a state where it can accept inbound PCI Express configuration transactions, the boot code should set the

CFG_READY bit in the PEX_CFG_READY register after which inbound PCI Express configuration transactions are accepted. Refer to [Configuration Ready Register \(Configuration_Ready_Register\)](#), for more information about the CFG_READY bit.

In boot hold-off mode (RCW[BOOT_HO] = 1), the core is prevented from fetching its first instruction by withholding its internal bus grant. During this time, the PCI Express interface accepts all inbound PCI Express configuration transactions which allows an external host/RC to configure the device. When the external host/RC has configured the device to a state where it can allow the core to fetch code from the boot vector, it sets the BRR[CR0] bit after which the core is granted the internal bus. See [Boot release register](#), for more information.

18.13.2 Automatic link retraining during initialization

During initialization, if the link partner is down or the device has not received DLLP updates within timeout interval specified in PEX_FC_UPDATE_TOR[FC_UPDATE_TIMEOUT], it will fail link training.

It is recommended that software clear PEX_FC_UPDATE_TOR[DIS_LR] to allow for automatic retraining of the link. See [Flow Control Update Timeout Register \(Flow_Control_Update_Timeout_Register\)](#), for more information.

18.13.3 Configuration accesses and inbound writes to CCSR space

In RC mode, when the core issues an outbound configuration access to an external device using the PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA), there is a potential deadlock if several inbound memory write transactions targeting the CCSR space are received before the completion for the configuration access is returned.

Inbound writes to CCSR space include MSIs and message interrupts to the MPIC. The deadlock also can occur when the endpoint generates a single write to CCSR followed by multiple writes to another target before the original configuration access by the RC completes. When the deadlock occurs the configuration access will time out and an error will be reported (if enabled). This deadlock can be avoided by any of the following methods:

1. Configure external agents to allow completions to bypass memory write requests.
Many devices allow this when relaxed ordering (RO) is in effect; however, this

ability is optional under the *PCI Express Base Specification* ordering rules, so it is possible that some devices may not support it.

2. For the RC, restrict external configuration accesses through PEX_CONFIG_ADDR/PEX_CONFIG_DATA to the enumeration of the bus and other initialization functions; do not use PEX_CONFIG_ADDR/PEX_CONFIG_DATA to access external devices during periods when PCI Express endpoints are generating inbound writes to CCSR space.
3. Govern the generation of write transactions by the endpoint device such that after a write to the CCSR space no other writes are generated until pending configuration accesses are completed or the write is guaranteed to have completed by the completion of a subsequent read request by the endpoint devices.
4. Use the PCI Express outbound ATMU window to generate outbound configuration accesses during periods when an endpoint device may be writing to CCSR space. See [RC configuration register access](#) for more information. The deadlock does not occur when the ATMU window is used for configuration accesses.

18.13.4 Configuring ACK replay time-out when ASPM is enabled

When ASPM is enabled, the REPLAY_TIMEOUT value in the PCI Express ACK replay time-out register must be adjusted to avoid replay timeout errors. In this case, the REPLAY_TIMEOUT value should be calculated as follows:

1. Read the PCI Express N_FTS control register (PEX_N_FTS_CTL) and take the appropriate N_FTS value:
2. Add 1024 + 16 to the value in Step 1.
3. Multiply the result of Step 2 by 4 symbol times or 4/PIPE clock frequency, where PIPE clock frequency is either 250 MHz for 2.5 Gbps operation or 500 MHz for 5.0 Gbps operation.
4. Multiply the result of Step 3 by the PCI Express controller clock frequency/PIPE clock frequency. See [PCI Express Controller Core Clock Ratio Register \(Controller_Core_Clock_Ratio_Register\)](#) for information about the PCI Express controller clock frequency.

Use the result of Step 4 to update the REPLAY_TIMEOUT value in the PCI Express ACK replay time-out register. As an equation, this value can be expressed:

$$\left(N_FTS + 1024 + 16 \right) * \frac{4}{PIPE_clk} * \frac{PCI_Express_controller_clk}{PIPE_clk}$$

Equation 1. REPLAY_TIMEOUT value when ASPM is enabled

For example, given the following conditions:

- The platform clock frequency is 600 MHz
- The PCIe link speed is 5.0 Gbps
- The PCI Express controller clock is platform_freq/2
- The reset value of PEX_N_FTS_CTL is 40404040h
- The common clock configuration bit (CCC) in the Link Control Register is set

Determine the new value for REPLAY_TIMEOUT by following the steps:

1. $N_FTS_GEN2 = 40h = 64d$

Use N_FTS_CM_GEN2 because the link speed is 5.0 Gbps and the common clock configuration bit is set.

2. $64 + 1024 + 16 = 1104$

3. $1104 \times (4/500 \text{ MHz}) = 1104 \times 8 \text{ ns} = 8832 \text{ ns}$

Use 500 MHz for the PIPE_clk because the link speed is 5.0 Gbps.

4. $8832 \times (300/500) = 5299.2$

Use 300 MHz for PCI_Express_controller_clk because the platform clock frequency is 600 MHz and the PCI Express controller clock frequency is platform_freq/2. Use 500 MHz for the PIPE_clk because the link speed is 5.0 Gbps.

Rounding up to 5300, the new value for REPLAY_TIMEOUT should be 14B4h.

Chapter 19

Serial RapidIO Interface

19.1 Serial RapidIO overview

The serial RapidIO (SRIO) controller consists of a RapidIO endpoint and the RapidIO messaging unit (RMU). Both portions are compliant with the *RapidIO Interconnect Specification, Revision 1.2*.

The serial RapidIO interface provides a RapidIO port and message unit to communicate with other RapidIO devices. This figure shows the RapidIO port and message unit as they interface with OCeaN and with each other.

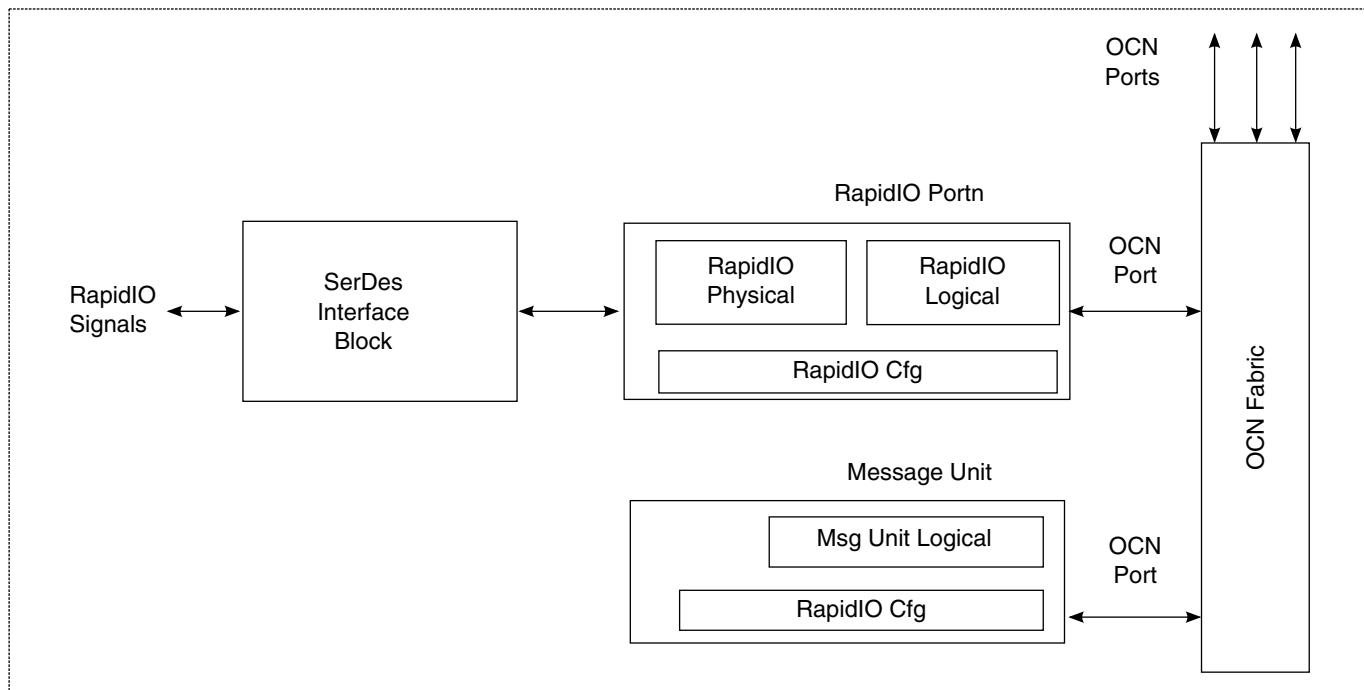


Figure 19-1. RapidIO endpoint and RMU

19.2 Serial RapidIO features summary

The RapidIO port supports the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Small or large size transport information field
- 34-bit addressing
- Up to 256-byte data payload
- Up to eight outstanding unacknowledged RapidIO transactions
- Hardware recovery only
- All transaction flows and all priorities
- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.2, Part VIII: Error Management Extensions Specification*.
- Hot swap
- ATOMIC set/clr/inc/dec for read-modify-write operations
- IO_READ_HOME and FLUSH with data for accessing cache-coherent data from a remote memory system
- Only supports receiver-controlled flow control
- Inbound transactions to the configuration registers are limited to 32-bit accesses only.
- Outbound maintenance transactions can be any valid size.

The RMU supports the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- Two outbound message controllers
- Two inbound message controllers
- One outbound doorbell controller
- One inbound doorbell controller
- One inbound port-write controller

RapidIO endpoint supports the following user-defined features:

- Nine outbound ATMU windows with each window having up to 32 subwindows (except for the default window 0)
- Five inbound ATMU windows
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RIO interface fails
- Accept-all mode of operation for failover support
- RapidIO random bit error injection
- Performance monitor interface

The RMU supports the following user-defined features:

- Performance monitor interface

RapidIO endpoint does not support or has limited support of the following features of the *RapidIO Interconnect Specification, Revision 1.2*:

- No support for 50- and 66-bit addressing
- No support for software assisted error recovery
- No support for ATOMIC test-and-swap transaction
- No support for coherent (CC-NUMA) transactions with the exception of IO_READ_HOME and FLUSH with data transactions
- No support for transmitter-controlled flow control
- No decrementing of a maintenance packet hop count
- No support for multicast event control symbols

RapidIO endpoint supports the following features of RapidIO LP-Serial:

- 1x and 4x LP-Serial link interfaces
- Transmission rates of 2.5, 3.125 Gbaud (data rates of 2.0, 2.5 Gbps) per lane
- Auto detection of 1x and 4x mode operation during port initialization
- Error detection for packets and control symbols
- Support for link initialization, synchronization, error recovery, and time-out

RapidIO endpoint does not support the following features of RapidIO LP-Serial:

- RapidIO endpoint cannot be configured as four 1x ports

19.3 Serial RapidIO modes of operation

This section provides information on the Serial RapidIO modes of operation.

19.3.1 RapidIO port

The RapidIO port's primary operating modes are the following:

- 1x or 4x LP-Serial link interfaces
- Transmission rates of 2.5, 3.125 Gbaud (data rates of 2.0, 2.5 Gbps) per lane
- Small or large size transport information field
- Accept-all mode of operation-all packets are accepted regardless of the target ID

19.3.2 Message unit

The message unit's primary operating modes are the following:

- Outbound message controller
 - Direct mode. No descriptors are involved. Software must initialize the required fields before starting a transfer.
 - Chaining mode. Software must initialize descriptors in memory and the required fields before starting a transfer.
 - Multicast mode. Single segment messages can be transferred to up to 32 destinations.

19.4 LP-serial signal descriptions

The high-speed interface signals used for the serial RapidIO interface are shared with other I/O ports and must be configured at power-on reset for use with the serial RapidIO controller.

See [SerDes Lane Assignments and Multiplexing](#) for specific configuration details.

The following sections describe the serial RapidIO signal functionality. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x , Chapter 8, Electrical Specifications*, for electrical characteristic details.

19.4.1 Serial Rapid I/O overview

The serial Rapid I/O interface is compatible with the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x LP-Serial Specifications*.

19.4.2 Serial Rapid I/O detailed signal descriptions

19.4.2.1 SD_TXn/SD_TX n_B-outputs

These are the serial data outputs. They are differential pairs, one for each lane in 4x mode of operation. Please refer to the Reset and Signals chapters of the device reference manual for external signal mapping and configuration information.

See *Part VI: Physical Layer 1x/4x LP-Serial Specification, RapidIO Interconnect Specification, Revision 1.2*, for complete details.

These outputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 1.2*. This implementation supports data rates of 2.5, 3.125 Gbaud.

19.4.2.2 SD_RXn/SD_RX n_B-inputs

These are the serial data input pads. They are differential pairs, one for each lane in 4x mode of operation.

See *Part VI: Physical Layer 1x/4x LP-Serial Specification, RapidIO Interconnect Specification, Revision 1.2*. for complete details.

These inputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 1.2*. This implementation supports data rates of 2.5, 3.125 Gbaud.

19.5 Serial RapidIO Memory Map/Register Definition

The following table is the memory map of the RapidIO configuration registers .

SRIo memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_0000	Device identity capability register (SRIO_DIDCAR)	32	R	See section	19.5.1/1306
C_0004	Device information capability register (SRIO_DICAR)	32	R	See section	19.5.2/1307
C_0008	Assembly identity capability register (SRIO_AIDCAR)	32	R/W	0000_0000h	19.5.3/1307
C_000C	Assembly information capability register (SRIO_AICAR)	32	R/W	0000_0100h	19.5.4/1308
C_0010	Processing element features capability register (SRIO_PEFCAR)	32	R	See section	19.5.5/1309
C_0018	Source operations capability register (SRIO_SOCAR)	32	R	0600_FCF0h	19.5.6/1311
C_001C	Destination operations capability register (SRIO_DOCAR)	32	R	0000_FCF4h	19.5.7/1314
C_0040	Mailbox command and status register (SRIO_MCSR)	32	R	2020_0000h	19.5.8/1317
C_0044	Port -Write and doorbell command and status register (SRIO_PWDCSR)	32	R	2000_0020h	19.5.9/1319
C_004C	Processing element logical layer control command and status register (SRIO_PELLCCSR)	32	R	0000_0001h	19.5.10/ 1321
C_005C	Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR)	32	R/W	0000_0000h	19.5.11/ 1322
C_0060	Base device ID command and status register (SRIO_BDIDCSR)	32	R/W	See section	19.5.12/ 1323
C_0068	Host base device ID lock command and status register (SRIO_HBDIDLCSR)	32	R/W	0000_FFFFh	19.5.13/ 1324

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_006C	Component tag command and status register (SRIO_CTCCSR)	32	R/W	0000_0000h	19.5.14/ 1324
C_0100	Port maintenance block header 0 (SRIO_PMBH0)	32	R	0600_0001h	19.5.15/ 1325
C_0120	Port link time-out control command and status register (SRIO_PLTOCCSR)	32	R/W	FFFF_FF00h	19.5.16/ 1325
C_0124	Port response time-out control command and status register (SRIO_PRTOCCSR)	32	R/W	FFFF_FF00h	19.5.17/ 1326
C_013C	Port General control command and status register (SRIO_PGCCSR)	32	R/W	See section	19.5.18/ 1327
C_0140	Port 1 Link maintenance request command and status register (SRIO_P1LMREQCSR)	32	R/W	0000_0000h	19.5.19/ 1328
C_0144	Port 1 Link maintenance response command and status register (SRIO_P1LMRESPCSR)	32	R	0000_0000h	19.5.20/ 1328
C_0148	Port 1 Local ackID status command and status register (SRIO_P1LASCSR)	32	R/W	0000_0000h	19.5.21/ 1329
C_0158	Port 1 Error and status command and status register (SRIO_P1ESCSR)	32	R/W	0000_0001h	19.5.22/ 1331
C_015C	Port 1 Control command and status register (SRIO_P1CCSR)	32	R/W	5060_0001h	19.5.23/ 1334
C_0160	Port 2 Link maintenance request command and status register (SRIO_P2LMREQCSR)	32	R/W	0000_0000h	19.5.24/ 1336
C_0164	Port 2 Link maintenance response command and status register (SRIO_P2LMRESPCSR)	32	R	0000_0000h	19.5.25/ 1336
C_0168	Port 2 Local ackID status command and status register (SRIO_P2LASCSR)	32	R/W	0000_0000h	19.5.26/ 1337
C_0178	Port 2 Error and status command and status register (SRIO_P2ESCSR)	32	R/W	0000_0001h	19.5.27/ 1339
C_017C	Port 2 Control command and status register (SRIO_P2CCSR)	32	R/W	5060_0001h	19.5.28/ 1342
C_0600	Error reporting block header (SRIO_ERBH)	32	R	0000_0007h	19.5.29/ 1344
C_0608	Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR)	32	R/W	0000_0000h	19.5.30/ 1344
C_060C	Logical/Transport layer error enable command and status register (SRIO_LTLEECSR)	32	R/W	0000_0000h	19.5.31/ 1347
C_0614	Logical/Transport layer address capture command and status register (SRIO_LTLACCSR)	32	R/W	0000_0000h	19.5.32/ 1348
C_0618	Logical/Transport layer device ID capture command and status register (SRIO_LTLDIDCCSR)	32	R/W	0000_0000h	19.5.33/ 1349
C_061C	Logical/Transport layer control capture command and status register (SRIO_LTLCCSR)	32	R/W	0000_0000h	19.5.34/ 1350
C_0640	Port 1 Error detect command and status register (SRIO_P1EDCSR)	32	R/W	0000_0000h	19.5.35/ 1351

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_0644	Port 1 Error rate enable command and status register (SRIO_P1ERCSR)	32	R/W	0000_0000h	19.5.36/ 1352
C_0648	Port 1 Error capture attributes command and status register (SRIO_P1ECACSR)	32	R/W	0000_0000h	19.5.37/ 1354
C_064C	Port 1 Packet/control symbol error capture command and status register 0 (SRIO_P1PCSECCSR0)	32	R/W	0000_0000h	19.5.38/ 1355
C_0650	Port 1 Packet error capture command and status register 1 (SRIO_P1PECCSR1)	32	R/W	0000_0000h	19.5.39/ 1356
C_0654	Port 1 Packet error capture command and status register 2 (SRIO_P1PECCSR2)	32	R/W	0000_0000h	19.5.40/ 1356
C_0658	Port 1 Packet error capture command and status register 3 (SRIO_P1PECCSR3)	32	R/W	0000_0000h	19.5.41/ 1357
C_0668	Port 1 Error rate command and status register (SRIO_P1ERCSR)	32	R/W	8000_0000h	19.5.42/ 1357
C_066C	Port 1 Error rate threshold command and status register (SRIO_P1ERTCSR)	32	R/W	FFFF_0000h	19.5.43/ 1358
C_0680	Port 2 Error detect command and status register (SRIO_P2EDCSR)	32	R/W	0000_0000h	19.5.44/ 1359
C_0684	Port 2 Error rate enable command and status register (SRIO_P2ERCSR)	32	R/W	0000_0000h	19.5.45/ 1361
C_0688	Port 2 Error capture attributes command and status register (SRIO_P2ECACSR)	32	R/W	0000_0000h	19.5.46/ 1362
C_068C	Port 2 Packet/control symbol error capture command and status register 0 (SRIO_P2PCSECCSR0)	32	R/W	0000_0000h	19.5.47/ 1363
C_0690	Port 2 Packet error capture command and status register 1 (SRIO_P2PECCSR1)	32	R/W	0000_0000h	19.5.48/ 1364
C_0694	Port 2 Packet error capture command and status register 2 (SRIO_P2PECCSR2)	32	R/W	0000_0000h	19.5.49/ 1364
C_0698	Port 2 Packet error capture command and status register 3 (SRIO_P2PECCSR3)	32	R/W	0000_0000h	19.5.50/ 1365
C_06A8	Port 2 Error rate command and status register (SRIO_P2ERCSR)	32	R/W	8000_0000h	19.5.51/ 1365
C_06AC	Port 2 Error rate threshold command and status register (SRIO_P2ERTCSR)	32	R/W	FFFF_0000h	19.5.52/ 1366
D_0004	Logical layer configuration register (SRIO_LLCSR)	32	R/W	0000_0000h	19.5.53/ 1367
D_0010	Error / port-write interrupt status register (SRIO_EPWISR)	32	R	0000_0000h	19.5.54/ 1369
D_0020	Logical retry error threshold configuration register (SRIO_LRETCR)	32	R/W	0000_00FFh	19.5.55/ 1370
D_0080	Physical retry error threshold configuration register (SRIO_PRETCR)	32	R/W	0000_00FFh	19.5.56/ 1370
D_0100	Port 1 Alternate device ID command and status register (SRIO_P1ADIDCSR)	32	R/W	0000_0000h	19.5.57/ 1371

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0120	Port 1 accept-all configuration register (SRIO_P1AACR)	32	R/W	1000_0000h	19.5.58/ 1372
D_0124	Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR)	32	R/W	0000_0000h	19.5.59/ 1372
D_0130	Port 1 Implementation error command and status register (SRIO_P1IECSR)	32	w1c	0000_0000h	19.5.60/ 1374
D_0140	Port 1 Physical configuration register (SRIO_P1PCR)	32	R/W	0000_8010h	19.5.61/ 1375
D_0158	Port 1 Serial link command and status register (SRIO_P1SLCSR)	32	w1c	0000_0000h	19.5.62/ 1376
D_0160	Port 1 Serial link error injection configuration register (SRIO_P1SLEICR)	32	R/W	0000_0000h	19.5.63/ 1377
D_0180	Port 2 Alternate device ID command and status register (SRIO_P2ADIDCSR)	32	R/W	0000_0000h	19.5.64/ 1378
D_01A0	Port 2 accept-all configuration register (SRIO_P2AACR)	32	R/W	1000_0000h	19.5.65/ 1379
D_01A4	Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR)	32	R/W	0000_0000h	19.5.66/ 1379
D_01B0	Port 2 Implementation error command and status register (SRIO_P2IECSR)	32	w1c	0000_0000h	19.5.67/ 1380
D_01C0	Port 2 Physical configuration register (SRIO_P2PCR)	32	R/W	0000_8010h	19.5.68/ 1381
D_01D8	Port 2 Serial link command and status register (SRIO_P2SLCSR)	32	w1c	0000_0000h	19.5.69/ 1382
D_01E0	Port 2 Serial link error injection configuration register (SRIO_P2SLEICR)	32	R/W	0000_0000h	19.5.70/ 1383
D_0BF8	IP Block Revision Register 1 (SRIO_IPBRR1)	32	R	01C0_0102h	19.5.71/ 1384
D_0BFC	IP Block Revision Register 2 (SRIO_IPBRR2)	32	R	0000_0000h	19.5.72/ 1384
D_0C00	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR0)	32	R/W	FF80_0000h	19.5.73/ 1385
D_0C04	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR0)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0C10	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR0)	32	R/W	8004_4023h	19.5.75/ 1387
D_0C20	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR1)	32	R/W	0000_0000h	19.5.76/ 1390
D_0C24	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR1)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0C28	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR1)	32	R/W	0000_0000h	19.5.77/ 1391
D_0C30	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR1)	32	R/W	8004_4023h	19.5.75/ 1387

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0C34	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R1)	32	R/W	0044_0000h	19.5.78/ 1392
D_0C38	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R1)	32	R/W	0044_0000h	19.5.79/ 1394
D_0C3C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R1)	32	R/W	0044_0000h	19.5.80/ 1396
D_0C40	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR2)	32	R/W	0000_0000h	19.5.76/ 1390
D_0C44	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR2)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0C48	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR2)	32	R/W	0000_0000h	19.5.77/ 1391
D_0C50	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR2)	32	R/W	8004_4023h	19.5.75/ 1387
D_0C54	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R2)	32	R/W	0044_0000h	19.5.78/ 1392
D_0C58	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R2)	32	R/W	0044_0000h	19.5.79/ 1394
D_0C5C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R2)	32	R/W	0044_0000h	19.5.80/ 1396
D_0C60	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR3)	32	R/W	0000_0000h	19.5.76/ 1390
D_0C64	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR3)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0C68	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR3)	32	R/W	0000_0000h	19.5.77/ 1391
D_0C70	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR3)	32	R/W	8004_4023h	19.5.75/ 1387
D_0C74	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R3)	32	R/W	0044_0000h	19.5.78/ 1392
D_0C78	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R3)	32	R/W	0044_0000h	19.5.79/ 1394
D_0C7C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R3)	32	R/W	0044_0000h	19.5.80/ 1396
D_0C80	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR4)	32	R/W	0000_0000h	19.5.76/ 1390
D_0C84	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR4)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0C88	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR4)	32	R/W	0000_0000h	19.5.77/ 1391
D_0C90	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR4)	32	R/W	8004_4023h	19.5.75/ 1387
D_0C94	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R4)	32	R/W	0044_0000h	19.5.78/ 1392

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0C98	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R4)	32	R/W	0044_0000h	19.5.79/ 1394
D_0C9C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R4)	32	R/W	0044_0000h	19.5.80/ 1396
D_0CA0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR5)	32	R/W	0000_0000h	19.5.76/ 1390
D_0CA4	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR5)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0CA8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR5)	32	R/W	0000_0000h	19.5.77/ 1391
D_0CB0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWWAR5)	32	R/W	8004_4023h	19.5.75/ 1387
D_0CB4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R5)	32	R/W	0044_0000h	19.5.78/ 1392
D_0CB8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R5)	32	R/W	0044_0000h	19.5.79/ 1394
D_0CBC	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R5)	32	R/W	0044_0000h	19.5.80/ 1396
D_0CC0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR6)	32	R/W	0000_0000h	19.5.76/ 1390
D_0CC4	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR6)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0CC8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR6)	32	R/W	0000_0000h	19.5.77/ 1391
D_0CD0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWWAR6)	32	R/W	8004_4023h	19.5.75/ 1387
D_0CD4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R6)	32	R/W	0044_0000h	19.5.78/ 1392
D_0CD8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R6)	32	R/W	0044_0000h	19.5.79/ 1394
D_0CDC	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R6)	32	R/W	0044_0000h	19.5.80/ 1396
D_0CE0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR7)	32	R/W	0000_0000h	19.5.76/ 1390
D_0CE4	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR7)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0CE8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR7)	32	R/W	0000_0000h	19.5.77/ 1391
D_0CF0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWWAR7)	32	R/W	8004_4023h	19.5.75/ 1387
D_0CF4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R7)	32	R/W	0044_0000h	19.5.78/ 1392
D_0CF8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R7)	32	R/W	0044_0000h	19.5.79/ 1394

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0CF0	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R7)	32	R/W	0044_0000h	19.5.80/ 1396
D_0D00	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR8)	32	R/W	0000_0000h	19.5.76/ 1390
D_0D04	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR8)	32	R/W	0000_003Fh	19.5.74/ 1386
D_0D08	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR8)	32	R/W	0000_0000h	19.5.77/ 1391
D_0D10	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR8)	32	R/W	8004_4023h	19.5.75/ 1387
D_0D14	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R8)	32	R/W	0044_0000h	19.5.78/ 1392
D_0D18	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R8)	32	R/W	0044_0000h	19.5.79/ 1394
D_0D1C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R8)	32	R/W	0044_0000h	19.5.80/ 1396
D_0D60	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR4)	32	R/W	0000_0000h	19.5.81/ 1398
D_0D68	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR4)	32	R/W	0000_0000h	19.5.82/ 1399
D_0D70	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR4)	32	R/W	0004_4021h	19.5.83/ 1400
D_0D80	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR3)	32	R/W	0000_0000h	19.5.81/ 1398
D_0D88	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR3)	32	R/W	0000_0000h	19.5.82/ 1399
D_0D90	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR3)	32	R/W	0004_4021h	19.5.83/ 1400
D_0DA0	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR2)	32	R/W	0000_0000h	19.5.81/ 1398
D_0DA8	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR2)	32	R/W	0000_0000h	19.5.82/ 1399
D_0DB0	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR2)	32	R/W	0004_4021h	19.5.83/ 1400
D_0DC0	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR1)	32	R/W	0000_0000h	19.5.81/ 1398
D_0DC8	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR1)	32	R/W	0000_0000h	19.5.82/ 1399
D_0DD0	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR1)	32	R/W	0004_4021h	19.5.83/ 1400
D_0DE0	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR0)	32	R/W	0000_0000h	19.5.81/ 1398
D_0DF0	Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0)	32	R/W	8004_4021h	19.5.84/ 1402

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0E00	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR0)	32	R/W	FF80_0000h	19.5.85/ 1404
D_0E04	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR0)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0E10	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR0)	32	R/W	8004_4023h	19.5.87/ 1407
D_0E20	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR1)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0E24	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR1)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0E28	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR1)	32	R/W	0000_0000h	19.5.89/ 1411
D_0E30	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR1)	32	R/W	8004_4023h	19.5.87/ 1407
D_0E34	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R1)	32	R/W	0044_0000h	19.5.90/ 1412
D_0E38	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R1)	32	R/W	0044_0000h	19.5.91/ 1414
D_0E3C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R1)	32	R/W	0044_0000h	19.5.92/ 1416
D_0E40	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR2)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0E44	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR2)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0E48	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR2)	32	R/W	0000_0000h	19.5.89/ 1411
D_0E50	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR2)	32	R/W	8004_4023h	19.5.87/ 1407
D_0E54	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R2)	32	R/W	0044_0000h	19.5.90/ 1412
D_0E58	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R2)	32	R/W	0044_0000h	19.5.91/ 1414
D_0E5C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R2)	32	R/W	0044_0000h	19.5.92/ 1416
D_0E60	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR3)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0E64	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR3)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0E68	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR3)	32	R/W	0000_0000h	19.5.89/ 1411
D_0E70	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR3)	32	R/W	8004_4023h	19.5.87/ 1407
D_0E74	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R3)	32	R/W	0044_0000h	19.5.90/ 1412

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0E78	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R3)	32	R/W	0044_0000h	19.5.91/ 1414
D_0E7C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R3)	32	R/W	0044_0000h	19.5.92/ 1416
D_0E80	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR4)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0E84	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR4)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0E88	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR4)	32	R/W	0000_0000h	19.5.89/ 1411
D_0E90	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR4)	32	R/W	8004_4023h	19.5.87/ 1407
D_0E94	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R4)	32	R/W	0044_0000h	19.5.90/ 1412
D_0E98	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R4)	32	R/W	0044_0000h	19.5.91/ 1414
D_0E9C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R4)	32	R/W	0044_0000h	19.5.92/ 1416
D_0EA0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR5)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0EA4	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR5)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0EA8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR5)	32	R/W	0000_0000h	19.5.89/ 1411
D_0EB0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR5)	32	R/W	8004_4023h	19.5.87/ 1407
D_0EB4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R5)	32	R/W	0044_0000h	19.5.90/ 1412
D_0EB8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R5)	32	R/W	0044_0000h	19.5.91/ 1414
D_0EBC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R5)	32	R/W	0044_0000h	19.5.92/ 1416
D_0EC0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR6)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0EC4	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR6)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0EC8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR6)	32	R/W	0000_0000h	19.5.89/ 1411
D_0ED0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWWAR6)	32	R/W	8004_4023h	19.5.87/ 1407
D_0ED4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R6)	32	R/W	0044_0000h	19.5.90/ 1412
D_0ED8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R6)	32	R/W	0044_0000h	19.5.91/ 1414

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0EDC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R6)	32	R/W	0044_0000h	19.5.92/ 1416
D_0EE0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR7)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0EE4	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR7)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0EE8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR7)	32	R/W	0000_0000h	19.5.89/ 1411
D_0EF0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR7)	32	R/W	8004_4023h	19.5.87/ 1407
D_0EF4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R7)	32	R/W	0044_0000h	19.5.90/ 1412
D_0EF8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R7)	32	R/W	0044_0000h	19.5.91/ 1414
D_0EFC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R7)	32	R/W	0044_0000h	19.5.92/ 1416
D_0F00	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR8)	32	R/W	FF80_0000h	19.5.88/ 1410
D_0F04	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR8)	32	R/W	0000_003Fh	19.5.86/ 1406
D_0F08	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR8)	32	R/W	0000_0000h	19.5.89/ 1411
D_0F10	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR8)	32	R/W	8004_4023h	19.5.87/ 1407
D_0F14	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R8)	32	R/W	0044_0000h	19.5.90/ 1412
D_0F18	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R8)	32	R/W	0044_0000h	19.5.91/ 1414
D_0F1C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R8)	32	R/W	0044_0000h	19.5.92/ 1416
D_0F60	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR4)	32	R/W	0000_0000h	19.5.93/ 1418
D_0F68	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR4)	32	R/W	0000_0000h	19.5.94/ 1419
D_0F70	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR4)	32	R/W	0004_4021h	19.5.95/ 1420
D_0F80	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR3)	32	R/W	0000_0000h	19.5.93/ 1418
D_0F88	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR3)	32	R/W	0000_0000h	19.5.94/ 1419
D_0F90	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR3)	32	R/W	0004_4021h	19.5.95/ 1420
D_0FA0	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR2)	32	R/W	0000_0000h	19.5.93/ 1418

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0FA8	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR2)	32	R/W	0000_0000h	19.5.94/ 1419
D_0FB0	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR2)	32	R/W	0004_4021h	19.5.95/ 1420
D_0FC0	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR1)	32	R/W	0000_0000h	19.5.93/ 1418
D_0FC8	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR1)	32	R/W	0000_0000h	19.5.94/ 1419
D_0FD0	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR1)	32	R/W	0004_4021h	19.5.95/ 1420
D_0FE0	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR0)	32	R/W	0000_0000h	19.5.93/ 1418
D_0FF0	Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0)	32	R/W	8004_4021h	19.5.96/ 1422
D_3000	Outbound message n mode register (SRIO_OM0MR)	32	R/W	0000_0000h	19.5.97/ 1424
D_3004	Outbound message n status register (SRIO_OM0SR)	32	R/W	0000_0000h	19.5.98/ 1427
D_3008	Extended outbound message n descriptor queue dequeue pointer address register (SRIO_EOM0DQDPAR)	32	R/W	0000_0000h	19.5.99/ 1429
D_300C	Outbound message n descriptor queue dequeue pointer address register (SRIO_OM0DQDPAR)	32	R/W	0000_0000h	19.5.100/ 1430
D_3010	Extended outbound message n source address register (SRIO_EOM0SAR)	32	R/W	0000_0000h	19.5.101/ 1431
D_3014	Outbound message n source address register (SRIO_OM0SAR)	32	R/W	0000_0000h	19.5.102/ 1431
D_3018	Outbound message n destination port register (SRIO_OM0DPR)	32	R/W	0000_0000h	19.5.103/ 1432
D_301C	Outbound message n destination attributes Register (SRIO_OM0DATR)	32	R/W	0000_0000h	19.5.104/ 1433
D_3020	Outbound message n double-word count register (SRIO_OM0DCR)	32	R/W	0000_0000h	19.5.105/ 1434
D_3024	Extended outbound message n descriptor queue enqueue pointer address register (SRIO_EOM0DQEPR)	32	R/W	0000_0000h	19.5.106/ 1435
D_3028	Outbound message n descriptor queue enqueue pointer address register (SRIO_OM0DQEPR)	32	R/W	0000_0000h	19.5.107/ 1435
D_302C	Outbound message n retry error threshold configuration register (SRIO_OM0RETCR)	32	R/W	0000_0000h	19.5.108/ 1436
D_3030	Outbound message n multicast group register (SRIO_OM0MGR)	32	R/W	0000_0000h	19.5.109/ 1437
D_3034	Outbound message n multicast list register (SRIO_OM0MLR)	32	R/W	0000_0000h	19.5.110/ 1438
D_3060	Inbound message n mode register (SRIO_IM0MR)	32	R/W	0000_0000h	19.5.111/ 1439

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_3064	Inbound message n status register (SRIO_IM0SR)	32	R/W	0000_0002h	19.5.112/ 1442
D_3068	Extended inbound message n frame queue dequeue pointer address register (SRIO_EIM0FQDPAR)	32	R/W	0000_0000h	19.5.113/ 1444
D_306C	Inbound message n frame queue dequeue pointer address register (SRIO_IM0FQDPAR)	32	R/W	0000_0000h	19.5.114/ 1444
D_3070	Extended inbound message n frame queue enqueue pointer address register (SRIO_EIM0FQEPAR)	32	R/W	0000_0000h	19.5.115/ 1445
D_3074	Inbound message n frame queue enqueue pointer address register (SRIO_IM0FQEPAR)	32	R/W	0000_0000h	19.5.116/ 1446
D_3078	Inbound message n maximum interrupt report interval register (SRIO_IM0MIRIR)	32	R/W	FFFF_FF00h	19.5.117/ 1447
D_3100	Outbound message n mode register (SRIO_OM1MR)	32	R/W	0000_0000h	19.5.97/ 1424
D_3104	Outbound message n status register (SRIO_OM1SR)	32	R/W	0000_0000h	19.5.98/ 1427
D_3108	Extended outbound message n descriptor queue dequeue pointer address register (SRIO_EOM1DQDPAR)	32	R/W	0000_0000h	19.5.99/ 1429
D_310C	Outbound message n descriptor queue dequeue pointer address register (SRIO_OM1DQDPAR)	32	R/W	0000_0000h	19.5.100/ 1430
D_3110	Extended outbound message n source address register (SRIO_EOM1SAR)	32	R/W	0000_0000h	19.5.101/ 1431
D_3114	Outbound message n source address register (SRIO_OM1SAR)	32	R/W	0000_0000h	19.5.102/ 1431
D_3118	Outbound message n destination port register (SRIO_OM1DPR)	32	R/W	0000_0000h	19.5.103/ 1432
D_311C	Outbound message n destination attributes Register (SRIO_OM1DATR)	32	R/W	0000_0000h	19.5.104/ 1433
D_3120	Outbound message n double-word count register (SRIO_OM1DCR)	32	R/W	0000_0000h	19.5.105/ 1434
D_3124	Extended outbound message n descriptor queue enqueue pointer address register (SRIO_EOM1DQEPR)	32	R/W	0000_0000h	19.5.106/ 1435
D_3128	Outbound message n descriptor queue enqueue pointer address register (SRIO_OM1DQEPR)	32	R/W	0000_0000h	19.5.107/ 1435
D_312C	Outbound message n retry error threshold configuration register (SRIO_OM1RETCR)	32	R/W	0000_0000h	19.5.108/ 1436
D_3130	Outbound message n multicast group register (SRIO_OM1MGR)	32	R/W	0000_0000h	19.5.109/ 1437
D_3134	Outbound message n multicast list register (SRIO_OM1MLR)	32	R/W	0000_0000h	19.5.110/ 1438
D_3160	Inbound message n mode register (SRIO_IM1MR)	32	R/W	0000_0000h	19.5.111/ 1439
D_3164	Inbound message n status register (SRIO_IM1SR)	32	R/W	0000_0002h	19.5.112/ 1442

Table continues on the next page...

SRIo memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_3168	Extended inbound message n frame queue dequeue pointer address register (SRIO_EIM1FQDPAR)	32	R/W	0000_0000h	19.5.113/ 1444
D_316C	Inbound message n frame queue dequeue pointer address register (SRIO_IM1FQDPAR)	32	R/W	0000_0000h	19.5.114/ 1444
D_3170	Extended inbound message n frame queue enqueue pointer address register (SRIO_EIM1FQEPR)	32	R/W	0000_0000h	19.5.115/ 1445
D_3174	Inbound message n frame queue enqueue pointer address register (SRIO_IM1FQEPR)	32	R/W	0000_0000h	19.5.116/ 1446
D_3178	Inbound message n maximum interrupt report interval register (SRIO_IM1MIRIR)	32	R/W	FFFF_FF00h	19.5.117/ 1447
D_3400	Outbound doorbell mode register (SRIO_ODMR)	32	R/W	0000_0000h	19.5.118/ 1447
D_3404	Outbound doorbell status register (SRIO_ODSR)	32	R/W	0000_0000h	19.5.119/ 1449
D_3418	Outbound doorbell n destination port register (SRIO_ODDPR)	32	R/W	0000_0000h	19.5.120/ 1451
D_341C	Outbound doorbell n destination attributes register (SRIO_ODDATR)	32	R/W	0000_0000h	19.5.121/ 1452
D_342C	Outbound doorbell n retry error threshold configuration register (SRIO_ODRETCR)	32	R/W	0000_0000h	19.5.122/ 1453
D_3460	Inbound doorbell n mode register (SRIO_IDMR)	32	R/W	0000_0000h	19.5.123/ 1454
D_3464	Inbound doorbell n status register (SRIO_IDSRA)	32	R/W	0000_0002h	19.5.124/ 1457
D_3468	Extended inbound doorbell n queue dequeue pointer address register (SRIO_EIDQDPAR)	32	R/W	0000_0000h	19.5.125/ 1458
D_346C	Inbound doorbell n queue dequeue Pointer address register (SRIO_IDQDPAR)	32	R/W	0000_0000h	19.5.126/ 1459
D_3470	Extended inbound doorbell n queue enqueue pointer address register (SRIO_EIDQEPR)	32	R/W	0000_0000h	19.5.127/ 1460
D_3474	Inbound doorbell n Queue enqueue pointer address register (SRIO_IDQEPR)	32	R/W	0000_0000h	19.5.128/ 1460
D_3478	Inbound doorbell n maximum interrupt report interval register (SRIO_IDMIRIR)	32	R/W	FFFF_FF00h	19.5.129/ 1461
D_34E0	Inbound port-write n mode register (SRIO_IPWMR)	32	R/W	0000_0000h	19.5.130/ 1462
D_34E4	Inbound port-write n status register (SRIO_IPWSR)	32	R/W	0000_0000h	19.5.131/ 1463
D_34E8	Extended inbound port-write n queue base address register (SRIO_EIPWQBAR)	32	R/W	0000_0000h	19.5.132/ 1464
D_34EC	Inbound port-write n queue base address register (SRIO_IPWQBAR)	32	R/W	0000_0000h	19.5.133/ 1464

19.5.1 Device identity capability register (SRIODIDCAR)

The device vendor identity field (DVI) identifies the vendor that manufactured the device containing the processing element. A value for DVI is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association.

The device identity field (DI) is intended to uniquely identify the type of device from the vendor specified by DVI. The values for DI are assigned and managed by the respective vendor.

DIDCAR is a read-only register.

Address: C_0000h base + 0h offset = C_0000h

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

SRIODIDCAR field descriptions

Field	Description
0–15 DI	Device identity 0x0400 = P4080E (with security) 0x0401 = P4080 (no security) 0x0408 = P4040E (with security) 0x0409 = P4040 (no security)
16–31 DVI	Device vendor identity (Freescale = 0x0002)

19.5.2 Device information capability register (SRIO_DICAR)

The device revision field (DR) is intended to identify the revision level of the device. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI]. DICAR represents a copy of the device's system version register (SVR). See System Version Register (SVR) for more information.

Address: C_0000h base + 4h offset = C_0004h

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

SRIO_DICAR field descriptions

Field	Description
0–31 DR	Device revision. This is a copy of the device's system version register (SVR). See System version register (DCFG_SVR) for additional information. 0x8208_0010 P4080 0x8209_0010 P4040

19.5.3 Assembly identity capability register (SRIO_AIDCAR)

The assembly vendor identity field (AVI) identifies the vendor that manufactured the assembly or subsystem containing the device. A value for AVI is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association.

The assembly identity field (AI) is intended to uniquely identify the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.

Address: C_0000h base + 8h offset = C_0008h

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

SARIO_AIDCAR field descriptions

Field	Description
0–15 AI	Assembly identity (all zeros)
16–31 AVI	Assembly vendor identity (all zeros)

19.5.4 Assembly information capability register (SRIO_AICAR)

AICAR contains additional information about the assembly and the pointer to the first entry in the extended features list.

Address: C_0000h base + Ch offset = C_000Ch

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

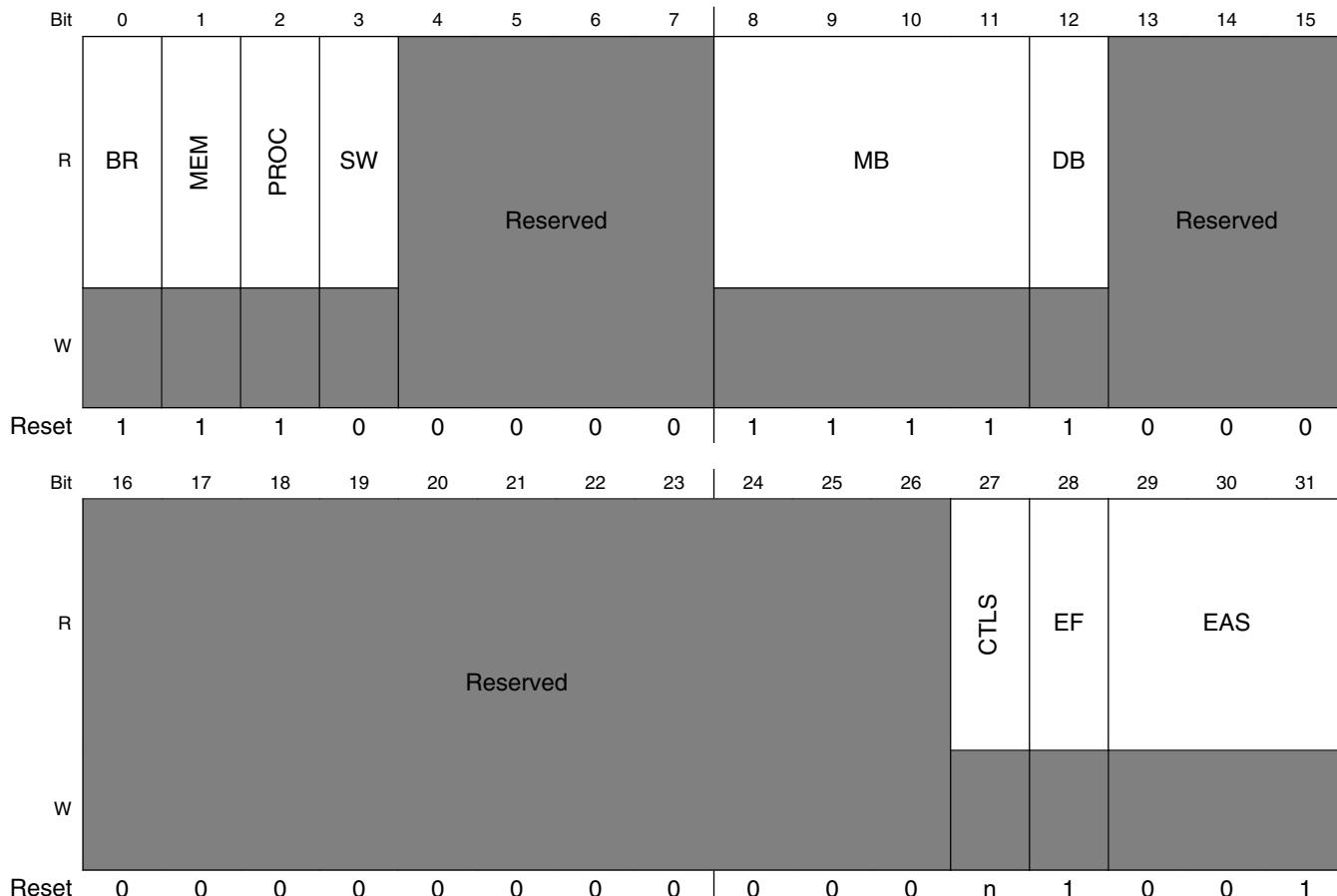
SARIO_AICAR field descriptions

Field	Description
0–15 AR	AssyRev field (all zeros)
16–31 EFP	ExtendedFeaturesPtr field (0x100)

19.5.5 Processing element features capability register (SRI0_PEFCAR)

The processing element features capability register (PEFCAR) identifies the major functionality provided by the processing element. PEFCAR is a read-only register.

Address: C_0000h base + 10h offset = C_0010h



SRI0_PEFCAR field descriptions

Field	Description
0 BR	Bridge. PE can bridge to another interface. (BR = 1)
1 MEM	Memory. PE has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations. (MEM = 1)
2 PROC	Processor. PE physically contains a local processor that executes code. (PROC = 1)
3 SW	Switch. PE does not bridge to another external RapidIO interface. (SW = 0)

Table continues on the next page...

SRIO_PEFCAR field descriptions (continued)

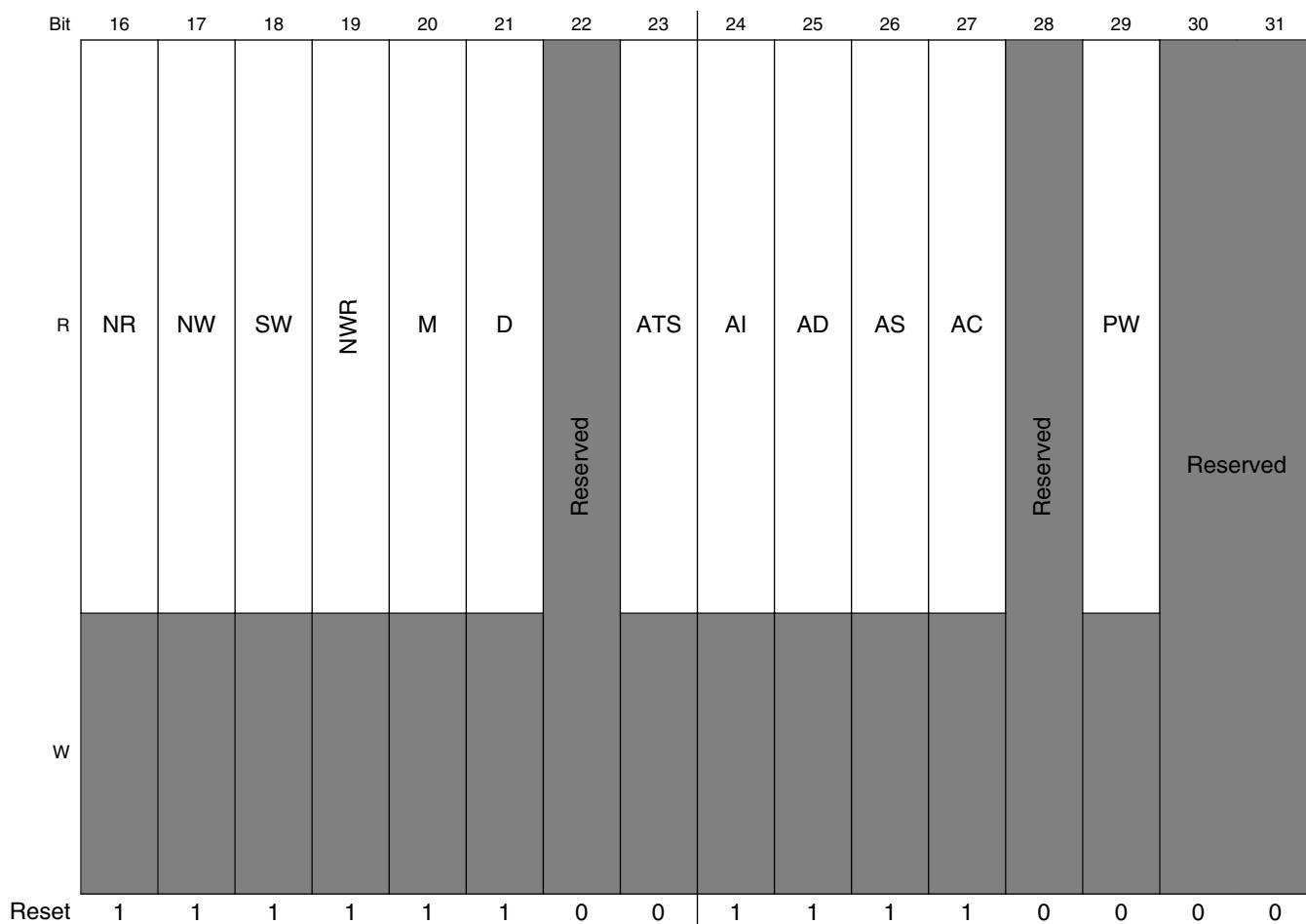
Field	Description
4–7 -	This field is reserved. Reserved
8–11 MB	Mailbox 0-3. (MB = 1111) 0 indicates PE supports inbound mailbox 0. 1 indicates PE supports inbound mailbox 1. 2 indicates PE supports inbound mailbox 2. 3 indicates PE supports inbound mailbox 3.
12 DB	Doorbell. The RapidIO controller supports inbound doorbells. (DB = 1)
13–26 -	This field is reserved. Reserved
27 CTLS	This bit reflects the selected RapidIO common transport system size. The RapidIO common transport system size is determined at power-on reset. 0 PE only supports common transport small systems (up to 256 devices). 1 PE supports common transport large systems (up to 65,536 devices).
28 EF	Extended features pointer is valid. (EF = 1)
29–31 EAS	Indicates the number of address bits supported by the PE both as a source and target of an operation. EAS = 001(34-bit addresses)

19.5.6 Source operations capability register (SRI0_SOCAR)

The source operations capability register (SOCAR) defines the set of RapidIO I/O logical operations that can be issued by this processing element. SOCAR is a read-only register.

Address: C_0000h base + 18h offset = C_0018h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES						
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0



SARIO_SOCAR field descriptions

Field	Description
0 R	PE does not support a Read operation. (R = 0)
1 IR	PE does not support an IRead operation. (IR = 0)
2 RO	PE does not support a Read_To_Own operation. (RO = 0)
3 DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4 C	PE does not support a Castout operation. (C = 0)
5 F	PE supports a Flush operation. (F = 1)
6 IOR	PE supports an I/O-Read operation. (IOR = 1)
7 ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8 TIE	PE does not support a TLBIE operation. (TIE = 0)

Table continues on the next page...

SRIO_SOCAR field descriptions (continued)

Field	Description
9 TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15 -	This field is reserved. Reserved
16 NR	PE supports a Nread operation. (NR = 1)
17 NW	PE supports a Nwrite operation. (NW = 1)
18 SW	PE supports an Swrite operation. (SW = 1)
19 NWR	PE supports a Nwrite_R operation. (NWR = 1)
20 M	PE supports a Message operation. (M = 1)
21 D	PE supports a Doorbell operation. (D = 1)
22 -	This field is reserved. Reserved
23 ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24 AI	PE supports an Atomic-Inc operation. (AI = 1)
25 AD	PE supports an Atomic-Dec operation. (AD = 1)
26 AS	PE supports an Atomic-Set operation. (AS = 1)
27 AC	PE supports an Atomic-Clr operation. (AC = 1)
28 -	This field is reserved. Reserved
29 PW	PE does not support a Port-Write operation. (PW = 0)
30–31 -	This field is reserved. Reserved

19.5.7 Destination operations capability register (SRI0_DOCAR)

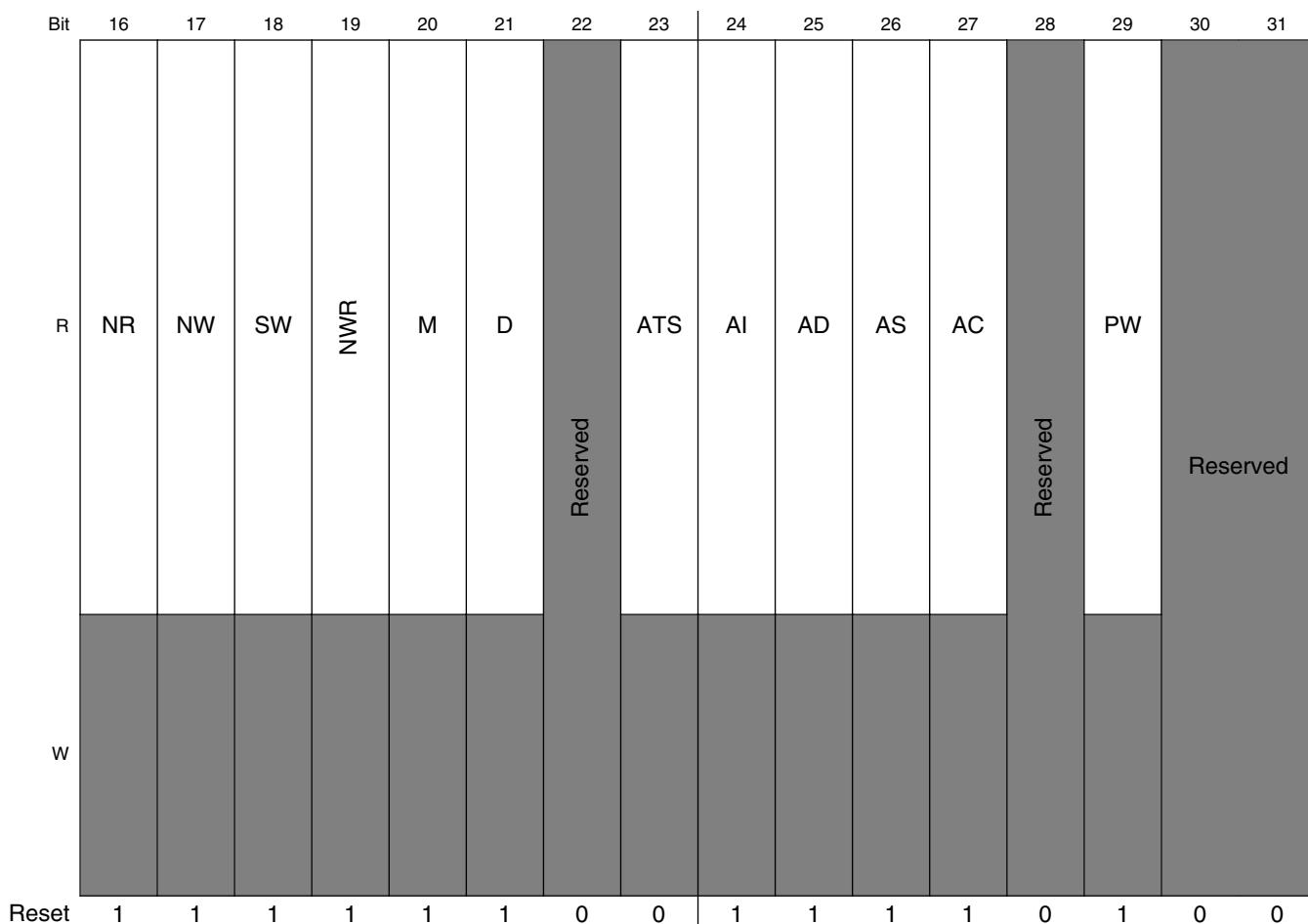
The destination operations capability register (DOCAR) defines the set of RapidIO I/O operations that can be supported by this processing element. DOCAR is a read-only register.

Address: C_0000h base + 1Ch offset = C_001Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES						
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Reserved



SRIO_DOCAR field descriptions

Field	Description
0 R	PE does not support a Read operation. (R = 0)
1 IR	PE does not support an IRead operation. (IR = 0)
2 RO	PE does not support a Read_To_Own operation. (RO = 0)
3 DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4 C	PE does not support a Castout operation. (C = 0)
5 F	PE does not support a Flush operation. (F = 0)
6 IOR	PE does not support an I/O-Read operation. (IOR = 0)
7 ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8 TIE	PE does not support a TLBIE operation. (TIE = 0)

Table continues on the next page...

SRIO_DOCAR field descriptions (continued)

Field	Description
9 TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15 -	This field is reserved. Reserved
16 NR	PE supports a Nread operation. (NR = 1)
17 NW	PE supports a Nwrite operation. (NW = 1)
18 SW	PE supports an Swrite operation. (SW = 1)
19 NWR	PE supports a Nwrite_R operation. (NWR = 1)
20 M	PE supports a Message operation. (M = 1)
21 D	PE supports a Doorbell operation. (D = 1)
22 -	This field is reserved. Reserved
23 ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24 AI	PE supports an Atomic-Inc operation. (AI = 1)
25 AD	PE supports an Atomic-Dec operation. (AD = 1)
26 AS	PE supports an Atomic-Set operation. (AS = 1)
27 AC	PE supports an Atomic-Clr operation. (AC = 1)
28 -	This field is reserved. Reserved
29 PW	PE supports a Port-Write operation. (PW = 1)
30–31 -	This field is reserved. Reserved

19.5.8 Mailbox command and status register (SRIOMCSR)

The MCSR reflects the status of the RapidIO message controllers on this device.

Address: C_0000h base + 40h offset = C_0040h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A0	FU0	EM0	B0	FA0	ERR0	Reserved		A1	FU1	EM1	B1	FA1	ERR1	Reserved	
W																
Reset	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOMCSR field descriptions

Field	Description
0 A0	Available 0 Message Controller 0 is not ready to accept messages. All incoming message transactions return error responses. 1 Message Controller 0 is initialized and ready to accept messages.
1 FU0	Full 0 Message Controller 0 is not full. 1 Message Controller 0 is full. New messages are retried.
2 EM0	Empty 0 Message Controller 0 contains outstanding messages. 1 Message Controller 0 contains no outstanding messages.
3 B0	Busy 0 Message Controller 0 is not busy processing a message. 1 Message Controller 0 is busy processing a message. New message operations return retry responses.
4 FA0	Failure 0 Message Controller 0 has not encountered an internal error. 1 Message Controller 0 had an internal fault or error condition and is waiting for assistance. All incoming message transactions return error responses.
5 ERR0	Error This field always returns a 0.
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_MCSR field descriptions (continued)

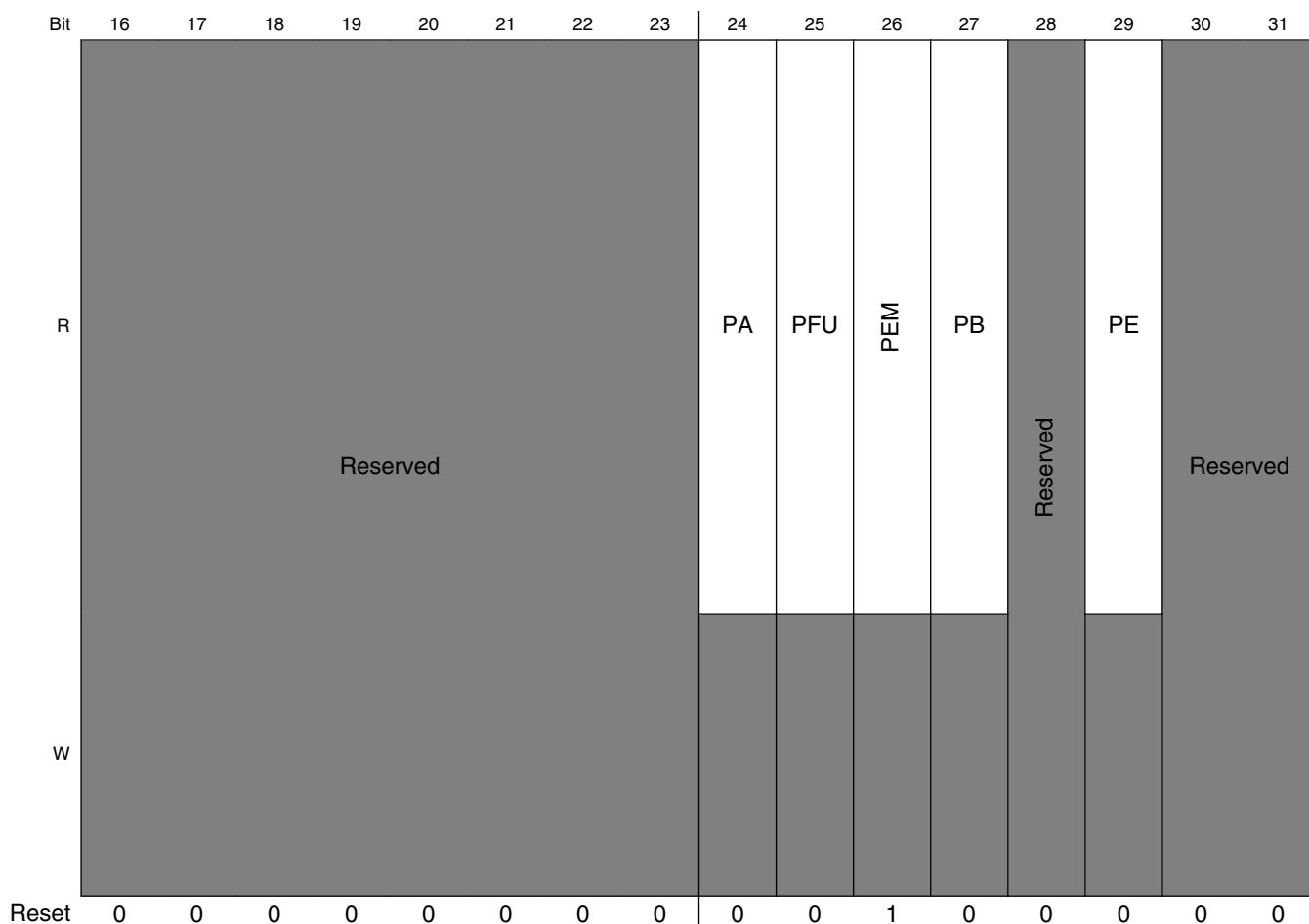
Field	Description
8 A1	Available 0 Message Controller 1 is not ready to accept messages. All incoming message transactions return error responses. 1 Message Controller 1 is initialized and ready to accept messages.
9 FU1	Full 0 Message Controller 1 is not full. 1 Message Controller 1 is full. New messages are retried.
10 EM1	Empty 0 Message Controller 1 contains outstanding messages. 1 Message Controller 1 contains no outstanding messages.
11 B1	Busy 0 Message Controller 1 is not busy processing a message. 1 Message Controller 1 is busy processing a message. New message operations return retry responses.
12 FA1	Failure 0 Message Controller 1 has not encountered an internal error. 1 Message Controller 1 had an internal fault or error condition and is waiting for assistance. All incoming message transactions return error responses.
13 ERR1	Error This field always returns a 0.
14–31 -	This field is reserved. Reserved

19.5.9 Port -Write and doorbell command and status register (SRIO_PWDCSR)

The PWDCSR reflects the status of the RapidIO doorbell and port-write hardware on this device. Additional details can be found in the *RapidIO Interconnect Specification, Revision 1.2*, in sections "Doorbell CSR" and "Write Port CSR."

Address: C_0000h base + 44h offset = C_0044h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	A	FU	EM	B	FA	ERR										
W																
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

**SRIO_PWDCSR field descriptions**

Field	Description
0 A	Available 0 Doorbell unit is not ready to accept doorbell messages. All incoming doorbell transactions return error responses. 1 Doorbell unit is initialized and ready to accept doorbell messages.
1 FU	Full 0 Doorbell unit is not full. 1 Doorbell unit is full. New doorbell messages are retried.
2 EM	Empty 0 Doorbell unit has outstanding doorbell messages. 1 Doorbell unit has no outstanding doorbell messages.
3 B	Busy 0 Doorbell unit is not busy processing a doorbell message. 1 Doorbell unit is busy processing a doorbell message. Incoming transactions are not retried.
4 FA	Failure

Table continues on the next page...

SRIO_PWDCSR field descriptions (continued)

Field	Description
	0 Doorbell unit has not encountered an internal error. 1 Doorbell unit had an internal fault or error condition and is waiting for assistance. All incoming doorbell transactions return error responses.
5 ERR	Error This field always returns a 0.
6–23 -	This field is reserved. Reserved
24 PA	Port-write unit available. 0 Port-write unit is not available. All incoming port-write packets are discarded. 1 Port-write unit is initialized and ready to accept a port-write packet.
25 PFU	Port-write unit full 0 Port-write unit is not full. 1 Port-write unit is full. All incoming port-write packets are discarded.
26 PEM	Port-write unit empty This field always returns a 1.
27 PB	Port-write unit busy 0 Port-write unit is not busy. 1 Port-write unit is busy processing a port-write packet. Incoming port-write packets are discarded.
28 -	This field is reserved. Reserved
29 PE	Error This field always returns a 0.
30–31 -	This field is reserved. Reserved

19.5.10 Processing element logical layer control command and status register (SRIO_PELLCCSR)

The processing element logical layer control command and status register (PELLCCSR) controls the extended addressing abilities. The RapidIO endpoint only supports 34-bit addressing. PELLCCSR is a read-only register.

Address: C_0000h base + 4Ch offset = C_004Ch

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

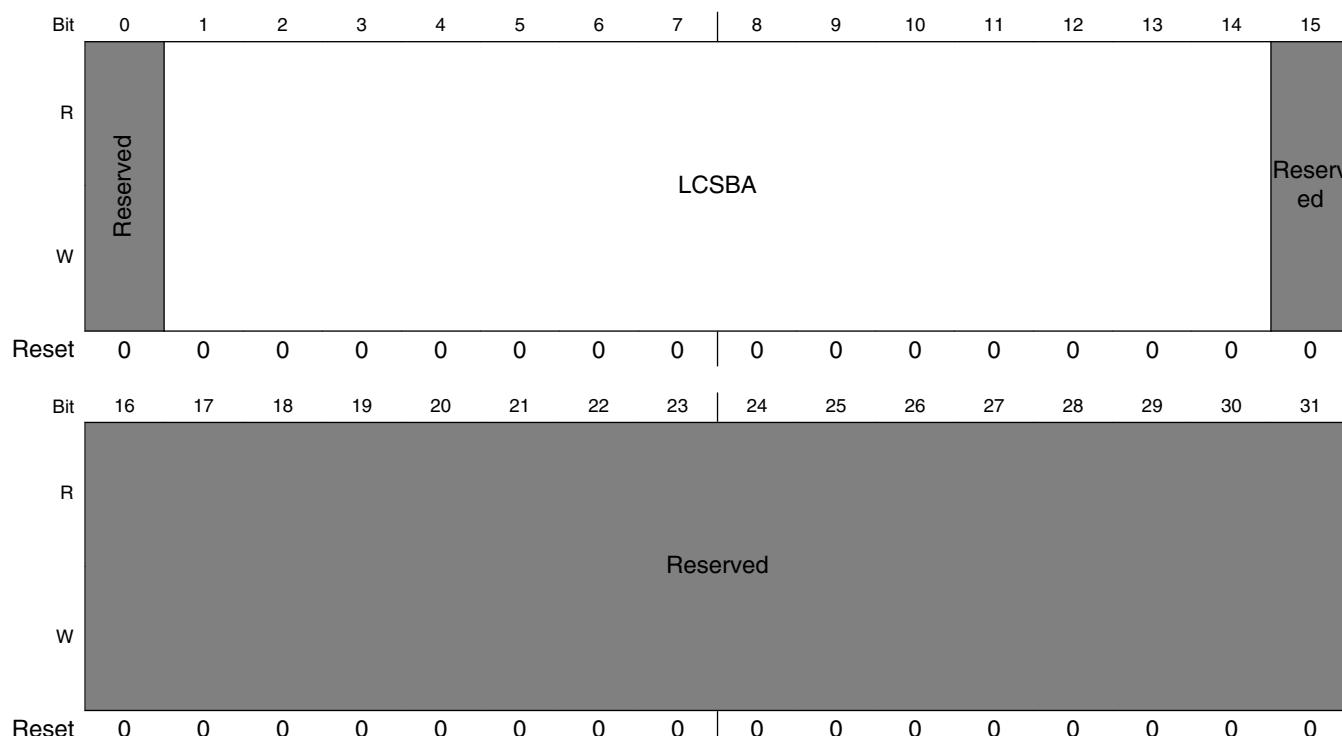
SRIO_PELLCCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 EAC	Extended addressing control. Read-only value is 0b001.

19.5.11 Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR)

The local configuration space base address 1 command and status register (LCSBA1CSR) specifies the least-significant bits of the local physical address double-word offset for the processing element's configuration register space, allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of a processing element through regular read and write operations rather than maintenance operations. The double-word offset is right-justified in the register. This window has priority over all ATMU windows. As is the case with all registers, an external processor writing to LCSBA1CSR should not assume it has been written until a response has been received.

Address: C_0000h base + 5Ch offset = C_005Ch



SRIO_LCSBA1CSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–14 LCSBA	Local configuration space base address. These bits correspond to the highest 14 bits of the 34-bit RapidIO address space.
15–31 -	This field is reserved. Reserved

19.5.12 Base device ID command and status register (SRIO_BDIDCSR)

BDIDCSR contains the base device ID values for the processing element.

Address: C 0000h base + 60h offset = C 0060h

SRIODIDCSR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 BDID	<p>Base device ID of the device in a small common transport system (RapidIO device ID).</p> <p>If RapidIO is configured as a host, BDID = {0b0000_0, RCW[RIO_DEVICE_ID](260-262) }.</p> <p>If RapidIO is configured as an agent, BDID = {0b1111_111, RCW[RIO_DEVICE_ID](262) }.</p> <p>(See RCW Field Definitions for details. Specifically, see RCW fields RIO_DEVICE_ID(260-262), RIO_SYS_SIZE(263), and HOST_AGT_B1(264-266).)</p>
16–31 LBDID	<p>Large base device ID of the device in a large common transport system. This field is valid only if PEFCAR[CTLS] is set.</p> <p>If RapidIO is configured as a host, LBDID = 0b0000_0000_0000_0, RCW[RIO_DEVICE_ID](260-262) .</p> <p>If RapidIO is configured as an agent, LBDID = 0b1111_1111_1111_111, RCW[RIO_DEVICE_ID](262) .</p> <p>If RapidIO is configured as a host, LBDID = 0b0000_0000.</p> <p>If RapidIO is configured as an agent, LBDID = 0b1111_1111.</p> <p>(See RCW Field Definitions for details. Specifically, see RCW fields RIO_SYS_SIZE(263), and HOST_AGT_B1(264-266).)</p>

19.5.13 Host base device ID lock command and status register (SRI0_HBDIDLCSR)

The host base device ID lock command and status register (HBDIDLCSR) contains the base device ID value for the processing element in the system that is responsible for initializing this processing element. The HBDID field is a write-once/resettable field which provides a lock function. Once HBDID is written, all subsequent writes to the field are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF. After writing HBDID, a processing element must then read the host base device ID lock CSR to verify that it owns the lock before attempting to initialize this processing element.

Address: C_0000h base + 68h offset = C_0068h

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

SRI0_HBDIDLCSR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 HBDID	This is the host base device ID for the processing element that is responsible for initializing this device. Only the first write to this field is accepted; all other writes are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF.

19.5.14 Component tag command and status register (SRI0_CTCsr)

The component tag command and status register (CTCSR) contains a component tag value for the processing element and can be assigned by software when the device is initialized. It is unused internally in the RapidIO endpoint.

Address: C_0000h base + 6Ch offset = C_006Ch

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

SRIO_CTCsr field descriptions

Field	Description
0–31 CT	Component tag

19.5.15 Port maintenance block header 0 (SRIO_PMBH0)

The port maintenance block header 0 register (PMBH0) contains a pointer (EF_PTR) to the next EF_BLK (Extended Features Space, Error Management) and the EF_ID that identifies this as the generic end point port maintenance block header. Note that while registers defined by software-assisted error recovery are supported, software-assisted error recovery is not (these registers are included for hot insertion only); therefore, the RapidIO endpoint is defined here as not supporting software-assisted error recovery. PMBH0 is a read-only register.

Address: C_0000h base + 100h offset = C_0100h

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

SRIO_PMBH0 field descriptions

Field	Description
0–15 EF_PTR	Extended features pointer
16–31 EF_ID	Extended features ID

19.5.16 Port link time-out control command and status register (SRIO_PLTOCCSR)

The port link time-out control command and status register (PLTOCCSR) contains the link time-out value for all ports on a device. This time-out is for link events such as sending a packet to receiving the corresponding acknowledge and sending a link-request to receiving the corresponding link-response. The reset, or default, value of this counter is the maximum time-out interval . Minimal programmable TV value supported is 0x200.

The resolution of this timer is 152/(platform frequency). For example, at a platform frequency of 600 MHz, the maximum timeout value is $0xFF_FFFF * 152/600MHz = 4.25$ seconds.

Address: C_0000h base + 120h offset = C_0120h

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

Reset 1 0 0 0 0 0 0 0 0 0

SRIO_PLTOCCSR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the link time-out timer. This value is loaded each time the link time-out timer starts.
24–31 -	This field is reserved. Reserved

19.5.17 Port response time-out control command and status register (SRIO_PRTOCCSR)

The port response time-out control command and status register (PRTOCCSR) contains the time-out timer count for all ports on a device. This time-out is for sending a request packet to receiving the corresponding response packet. Note that this applies to the RapidIO endpoint and the messaging unit. The reset, or default, value of this counter is the maximum time-out interval .

The resolution of this timer is 152/(platform frequency). For example, at a platform frequency of 600 MHz, the maximum timeout value is $0xFF_FFFF * 152/600MHz = 4.25$ seconds.

Address: C_0000h base + 124h offset = C_0124h

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

Reset 1 0 0 0 0 0 0 0 0 0

SRIO_PRTOCCSR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the response time-out timer. This value is loaded each time the response time-out timer starts.
24–31 -	This field is reserved. Reserved

19.5.18 Port General control command and status register (SRIO_PGCCSR)

The port general control command and status register (PGCCSR) contains control register bits applicable to the RapidIO interface .

Address: C_0000h base + 13Ch offset = C_013Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	H	M	D													
Reset	n	n	n	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

-

SRIO_PGCCSR field descriptions

Field	Description
0 H	Host. On reset the value of this bit is assigned by the RCW field, "HOST_AGT_B1" (264-266). (See RCW Field Definitions .) 0 Agent device 1 Host device
1 M	Master. The value of this bit is identical to PGCCSR[H] which is assigned by the RCW field, "HOST_AGT_B1" (264-266). (See RCW Field Definitions .) For agent devices, this bit may be modified manually by software after device enumeration is complete. This bit controls whether or not a device is allowed to issue requests into the system. If M = 0, the device may only respond to requests. Software must ensure no requests are sent to the device by disabling any local access windows and disabling inbound ATMU windows from other devices targeting SRIO while M=0. 0 Device is not enabled to issue requests into the system. 1 Device is enabled to issue requests into the system.
2 D	Discovered. The value of this bit is identical to PGCCSR[H] which is assigned by the RCW field, "HOST_AGT_B1" (264-266). (See RCW Field Definitions .) For agent devices, this bit should be modified by the host device to indicate this device has been discovered and enumerated. 0 Device has not been discovered by system host. 1 Device has been discovered by system host.
3–31 -	This field is reserved. Reserved

19.5.19 Port 1 Link maintenance request command and status register (SRIO_P1LMREQCSR)

The port 1 link maintenance request command and status register (P1LMREQCSR) is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the corresponding RapidIO endpoint port interface.

Address: C_0000h base + 140h offset = C_0140h

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

SRIO_P1LMREQCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 C	LINK_REQUEST command to send. If read, this field returns the last written value. If written with a value other than 0b011 (reset-device) or 0b100 (input-status), the resulting operation is undefined, as all other values are reserved in the RapidIO specification.

19.5.20 Port 1 Link maintenance response command and status register (SRIO_P1LMRESPCSR)

The port 1 link maintenance response command and status register (P1LMRESPCSR) is accessible both by the local processor and an external device. A read to this register returns the status received in a link-response control symbol. This register is read only.

Address: C_0000h base + 144h offset = C_0144h

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

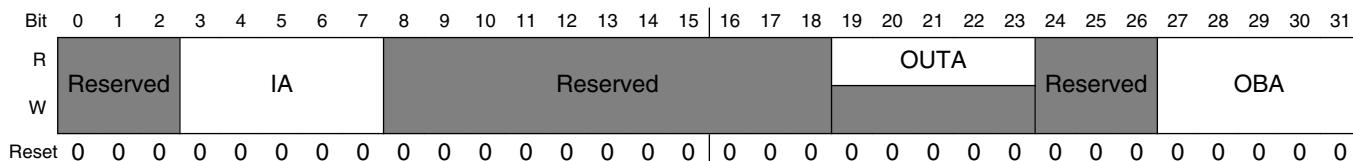
SRIO_P1LMRESPCSR field descriptions

Field	Description
0 RV	Response valid. If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. If the link-request does not cause a link-response, this bit indicates that the link-request has been transmitted. This bit clears on read, so be careful when polling its value.
1–21 -	This field is reserved. Reserved
22–26 AS	AckID_status field from LINK_RESPONSE
27–31 LS	Link_status field from LINK_RESPONSE

19.5.21 Port 1 Local ackID status command and status register (SRIO_P1LASCSR)

The port 1 local ackID status command and status register (P1LASCSR) is accessible both by the local processor and an external device. A read to this register returns the local ackID status for both the output and input ports of the device.

Address: C_0000h base + 148h offset = C_0148h

**SRIO_P1LASCSR field descriptions**

Field	Description
0–2 -	This field is reserved. Reserved
3–7 IA	Input port next expected ackID value.
8–18 -	This field is reserved. Reserved
19–23 OUTA	Outstanding port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
24–26 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1LASCSR field descriptions (continued)

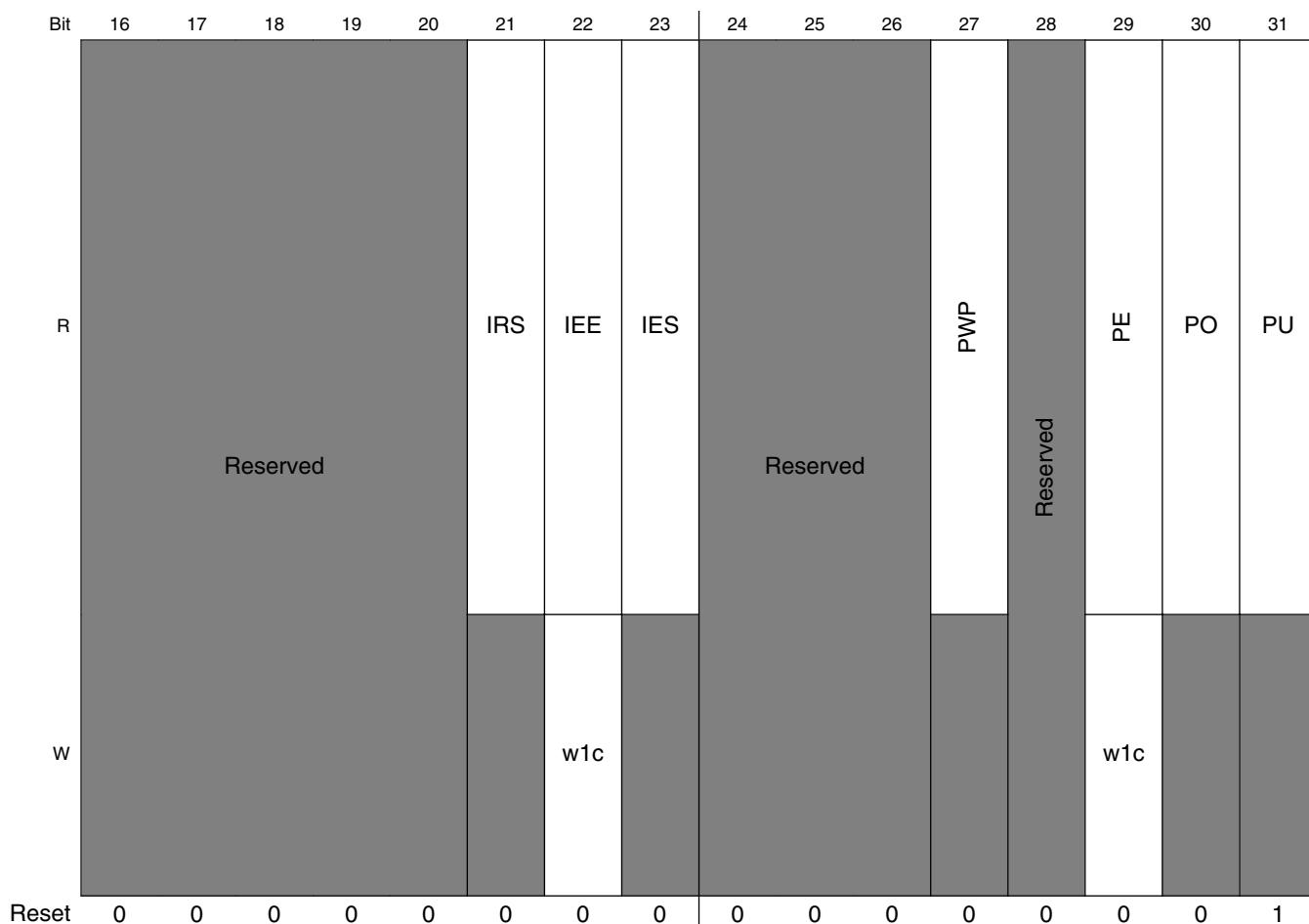
Field	Description
27–31 OBA	Outbound ackID output port next transmitted ackID value. This can be written by software but only if there are no outstanding unacknowledged packets. If there are, the newly-written value is ignored.

19.5.22 Port 1 Error and status command and status register (SRIO_P1ESCSR)

The port 1 error and status command and status register (P1ESCSR) is accessed when the local processor or an external device wishes to examine the port error and status information.

Address: C_0000h base + 158h offset = C_0158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R						OPD	OFE	ODE				ORE	OR	ORS	OEE	OES
	Reserved								Reserved							
W						w1c	w1c	w1c				w1c			w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**SARIO_P1ESCSR field descriptions**

Field	Description
0-4 -	This field is reserved. Reserved
5 OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: <ul style="list-style-type: none"> It is received while OFE is set and PmCCSR[DPE] (drop packet enable) is set and PmCCSR[SPF] (stop on port failed) is set. It is received while PmPCR[OBDEN] (output buffer drain enable) is set. It is not-accepted by the link-partner while PmERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and PmCCSR[DPE] is set and PmCCSR[SPF] is not set (and link-response returns expected ackID). Once OPD is set, it remains set until written with a logic 1 to clear.
6 OFE	Output Failed-encountered. Output port has encountered a failed condition, meaning that the Error Rate Counter has met or exceeded the port's failed error threshold (ERFTT). Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's failed error threshold and then meets or exceeds it again.
7 ODE	Output port has encountered a degraded condition, meaning that the Error Rate Counter has met or exceeded the port's degraded error threshold. Once set, remains set until written with a logic 1 to clear.

Table continues on the next page...

SRIO_P1ESCSR field descriptions (continued)

Field	Description
	Once cleared, does not assert again unless the Error Rate Counter dips below the port's degraded error threshold and then meets or exceeds it again.
8–10 -	This field is reserved. Reserved
11 ORE	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logic 1 to clear.
12 OR	Output port has received a packet retry control symbol and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read only)
13 ORS	Output port is stopped due to a retry (read only)
14 OEE	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logic 1 to clear.
15 OES	Output port is stopped due to a transmission error (read only)
16–20 -	This field is reserved. Reserved
21 IRS	Input port is stopped due to a retry (read only)
22 IEE	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logic 1 to clear.
23 IES	Input port is stopped due to a transmission error (read-only)
24–26 -	This field is reserved. Reserved
27 PWP	Port has encountered a condition which required it to initiate a maintenance port-write operation. This bit is only valid if the device is capable of issuing a maintenance port-write transaction. The RapidIO endpoint is not capable of issuing port-writes. This bit is hardwired to 0.
28 -	This field is reserved. Reserved
29 PE	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logic 1 to clear. This bit indicates that OFE is set while PmCCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.
30 PO	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. (read-only).
31 PU	Input and output ports are not initialized . This bit and bit 30 are mutually exclusive (read-only).

19.5.23 Port 1 Control command and status register (SRI0_P1CCSR)

The port 1 control command and status register (P1CCSR) contains control register bits for the RapidIO port .

Address: C_0000h base + 15Ch offset = C_015Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PW		IPW				PWO		PD	OPE	IPE	ECD	MEP			
W																
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R														SPF	DPE	PL
W																PT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SRI0_P1CCSR field descriptions

Field	Description
0–1 PW	Hardware width of the port (read-only). 00 Single-lane port 01 Four-lane port 10-11 Reserved
2–4 IPW	Width of the ports after initialized (read-only). 000 Single-lane port, lane 0 001 Single-lane port, lane 2 010 Four-lane port 011-111 Reserved
5–7 PWO	Soft port configuration to override the hardware size. This field should be changed only when the port is uninitialized. To achieve this, first disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port. To achieve this, first set PmCCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear PmCCSR[PD] (enabled). 000 No override 001 Reserved 010 Force single lane, lane 0 011 Force single lane, lane 2 100-111 Reserved, causes undefined operation
8 PD	Port disable.

Table continues on the next page...

SRIOP1CCSR field descriptions (continued)

Field	Description
	<p>0 Input error state machine operates normally 1 Input error state machine is forced to normal state</p>
9 OPE	<p>Output port transmit enable. OPE is ignored by RapidIO endpoints. It is expected that if OPE = 0, software will not send packets out of outbound. If packets are sent by OCN to outbound, they are sent out of RapidIO endpoints, regardless of the value of OPE.</p> <p>Initial value read from configuration pins.</p> <p>0 Port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. 1 Port is enabled to issue any packets.</p>
10 IPE	<p>Input port receive enable. This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly.</p> <p>Initial value read from configuration pins.</p> <p>0 Port is stopped and only enabled to route or respond to I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 1 Port is enabled to respond to any packet.</p>
11 ECD	<p>Error checking disable. This bit is hardwired to 0.</p> <p>This bit disables all RapidIO transmission error checking</p> <p>0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.</p>
12 MEP	<p>Multicast-event participant. This bit is hard-wired to 0.</p>
13–27 -	<p>This field is reserved. Reserved</p>
28 SPF	<p>Stop on port failed-encountered enable. This bit is used with the drop packet enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.</p>
29 DPE	<p>Drop packet enable. This bit is used with the stop on port failed-encountered enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.</p>
30 PL	<p>Port lockout.</p> <p>0 The packets that may be received and issued are controlled by the state of the OPE and IPE bits. 1 This port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.</p>
31 PT	<p>Port type (read-only).</p> <p>0 Reserved 1 Serial port.</p>

19.5.24 Port 2 Link maintenance request command and status register (SRIO_P2LMREQCSR)

The port 2 link maintenance request command and status register (P2LMREQCSR) is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the corresponding RapidIO endpoint port interface.

Address: C_0000h base + 160h offset = C_0160h

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

SRIO_P2LMREQCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 C	LINK_REQUEST command to send. If read, this field returns the last written value. If written with a value other than 0b011 (reset-device) or 0b100 (input-status), the resulting operation is undefined, as all other values are reserved in the RapidIO specification.

19.5.25 Port 2 Link maintenance response command and status register (SRIO_P2LMRESPCSR)

The port 2 link maintenance response command and status register (P2LMRESPCSR) is accessible both by the local processor and an external device. A read to this register returns the status received in a link-response control symbol. This register is read only.

Address: C_0000h base + 164h offset = C_0164h

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									AS				LS			
W																

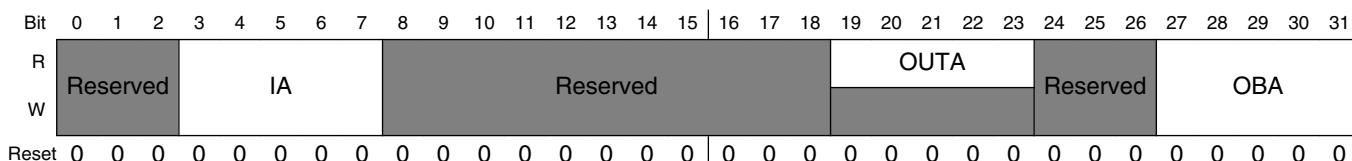
SRIO_P2LMRESPCSR field descriptions

Field	Description
0 RV	Response valid. If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. If the link-request does not cause a link-response, this bit indicates that the link-request has been transmitted. This bit clears on read, so be careful when polling its value.
1–21 -	This field is reserved. Reserved
22–26 AS	AckID_status field from LINK_RESPONSE
27–31 LS	Link_status field from LINK_RESPONSE

19.5.26 Port 2 Local ackID status command and status register (SRIO_P2LASCSR)

The port 2 local ackID status command and status register (P2LASCSR) is accessible both by the local processor and an external device. A read to this register returns the local ackID status for both the output and input ports of the device.

Address: C_0000h base + 168h offset = C_0168h

**SRIO_P2LASCSR field descriptions**

Field	Description
0–2 -	This field is reserved. Reserved
3–7 IA	Input port next expected ackID value.
8–18 -	This field is reserved. Reserved
19–23 OUTA	Outstanding port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
24–26 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2LASCSR field descriptions (continued)

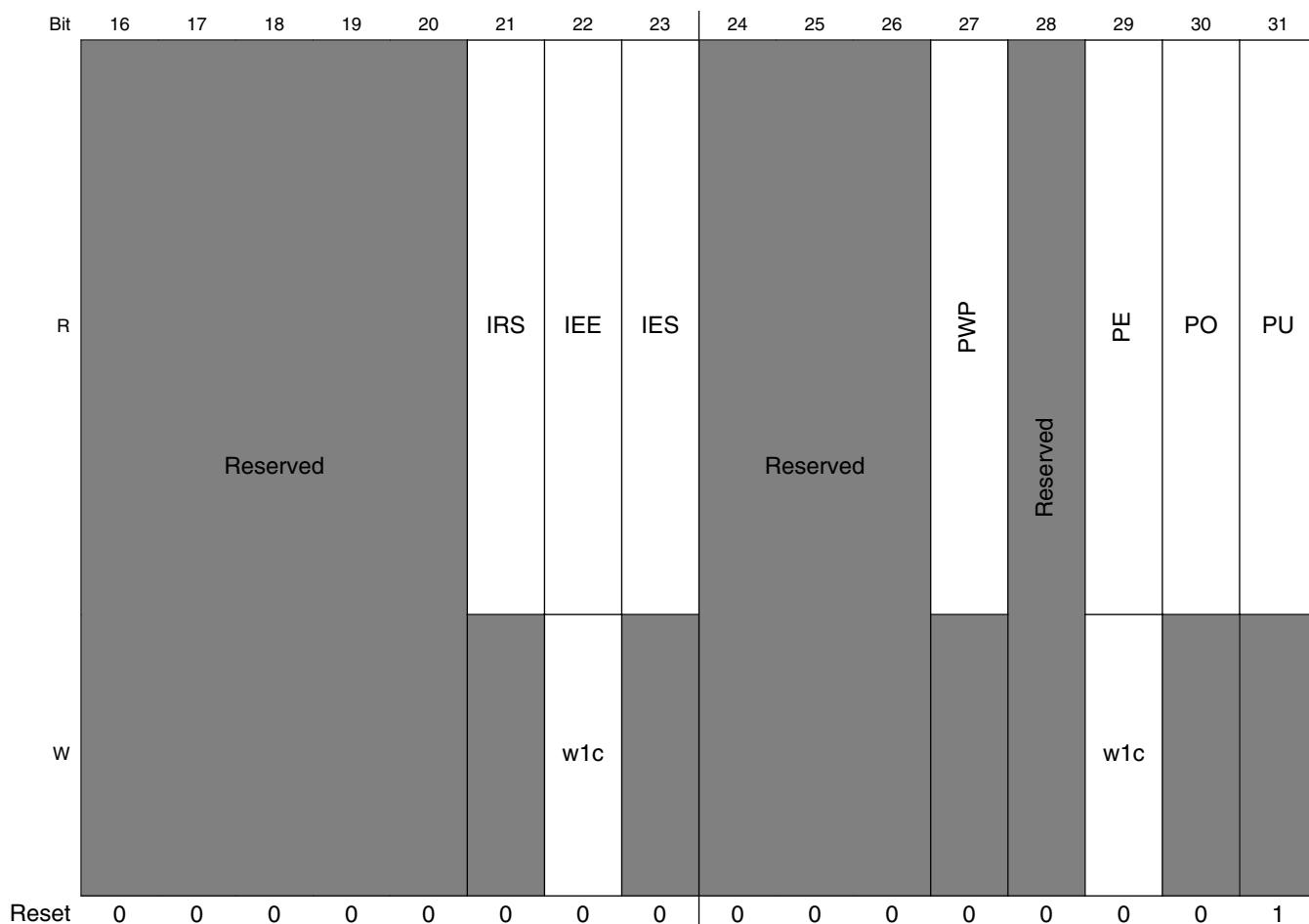
Field	Description
27–31 OBA	Outbound ackID output port next transmitted ackID value. This can be written by software but only if there are no outstanding unacknowledged packets. If there are, the newly-written value is ignored.

19.5.27 Port 2 Error and status command and status register (SRIO_P2ESCSR)

The port 2 error and status command and status register (P2ESCSR) is accessed when the local processor or an external device wishes to examine the port error and status information.

Address: C_0000h base + 178h offset = C_0178h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R						OPD	OFE	ODE				ORE	OR	ORS	OEE	OES
	Reserved								Reserved							
W						w1c	w1c	w1c				w1c			w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRIO_P2ESCSR field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5 OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: <ul style="list-style-type: none"> It is received while OFE is set and PmCCSR[DPE] (drop packet enable) is set and PmCCSR[SPF] (stop on port failed) is set. It is received while PmPCR[OBDEN] (output buffer drain enable) is set. It is not-accepted by the link-partner while PmERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and PmCCSR[DPE] is set and PmCCSR[SPF] is not set (and link-response returns expected ackID). Once OPD is set, it remains set until written with a logic 1 to clear.
6 OFE	Output Failed-encountered. Output port has encountered a failed condition, meaning that the Error Rate Counter has met or exceeded the port's failed error threshold (ERFTT). Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's failed error threshold and then meets or exceeds it again.
7 ODE	Output port has encountered a degraded condition, meaning that the Error Rate Counter has met or exceeded the port's degraded error threshold. Once set, remains set until written with a logic 1 to clear.

Table continues on the next page...

SRIO_P2ESCSR field descriptions (continued)

Field	Description
	Once cleared, does not assert again unless the Error Rate Counter dips below the port's degraded error threshold and then meets or exceeds it again.
8–10 -	This field is reserved. Reserved
11 ORE	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logic 1 to clear.
12 OR	Output port has received a packet retry control symbol and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read only)
13 ORS	Output port is stopped due to a retry (read only)
14 OEE	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logic 1 to clear.
15 OES	Output port is stopped due to a transmission error (read only)
16–20 -	This field is reserved. Reserved
21 IRS	Input port is stopped due to a retry (read only)
22 IEE	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logic 1 to clear.
23 IES	Input port is stopped due to a transmission error (read-only)
24–26 -	This field is reserved. Reserved
27 PWP	Port has encountered a condition which required it to initiate a maintenance port-write operation. This bit is only valid if the device is capable of issuing a maintenance port-write transaction. The RapidIO endpoint is not capable of issuing port-writes. This bit is hardwired to 0.
28 -	This field is reserved. Reserved
29 PE	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logic 1 to clear. This bit indicates that OFE is set while PmCCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.
30 PO	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. (read-only).
31 PU	Input and output ports are not initialized . This bit and bit 30 are mutually exclusive (read-only).

19.5.28 Port 2 Control command and status register (SRI0_P2CCSR)

The port 2 control command and status register (P2CCSR) contains control register bits for the RapidIO port .

Address: C_0000h base + 17Ch offset = C_017Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PW		IPW				PWO		PD	OPE	IPE	ECD	MEP			
W																
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R														SPF	DPE	PL
W																PT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SRI0_P2CCSR field descriptions

Field	Description
0–1 PW	Hardware width of the port (read-only). 00 Single-lane port 01 Four-lane port 10-11 Reserved
2–4 IPW	Width of the ports after initialized (read-only). 000 Single-lane port, lane 0 001 Single-lane port, lane 2 010 Four-lane port 011-111 Reserved
5–7 PWO	Soft port configuration to override the hardware size. This field should be changed only when the port is uninitialized. To achieve this, first disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port. To achieve this, first set PmCCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear PmCCSR[PD] (enabled). 000 No override 001 Reserved 010 Force single lane, lane 0 011 Force single lane, lane 2 100-111 Reserved, causes undefined operation
8 PD	Port disable.

Table continues on the next page...

SRIOP2CCSR field descriptions (continued)

Field	Description
	<p>0 Input error state machine operates normally 1 Input error state machine is forced to normal state</p>
9 OPE	<p>Output port transmit enable. OPE is ignored by RapidIO endpoints. It is expected that if OPE = 0, software will not send packets out of outbound. If packets are sent by OCN to outbound, they are sent out of RapidIO endpoints, regardless of the value of OPE.</p> <p>Initial value read from configuration pins.</p> <p>0 Port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. 1 Port is enabled to issue any packets.</p>
10 IPE	<p>Input port receive enable. This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly.</p> <p>Initial value read from configuration pins.</p> <p>0 Port is stopped and only enabled to route or respond to I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 1 Port is enabled to respond to any packet.</p>
11 ECD	<p>Error checking disable. This bit is hardwired to 0.</p> <p>This bit disables all RapidIO transmission error checking</p> <p>0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.</p>
12 MEP	<p>Multicast-event participant. This bit is hard-wired to 0.</p>
13–27 -	<p>This field is reserved. Reserved</p>
28 SPF	<p>Stop on port failed-encountered enable. This bit is used with the drop packet enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.</p>
29 DPE	<p>Drop packet enable. This bit is used with the stop on port failed-encountered enable bit to force certain behavior when the error rate failed threshold has been met or exceeded.</p>
30 PL	<p>Port lockout.</p> <p>0 The packets that may be received and issued are controlled by the state of the OPE and IPE bits. 1 This port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.</p>
31 PT	<p>Port type (read-only).</p> <p>0 Reserved 1 Serial port.</p>

19.5.29 Error reporting block header (SRIO_ERBH)

The error reporting block header register contains the EF_PTR to the next EF_BLK (the next EF_PTR is 0x0000 since this is the last set of registers in the extended features space) and the EF_ID that identifies this as the error reporting block header. ERBH is a read-only register.

Address: C_0000h base + 600h offset = C_0600h

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

SRIO_ERBH field descriptions

Field	Description
0–15 EF_PTR	Extended features pointer
16–31 EF_ID	Extended features ID

19.5.30 Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR)

This register indicates the error that was detected by the Logical or Transport logic layer. Software should write this register with all 0s to clear any detected errors and unlock the capture registers.

Error information that corresponds to two or more different error events are not captured on the same clock cycle. However, one error event (such as a corrupted inbound packet) can cause multiple bits to be set. The priority of errors is PRT and then all other errors. OACB error results in PRT. When PRT bit is set with OACB bit, error capture is for an OCN transaction for which an Outbound ATMU window boundary was crossed or a segment or subsegment boundary was crossed.

An error that is not enabled will set the detect bit in this register as long as a capture has not yet occurred.

Note that fields in this register can be set by writing to this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual Logical/Transport Layer error is being detected. Also, note that this register can be written with an invalid combination of bits set and care should be taken to avoid this.

Address: C_0000h base + 608h offset = C_0608h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IER	MER	GER	MFE	ITD	ITTE	MRT	PRT	UR	UT	Reserved					
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved									IACB	OACB	DMAMER	RETE	TSE	PTTL	Reserved
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_LTLEDCSR field descriptions

Field	Description
0 IER	IO error response. Received a response of ERROR for an IO logical layer request.
1 MER	Reserved for message error response. Received a response of ERROR for an MSG logical layer request. Error detected and captured in the message unit, if one exists.
2 GER	GSM error response. Received a response of ERROR for a GSM logical layer request.
3 MFE	Reserved for message format error. Received MESSAGE packet data payload with an invalid size or segment. Error detected and captured in the message unit, if one exists.
4 ITD	Illegal transaction decode. Received illegal fields in the request/response packet for a supported transaction (IO/MSG/GSM logical)
5 ITTE	Illegal transaction target error. Received a packet that contained a destination ID that is not defined for this end point . Endpoints with multiple ports and a built-in switch function may not report this as an error (transport)
6 MRT	Reserved for message request time-out. A required message request has not been received within the specified time-out interval. Error detected and captured in the message unit, if one exists.

Table continues on the next page...

SRIO_LTLEDCSR field descriptions (continued)

Field	Description
7 PRT	Packet response time-out. A required response has not been received within the specified time out interval (IO/MSG/GSM logical)
8 UR	Unsolicited response. An unsolicited/unexpected response packet was received (IO/MSG/GSM logical; only maintenance response for switches)
9 UT	Unsupported transaction. A transaction is received that is not supported in the destination operations CAR (IO/MSG/GSM logical; only maintenance port-write for switches)
10–23 -	This field is reserved. Reserved
24 IACB	Inbound ATMU crossed boundary. A transaction is received that crosses an inbound ATMU boundary.
25 OACB	Outbound ATMU crossed boundary. A transaction is being sent that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary.
26 DMAMER	DMA message error response. An error response was received for a DMA message (detected in the RapidIO endpoint, not the message unit).
27 RETE	Retry error threshold exceeded. The allowed number of logical retries (given by LRETCR[RET] has been exceeded. This bit is also driven by the Message Unit when the allowed number of message retries has been exceeded.
28 TSE	Transport size error. The tt field is not consistent with bit 27 of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29 PTTL	Packet time-to-live error. A packet time-to-live error occurred (a packet could not be successfully transmitted before the packet time-to-live counter expired).
30–31 -	This field is reserved. Reserved

19.5.31 Logical/Transport layer error enable command and status register (SRIO_LTLEECSR)

This register contains the bits that control whether an error condition locks the logical/transport layer error detect and capture registers and is reported to the system host. LTLEEDCSR is stored in all ports and the message unit.

Address: C_0000h base + 60Ch offset = C_060Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IER	MER	GER	MFE	ITD	ITTE	MRT	PRT	UR	UT	Reserved					
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								IACB	OACB	DMAMER	RETE	TSE	PTTL	Reserved	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_LTLEECSR field descriptions

Field	Description
0 IER	IO error response enable. Enable reporting of an IO error response. Capture and lock the error.
1 MER	Message error response enable. Enable reporting of a Message error response. Capture and lock the error (capture done in Message Unit, if one exists).
2 GER	GSM error response enable. Enable reporting of a GSM error response. Capture and lock the error.
3 MFE	Message format error enable. Enable reporting of a message format error. Capture and lock the error.(capture done in Message Unit, if one exists).
4 ITD	Illegal transaction decode enable. Enable reporting of an illegal transaction decode error. Capture and lock the error.
5 ITTE	Illegal transaction target error enable. Enable reporting of an illegal transaction target error. Capture and lock the error.
6 MRT	Message request time-out enable. Enable reporting of a Message Request time-out error. Capture and lock the error. (capture done in Message Unit, if one exists)

Table continues on the next page...

SRIO_LTLECSR field descriptions (continued)

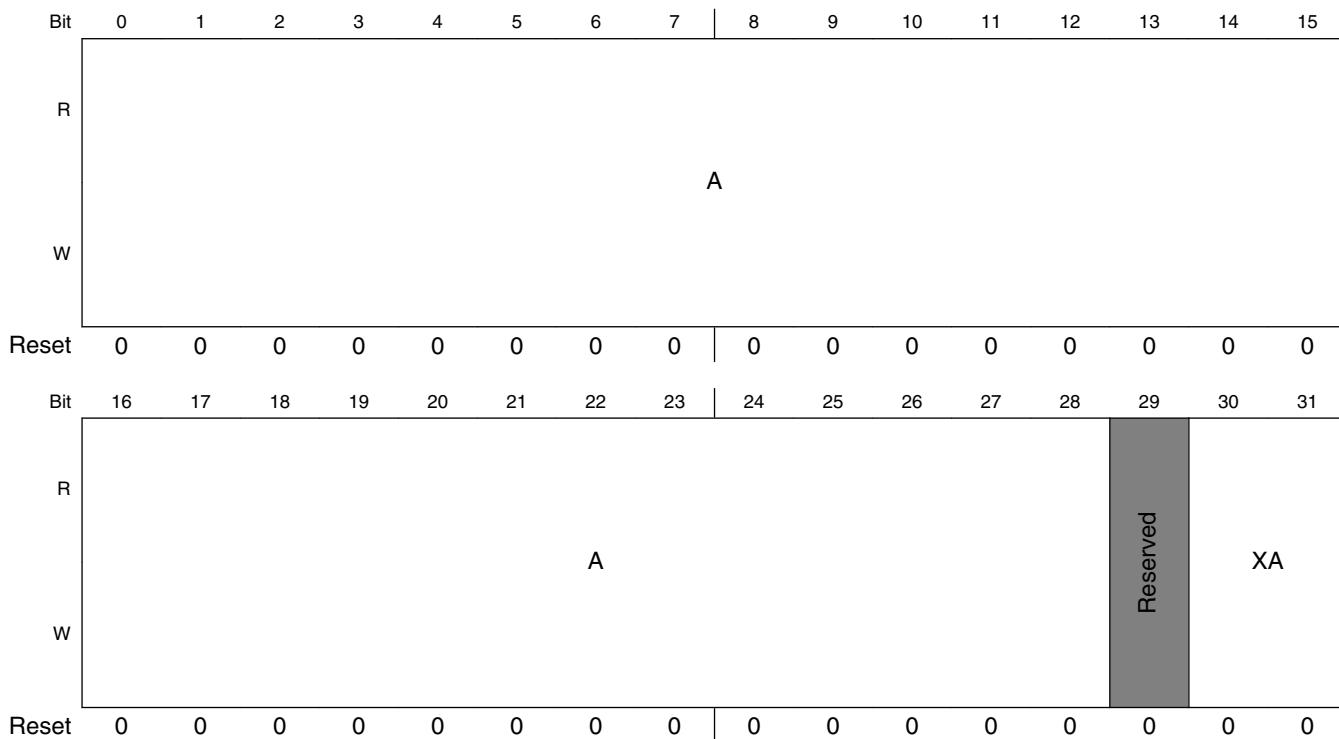
Field	Description
7 PRT	Packet response time-out error enable. Enable reporting of a packet response time-out error. Capture and lock the error.
8 UR	Unsolicited response error enable. Enable reporting of an unsolicited response error. Capture and lock the error.
9 UT	Unsupported transaction error enable. Enable reporting of an unsupported transaction error. Capture and lock the error.
10–23 -	This field is reserved. Reserved
24 IACB	Inbound ATMU crossed boundary error enable. Enable reporting of a received transaction that crosses an inbound ATMU boundary. Capture and lock the error.
25 OACB	Outbound ATMU crossed boundary error enable. Enable reporting of a transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary. Capture and lock the error.
26 DMAMER	DMA message error response. Enable error reporting of an error response for a DMA message. Capture and lock the error.
27 RETE	Retry error threshold exceeded. Enable error reporting when the allowed number of logical retries has been exceeded.
28 TSE	Transport size error. Enable error reporting when the tt field is not consistent with bit 27 of the Processing Element Features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29 PTTL	Packet time-to-live error. Enable reporting of a packet time-to-live time-out error. Capture and lock the error.
30–31 -	This field is reserved. Reserved

19.5.32 Logical/Transport layer address capture command and status register (SRIO_LTLACCSR)

The LTLACCSR provides error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. LTLACCSR is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLACCSR cannot lock if the message unit or any other port has locked.

Note that fields in this register can be set by writing this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Address: C_0000h base + 614h offset = C_0614h



SRIO_LTLACCSR field descriptions

Field	Description
0–28 A	Normally the least significant 29 bits of the address associated with the error (for requests, for responses if available). See Logical layer errors and error handling for details.
29 -	This field is reserved. Reserved
30–31 XA	xamsbs. Normally the extended address bits of the address associated with the error (for requests, responses, if available). See Logical layer errors and error handling for details.

19.5.33 Logical/Transport layer device ID capture command and status register (SRIO_LTLDIDCCSR)

LTLDIDCCSR contains error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. LTLDIDCCSR is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLDIDCCSR cannot lock if the message unit or any other port has locked.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Serial RapidIO Memory Map/Register Definition

Address: C_0000h base + 618h offset = C_0618h

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

SRIO_LTLDIDCCSR field descriptions

Field	Description
0–7 DIDMSB	Normally the most significant byte of the destinationID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). See Logical layer errors and error handling for details.
8–15 DID	Normally the destinationID (or least significant byte of the destination ID if large transport system) associated with the error. See Logical layer errors and error handling for details.
16–23 SIDMSB	Normally the most significant byte of the sourceID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). See Logical layer errors and error handling for details.
24–31 SID	Normally the sourceID (or least significant byte of the source ID if large transport system) associated with the error. See Logical layer errors and error handling for details.

19.5.34 Logical/Transport layer control capture command and status register (SRIO_LTLCCCSR)

LTLCCCS contains error information. It is stored in each port and in the message unit, although the values in this register can differ between each port and message unit. The message unit LTLCCCSR cannot lock if the message unit or any other port has locked.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Address: C_0000h base + 61Ch offset = C_061Ch

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

SRIO_LTLCCCSR field descriptions

Field	Description
0–3 FT	Normally the format type associated with the error. See Logical layer errors and error handling for details.
4–7 TT	Normally the transaction type associated with the error. See Logical layer errors and error handling for details.
8–15 MI	Normally the message Information: letter, mbox, and msgseg for the last message request received for the mailbox that had an error (message errors only). See Logical layer errors and error handling for details.
16–31 -	This field is reserved. Reserved

19.5.35 Port 1 Error detect command and status register (SRIO_P1EDCSR)

The port m error detect command and status register (P m EDCSR) indicates transmission errors that are detected by the hardware. Software can write bits in this register with 1 to cause the Error Rate Counter to increment. Undefined results occur if this register is written while actual physical layer errors are being detected by the port.

Address: C_0000h base + 640h offset = C_0640h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved								CCS	AUA	PNA	UA	CRC	EM	Reserv ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								NOA	PE	-	DE	UCS	LTO		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1EDCSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Received a control symbol with a bad CRC value.
10 AUA	Received acknowledge control symbol with unexpected ackID (packet-accepted or packet-retry).
11 PNA	Received packet-not-accepted acknowledge control symbol
12 UA	Received packet with unexpected ackID value.
13 CRC	Received a packet with a bad CRC value
14 EM	Received packet which exceed the maximum allowed size (276 bytes).
15–25 -	This field is reserved. Reserved
26 NOA	Link-response received with an ackID that is not outstanding.
27 PE	Protocol Error: An unexpected packet or control symbol was received.
28 -	Reserved
29 DE	Received unaligned /SC/ or /PD/ or undefined code-group.

Table continues on the next page...

SRIO_P1EDCSR field descriptions (continued)

Field	Description
30 UCS	An unexpected acknowledge control symbol was received.
31 LTO	An acknowledge or link-response control symbol is not received within the specified time-out interval.

19.5.36 Port 1 Error rate enable command and status register (SRIO_P1ERECSR)

The PmERECSR contains the bits that control when an error condition is allowed to increment the error rate counter in the port m error rate threshold register and lock the port m error capture registers.

Address: C_0000h base + 644h offset = C_0644h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								CCS	AUA	PNA	UA	CRC	EM	Reserved	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								NOA	PE	-	DE	UCS	LTO		
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ERECSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Enable error rate counting of a corrupt control symbol
10 AUA	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11 PNA	Enable error rate counting of received packet-not-accepted control symbols.
12 UA	Enable error rate counting of packet with unexpected ackID value.
13 CRC	Enable error rate counting of packet with a bad CRC value.
14 EM	Enable error rate counting of packet which exceeds the maximum allowed size
15–25 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1ERECSR field descriptions (continued)

Field	Description
26 NOA	Enable error rate counting of link-responses received with an ackID that is not outstanding.
27 PE	Enable error rate counting of protocol errors
28 -	Reserved
29 DE	Enable error rate counting of delineation errors.
30 UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31 LTO	Enable error rate counting of link time-out errors.

19.5.37 Port 1 Error capture attributes command and status register (SRIO_P1ECACSR)

The error capture attribute register indicates the type of information contained in the port m error capture registers. In the case of multiple detected errors during the same clock cycle one of the errors must be reflected in the Error type field. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 648h offset = C_0648h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	IT		Reserved		ET								ECI			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R													ECI			
W													Reserved			CVI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ECACSR field descriptions

Field	Description
0–1 IT	Type of information logged: 00 Packet (error capture registers hold the first 4 words of the packet, or the entire packet if it is less than 4 words long). 01 Control symbol (only error capture register 0 is valid) 10 Reserved 11 Undefined (not clearly a control symbol or packet error. Error capture registers hold the symbol that caused the error and the next 3 symbols.)

Table continues on the next page...

SRIO_P1ECACSR field descriptions (continued)

Field	Description
2 -	This field is reserved. Reserved
3–7 ET	The encoded value of the bit in the port m error detect CSR that describes the error captured in the port m error capture CSRs
8–23 ECI	Extended capture information [0:15]. ECI contains the control/data character signal corresponding to each byte of captured data.
24–30 -	This field is reserved. Reserved
31 CVI	This bit is set by hardware to indicate that the Packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.

19.5.38 Port 1 Packet/control symbol error capture command and status register 0 (SRIO_P1PCSECCSR0)

P m PCSECCSR0 contains the first four bytes of captured packet symbol information or a control character and control symbol. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 64Ch offset = C_064Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	C0															
W																																

Reset 0

SRIO_P1PCSECCSR0 field descriptions

Field	Description
0–31 C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

19.5.39 Port 1 Packet error capture command and status register 1 (SRIO_P1PECCSR1)

Error capture register 1 contains bytes 4 through 7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 650h offset = C_0650h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	C1															
W																																

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

SRIO_P1PECCSR1 field descriptions

Field	Description
0–31 C1	Capture 1. Bytes 4 to 7 of the packet header

19.5.40 Port 1 Packet error capture command and status register 2 (SRIO_P1PECCSR2)

Error capture register 2 contains bytes 8 through 11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 654h offset = C_0654h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	C2															
W																																

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

SRIO_P1PECCSR2 field descriptions

Field	Description
0–31 C2	Capture 2. Bytes 8 to 11 of the packet header

19.5.41 Port 1 Packet error capture command and status register 3 (SRIO_P1PECCSR3)

Error capture register 3 contains bytes 12 through 15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C 0000h base + 658h offset = C 0658h

SRI0 P1PECCSR3 field descriptions

Field	Description
0-31 C3	Capture 3. Bytes 12 to 15 of the packet header

19.5.42 Port 1 Error rate command and status register (SRIO_P1ERCSR)

The port m error rate register is a 32-bit register used with the port m error rate threshold register to monitor and control the reporting of transmission errors.

Address: C 0000h base + 668h offset = C 0668h

SRIO P1ERCSR field descriptions

Field	Description
0–7 ERB	<p>These bits provide the error rate bias value. Other values are reserved and cause undefined operation.</p> <ul style="list-style-type: none"> 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms ($\pm 34\%$) 0x02 Decrement every 10 ms ($\pm 34\%$) 0x04 Decrement every 100 ms ($\pm 34\%$) 0x08 Decrement every 1 s ($\pm 34\%$)

Table continues on the next page...

SRIO_P1ERCSR field descriptions (continued)

Field	Description
	0x10 Decrement every 10 s ($\pm 34\%$) 0x20 Decrement every 100 s ($\pm 34\%$) 0x40 Decrement every 1000 s ($\pm 34\%$) 0x80 Decrement every 10000 s ($\pm 34\%$)
8–13 -	This field is reserved. Reserved
14–15 ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. Note that the Error Rate Counter never increments above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger imply that it might. 00 Only count 2 errors above 01 Only count 4 errors above 10 Only count 16 error above 11 Do not limit incrementing the error rate count
16–23 PER	Peak error rate. Contains the peak value attained by the error rate counter
24–31 ERC	Error rate counter. These bits maintain a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate. Software should not attempt to write this field to a value higher than failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).

19.5.43 Port 1 Error rate threshold command and status register (SRIO_P1ERTCSR)

The port m error rate threshold register is a 32-bit register used to control the reporting of the link status to the system host.

Address: C_0000h base + 66Ch offset = C_066Ch

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

Reset 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SRIO_P1ERTCSR field descriptions

Field	Description
0–7 ERFTT	Error rate failed threshold trigger. These bits provide the threshold value for reporting an error condition due to a possibly broken link. The P m ESCSR[OFE] bit is not set if ERFTT is written to a value lower than or equal to the P m ERCSR[ERC]. See Table 19-324 for more details.

Table continues on the next page...

SRIO_P1ERTCSR field descriptions (continued)

Field	Description
	0x00 Disable the Error Rate Failed Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255.
8–15 ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. The P m ESCSR[ODE] bit is not set if ERDTT is written to a value lower than or equal to the P m ERCSR[ERC]. See Table 19-324 for more details. 0x00 Disable the Error Rate Degraded Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255.
16–31 -	This field is reserved. Reserved

19.5.44 Port 2 Error detect command and status register (SRIO_P2EDCSR)

The port m error detect command and status register (P m EDCSR) indicates transmission errors that are detected by the hardware. Software can write bits in this register with 1 to cause the Error Rate Counter to increment. Undefined results occur if this register is written while actual physical layer errors are being detected by the port.

Address: C_0000h base + 680h offset = C_0680h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								CCS	AUA	PNA	UA	CRC	EM	Reserved	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								NOA	PE	-	DE	UCS	LTO		
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2EDCSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Received a control symbol with a bad CRC value.
10 AUA	Received acknowledge control symbol with unexpected ackID (packet-accepted or packet-retry).
11 PNA	Received packet-not-accepted acknowledge control symbol
12 UA	Received packet with unexpected ackID value.
13 CRC	Received a packet with a bad CRC value
14 EM	Received packet which exceed the maximum allowed size (276 bytes).
15–25 -	This field is reserved. Reserved
26 NOA	Link-response received with an ackID that is not outstanding.
27 PE	Protocol Error: An unexpected packet or control symbol was received.
28 -	Reserved
29 DE	Received unaligned /SC/ or /PD/ or undefined code-group.
30 UCS	An unexpected acknowledge control symbol was received.
31 LTO	An acknowledge or link-response control symbol is not received within the specified time-out interval.

19.5.45 Port 2 Error rate enable command and status register (SRIO_P2ERECSR)

The P_m ERECSR contains the bits that control when an error condition is allowed to increment the error rate counter in the port *m* error rate threshold register and lock the port *m* error capture registers.

Address: C_0000h base + 684h offset = C_0684h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved								CCS	AUA	PNA	UA	CRC	EM	Reserv ed	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								NOA	PE	-	DE	UCS	LTO		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ERECSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Enable error rate counting of a corrupt control symbol
10 AUA	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11 PNA	Enable error rate counting of received packet-not-accepted control symbols.
12 UA	Enable error rate counting of packet with unexpected ackID value.
13 CRC	Enable error rate counting of packet with a bad CRC value.
14 EM	Enable error rate counting of packet which exceeds the maximum allowed size
15–25 -	This field is reserved. Reserved
26 NOA	Enable error rate counting of link-responses received with an ackID that is not outstanding.
27 PE	Enable error rate counting of protocol errors
28 -	Reserved
29 DE	Enable error rate counting of delineation errors.

Table continues on the next page...

SRIO_P2ERECSR field descriptions (continued)

Field	Description
30 UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31 LTO	Enable error rate counting of link time-out errors.

19.5.46 Port 2 Error capture attributes command and status register (SRIO_P2ECACSR)

The error capture attribute register indicates the type of information contained in the port m error capture registers. In the case of multiple detected errors during the same clock cycle one of the errors must be reflected in the Error type field. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 688h offset = C_0688h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	IT		Reserved		ET								ECI			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
				ECI									Reserved			CVI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRI0 P2ECACSR field descriptions

Field	Description
0–1 IT	<p>Type of information logged:</p> <ul style="list-style-type: none"> 00 Packet (error capture registers hold the first 4 words of the packet, or the entire packet if it is less than 4 words long). 01 Control symbol (only error capture register 0 is valid) 10 Reserved 11 Undefined (not clearly a control symbol or packet error. Error capture registers hold the symbol that caused the error and the next 3 symbols.)
2 - Reserved	This field is reserved. Reserved
3–7 ET	The encoded value of the bit in the port m error detect CSR that describes the error captured in the port m error capture CSRs
8–23 ECI	<p>Extended capture information [0:15].</p> <p>ECI contains the control/data character signal corresponding to each byte of captured data.</p>
24–30 - Reserved	This field is reserved. Reserved
31 CVI	This bit is set by hardware to indicate that the Packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.

19.5.47 Port 2 Packet/control symbol error capture command and status register 0 (SRIO_P2PCSECCSR0)

P_m PCSECCSR0 contains the first four bytes of captured packet symbol information or a control character and control symbol. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 68Ch offset = C_068Ch

SRIO P2PCSECCSR0 field descriptions

Field	Description
0-31 C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

19.5.48 Port 2 Packet error capture command and status register 1 (SRIO_P2PECCSR1)

Error capture register 1 contains bytes 4 through 7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 690h offset = C_0690h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	C1															
W																																

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

SRIO_P2PECCSR1 field descriptions

Field	Description
0–31 C1	Capture 1. Bytes 4 to 7 of the packet header

19.5.49 Port 2 Packet error capture command and status register 2 (SRIO_P2PECCSR2)

Error capture register 2 contains bytes 8 through 11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 694h offset = C_0694h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	C2															
W																																

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

SRIO_P2PECCSR2 field descriptions

Field	Description
0–31 C2	Capture 2. Bytes 8 to 11 of the packet header

19.5.50 Port 2 Packet error capture command and status register 3 (SRIO_P2PECCSR3)

Error capture register 3 contains bytes 12 through 15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P_m ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C 0000h base + 698h offset = C 0698h

SRI0 P2PECCSR3 field descriptions

Field	Description
0-31 C3	Capture 3. Bytes 12 to 15 of the packet header

19.5.51 Port 2 Error rate command and status register (SRIO_P2ERCSR)

The port m error rate register is a 32-bit register used with the port m error rate threshold register to monitor and control the reporting of transmission errors.

Address: C 0000h base + 6A8h offset = C 06A8h

SRIO P2ERCSR field descriptions

Field	Description
0–7 ERB	<p>These bits provide the error rate bias value. Other values are reserved and cause undefined operation.</p> <ul style="list-style-type: none"> 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms ($\pm 34\%$) 0x02 Decrement every 10 ms ($\pm 34\%$) 0x04 Decrement every 100 ms ($\pm 34\%$) 0x08 Decrement every 1 s ($\pm 34\%$)

Table continues on the next page...

SRIO_P2ERCSR field descriptions (continued)

Field	Description
	0x10 Decrement every 10 s ($\pm 34\%$) 0x20 Decrement every 100 s ($\pm 34\%$) 0x40 Decrement every 1000 s ($\pm 34\%$) 0x80 Decrement every 10000 s ($\pm 34\%$)
8–13 -	This field is reserved. Reserved
14–15 ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. Note that the Error Rate Counter never increments above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger imply that it might. 00 Only count 2 errors above 01 Only count 4 errors above 10 Only count 16 error above 11 Do not limit incrementing the error rate count
16–23 PER	Peak error rate. Contains the peak value attained by the error rate counter
24–31 ERC	Error rate counter. These bits maintain a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate. Software should not attempt to write this field to a value higher than failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).

19.5.52 Port 2 Error rate threshold command and status register (SRIO_P2ERTCSR)

The port m error rate threshold register is a 32-bit register used to control the reporting of the link status to the system host.

Address: C_0000h base + 6ACh offset = C_06ACh

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

Reset 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SRIO_P2ERTCSR field descriptions

Field	Description
0–7 ERFTT	Error rate failed threshold trigger. These bits provide the threshold value for reporting an error condition due to a possibly broken link. The P m ESCSR[OFE] bit is not set if ERFTT is written to a value lower than or equal to the P m ERCSR[ERC]. See Table 19-324 for more details.

Table continues on the next page...

SRIO_P2ERTCSR field descriptions (continued)

Field	Description
	0x00 Disable the Error Rate Failed Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255.
8–15 ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. The P _m ESCSR[ODE] bit is not set if ERDTT is written to a value lower than or equal to the P _m ERCSR[ERC]. See Physical layer RapidIO errors detected for more details. 0x00 Disable the Error Rate Degraded Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. ... 0xFF Set the error reporting threshold to 255.
16–31 -	This field is reserved. Reserved

19.5.53 Logical layer configuration register (SRIO_LLCR)

The logical layer configuration register contains general port-common logical layer mode enables.

Address: C_0000h base + 1_0004h offset = D_0004h

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

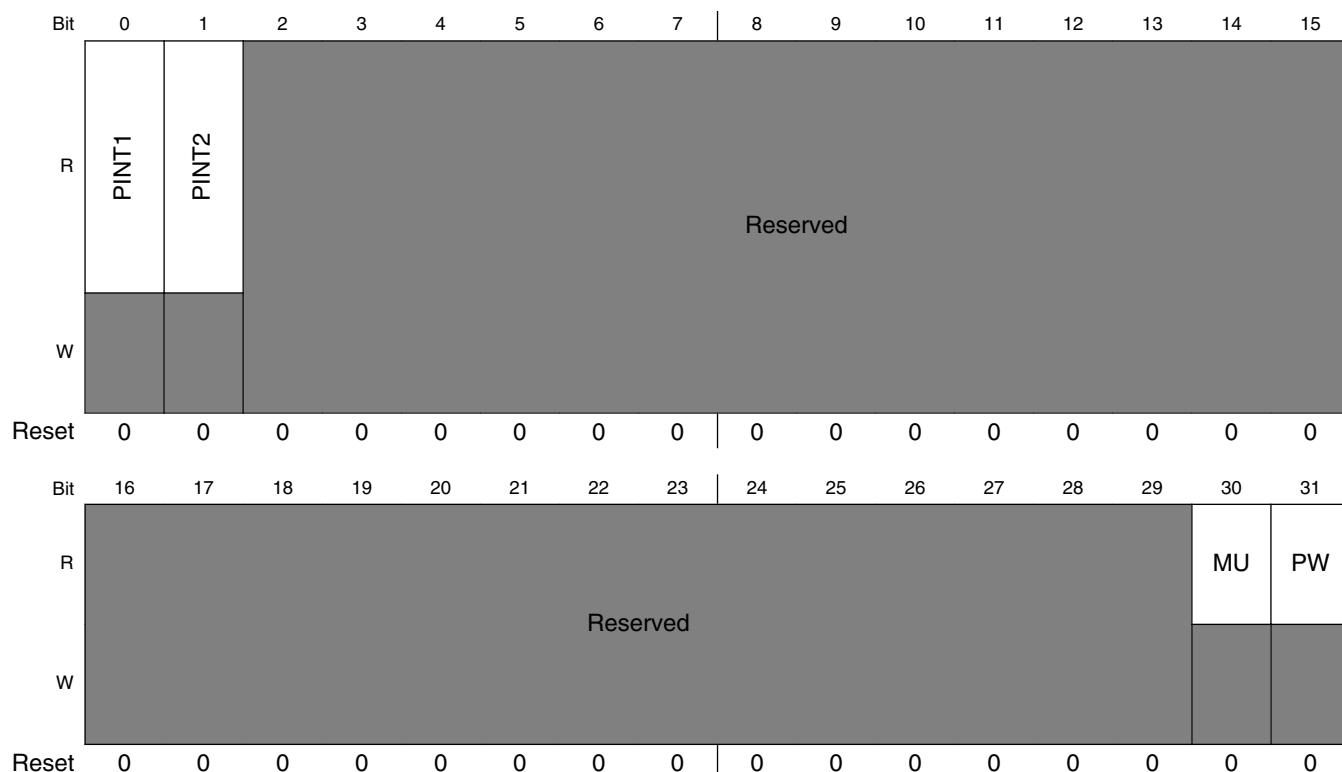
SRIO_LLCR field descriptions

Field	Description
0 Unused	This bit is unused. It is readable and writable.
1 DMAME	DMA message enable (for outbound DMA messages). 0 Message Unit messages can be assigned to letter 0,1,2,3. No DMA Messages. 1 DMA messages are assigned letter 3, Message Unit messages 0,1,2
2 ECRAB	External configuration register access block. When set, all maintenance requests and accesses that hit LCSBA1CSR are blocked; reads return all 0's, and writes are ignored (both return done response). When clear, any external RapidIO device can access registers.
3–31 -	This field is reserved. Reserved

19.5.54 Error / port-write interrupt status register (SRI0_EPWISR)

The EPWISR register contains status bits of the interrupts that have been generated by any port or the message unit for physical or logical/transport layer errors or inbound port-writes. Because errors from all ports are reported to the core with one interrupt signal, this register provides the core with quick access to where the error occurred. This register is read only and is stored in each port and the message unit as a logically equivalent copy.

Address: C_0000h base + 1_0010h offset = D_0010h



SRI0_EPWISR field descriptions

Field	Description
0 PINT1	A physical or logical/transport error interrupt was generated for port 1 . This bit is also set for outbound doorbell packet response time-out (PRT) errors.
1 PINT2	A physical or logical/transport error interrupt was generated for port 2. This bit is also set for outbound doorbell packet response time-out (PRT) errors.
2–29 -	This field is reserved. Reserved
30 MU	A logical/transport layer error interrupt was generated in the message unit.
31 PW	An inbound port-write was received.

19.5.55 Logical retry error threshold configuration register (SRIO_LRETCR)

The LRETCR register contains the retry error threshold for the logical layer. When the number of consecutive logical retries for a given packet is greater than this value, an error interrupt is generated. Note that the number of retries must be greater than this value unlike other registers that define a retry threshold.

Address: C_0000h base + 1_0020h offset = D_0020h

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

SRIO_LRETCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 RET	Retry error threshold. These bits provide the threshold value for the number of consecutive logical retries (for GSM responses) received for a given packet that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 ... 0xFF Set the error reporting threshold to 255

19.5.56 Physical retry error threshold configuration register (SRIO_PRETCR)

The PRETCR register contains the retry error threshold for the physical layer. When the number of consecutive ACK-retries is greater than or equal to this value, an error interrupt is generated.

Address: C_0000h base + 1_0080h offset = D_0080h

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

SRIO_PRETCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 RET	Retry error threshold. These bits provide the threshold value for the number of consecutive ACK retries received that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 ... 0xFF Set the error reporting threshold to 255

19.5.57 Port 1 Alternate device ID command and status register (SRIO_P1ADIDCSR)

The port m alternate device id CSR contains an alternate deviceID . This register should be used on a multi-port device to enable separate deviceIDs for each port. It is intended that this register should be enabled before the master enabled bit of the Pm GCCSR is set, such that when it is enabled, all other devices in the RapidIO system (including switches) send packets to and receive packets from the deviceID contained in this register, instead of the deviceID contained in BDIDCSR.

When the alternate deviceID is enabled, the inbound RapidIO endpoint only accepts packets sent with the deviceID contained in $P m$ ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint accepts packets using the same common transport system). In addition, the outbound RapidIO endpoint only generates requests using the deviceID contained in $P m$ ADIDCSR; it generates responses with the deviceID given in the original request packet (either from $P m$ ADIDCSR or BDIDCSR).

Address: C_0000h base + 1_0100h offset = D_0100h

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

SRIO_P1ADIDCSR field descriptions

Field	Description
0 ADE	Alternate device ID enable When set, this bit causes the port to use the deviceId specified in this register instead of the deviceId specified in BDIDCSR
1–7 -	This field is reserved. Reserved
8–15 ADID	Alternate device ID for the device in a small transport system
16–31 LADID	Alternate device ID for the device in a large transport system

19.5.58 Port 1 accept-all configuration register (SRIO_P1AACR)

The port m accept-all configuration register contains information on accept-all mode.

Address: C_0000h base + 1_0120h offset = D_0120h

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

SRIO_P1AACR field descriptions

Field	Description
0–30 -	This field is reserved. Reserved
31 AA	Accept all. 1 All packets are accepted without checking the target ID. However, the tt field must be consistent with the common transport system specified by bit 27 of the processing element features CAR (PEFCAR). 0 Normal RapidIO acceptance based on target ID.

19.5.59 Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR)

The port m logical outbound packet time-to-live configuration register contains the time-to-live count for all ports on a device. This packet time-to-live counter starts when a packet is ready to be transmitted. If the packet is not successfully transmitted before the timer expires, the packet is discarded. Successfully transmitted means that a packet accept was received for the packet on the RIO interface. If the packet requires a response, an internal error response is returned after the response time-out occurs (PRTOCCSR). The packet time-to-live counter prevents the local processor from being stalled when packets cannot be successfully transmitted (acknowledged with an accept by the link partner at the physical level). The value of this register should always be larger than the link time-out value (PLTOCCSR). When the packet time-to-live counter expires, P_m PCR[OBDEN] is automatically set. P_m PCR[OBDEN] must be cleared by software. By default, this time-out value is disabled (all zeros).

The resolution of this timer is $152/(\text{platform frequency})$. For example, at a platform frequency of 600 MHz, the maximum timeout value is $0xFF_FFFF * 152/600\text{MHz} = 4.25$ seconds.

When the packet time-to-live counter expires, P_m PCR[OBDEN] is automatically set. P_m PCR[OBDEN] must be cleared by software.

Address: C 0000h base + 1 0124h offset = D 0124h

SRI0 P1LOPTTLCR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
24–31 -	This field is reserved. Reserved

19.5.60 Port 1 Implementation error command and status register (SRIOP1IECSR)

The P_m IECSR register contains status bits that are asserted whenever an implementation-defined error occurs.

Address: C_0000h base + 1_0130h offset = D_0130h

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

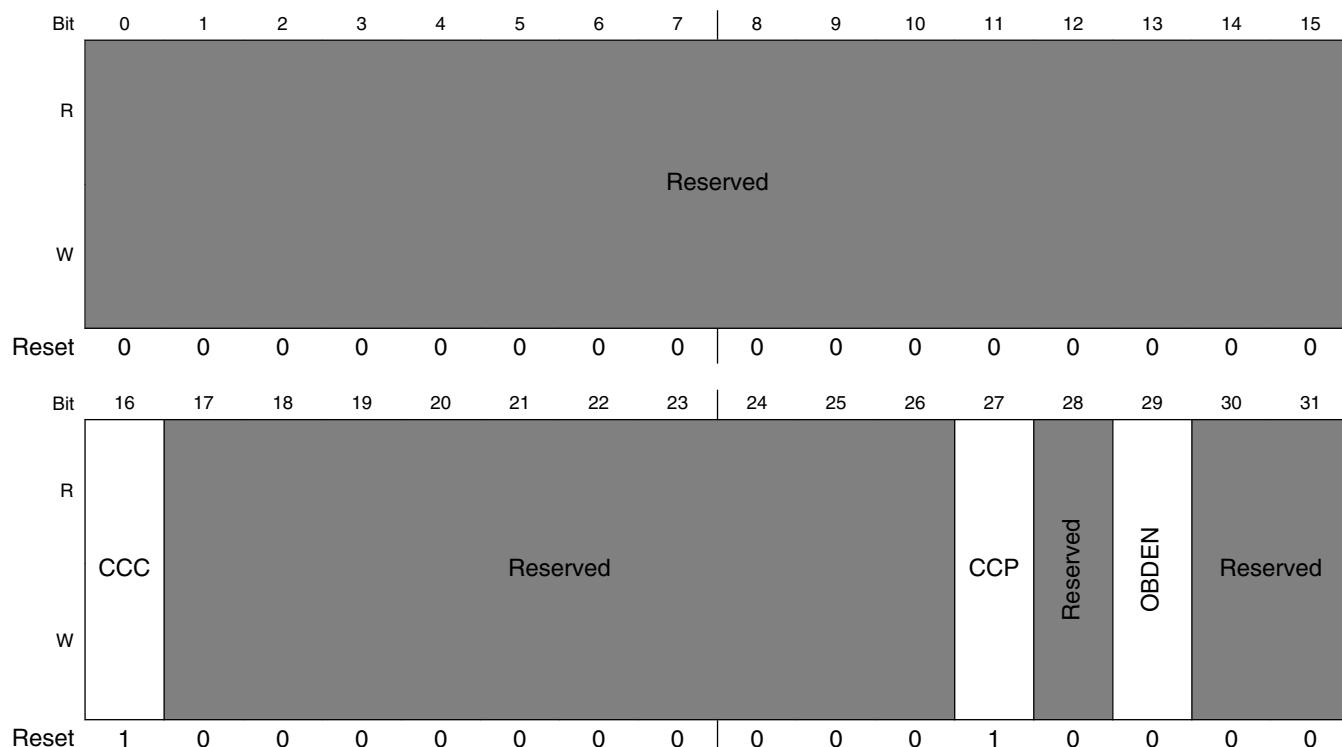
SRIOP1IECSR field descriptions

Field	Description
0 RETE	<p>Retry error threshold exceeded.</p> <p>This bit is asserted when the number of consecutive retries has reached retry error threshold in the retry error threshold register. This bit is cleared by writing a 1 to it.</p> <p>This bit sets again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.</p>
1–31 -	This field is reserved. Reserved

19.5.61 Port 1 Physical configuration register (SRIO_P1PCR)

The P_m PCR contains general physical layer protocol and link mode enables.

Address: C_0000h base + 1_0140h offset = D_0140h



SRIO_P1PCR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 CCC	CRC checking enable-control symbol. When set, CRC is checked on received control symbols. When cleared, no CRC is checked on received control symbols.
17–26 -	This field is reserved. Reserved
27 CCP	CRC checking enable-packet. When set, CRC is checked on received packets. When cleared, no CRC is checked on received packets
28 -	This field is reserved. Reserved
29 OBDEN	Output buffer drain enable.

Table continues on the next page...

SRIO_P1PCR field descriptions (continued)

Field	Description
	When set, the output drains packets from the outbound buffer and does not send them out. This intentionally causes the inbound to time-out (when a response on a drained request was expected) and send an error response to OCN. A packet time-to-live time-out causes this bit to be set (See Inbound port-write n queue base address register).
30–31 -	This field is reserved. Reserved

19.5.62 Port 1 Serial link command and status register (SRIO_P1SLCSR)

The P_m SLCSR contains status of the of the serial physical link.

Address: C_0000h base + 1_0158h offset = D_0158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LS0	LS1	LS2	LS3	Reserved			LA	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1SLCSR field descriptions

Field	Description
0 LS0	Lane sync achieved for lane 0. Write with 1 to clear
1 LS1	Lane sync achieved for lane 1. Write with 1 to clear
2 LS2	Lane sync achieved for lane 2. Write with 1 to clear
3 LS3	Lane sync achieved for lane 3. Write with 1 to clear.
4–7 -	This field is reserved. Reserved
8 LA	Lane alignment achieved. Write with 1 to clear.
9–31 -	This field is reserved. Reserved

19.5.63 Port 1 Serial link error injection configuration register (SRIO_P1SLEICR)

The PmSLEICR is used to control the injection of bit errors into the transmit bit stream.

The PmSLEICR register is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is any of the allowable non-zero values (as shown in the table above), then error injection is enabled for one lane or all four lanes, as selected by the EIC value. When enabled, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. This occurs only in the lane(s) that have error injection enabled. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field, as described above. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Address: C_0000h base + 1_0160h offset = D_0160h

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

SRIO_P1SLEICR field descriptions

Field	Description
0–4 EIC	Error injection control. Enables and controls serial link error injection as follows: All other values are reserved. 00000 Error injection is disabled. 10000 Error injection, lane 0 only 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all 4 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes
5–11 -	This field is reserved. Reserved
12–31 EIR	Error injection range. The value of EIR x 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 would indicate a maximum delay of 32 character times. Value within this register should be right-justified.

19.5.64 Port 2 Alternate device ID command and status register (SRIOP2ADIDCSR)

The port m alternate device id CSR contains an alternate deviceID. This register should be used on a multi-port device to enable separate deviceIDs for each port. It is intended that this register should be enabled before the master enabled bit of the Pm GCCSR is set, such that when it is enabled, all other devices in the RapidIO system (including switches) send packets to and receive packets from the deviceID contained in this register, instead of the deviceID contained in BDIDCSR.

When the alternate deviceID is enabled, the inbound RapidIO endpoint only accepts packets sent with the deviceID contained in Pm ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint accepts packets using the same common transport system). In addition, the outbound RapidIO endpoint only generates requests using the deviceID contained in Pm ADIDCSR; it generates responses with the deviceID given in the original request packet (either from Pm ADIDCSR or BDIDCSR).

Address: C_0000h base + 1_0180h offset = D_0180h

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

SRIOP2ADIDCSR field descriptions

Field	Description
0 ADE	Alternate device ID enable When set, this bit causes the port to use the deviceID specified in this register instead of the deviceid specified in BDIDCSR
1–7 -	This field is reserved. Reserved
8–15 ADID	Alternate device ID for the device in a small transport system
16–31 LADID	Alternate device ID for the device in a large transport system

19.5.65 Port 2 accept-all configuration register (SRIO_P2AACR)

The port m accept-all configuration register contains information on accept-all mode.

Address: C_0000h base + 1_01A0h offset = D_01A0h

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

SRIO_P2AACR field descriptions

Field	Description
0–30 -	This field is reserved. Reserved
31 AA	Accept all. 1 All packets are accepted without checking the target ID. However, the tt field must be consistent with the common transport system specified by bit 27 of the processing element features CAR (PEFCAR). 0 Normal RapidIO acceptance based on target ID.

19.5.66 Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR)

The port m logical outbound packet time-to-live configuration register contains the time-to-live count for all ports on a device. This packet time-to-live counter starts when a packet is ready to be transmitted. If the packet is not successfully transmitted before the timer expires, the packet is discarded. Successfully transmitted means that a packet accept was received for the packet on the RIO interface. If the packet requires a response, an internal error response is returned after the response time-out occurs (PRTOCCSR). The packet time-to-live counter prevents the local processor from being stalled when packets cannot be successfully transmitted (acknowledged with an accept by the link partner at the physical level). The value of this register should always be larger than the link time-out value (PLTOCCSR). When the packet time-to-live counter expires, P m PCR[OBDEN] is automatically set. P m PCR[OBDEN] must be cleared by software. By default, this time-out value is disabled (all zeros).

The resolution of this timer is 152/(platform frequency). For example, at a platform frequency of 600 MHz, the maximum timeout value is $0xFF_FFFF * 152/600MHz = 4.25$ seconds.

When the packet time-to-live counter expires, P_m PCR[OBDEN] is automatically set. P_m PCR[OBDEN] must be cleared by software.

Address: C_0000h base + 1_01A4h offset = D_01A4h

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

SRIO_P2LOPTTLCR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
24–31 -	This field is reserved. Reserved

19.5.67 Port 2 Implementation error command and status register (SRIO_P2IECSR)

The P_m IECSR register contains status bits that are asserted whenever an implementation-defined error occurs.

Address: C_0000h base + 1_01B0h offset = D_01B0h

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

SRIO_P2IECSR field descriptions

Field	Description
0 RETE	Retry error threshold exceeded. This bit is asserted when the number of consecutive retries has reached retry error threshold in the retry error threshold register. This bit is cleared by writing a 1 to it. This bit sets again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.

Table continues on the next page...

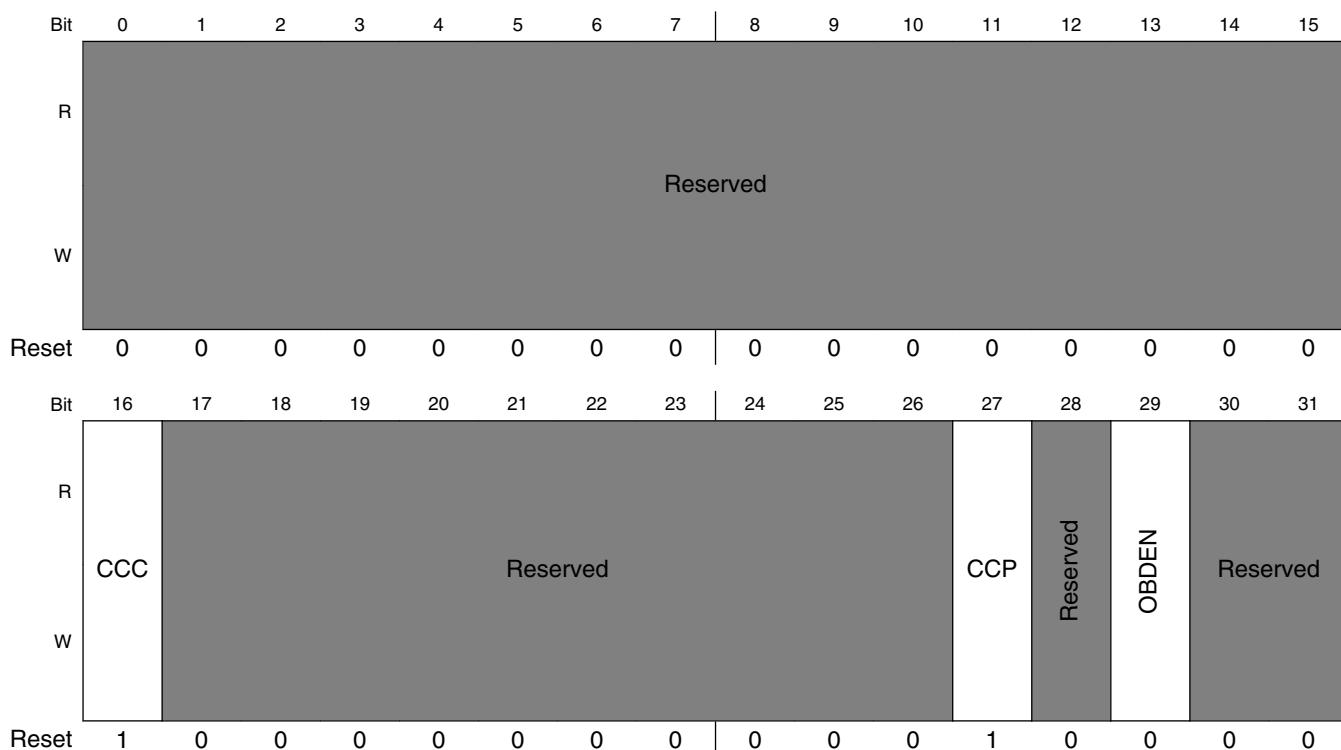
SRIO_P2IECSR field descriptions (continued)

Field	Description
1–31 -	This field is reserved. Reserved

19.5.68 Port 2 Physical configuration register (SRIO_P2PCR)

The P_m PCR contains general physical layer protocol and link mode enables.

Address: C_0000h base + 1_01C0h offset = D_01C0h

**SRIO_P2PCR field descriptions**

Field	Description
0–15 -	This field is reserved. Reserved
16 CCC	CRC checking enable-control symbol. When set, CRC is checked on received control symbols. When cleared, no CRC is checked on received control symbols.
17–26 -	This field is reserved. Reserved
27 CCP	CRC checking enable-packet.

Table continues on the next page...

SRIO_P2PCR field descriptions (continued)

Field	Description
	When set, CRC is checked on received packets. When cleared, no CRC is checked on received packets
28 -	This field is reserved. Reserved
29 OBDEN	Output buffer drain enable. When set, the output drains packets from the outbound buffer and does not send them out. This intentionally causes the inbound to time-out (when a response on a drained request was expected) and send an error response to OCN. A packet time-to-live time-out causes this bit to be set. (See Inbound port-write n queue base address register).
30–31 -	This field is reserved. Reserved

19.5.69 Port 2 Serial link command and status register (SRIO_P2SLCSR)

The P_m SLCSR contains status of the of the serial physical link.

Address: C_0000h base + 1_01D8h offset = D_01D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LS0	LS1	LS2	LS3	Reserved			LA	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2SLCSR field descriptions

Field	Description
0 LS0	Lane sync achieved for lane 0. Write with 1 to clear
1 LS1	Lane sync achieved for lane 1. Write with 1 to clear
2 LS2	Lane sync achieved for lane 2. Write with 1 to clear
3 LS3	Lane sync achieved for lane 3. Write with 1 to clear.
4–7 -	This field is reserved. Reserved
8 LA	Lane alignment achieved. Write with 1 to clear.

Table continues on the next page...

SRIO_P2SLCSR field descriptions (continued)

Field	Description
9–31 -	This field is reserved. Reserved

19.5.70 Port 2 Serial link error injection configuration register (SRIO_P2SLEICR)

The P_m SLEICR is used to control the injection of bit errors into the transmit bit stream.

The P_m SLEICR register is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is any of the allowable non-zero values (as shown in the table above), then error injection is enabled for one lane or all four lanes, as selected by the EIC value. When enabled, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. This occurs only in the lane(s) that have error injection enabled. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field, as described above. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Address: C_0000h base + 1_01E0h offset = D_01E0h

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

SRIO_P2SLEICR field descriptions

Field	Description
0–4 EIC	Error injection control. Enables and controls serial link error injection as follows: All other values are reserved. 00000 Error injection is disabled. 10000 Error injection, lane 0 only 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all 4 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes
5–11 -	This field is reserved. Reserved
12–31 EIR	Error injection range.

Table continues on the next page...

SRI0_P2SLEICR field descriptions (continued)

Field	Description
	The value of EIR x 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 would indicate a maximum delay of 32 character times. Value within this register should be right-justified.

19.5.71 IP Block Revision Register 1 (SRI0_IPBRR1)

IP block revision register 1 is used to track changes and revisions of the RapidIO endpoint. It is a read-only register.

Address: C_0000h base + 1_0BF8h offset = D_0BF8h

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

SRI0_IPBRR1 field descriptions

Field	Description
0–15 IPID	IP block ID = 0x01C0
16–23 IPMJ	Major revision of the IP block = 0x01
24–31 IPMN	Minor revision of the IP block = 0x 2

19.5.72 IP Block Revision Register 2 (SRI0_IPBRR2)

IP block revision register 2 is used to track changes and revisions of the RapidIO endpoint. It is a read-only register.

Address: C_0000h base + 1_0BFCh offset = D_0BFCh

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

SRIO_IPBRR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IPINT	IP block Integration options = 0x0
16–23 -	This field is reserved. Reserved
24–31 IPCFG	IP block Configuration options = 0x 1

19.5.73 Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR0)

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to OCN packets on inbound and to translate OCN packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4K and the largest window size is 16G for inbound translation and 64G for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the OCN address space has 36 bits. For outbound transactions, the default window maps each of the four 16G chunks to the RapidIO 16G address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

See [Segmented Outbound Window Description](#).

The port *m* RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Serial RapidIO Memory Map/Register Definition

Address: C_0000h base + 1_0C00h offset = D_0C00h

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

SRIOP1ROWTAR0 field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0-5 of the target ID are specified in the window's port <i>m</i> RapidIO outbound window translation extended address register.
2–11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8-9] is reserved.
12–31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. This field is reserved for default window 0.

19.5.74 Port 1 RapidIO outbound window translation extended address register 0 (SRIOP1ROWTEARn)

The port *m* RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0C04h offset + (32d × i), where i=0d to 8d

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

SRIO_P1ROWTEARn field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26–31 LTGTID	LTGTID correspond to bits 0–5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6–7 of the target ID are specified in the window's port <i>m</i> RapidIO outbound window translation address register.

19.5.75 Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWARn)

The port *m* RapidIO outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64G is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0C10h offset + (32d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
EN				Reserved		TFLOWLV	PCI		NSEG		NSSEG			RDTYP		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
WRTYP									Reserved					SIZE		
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

SRIO_P1ROWARn field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–3 -	This field is reserved. Reserved
4–5 TFLOWLVL	Transaction flow level. This field must be set to 00 if the PCI bit is set. Also, the RapidIO priority given by this field must always be greater than or equal to the OCN priority or a deadlock can occur. Normally, the OCN priority of all packets is 0 except for packets coming from a PCI type interface. Read type packets coming from a PCI type interface are priority 0 and write type packets are priority 1. Packets coming from a PCI type interface should set this field to 0 and set the PCI field to 1. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6 PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification The TFLOWLVL field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7 -	This field is reserved. Reserved
8–9 NSEG	Number of segments for this window. This field is reserved for default window 0. 00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved
10–11 NSSEG	Number of subsegments for this segment. This field is reserved for default window 0. Note that this field is valid only when ROWAR n [NSEG] contains a non-zero value (1 or 2). 00 One target deviceID for this segment 01 Two target deviceIDs for this segment 10 Four target deviceIDs for this segment 11 Eight target deviceIDs for this segment
12–15 RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved

Table continues on the next page...

SRIO_P1ROWARn field descriptions (continued)

Field	Description
	<p>...</p> <p>1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
16–19 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write.</p> <p>Write-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1111 Reserved</p>
20–25 -	This field is reserved. Reserved
26–31 SIZE	<p>Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes.</p> <p>This field is read only for default window 0.</p> <p>000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved ... 111111 Reserved</p>

19.5.76 Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTARn)

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to OCN packets on inbound and to translate OCN packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4K and the largest window size is 16G for inbound translation and 64G for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the OCN address space has 36 bits. For outbound transactions, the default window maps each of the four 16G chunks to the RapidIO 16G address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

See [Segmented Outbound Window Description](#).

The port m RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0C20h offset + (32d \times i), where i=0d to 7d

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

SRIO_P1ROWTARn field descriptions

Field	Description
0-1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set.

Table continues on the next page...

SRIO_P1ROWTARn field descriptions (continued)

Field	Description
	Bits 0-5 of the target ID are specified in the window's port <i>m</i> RapidIO outbound window translation extended address register.
2-11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8-9] is reserved.
12-31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. This field is reserved for default window 0.

19.5.77 Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBARn)

The port *m* RapidIO outbound window base address registers select the base address for the windows which are translated to an alternate system address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through the default register set. For transactions that cross more than one window, see [Outbound window boundary crossing errors](#) ."

Address: C_0000h base + 1_0C28h offset + (32d × i), where i=0d to 7d

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

SRIO_P1ROWBARn field descriptions

Field	Description
0-7 -	This field is reserved. Reserved
8-11 BEXTADD	Window base extended address. Corresponds to bits [0-3] of the 36-bit OCN base address.
12-31 BADD	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits [4-23] of the 36-bit OCN base address.

19.5.78 Port 1 RapidIO outbound window segment 1 register n (SRIOP1ROWS1Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C34h offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGT DID0_4				SGTGT DID5	SGTGT DID6	SGTGT DID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP1ROWS1Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P1ROWS1Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.79 Port 1 RapidIO outbound window segment 2 register n (SRIOP1ROWS2Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C38h offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGT DID0_4				SGTGT DID5	SGTGT DID6	SGTGT DID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP1ROWS2Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P1ROWS2Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.80 Port 1 RapidIO outbound window segment 3 register n (SRIOP1ROWS3Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C3Ch offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGT DID0_4				SGTGT DID5	SGTGT DID6	SGTGT DID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP1ROWS3Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P1ROWS3Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.81 Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTARn)

The port m RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0D60h offset + (32d \times i), where i=0d to 4d

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

SRIO_P1RIWTARn field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–11 TREXAD	Translation extended address. Corresponds to bits 0-3 of the 36-bit OCN translation address. TREXAD[2-3] are reserved for default window 0.
12–31 TRAD	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 4-23 of the 36-bit OCN translation address. TRAD is reserved for default window 0.

19.5.82 Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBArn)

The port m RapidIO inbound window n base address registers select the base address for the windows which are translated to an alternate target address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the transaction is forwarded to the interior of the chip using the default window. For transactions that cross more than one window, see [Outbound window boundary crossing errors](#). Note that the LCSBA1CSR register (See [Inbound port-write n queue base address register](#)) has priority over all ATMU windows if both are configured for the same address space.

Address: C_0000h base + 1_0D68h offset + (32d \times i), where i=0d to 3d

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

SRIO_P1RIWBArn field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10–11 BEXAD	Base extended address. BEXAD represents bits 0-1 of the 34-bit RapidIO address.
12–31 BADD	Base address. System address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits 2-21 of the 34-bit RapidIO base address.

19.5.83 Port 1 RapidIO inbound window attributes register n (SRIOP1RIWARn)

The port m RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, P m RIWAR n does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0D70h offset + (32d \times i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	PW	Reserved						TGINT			RDTYP				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRTYP			Reserved						SIZE						
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

SRIOP1RIWARn field descriptions

Field	Description																
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.																
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.																
2–7 -	This field is reserved. Reserved																
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. <table> <tr><td>0000</td><td>PCI Express 1</td></tr> <tr><td>0001</td><td>PCI Express 2</td></tr> <tr><td>0010</td><td>PCI Express 3</td></tr> <tr><td>0011-0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO port 1</td></tr> <tr><td>1001</td><td>Serial RapidIO port 2</td></tr> <tr><td>1010-1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1	0001	PCI Express 2	0010	PCI Express 3	0011-0111	Reserved	1000	Serial RapidIO port 1	1001	Serial RapidIO port 2	1010-1110	Reserved	1111	Local memory space
0000	PCI Express 1																
0001	PCI Express 2																
0010	PCI Express 3																
0011-0111	Reserved																
1000	Serial RapidIO port 1																
1001	Serial RapidIO port 2																
1010-1110	Reserved																
1111	Local memory space																
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ...																

Table continues on the next page...

SRIO_P1RIWARn field descriptions (continued)

Field	Description
	0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size

Table continues on the next page...

SRIOP1RIWARn field descriptions (continued)

Field	Description	
	011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved ... 111111 Reserved	

19.5.84 Port 1 RapidIO inbound window attributes register 0 (SRIOP1RIWAR0)

The port m RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, P m RIWAR n does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0DF0h offset = D_0DF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	PW	Reserved					TGINT	RDTYP							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRTYP				Reserved				SIZE							
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

SRIOP1RIWAR0 field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved

Table continues on the next page...

SRI0_P1RIWAR0 field descriptions (continued)

Field	Description																												
8–11 TGINT	<p>Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.</p> <table> <tr><td>0000</td><td>PCI Express 1</td></tr> <tr><td>0001</td><td>PCI Express 2</td></tr> <tr><td>0010</td><td>PCI Express 3</td></tr> <tr><td>0011–0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO port 1</td></tr> <tr><td>1001</td><td>Serial RapidIO port 2</td></tr> <tr><td>1010–1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1	0001	PCI Express 2	0010	PCI Express 3	0011–0111	Reserved	1000	Serial RapidIO port 1	1001	Serial RapidIO port 2	1010–1110	Reserved	1111	Local memory space												
0000	PCI Express 1																												
0001	PCI Express 2																												
0010	PCI Express 3																												
0011–0111	Reserved																												
1000	Serial RapidIO port 1																												
1001	Serial RapidIO port 2																												
1010–1110	Reserved																												
1111	Local memory space																												
12–15 RDTYP	<p>Transaction type to run on the I/O interface if access is a read.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>Transaction type to run on local memory if access is a read.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read, do not snoop local processor</td></tr> <tr><td>0101</td><td>Read, snoop local processor</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table>	0000	Reserved	...		0100	Read	0101	Reserved	...		1111	Reserved	0000	Reserved	...		0100	Read, do not snoop local processor	0101	Read, snoop local processor	0110	Reserved	0111	Reserved	...		1111	Reserved
0000	Reserved																												
...																													
0100	Read																												
0101	Reserved																												
...																													
1111	Reserved																												
0000	Reserved																												
...																													
0100	Read, do not snoop local processor																												
0101	Read, snoop local processor																												
0110	Reserved																												
0111	Reserved																												
...																													
1111	Reserved																												
16–19 WRTYP	<p>Transaction type to run on I/O interface if access is a write.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>write</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>Transaction type to run on local memory if access is a write.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0011</td><td>Reserved</td></tr> <tr><td>0100</td><td>Write, do not snoop local processor</td></tr> <tr><td>0101</td><td>Write, snoop local processor</td></tr> </table>	0000	Reserved	...		0100	write	0101	Reserved	...		1111	Reserved	0000	Reserved	...		0011	Reserved	0100	Write, do not snoop local processor	0101	Write, snoop local processor						
0000	Reserved																												
...																													
0100	write																												
0101	Reserved																												
...																													
1111	Reserved																												
0000	Reserved																												
...																													
0011	Reserved																												
0100	Write, do not snoop local processor																												
0101	Write, snoop local processor																												

Table continues on the next page...

SRIO_P1RIWAR0 field descriptions (continued)

Field	Description
	0110 Reserved ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved ... 111111 Reserved

19.5.85 Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR0)

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to OCN packets on inbound and to translate OCN packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4K and the largest window size is 16G for inbound translation and 64G for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the OCN address space has 36 bits. For outbound transactions, the default window maps each of the four 16G chunks to the RapidIO 16G address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

See [Segmented Outbound Window Description](#).

The port *m* RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0E00h offset = D_0E00h

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

SRIOP2ROWTAR0 field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0-5 of the target ID are specified in the window's port <i>m</i> RapidIO outbound window translation extended address register.
2–11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8-9] is reserved.
12–31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. This field is reserved for default window 0.

19.5.86 Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEARn)

The port m RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0E04h offset + (32d × i), where i=0d to 8d

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

SRIO_P2ROWTEARn field descriptions

Field	Description
0-25 -	This field is reserved. Reserved
26-31 LTGTID	LTGTID correspond to bits 0-5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6-7 of the target ID are specified in the window's port m RapidIO outbound window translation address register.

19.5.87 Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWARn)

The port m RapidIO outbound window attributes registers define the window sizes to translate and other attributes for the translations. 64G is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0E10h offset + (32d \times i), where i=0d to 8d

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

SRIO_P2ROWARn field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–3 -	This field is reserved. Reserved
4–5 TFLQLV	Transaction flow level. This field must be set to 00 if the PCI bit is set. Also, the RapidIO priority given by this field must always be greater than or equal to the OCN priority or a deadlock can occur. Normally, the OCN priority of all packets is 0 except for packets coming from a PCI type interface. Read type packets coming from a PCI type interface are priority 0 and write type packets are priority 1. Packets coming from a PCI type interface should set this field to 0 and set the PCI field to 1.

Table continues on the next page...

SRIO_P2ROWARn field descriptions (continued)

Field	Description
	<p>00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved</p>
6 PCI	<p>PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification</p> <p>The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.</p>
7 -	This field is reserved. Reserved
8–9 NSEG	<p>Number of segments for this window.</p> <p>This field is reserved for default window 0.</p> <p>00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved</p>
10–11 NSSEG	<p>Number of subsegments for this segment.</p> <p>This field is reserved for default window 0.</p> <p>Note that this field is valid only when ROWAR n [NSEG] contains a non-zero value (1 or 2).</p> <p>00 One target deviceID for this segment 01 Two target deviceIDs for this segment 10 Four target deviceIDs for this segment 11 Eight target deviceIDs for this segment</p>
12–15 RDTYP	<p>Transaction type to run on RapidIO interface if access is a read.</p> <p>0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
16–19 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write.</p> <p>Write-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p>

Table continues on the next page...

SRIO_P2ROWARn field descriptions (continued)

Field	Description
	0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 Reserved ... 111111 Reserved

19.5.88 Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTARn)

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to OCN packets on inbound and to translate OCN packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4K and the largest window size is 16G for inbound translation and 64G for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the OCN address space has 36 bits. For outbound transactions, the default window maps each of the four 16G chunks to the RapidIO 16G address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

See [Segmented Outbound Window Description](#).

The port m RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0E20h offset + (32d \times i), where i=0d to 7d

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

SRIO_P2ROWTARn field descriptions

Field	Description
0-1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set.

Table continues on the next page...

SRIO_P2ROWTARn field descriptions (continued)

Field	Description
	Bits 0-5 of the target ID are specified in the window's port <i>m</i> RapidIO outbound window translation extended address register.
2-11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions and default window 0, TREXAD[8-9] is reserved.
12-31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. This field is reserved for default window 0.

19.5.89 Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBARn)

The port *m* RapidIO outbound window base address registers select the base address for the windows which are translated to an alternate system address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through the default register set. For transactions that cross more than one window, see Section 15.9.6.2, "Window Boundary Crossing Errors ."

Address: C_0000h base + 1_0E28h offset + (32d × i), where i=0d to 7d

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

SRIO_P2ROWBARn field descriptions

Field	Description
0-7 -	This field is reserved. Reserved
8-11 BEXTADD	Window base extended address. Corresponds to bits [0-3] of the 36-bit OCN base address.
12-31 BADD	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits [4-23] of the 36-bit OCN base address.

19.5.90 Port 2 RapidIO outbound window segment 1 register n (SRIOP2ROWS1Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E34h offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGT DID0_4				SGTGT DID5	SGTGT DID6	SGTGT DID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP2ROWS1Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P2ROWS1Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.91 Port 2 RapidIO outbound window segment 2 register n (SRIOP2ROWS2Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E38h offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGT DID0_4				SGTGT DID5	SGTGT DID6	SGTGT DID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP2ROWS2Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P2ROWS2Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.92 Port 2 RapidIO outbound window segment 3 register n (SRIOP2ROWS3Rn)

The port m RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E3Ch offset + (32d \times i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				TFLOWLV	Reserved		RDTYP				WRTYP				
W	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGTID0_4				SGTGTID5	SGTGTID6	SGTGTID7	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIOP2ROWS3Rn field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P2ROWS3Rn field descriptions (continued)

Field	Description
	<p>0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear</p>
12–15 WRTYP	<p>Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from OCN must generate a write-requiring-response to RapidIO. Therefore, if an OCN write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.</p> <p>0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved</p>
16–23 -	This field is reserved. Reserved
24–28 SGTGTID0_4	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTID5	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTID6	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTID7	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

19.5.93 Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTARn)

The port m RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0F60h offset + (32d \times i), where i=0d to 4d

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

Reset 0

SRIO_P2RIWTARn field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–11 TREXAD	Translation extended address. Corresponds to bits 0-3 of the 36-bit OCN translation address. TREXAD[2-3] are reserved for default window 0.
12–31 TRAD	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 4-23 of the 36-bit OCN translation address. TRAD is reserved for default window 0.

19.5.94 Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBArn)

The port m RapidIO inbound window n base address registers select the base address for the windows which are translated to an alternate target address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the transaction is forwarded to the interior of the chip using the default window. For transactions that cross more than one window, see Section 15.9.7.2, "Window Boundary Crossing Errors ." Note that the LCSBA1CSR register (See [Inbound port-write n queue base address register](#)) has priority over all ATMU windows if both are configured for the same address space.

Address: C_0000h base + 1_0F68h offset + (32d \times i), where i=0d to 3d

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

SRIO_P2RIWBArn field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10–11 BEXAD	Base extended address. BEXAD represents bits 0-1 of the 34-bit RapidIO address.
12–31 BADD	Base address. System address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits 2-21 of the 34-bit RapidIO base address.

19.5.95 Port 2 RapidIO inbound window attributes register n (SRIOP2RIWARn)

The port m RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, P m RIWAR n does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0F70h offset + (32d \times i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	PW	Reserved						TGINT			RDTYP				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRTYP			Reserved						SIZE						
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

SRIOP2RIWARn field descriptions

Field	Description																
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.																
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.																
2–7 -	This field is reserved. Reserved																
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. <table> <tr><td>0000</td><td>PCI Express 1</td></tr> <tr><td>0001</td><td>PCI Express 2</td></tr> <tr><td>0010</td><td>PCI Express 3</td></tr> <tr><td>0011-0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO port 1</td></tr> <tr><td>1001</td><td>Serial RapidIO port 2</td></tr> <tr><td>1010-1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1	0001	PCI Express 2	0010	PCI Express 3	0011-0111	Reserved	1000	Serial RapidIO port 1	1001	Serial RapidIO port 2	1010-1110	Reserved	1111	Local memory space
0000	PCI Express 1																
0001	PCI Express 2																
0010	PCI Express 3																
0011-0111	Reserved																
1000	Serial RapidIO port 1																
1001	Serial RapidIO port 2																
1010-1110	Reserved																
1111	Local memory space																
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ...																

Table continues on the next page...

SRIO_P2RIWARn field descriptions (continued)

Field	Description
	0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size

Table continues on the next page...

SRIOP2RIWARn field descriptions (continued)

Field	Description	
	011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved ... 111111 Reserved	

19.5.96 Port 2 RapidIO inbound window attributes register 0 (SRIOP2RIWAR0)

The port m RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, P m RIWAR n does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0FF0h offset = D_0FF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	PW	Reserved					TGINT	RDTYP							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRTYP				Reserved				SIZE							
W																
Reset	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

SRIOP2RIWAR0 field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIOP2RIWAR0 field descriptions (continued)

Field	Description																												
8–11 TGINT	<p>Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target.</p> <table> <tr><td>0000</td><td>PCI Express 1</td></tr> <tr><td>0001</td><td>PCI Express 2</td></tr> <tr><td>0010</td><td>PCI Express 3</td></tr> <tr><td>0011–0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Serial RapidIO port 1</td></tr> <tr><td>1001</td><td>Serial RapidIO port 2</td></tr> <tr><td>1010–1110</td><td>Reserved</td></tr> <tr><td>1111</td><td>Local memory space</td></tr> </table>	0000	PCI Express 1	0001	PCI Express 2	0010	PCI Express 3	0011–0111	Reserved	1000	Serial RapidIO port 1	1001	Serial RapidIO port 2	1010–1110	Reserved	1111	Local memory space												
0000	PCI Express 1																												
0001	PCI Express 2																												
0010	PCI Express 3																												
0011–0111	Reserved																												
1000	Serial RapidIO port 1																												
1001	Serial RapidIO port 2																												
1010–1110	Reserved																												
1111	Local memory space																												
12–15 RDTYP	<p>Transaction type to run on the I/O interface if access is a read.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>Transaction type to run on local memory if access is a read.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>Read, do not snoop local processor</td></tr> <tr><td>0101</td><td>Read, snoop local processor</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table>	0000	Reserved	...		0100	Read	0101	Reserved	...		1111	Reserved	0000	Reserved	...		0100	Read, do not snoop local processor	0101	Read, snoop local processor	0110	Reserved	0111	Reserved	...		1111	Reserved
0000	Reserved																												
...																													
0100	Read																												
0101	Reserved																												
...																													
1111	Reserved																												
0000	Reserved																												
...																													
0100	Read, do not snoop local processor																												
0101	Read, snoop local processor																												
0110	Reserved																												
0111	Reserved																												
...																													
1111	Reserved																												
16–19 WRTYP	<p>Transaction type to run on I/O interface if access is a write.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0100</td><td>write</td></tr> <tr><td>0101</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table> <p>Transaction type to run on local memory if access is a write.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0011</td><td>Reserved</td></tr> <tr><td>0100</td><td>Write, do not snoop local processor</td></tr> <tr><td>0101</td><td>Write, snoop local processor</td></tr> </table>	0000	Reserved	...		0100	write	0101	Reserved	...		1111	Reserved	0000	Reserved	...		0011	Reserved	0100	Write, do not snoop local processor	0101	Write, snoop local processor						
0000	Reserved																												
...																													
0100	write																												
0101	Reserved																												
...																													
1111	Reserved																												
0000	Reserved																												
...																													
0011	Reserved																												
0100	Write, do not snoop local processor																												
0101	Write, snoop local processor																												

Table continues on the next page...

SRIOP2RIWAR0 field descriptions (continued)

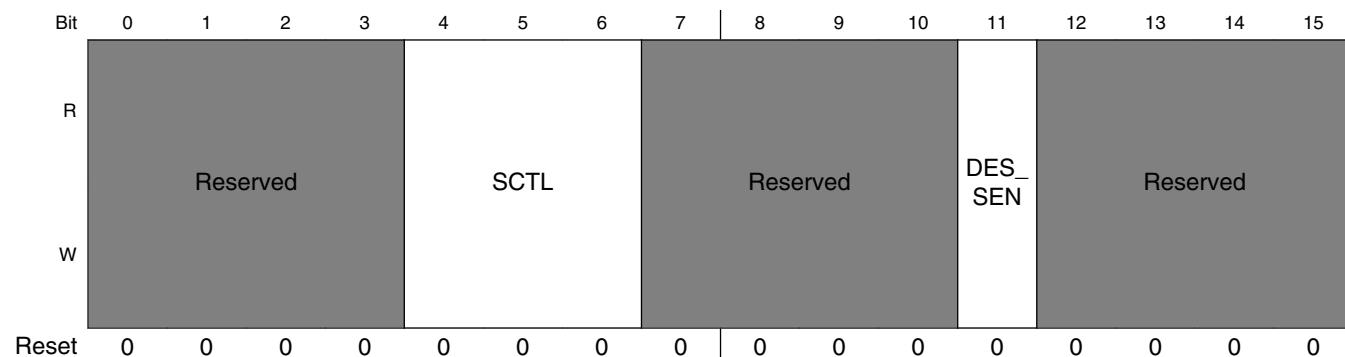
Field	Description
	0110 Reserved ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved ... 111111 Reserved

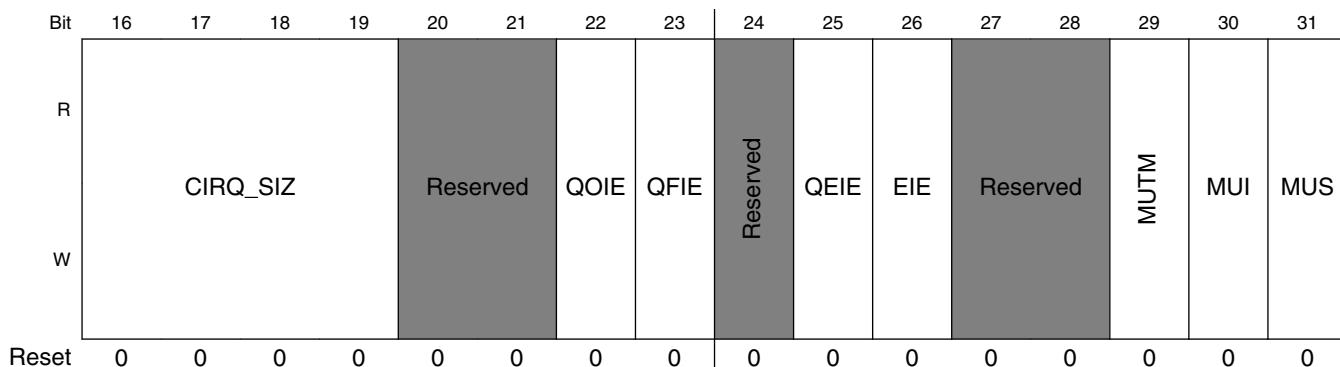
19.5.97 Outbound message n mode register (SRIOMnMR)

RapidIO Outbound Message 0 Registers control RapidIO outbound messages. The following provide partial descriptions of these registers.

The outbound message mode register allows software to start a message operation and to control various message operation characteristics.

Address: C_0000h base + 1_3000h offset + (256d × i), where i=0d to 1d





SRIO_OMnMR field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–6 SCTL	Service control. Determines the number of descriptors to process before servicing the next queue. Note that if one queue has SCTL set to fixed priority, all SCTL values in other queues are ignored. For example, of the two outbound message units, if unit 1's service control value is set to 0b000, fixed priority results with unit 0 the highest priority results. If a queue is in direct mode only, one message operation is serviced before servicing the next queue. For proper operation, this field should only be modified when the outbound message controller is not enabled. The value of this field cannot be changed unless both units are disabled. 000 Fixed priority based on outbound message unit number 001 1 descriptors 010 2 descriptors 011 4 descriptors 100 8 descriptors 101 16 descriptors 110 32 descriptors 111 64 descriptors
7–10 -	This field is reserved. Reserved
11 DES_SEN	Descriptor snoop enable. When set enables snooping of the local processor when reading descriptors from memory. For proper operation, this field should only be modified when the outbound message controller is not enabled
12–15 -	This field is reserved. Reserved
16–19 CIRQ_SIZ	Circular descriptor queue size. Determines the number of descriptors that can be placed on the circular queue without overflow. For proper operation, this field should only be modified when the outbound message controller is not enabled 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64

Table continues on the next page...

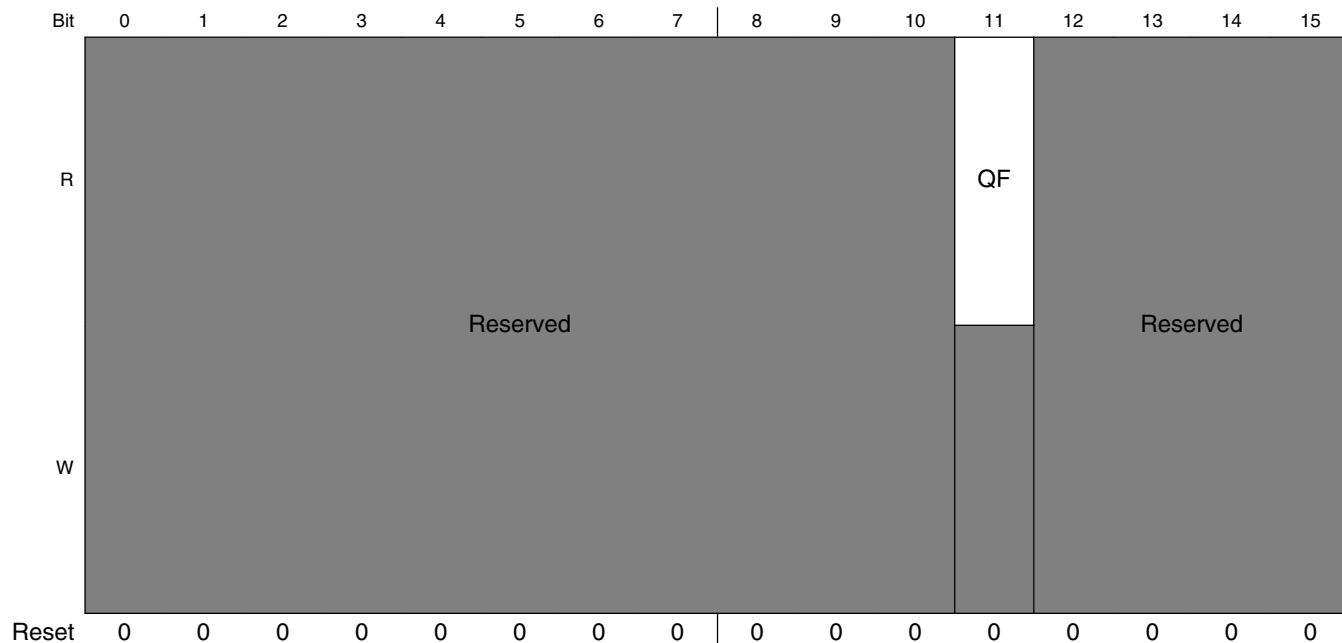
SRIO_OMnMR field descriptions (continued)

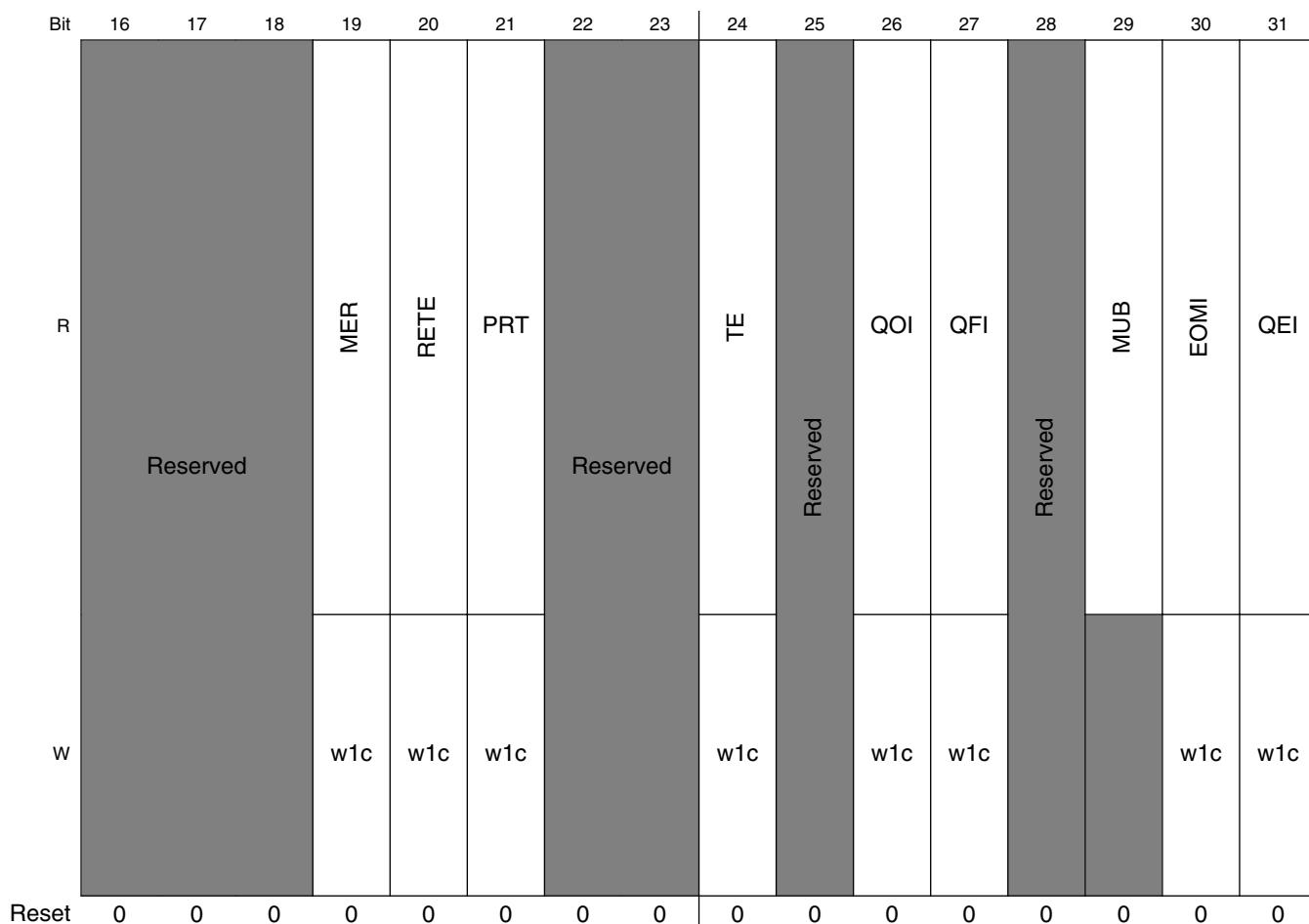
Field	Description
	0110 128 0111 256 1000 512 1001 1024 1010 2048 1011 Reserved ... 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved
20–21 -	This field is reserved. Reserved
22 QOIE	Queue overflow interrupt enable. When set, bit generates an interrupt on detection of a queue overflow (that is, the enqueue and dequeue pointers are no longer equal after being incremented by the processor and the queue was full). No queue overflow interrupt is generated if this bit is cleared. (only applicable to chaining mode). If this bit is not set and the queue overflows, the result is undefined.
23 QFIE	Queue full interrupt enable. When set will generate an interrupt when the queue transitions to full (that is, the enqueue and dequeue pointers are equal after being incremented by the processor). No QF interrupt is generated if this bit is cleared. If this bit is set and OM n SR[QF] is a 1, OM n SR[QFI] will assert.
24 -	This field is reserved. Reserved
25 QEIE	Queue empty interrupt enable. When set, bit generates an interrupt at the completion of all outstanding message operations (that is, the enqueue and dequeue pointers are equal after an increment by the message unit controller). No QE interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the outbound message controller is not enabled
26 EIE	Error interrupt enable. When set, bit generates the port-write/error interrupt when a transfer error (OM n SR[TE]), a message error response (OM n SR[MER]), a packet response time-out (OM n SR[PRT]), or a retry threshold event exceeded (OM n SR[RETE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–28 -	This field is reserved. Reserved
29 MUTM	Message unit transfer mode. Setting this bit puts the message unit into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the message transmission. Clearing this bit configures the message unit in chaining mode.
30 MUI	Message unit increment. Software sets this bit after writing a descriptor to memory. Hardware then increments the OM n DQEPCR and clear this bit. Always reads as 0 when MUS is set.
31 MUS	Message unit start. Direct mode-A 0 to 1 transition when the message unit is not busy (MUB bit is 0) starts the message unit. A 1 to 0 transition will have no effect. Chaining mode-If this bit is set, the message unit starts whenever the enqueue and dequeue pointers are not equal.

19.5.98 Outbound message n status register (SRI0_0MnSR)

The outbound message status register reports various message unit conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

Address: C_0000h base + 1_3004h offset + (256d × i), where i=0d to 1d





SRIO_OMnSR field descriptions

Field	Description
0–10 -	This field is reserved. Reserved
11 QF	Queue full-If the queue becomes full, then this bit is set. (Read-only)
12–18 -	This field is reserved. Reserved
19 MER	Message error response. This bit is set when an ERROR response is received from the message target. The Error response received field indicates value of the error response status bits when an error response is received. This bit is cleared by writing a 1.
20 RETE	Retry error threshold exceeded. This bit is set when the message unit has been unable to complete a message operation because the retry error threshold value has been exceeded due to RapidIO retry response. This bit is cleared by writing a 1.
21 PRT	Packet response time-out. This bit is set when the message unit has been unable to complete a message operation and a packet response time-out occurred. This bit is cleared by writing a 1.
22–23 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_OMnSR field descriptions (continued)

Field	Description
24 TE	Transaction error. This bit is set when an internal error condition occurs during the message operation. This bit is cleared by writing a 1. For proper operation, this field should only be modified when the outbound message controller is not enabled
25 -	This field is reserved. Reserved
26 QOI	Queue overflow interrupt. This bit is set when a queue overflow condition is detected. This bit is cleared by writing a 1. (only applicable to chaining mode)
27 QFI	Queue full interrupt. If the queue becomes full, if the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28 -	This field is reserved. Reserved
29 MUB	Message unit busy. When set indicates that a message operation is currently in progress. This bit is cleared as a result of an error or the message operation is finished. (Read-only)
30 EOMI	End-Of-Message interrupt. After finishing this message operation, if the EOMIE bit in the Destination Attributes Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
31 QEI	Queue Empty Interrupt. When the last message operation in the outbound descriptor queue is finished, if the QEIE bit in the Mode Register is set, then this bit is set and an interrupt is generated. Otherwise, no interrupt is generated. This bit is cleared by writing a 1.

19.5.99 Extended outbound message n descriptor queue dequeue pointer address register (SRIO_EOMnDQDPAR)

Used with OM n DQDPAR; see description in [Outbound message n descriptor queue dequeue pointer address register](#).

Address: C_0000h base + 1_3008h offset + (256d × i), where i=0d to 1d

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

Reset 0

SRIO_EOMnDQDPAR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EDQDPA	Extended descriptor queue dequeue pointer address bits. These are the highest order address bits. For proper operation, this field should only be modified when the outbound message controller is not enabled.

19.5.100 Outbound message n descriptor queue dequeue pointer address register (SRIO_OMnDQDPAR)

The outbound message descriptor queue dequeue pointer address registers contain the address of the first descriptor in memory to be processed. Software must initialize these registers to point to the first descriptor in memory. After processing this descriptor, the message unit controller increments the outbound message descriptor queue dequeue pointer address in OM n DQDPAR and EOM n DQDPAR to point to the next descriptor. If the outbound message descriptor queue enqueue pointer and the outbound message descriptor queue dequeue pointer are not equal (indicating that the queue is not empty), the message unit controller reads the next descriptor from memory for processing. If the enqueue and dequeue pointers are equal after the dequeue pointer has been incremented by the message unit controller, the queue is empty and the message unit halts until the enqueue pointer is incremented by the processor. Incrementing the pointer indicates that a new descriptor has been added to the queue and is ready for processing. If the queue becomes empty and OM n MR[QEIE] is set, OM n SR[QEI] is set and an interrupt is generated.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to the number of queue entries \times 32 bytes (size of each queue descriptor). For example, if there are eight entries in the queue, the queue must be 256-byte aligned.

The number of queue entries is set in OM n MR[CIRQ_SIZ]; see [Outbound message n mode register](#).

Address: C_0000h base + 1_300Ch offset + (256d \times i), where i=0d to 1d

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

SRIO_OMnDQDPAR field descriptions

Field	Description
0–26 DQDPA	Descriptor queue dequeue pointer address. Contains the address of the first descriptor in memory to process. Each descriptor must be size aligned to a 32 byte boundary as each descriptor is 32 bytes in size. For proper operation, this field should only be modified when the outbound message controller is not enabled. When writing to this register, it is not necessary to shift the address, since the lower 5 bits are merely ignored to ensure 32 byte alignment.
27–31 -	This field is reserved. Reserved

19.5.101 Extended outbound message n source address register (SRIO_EOMnSAR)

Address: C_0000h base + 1_3010h offset + (256d × i), where i=0d to 1d

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

Reset 0

SRIO_EOMnSAR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 ESAD	Extended source address bits. These are the highest order address bits. For proper operation, this field should only be modified when the outbound message controller is not enabled

19.5.102 Outbound message n source address register (SRIO_OMnSAR)

The outbound message unit source address registers indicate the address from which the message unit controller is to read data. Software must ensure that this is a valid local memory address. The source address must be aligned to a double-word boundary, so the least significant three bits are reserved.

Address: C_0000h base + 1_3014h offset + (256d × i), where i=0d to 1d

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

SAD

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

SAD

SNEN

Reserved

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SRIO_OMnSAR field descriptions

Field	Description
0–28 SAD	Source address. This is the source address of the message operation. The contents are updated after every memory read operation. For proper operation, this field should only be modified when the outbound message controller is not enabled
29 SNEN	Snoop enable. This bit enables snooping of the local processor caches for the data reads from local memory. For proper operation, this field should only be modified when the outbound message controller is not enabled
30–31 -	This field is reserved. Reserved

19.5.103 Outbound message n destination port register (SRIO_OMnDPR)

The destination port register indicates the RapidIO destination ID and mailbox to which the message unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

Address: C_0000h base + 1_3018h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EDTGTROUTE								DTGTROUTE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								XMAILBOX				MAILBOX			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

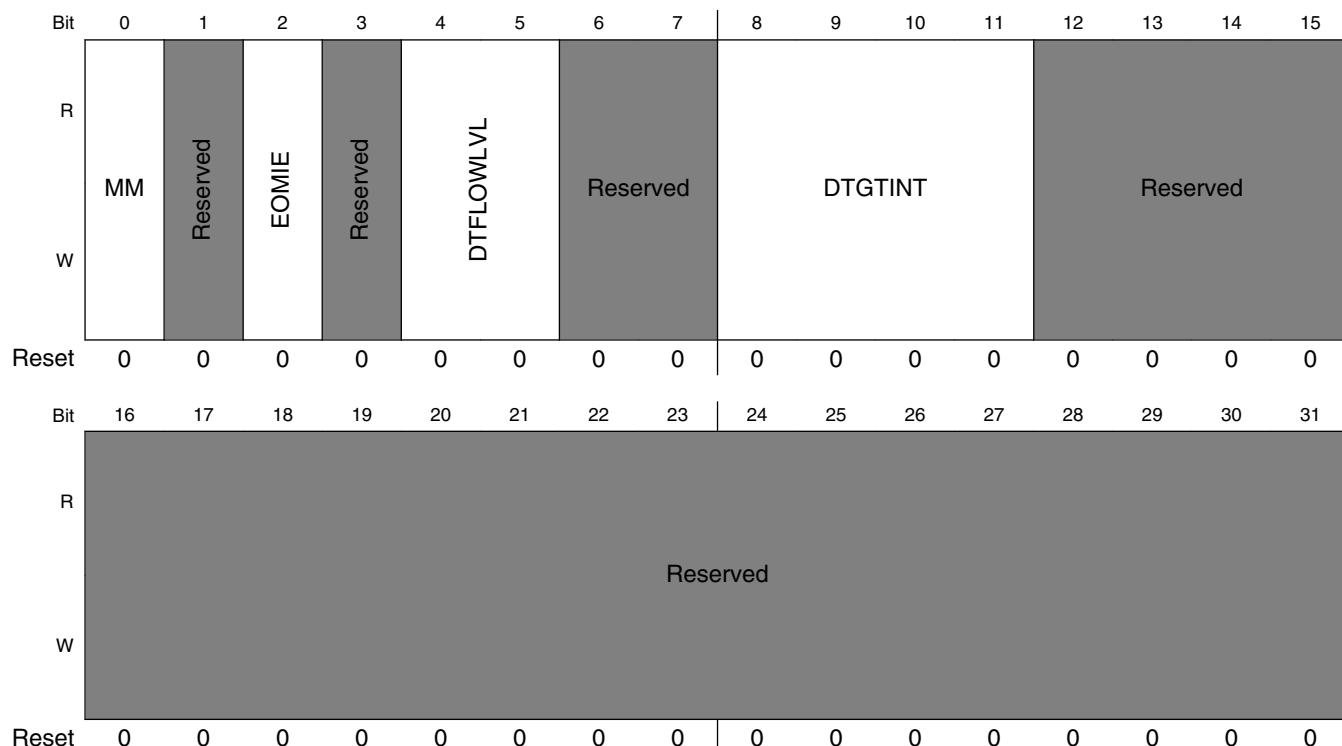
SRIO_OMnDPR field descriptions

Field	Description
0–7 EDTGTROUTE	Extended destination target route. Most significant byte of a 16-bit target route when operating in large transport mode. Reserved when operating in small transport mode. For proper operation, this field should only be modified when the outbound message controller is not enabled
8–15 DTGTROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target). This value is overridden by the multicast group and list if multicast mode is enabled. On error, if multicast mode is enabled, this field is loaded with the destination of the failed operation. For proper operation, this field should only be modified when the outbound message controller is not enabled
16–25 -	This field is reserved. Reserved
26–29 XMAILBOX	Value for 'xmb' field in MESSAGE packet. This field is only used when OM n DATR[MM] is set. For proper operation, this field should only be modified when the outbound message controller is not enabled
30–31 MAILBOX	Value for 'mbox' field in MESSAGE packet. For proper operation, this field should only be modified when the outbound message controller is not enabled

19.5.104 Outbound message n destination attributes Register (SRIO_OMnDATR)

The outbound message destination attributes register contains the transaction attributes to be used for the message operation.

Address: C_0000h base + 1_301Ch offset + (256d × i), where i=0d to 1d



SRIO_OMnDATR field descriptions

Field	Description
0 MM	Multicast mode. When set, the message operation is sent to all of the targets indicated by the multicast group and list. Messages are limited to one segment and 256 bytes or less when this mode is enabled.
1 -	This field is reserved. Reserved
2 EOMIE	End-of-Message interrupt enable. When set, generates an interrupt when the current message operation is finished. For proper operation, this field should only be modified when the outbound message controller is not enabled.
3 -	This field is reserved. Reserved
4–5 DTFLOWLVL	Transaction flow level. For proper operation, this field should only be modified when the outbound message controller is not enabled

Table continues on the next page...

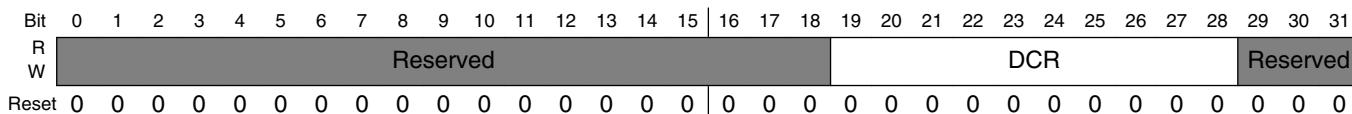
SRI0_OMnDATR field descriptions (continued)

Field	Description
	00 SRIO transaction priority of 1 01 SRIO transaction priority of 1 10 SRIO transaction priority of 2 11 Reserved
6-7 -	This field is reserved. Reserved
8-11 DTGTINT	Target interface. The value of this field should be set according to the following: 0000 RapidIO Port 1 0001 RapidIO Port 2
12-31 -	This field is reserved. Reserved

19.5.105 Outbound message n double-word count register (SRIO OM n DCR)

The outbound message double-word count register contains the number of double-words for the message operation. The maximum message operation size is 4 Kbytes and the minimum is 8 bytes.

Address: C_0000h base + 1_3020h offset + (256d × i), where i=0d to 1d



SRIO OMnDCR field descriptions

Field	Description
0–18 -	This field is reserved. Reserved
19–28 DCR	<p>Transfer count register. Contains the number of bytes for the message operation.</p> <p>All other values yield undefined behavior.</p> <p>For proper operation, this field should only be modified when the outbound message controller is not enabled.</p> <ul style="list-style-type: none"> 00 0000 0000 Reserved 00 0000 0001 8 bytes 00 0000 0010 16 bytes 00 0000 0100 32 bytes 00 0000 1000 64 bytes 00 0001 0000 128 bytes 00 0010 0000 256 bytes

Table continues on the next page...

SRIO_OMnDCR field descriptions (continued)

Field	Description
	00 0100 0000 512 bytes (multi segment mode only) 00 1000 0000 1024 bytes (multi segment mode only) 01 0000 0000 2048 bytes (multi segment mode only) 10 0000 0000 4096 bytes (multi segment mode only)
29–31 -	This field is reserved. Reserved

19.5.106 Extended outbound message n descriptor queue enqueue pointer address register (SRIO_EOMnDQEPA)

Used with OM n DQEPA; see description in [Outbound message n descriptor queue enqueue pointer address register](#).

Address: C_0000h base + 1_3024h offset + (256d \times i), where i=0d to 1d

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

SRIO_EOMnDQEPA field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EDQEPA	Extended descriptor queue enqueue pointer address. These are the highest-order address bits.

19.5.107 Outbound message n descriptor queue enqueue pointer address register (SRIO_OMnDQEPA)

The outbound message descriptor queue enqueue pointer address registers contain the address for the next descriptor in memory to be added to the queue. Software must initialize these registers to match the outbound message descriptor queue dequeue pointer address. When a message is ready to be sent, the processor writes a descriptor to the next location in the queue (indicated by the address in OM n DQEPA and EOM n DQEPA), and then either writes the OM n DQEPA to point to the next descriptor location in memory or sets OM n MR[MUI]. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full. If the OM n MR[QFIE] bit is set, then the OM n SR[QFI] bit is set, and an interrupt is generated.

- If the enqueue and dequeue pointer no longer match after the enqueue pointer has been incremented and the queue is full, then the queue overflows, and the message unit stops. If OM_n MR[QOIE] is set, OM_n SR[QOI] is set and an interrupt is generated. OM_n MR[MUS] must transition to a 0 to clear this error condition. If the enqueue pointer is directly written, the queue overflow condition is not detected.
- If the enqueue and dequeue pointers were the same before reading the register, the message unit controller will start (if enabled).

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to the number of queue entries x 32 bytes (size of each queue descriptor). For example, if there are eight entries in the queue, the queue must be 256-byte aligned.

The number of queue entries is set in OM_n MR[CIRQ_SIZ]; see [Outbound message n mode register](#).

Address: C_0000h base + 1_3028h offset + (256d x i), where i=0d to 1d

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

SRI0_OMnDQEPA field descriptions

Field	Description
0–26 DQEPA	Descriptor queue enqueue pointer address. Contains the address of the last descriptor in memory to process. The descriptor must be size aligned to a 32 byte boundary. When writing to this register it is not necessary to shift the address, since the lower 5 bits are merely ignored to ensure 32 byte alignment .
27–31 -	This field is reserved. Reserved

19.5.108 Outbound message n retry error threshold configuration register (SRI0_OMnRETCR)

The outbound message retry error threshold configuration register controls the number of times that the message unit attempts to transfer a message segment to a particular destination before reporting an error. A message segment is retransmitted if a RETRY response is received from the target.

Address: C_0000h base + 1_302Ch offset + (256d x i), where i=0d to 1d

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

SRI0 OMnRETCR field descriptions

Field	Description
0-23 -	This field is reserved. Reserved
24-31 RET	<p>Retry error threshold. This value is the number of times that the message unit attempts to transmit a message segment to a particular target.</p> <p>For proper operation, this field should only be modified when the outbound message controller is not enabled</p> <ul style="list-style-type: none"> 0x00 Disabled 0x01 Message segment transmitted only 1 time 0x02 Message segment transmitted up to 2 times ... 0xFF Message segment transmitted up to 255 times

19.5.109 Outbound message n multicast group register (SRIO OMnMGR)

The multicast group register contains a multicast group (MG) value and extended multicast group number (EMG) which, in combination with the multicast list register and the multicast enable (OM n DATR[MM]), indicates which deviceIDs are targets of a multicast operation. The multicast group represents the most significant three bits (bits[0-2]) of the RapidIO deviceIDs that are destinations of the message operation, whereas the multicast list indicates a list of targets within that group. Each individual set bit, when encoded, determines the least significant five bits of the RapidIO deviceIDs (bits[3-7]) that are targets of the message operation. Therefore, multicast group 0 (MG = 0) contains target deviceIDs (0,1,...,31), multicast group 1 (MG = 1) contains target deviceIDs (32,33,...,63), and so on.

If in large transport mode, the extended multicast group represents the eight most significant bits (bits[0-7]), the multicast group represents the next three bits (bits[8-10]) and the multicast list indicates a list of targets within that group.

If multicast is enabled, this information in the multicast group and mask register is used to determine the target of the message operation instead of the DTGTRoute and EDTGTRoute fields in the outbound message n destination attributes register.

Address: C 0000h base + 1 3030h offset + (256d × i), where i=0d to 1d

SRIOMnMGR field descriptions

Field	Description
0–20 -	This field is reserved. Reserved
21–28 EMG	Extended multicast group. This is the most significant eight bits of the target deviceIDs for the multicast operation when operating in large transport mode. For proper operation, this field should only be modified when the outbound message controller is not enabled
29–31 MG	Multicast group. This is the most significant three bits of the target deviceIDs for the multicast operation. For proper operation, this field should only be modified when the outbound message controller is not enabled

19.5.110 Outbound message n multicast list register (SRIO_OMnMLR)

See the multicast group register description for more information.

Address: C_0000h base + 1_3034h offset + (256d × i), where i=0d to 1d

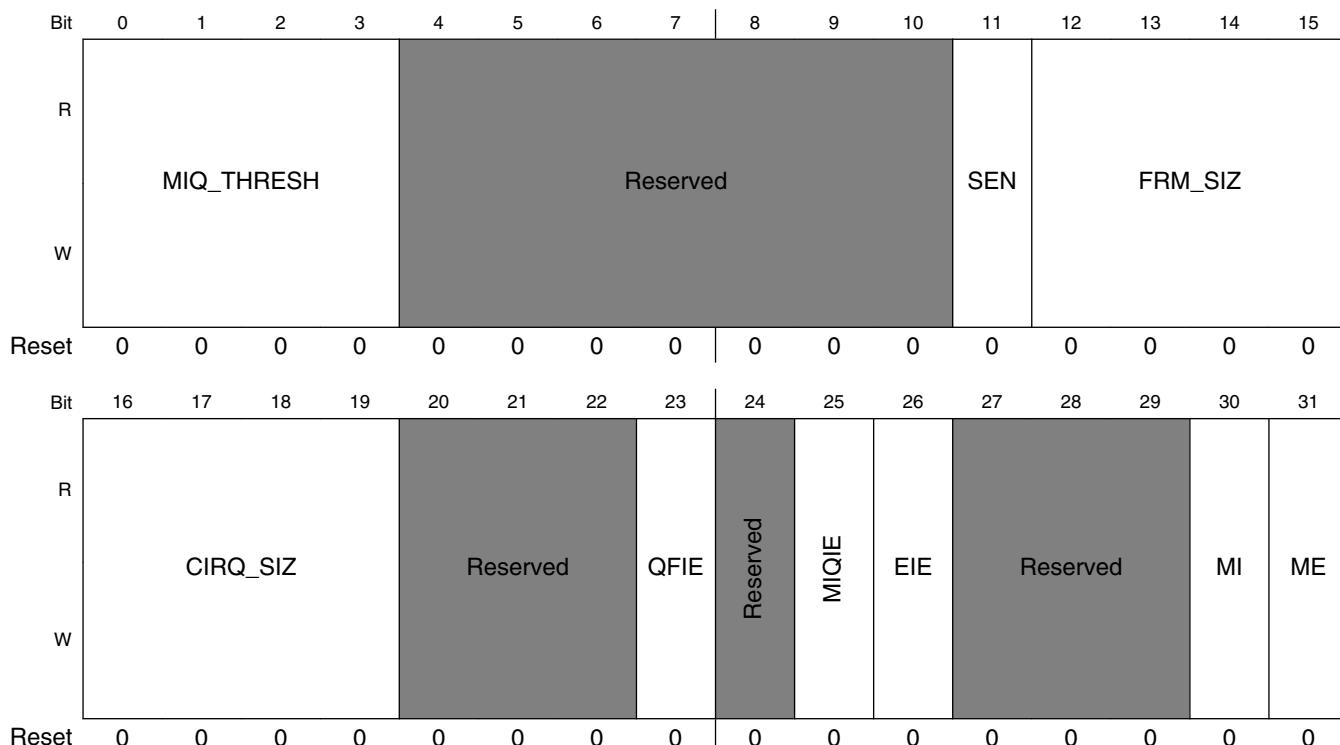
SRI0 OMnMLR field descriptions

Field	Description
0-31 ML	Multicast list. This is the group target list for the message operation. Depending on the value of the multicast group value, bit 0 corresponds to devicID 0, 32, 64, 96, and so on; bit 1 corresponds to devicID 1, 33, 65, 97, and so on. If none of the bits are set, bit 0 is assumed to be set. For proper operation, this field should only be modified when the outbound message controller is not enabled.

19.5.111 Inbound message n mode register (SRIO_IMnMR)

The inbound message mode register allows software to enable the mailbox controller and to control various message operation characteristics.

Address: C_0000h base + 1_3060h offset + (256d × i), where i=0d to 1d



SRIO_IMnMR field descriptions

Field	Description																		
0–3 MIQ_THRESH	<p>Message in queue threshold. Determines the number of message frames to be accumulated in the frame queue before Message In Queue is signaled. Undefined operation results if the message in queue threshold is set greater than or equal to the message queue size (IM n MR[CIRQ_SIZ]).</p> <p>For proper operation, this field should only be modified when the inbound message controller is not enabled.</p> <table> <tbody> <tr><td>0000</td><td>1</td></tr> <tr><td>0001</td><td>2</td></tr> <tr><td>0010</td><td>4</td></tr> <tr><td>0011</td><td>8</td></tr> <tr><td>0100</td><td>16</td></tr> <tr><td>0101</td><td>32</td></tr> <tr><td>0110</td><td>64</td></tr> <tr><td>0111</td><td>128</td></tr> <tr><td>1000</td><td>256</td></tr> </tbody> </table>	0000	1	0001	2	0010	4	0011	8	0100	16	0101	32	0110	64	0111	128	1000	256
0000	1																		
0001	2																		
0010	4																		
0011	8																		
0100	16																		
0101	32																		
0110	64																		
0111	128																		
1000	256																		

Table continues on the next page...

SRIO_IMnMR field descriptions (continued)

Field	Description																														
	1001 512 1010 1024 1011 Reserved ... 1111 Reserved																														
4–10 -	This field is reserved. Reserved																														
11 SEN	Snoop enable. When set, enables snooping the local processor when writing messages into memory. For proper operation, this field should only be modified when the inbound message controller is not enabled.																														
12–15 FRM_SIZ	<p>Message frame size. Determines the maximum message size that can be accepted by this mailbox without error. This parameter combined with CIRQ_SIZ determine the maximum contiguous memory space allocated to the mailbox.</p> <p>For proper operation, this field should only be modified when the inbound message controller is not enabled.</p> <table> <tr><td>0000</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>0010</td><td>8 bytes</td></tr> <tr><td>0011</td><td>16 bytes</td></tr> <tr><td>0100</td><td>32 bytes</td></tr> <tr><td>0101</td><td>64 bytes</td></tr> <tr><td>0110</td><td>128 bytes</td></tr> <tr><td>0111</td><td>256 bytes</td></tr> <tr><td>1000</td><td>512 bytes</td></tr> <tr><td>1001</td><td>1024 bytes</td></tr> <tr><td>1010</td><td>2048 bytes</td></tr> <tr><td>1011</td><td>4096 bytes</td></tr> <tr><td>1100</td><td>Reserved</td></tr> <tr><td>...</td><td></td></tr> <tr><td>1111</td><td>Reserved</td></tr> </table>	0000	Reserved	...		0010	8 bytes	0011	16 bytes	0100	32 bytes	0101	64 bytes	0110	128 bytes	0111	256 bytes	1000	512 bytes	1001	1024 bytes	1010	2048 bytes	1011	4096 bytes	1100	Reserved	...		1111	Reserved
0000	Reserved																														
...																															
0010	8 bytes																														
0011	16 bytes																														
0100	32 bytes																														
0101	64 bytes																														
0110	128 bytes																														
0111	256 bytes																														
1000	512 bytes																														
1001	1024 bytes																														
1010	2048 bytes																														
1011	4096 bytes																														
1100	Reserved																														
...																															
1111	Reserved																														
16–19 CIRQ_SIZ	<p>Circular frame queue size. Determines the number of message frames that can be place on the circular queue without overflow. This parameter combined with FRM_SIZ determine the maximum contiguous memory space allocated to the mailbox.</p> <p>For proper operation, this field should only be modified when the inbound message controller is not enabled.</p> <table> <tr><td>0000</td><td>2</td></tr> <tr><td>0001</td><td>4</td></tr> <tr><td>0010</td><td>8</td></tr> <tr><td>0011</td><td>16</td></tr> <tr><td>0100</td><td>32</td></tr> <tr><td>0101</td><td>64</td></tr> <tr><td>0110</td><td>128</td></tr> <tr><td>0111</td><td>256</td></tr> <tr><td>1000</td><td>512</td></tr> <tr><td>1001</td><td>1024</td></tr> <tr><td>1010</td><td>2048</td></tr> </table>	0000	2	0001	4	0010	8	0011	16	0100	32	0101	64	0110	128	0111	256	1000	512	1001	1024	1010	2048								
0000	2																														
0001	4																														
0010	8																														
0011	16																														
0100	32																														
0101	64																														
0110	128																														
0111	256																														
1000	512																														
1001	1024																														
1010	2048																														

Table continues on the next page...

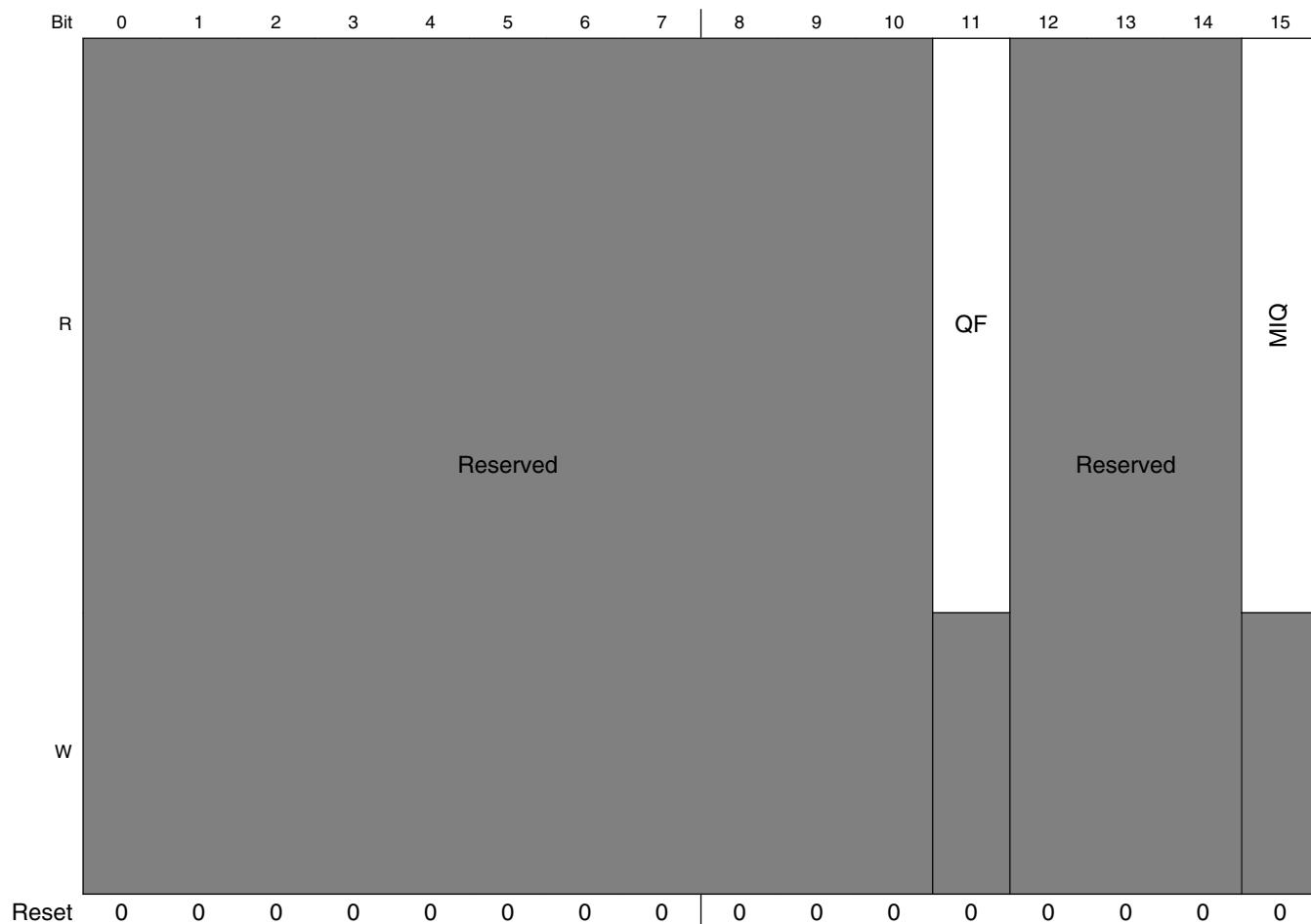
SRIO_IMnMR field descriptions (continued)

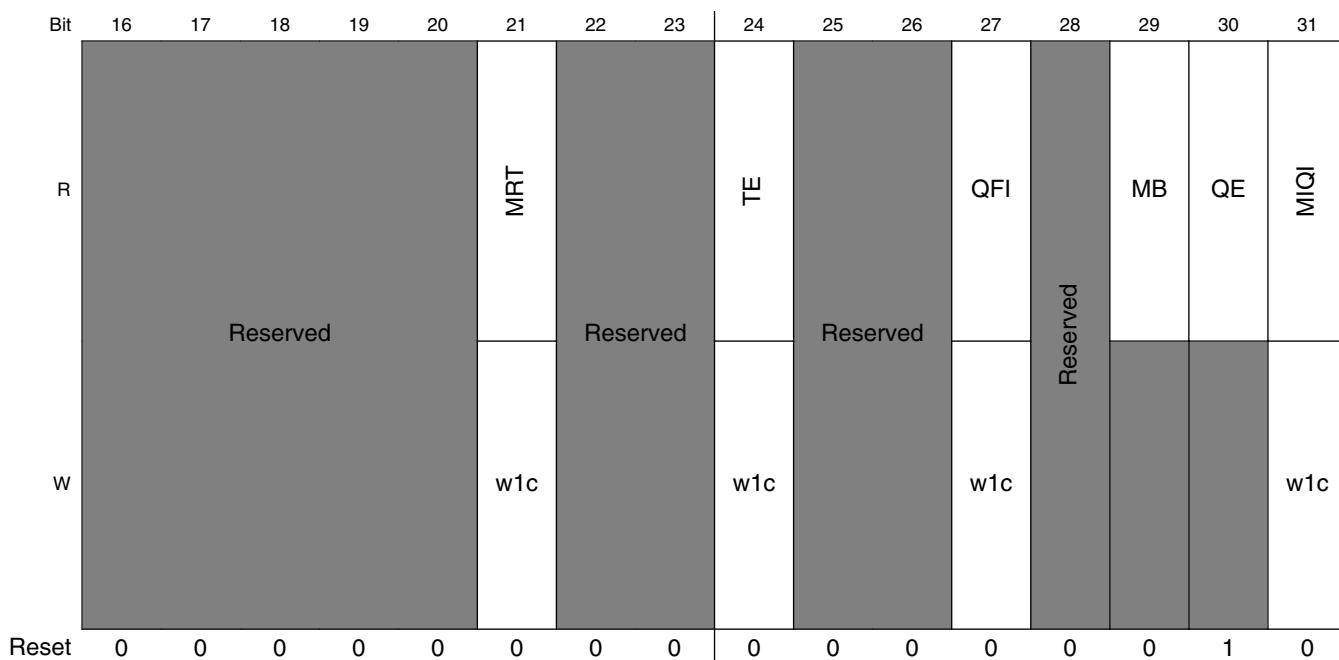
Field	Description
	1011 Reserved ... 1111 Reserved
20–22 -	This field is reserved. Reserved
23 QFIE	Queue full interrupt enable. When set, generates an interrupt when the queue is full (that is, the enqueue and dequeue pointers are equal after the dequeue pointer was incremented by the mailbox controller). No QFI interrupt is generated if this bit is cleared. If this bit is set and IM n SR[QF] is a 1, IM n SR[QFI] asserts.
24 -	This field is reserved. Reserved
25 MIQIE	Message in queue interrupt enable. When set, generates an interrupt when the queue has accumulated the number of messages specified by the IM n MR[MIQ_THRESH]. No MIQ interrupt is generated if this bit is cleared. If this bit is set and IM n SR[MIQ] is a 1, IM n SR[MIQI] asserts. If this bit is set and IM n MR[MI] is also set simultaneously, IM n SR[MIQI] indicates reflects the value of MIQ after the increment.
26 EIE	Error interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IM n SR[TE]) or a message request time-out (IM n SR[MRT]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29 -	This field is reserved. Reserved
30 MI	Mailbox increment. Software sets this bit after processing an inbound message. Hardware increments the IM n FQDPAR and clears this bit. Always reads as 0.
31 ME	Mailbox enable. If this bit is set, the mailbox is initialized and can service incoming message operations. If this bit is cleared after the first segment of a multi-segment message has arrived, a message request time-out results (IM n SR[MRT]) and the busy bit (IM n SR[MB]) clears as long as the port response timer value (PRTCCSR[TV]) is not set to the disabled value. If the port response timer value is set to the disabled value, the busy bit will not clear.

19.5.112 Inbound message n status register (SRI0_IMnSR)

The inbound message status register reports various mailbox conditions during and after a message operation. Writing a 1 to the corresponding set bit clears the bit.

Address: C_0000h base + 1_3064h offset + (256d × i), where i=0d to 1d





SRIO_IMnSR field descriptions

Field	Description
0–10 -	This field is reserved. Reserved
11 QF	Queue full. If the queue becomes full, then this bit will be set. This bit is cleared if the message controller is disabled. (Read-only)
12–14 -	This field is reserved. Reserved
15 MIQ	Message-In-Queue. If the queue has accumulated the number of messages specified by the IM n MR[MIQ_THRESH], then this bit will be set. This bit is cleared if the message controller is disabled. (Read-only)
16–20 -	This field is reserved. Reserved
21 MRT	Message request time-out. This bit is set when the message unit has not received another message segment for a multi segment message and a time-out occurred. This bit is cleared by writing a 1.
22–23 -	This field is reserved. Reserved
24 TE	Transaction error. This bit is set when there is an internal error condition occurs during the message operation. This bit is cleared by writing a 1. For proper operation, this field should only be modified when the inbound message controller is not enabled.
25–26 -	This field is reserved. Reserved
27 QFI	Queue full interrupt. If the queue is full and if the QFIE bit in the mode register is set, then this bit will be set and an interrupt generated. This bit is cleared by writing a 1.
28 -	This field is reserved. Reserved
29 MB	Mailbox busy. When set, indicates that a message operation is currently in progress. This bit is cleared as a result of an error or the message operation is finished. (Read-only)

Table continues on the next page...

SRI0_IMnSR field descriptions (continued)

Field	Description
30 QE	Queue empty. If the queue is empty, then this bit is set. This bit is also set when the message controller is disabled. (Read-only)
31 MIQI	Message-In-Queue interrupt. If the queue has accumulated the number of messages specified by the IM n MR[MIQ_THRESH] and if the MIQIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.

19.5.113 Extended inbound message n frame queue dequeue pointer address register (SRIO_EIMnFQDPAR)

Used with IM n FQDPAR; see description in [Inbound message n frame queue dequeue pointer address register](#).

Address: C 0000h base + 1 3068h offset + (256d x i), where i=0d to 1d

SRIO EIMnFQDPAR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EFQDPA	Extended frame queue dequeue pointer address bits. These are the highest order address bits.

19.5.114 Inbound message n frame queue dequeue pointer address register (SRIO_IMnFQDPAR)

The inbound message frame queue dequeue pointer address registers contain the address for the first message in memory to be processed. Software must initialize these registers to the first frame location in memory. When a message has been processed, the processor sets IM n MR[MI]. The mailbox hardware then increments IM n FQDPAR and EIM n FQDPAR to point to the next frame in memory and clears the IM n MR[MI] bit. If the inbound message frame queue enqueue pointer and the inbound message frame queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next message frame from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by the processor, the queue is empty and all outstanding messages have been processed.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to the number of queue entries \times frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IM n MR[CIRQ_SIZ], and the frame size is set in IM n MR[FRM_SIZ]. See [Inbound message n mode register](#).

Address: C_0000h base + 1_306Ch offset + (256d \times i), where i=0d to 1d

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

Reset 0

SRIO_IMnFQDPAR field descriptions

Field	Description
0–28 FQDPA	Frame queue dequeue pointer address. Contains the address of the first message in memory to process.
29–31 -	This field is reserved. Reserved

19.5.115 Extended inbound message n frame queue enqueue pointer address register (SRIO_EIMnFQEPA)

Used with IM n FQEPA; see description in [Inbound message n frame queue enqueue pointer address register](#).

Address: C_0000h base + 1_3070h offset + (256d \times i), where i=0d to 1d

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

Reset 0

SRIO_EIMnFQEPA field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EFQEPA	Extended frame queue enqueue pointer address bits. These are the highest order address bits. For proper operation, this field should only be modified when the inbound message controller is not enabled.

19.5.116 Inbound message n frame queue enqueue pointer address register (SRIO_IMnFQEPA)

The inbound message frame queue enqueue pointer address registers contain the address for the next message frame in memory to be added to the queue. Software must initialize these registers to match the frame queue dequeue pointer address. When a message is received by the mailbox controller, it writes the message data to the next location in the queue (indicated by the address in IM_n FQEPA and EIM_n FQEPA) and then increments IM_n FQEPA and EIM_n FQEPA to point to the next frame location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full and the mailbox controller does not accept any more incoming messages, returning RETRY responses to the sending devices until the queue is no longer full. If the IM_n MR[QFIE] bit is set, then the IM_n MR[QFI] bit is set and an interrupt is generated.
- If the enqueue and dequeue pointers were the same before the register was read, the queue has transitioned from empty to not empty. If IM_n MR[MIQIE] is set, IM_n SR[MIQI] is set, and an interrupt is generated.

When software initializes these registers, the queue to which they point must be aligned on a boundary equal to the number of queue entries x frame size. For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

The number of queue entries is set in IM_n MR[CIRQ_SIZ], and the frame size is set in IM_n MR[FRM_SIZ]. See [Inbound message n mode register](#).

Address: C_0000h base + 1_3074h offset + (256d × i), where i=0d to 1d

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

SRIO_IMnFQEPA field descriptions

Field	Description
0–28 FQEPA	Frame queue enqueue pointer address. Contains the address of the next message frame to be added to the queue. For proper operation, this field should only be modified when the inbound message controller is not enabled.
29–31 -	This field is reserved. Reserved

19.5.117 Inbound message n maximum interrupt report interval register (SRIO_IMnMIRIR)

The maximum interrupt interval register contains a time-out timer value to define the maximum amount of time that the mailbox controller is to wait from transitioning from not empty to signaling an interrupt (if enabled) to the local processor if the IM n MR[MIQ_THRESH] limit is not reached. The reset, or default, value of this counter is the maximum interrupt interval .

The resolution of this timer is 152/(platform frequency). For example, at a platform frequency of 600 MHz, the maximum interval value is 0xFF_FFFF * 152/600MHz = 4.25 seconds.

Address: C_0000h base + 1_3078h offset + (256d \times i), where i=0d to 1d

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

Reset 1 0 0 0 0 0 0 0 0

SRIO_IMnMIRIR field descriptions

Field	Description
0–23 MIRI	Maximum interrupt report interval. Maximum message-in-queue to interrupt generation time. A value of 0 disables the time-out timer. For proper operation, this field should only be modified when the inbound message controller is not enabled.
24–31 -	This field is reserved. Reserved

19.5.118 Outbound doorbell mode register (SRIO_ODMR)

The outbound mode register allows software to start a doorbell operation and to control various doorbell operation characteristics.

Address: C_0000h base + 1_3400h offset = D_3400h

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

Reserved

DUS

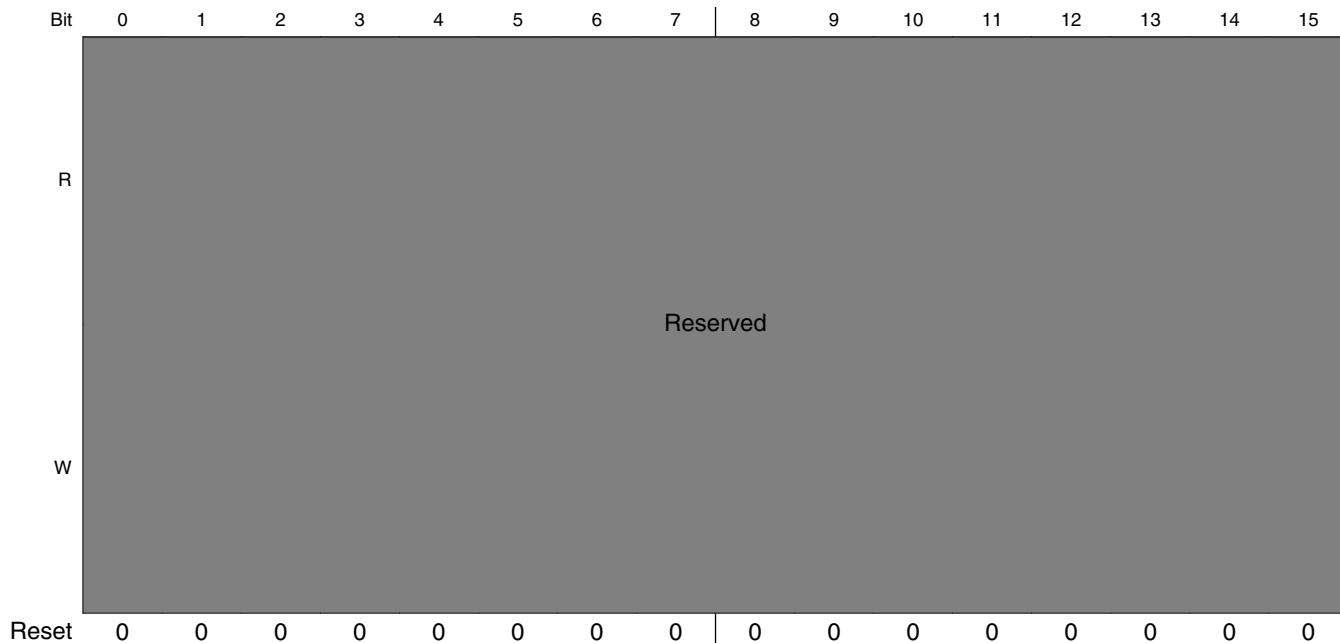
SRIO_ODMR field descriptions

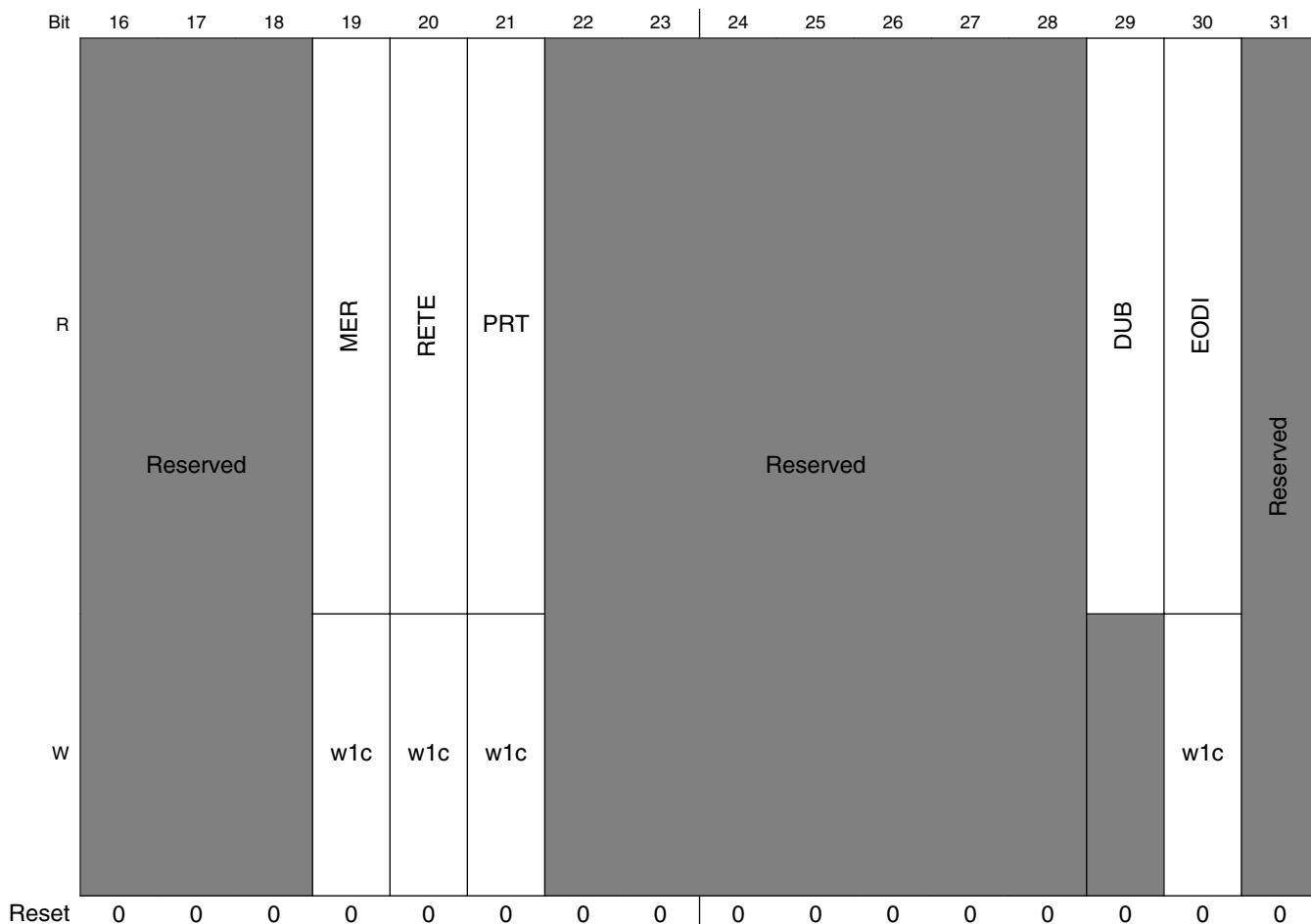
Field	Description
0–30 -	This field is reserved. Reserved
31 DUS	Doorbell unit start. Direct mode - A 0 to 1 transition when the doorbell unit is not busy (DUB bit is 0) starts the doorbell unit. A 1 to 0 transition has no effect.

19.5.119 Outbound doorbell status register (SRIODSR)

The outbound status register reports various doorbell unit conditions during and after a doorbell operation. For some bits, writing a 1 clears the bit.

Address: C_0000h base + 1_3404h offset = D_3404h



**SRIO_ODSR field descriptions**

Field	Description
0–18 -	This field is reserved. Reserved
19 MER	Message error response. This bit is set when an ERROR response is received from the doorbell target. The Error response received field indicates value of the error response status bits when an error response is received. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
20 RETE	Retry error threshold exceeded. This bit is set when the doorbell unit has been unable to complete a doorbell operation because the retry error threshold value has been exceeded due to RapidIO retry response. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
21 PRT	Packet response time-out. This bit is set when the doorbell unit has been unable to complete a doorbell operation and a packet response time-out occurred. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
22–28 -	This field is reserved. Reserved
29 DUB	Doorbell unit busy. When set, indicates that a doorbell operation is currently in progress. This bit is cleared as a result of an error or when the doorbell operation is finished. (Read-only)

Table continues on the next page...

SRIO_ODSR field descriptions (continued)

Field	Description
30 EODI	End-of-doorbell interrupt. After finishing this doorbell operation, if the EODIE bit in the destination attributes register is set, then this bit will be set and an interrupt generated. This bit is cleared by writing a 1. For proper operation, this bit should only be cleared when a doorbell operation is not in progress.
31 -	This field is reserved. Reserved

19.5.120 Outbound doorbell n destination port register (SRIO_ODDPR)

The destination port register indicates the RapidIO Destination ID and mailbox to which the doorbell unit controller is to send data. The software must ensure that this is a valid port in the receiving device.

Address: C_0000h base + 1_3418h offset = D_3418h

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

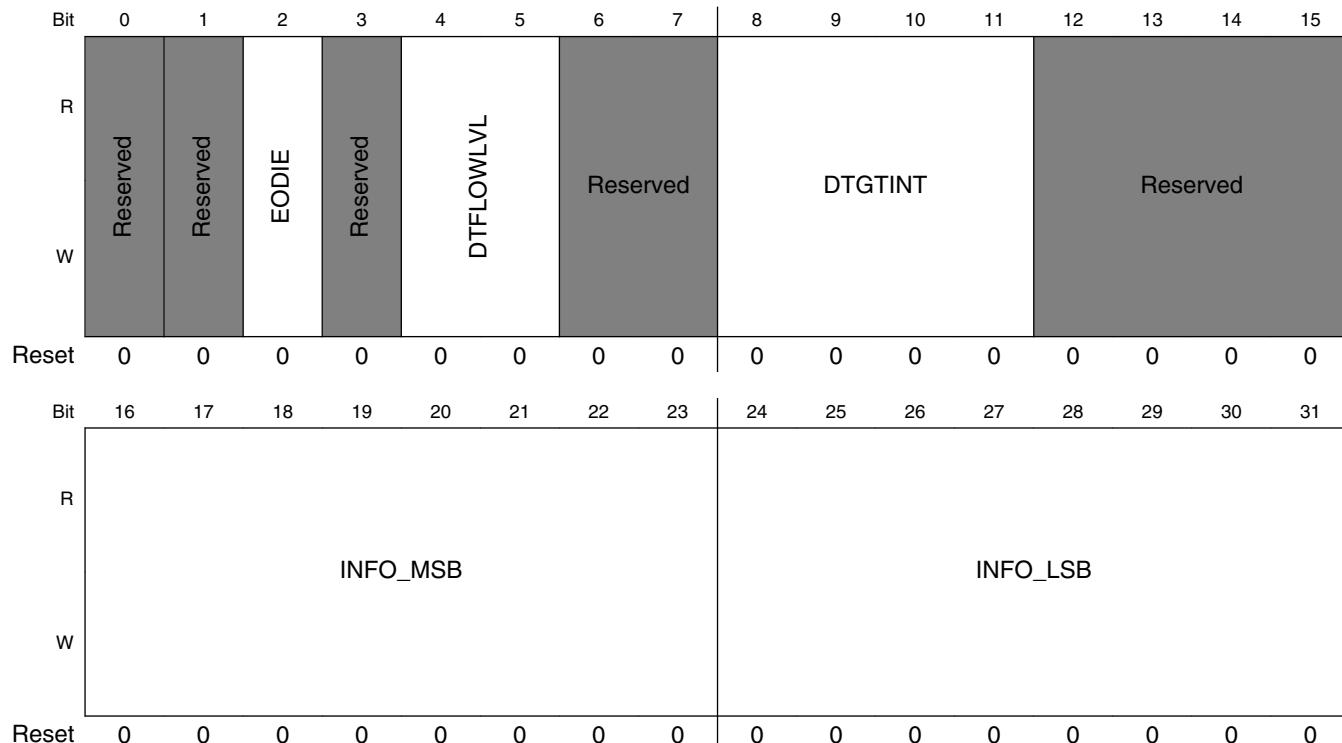
SRIO_ODDPR field descriptions

Field	Description
0–7 EDTGROUTE	Extended destination target route. Most significant byte of a 16-bit target route when operating in large transport mode. Reserved when operating in small transport mode. For proper operation, this field should only be modified when a doorbell operation is not in progress.
8–15 DTGROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target). For proper operation, this field should only be modified when a doorbell operation is not in progress.
16–31 -	This field is reserved. Reserved

19.5.121 Outbound doorbell n destination attributes register (SRIO_ODDATR)

The outbound destination attributes register contains the transaction attributes to be used for the doorbell operation.

Address: C_0000h base + 1_341Ch offset = D_341Ch



SRIO_ODDATR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1 -	This field is reserved. Reserved, hard wired to 0
2 EODIE	End-of-Doorbell interrupt enable. When set, generates an interrupt when the current doorbell operation is finished. For proper operation, this field should only be modified when a doorbell operation is not in progress.
3 -	This field is reserved. Reserved
4–5 DTFLOWLVL	Transaction flow level. For proper operation, this field should only be modified when a doorbell operation is not in progress. 00 Lowest priority transaction flow

Table continues on the next page...

SRIO_ODDATTR field descriptions (continued)

Field	Description
	01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved
6-7 -	This field is reserved. Reserved
8-11 DTGTINT	Target interface. The value of this field should be set according to the following: 0000 RapidIO Port 1 0001 RapidIO Port 2
12-15 -	This field is reserved. Reserved
16-23 INFO_MSB	Most significant byte of the doorbell "info" field. For proper operation, this field should only be modified when a doorbell operation is not in progress.
24-31 INFO_LSB	Least significant byte of the doorbell "info" field. For proper operation, this field should only be modified when a doorbell operation is not in progress.

19.5.122 Outbound doorbell n retry error threshold configuration register (SRIO_ODRETCR)

The retry error threshold configuration register controls the number of times that the doorbell unit attempts to complete a doorbell operation before reporting an error and moving on to the next task, if one is available. Failures to complete an operation are indicated by receiving a RapidIO logical layer RETRY response from the target. If the programmed count is exceeded, the ODSR[RETE] bit is set.

Address: C 0000h base + 1 342Ch offset = D 342Ch

SRIODRETCR field descriptions

Field	Description
0-23 -	This field is reserved. Reserved
24-31 RET	Retry error threshold. This value is the number of times that the doorbell unit attempts to transmit a doorbell. For proper operation, this field should only be modified when a doorbell operation is not in progress. 0x00 Disabled 0x01 Doorbell transmitted only 1 time

Table continues on the next page...

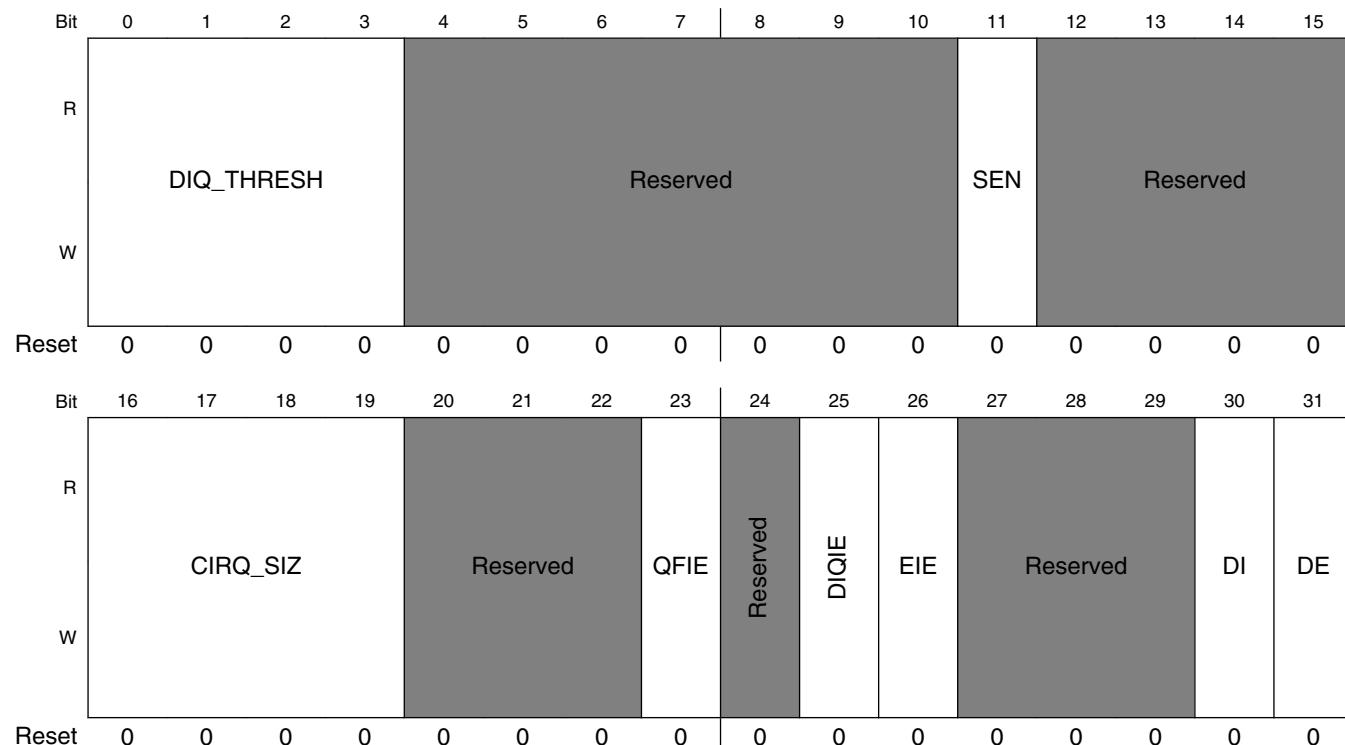
SRIODRETCR field descriptions (continued)

Field	Description	
	0x02	Doorbell transmitted up to 2 times
	...	
	0xFF	Doorbell transmitted up to 255 times

19.5.123 Inbound doorbell n mode register (SRIODIDMR)

The doorbell mode register allows software to enable the doorbell controller to control various doorbell operation characteristics.

Address: C_0000h base + 1_3460h offset = D_3460h

**SRIODIDMR field descriptions**

Field	Description
0–3 DIQ_THRESH	Doorbell-in-queue threshold. Determines the number of doorbells to be accumulated in the doorbell queue before the Doorbell-in-queue bit is set (IDSR[DIQ]). Undefined operation results if the actual number of entries defined by the doorbell-in-queue threshold is set greater than or equal to the actual size of the doorbell queue. For proper operation, this field should only be modified when the doorbell controller is not enabled. 0000 1 0001 2

Table continues on the next page...

SRIO_IDMR field descriptions (continued)

Field	Description
	0010 4 0011 8 0100 16 0101 32 0110 64 0111 128 1000 256 1001 512 1010 1024 1011 reserved ... 1111 reserved
4–10 -	This field is reserved. Reserved
11 SEN	Snoop enable. When set enables snooping the local processor when writing messages into memory. For proper operation, this field should only be modified when the doorbell controller is not enabled.
12–15 -	This field is reserved. Reserved
16–19 CIRQ_SIZ	Circular doorbell queue size. Determines the number of doorbell entries that can be placed on the circular queue. CIRQ_SIZ x 8 bytes determine the maximum contiguous memory space allocated to the doorbell unit. For proper operation, this field should only be modified when the doorbell controller is not enabled. For proper operation, this field should only be modified when the doorbell controller is not enabled. 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64 0110 128 0111 256 1000 512 (4-kbyte page boundary) 1001 1024 1010 2048 1011 reserved ... 1111 reserved
20–22 -	This field is reserved. Reserved
23 QFIE	Queue full interrupt enable. If this bit is set and ID SR[QF] is a 1, ID SR[QFI] is asserted. 0 No QF interrupt is generated 1 Generates an interrupt when the queue is full (that is, the enqueue and dequeue pointers are equal after the dequeue pointer was incremented by the doorbell controller).

Table continues on the next page...

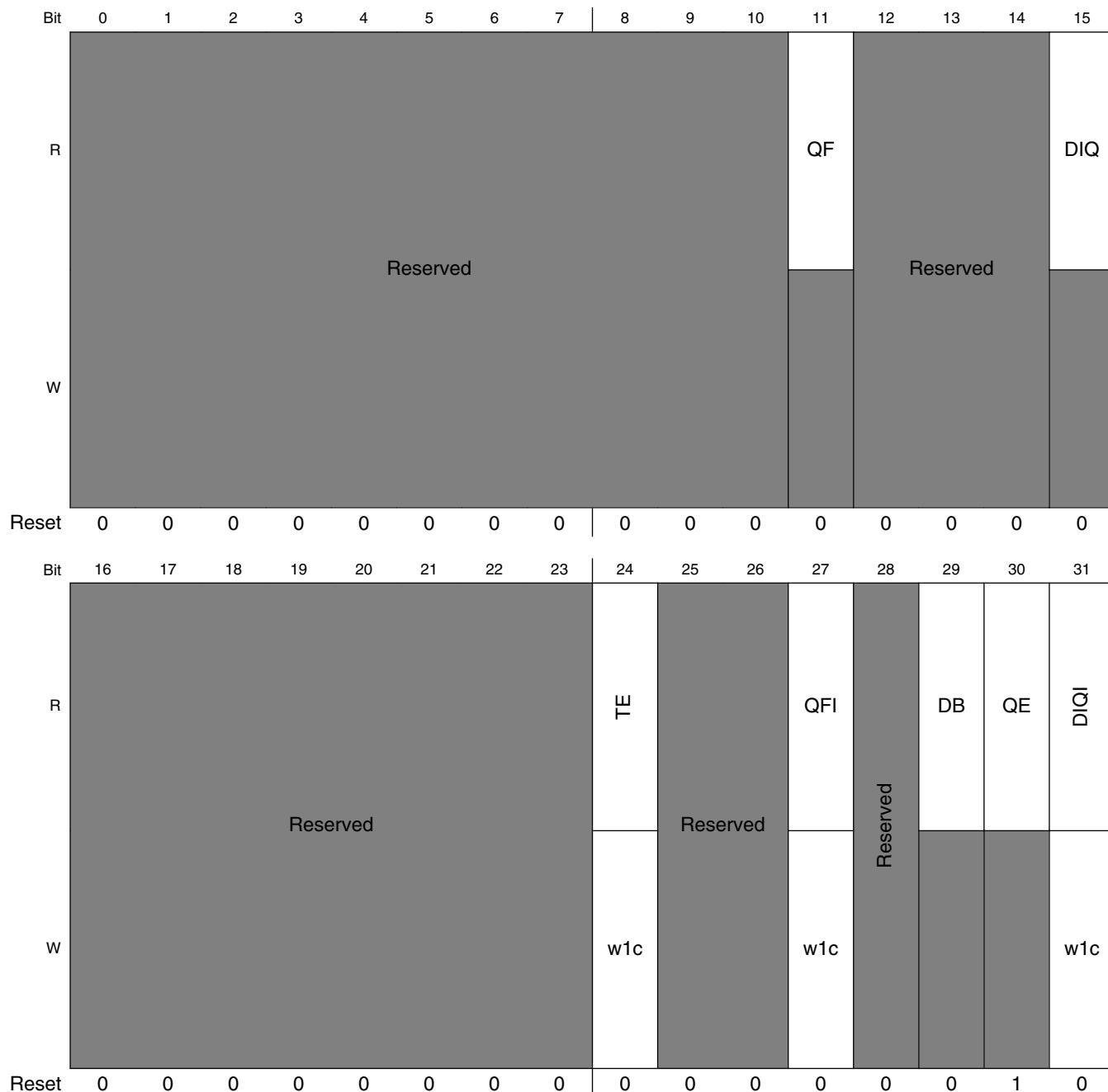
SRIO_IDMR field descriptions (continued)

Field	Description
24 -	This field is reserved. Reserved
25 DIQIE	Doorbell in queue interrupt enable. When set, allows the doorbell in queue interrupt to assert. The doorbell in queue interrupt cannot be asserted if this bit is cleared. Doorbell in queue interrupt enable. If this bit is set and ID SR[DIQ] is a 1, ID SR[DIQI] will assert. If this bit is set and ID MR[DI] is also set simultaneously, ID SR[DIQI] indicates reflects the value of DIQ after the increment. 0 No DIQ interrupt is generated 1 Generates an interrupt when the DIQ bit is set (IDSR[DIQ])
26 EIE	Error interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IDSR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29 -	This field is reserved. Reserved
30 DI	Doorbell increment. Software sets this bit after processing an inbound doorbell. Hardware then increments the OD QEPPAR and clears this bit. Always reads as 0.
31 DE	Doorbell enable. 0 Doorbell disabled 1 The doorbell has been initialized and can service incoming doorbell operations.

19.5.124 Inbound doorbell n status register (SRIO_IDSR)

The doorbell status register reports various doorbell conditions after a doorbell operation. Writing a 1 to the corresponding set bit will clear the bit.

Address: C_0000h base + 1_3464h offset = D_3464h



SRIO_IDSR field descriptions

Field	Description
0–10 -	This field is reserved. Reserved
11 QF	Queue full. If the queue is full, then this bit is set. This bit clears if the doorbell controller is disabled. (Read-only)
12–14 -	This field is reserved. Reserved
15 DIQ	Doorbell-In-Queue. If the queue has accumulated the number of doorbells specified by DIQ_THRESH, then this bit is set. Also, if a valid entry pointed to by the dequeue address pointers has not been serviced within the configured maximum interval, this bit is set. This bit clears if the doorbell controller is disabled. (Read-only)
16–23 -	This field is reserved. Reserved
24 TE	Transaction error. This bit is set when an internal error occurs during the doorbell operation. (Bit reset, write 1 clear). For proper operation, this field should only be modified when the doorbell controller is not enabled.
25–26 -	This field is reserved. Reserved
27 QFI	Queue full interrupt. If the queue becomes full and the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28 -	This field is reserved. Reserved
29 DB	Doorbell busy. When set, indicates that a doorbell has been received and the doorbell queue is currently being written. This bit clears when the write to memory has finished. Disabling the doorbell controller does not affect this bit. (Read-only)
30 QE	Queue empty. If the queue is empty, then this bit is set. This bit is also set if the doorbell controller is disabled.(Read-only)
31 DIQI	Doorbell-In-Queue interrupt. If DIQ is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.

19.5.125 Extended inbound doorbell n queue dequeue pointer address register (SRIO_EIDQDPAR)

Used with ID QDPAR; see description in [Inbound doorbell n queue dequeue Pointer address register](#).

Address: C_0000h base + 1_3468h offset = D_3468h

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

Reset 0

SARIO_EIDQDPAR field descriptions

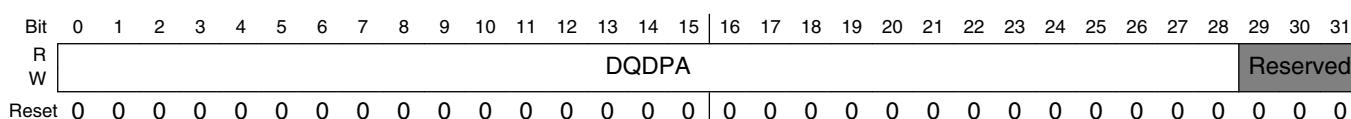
Field	Description
0–27 - Reserved	This field is reserved.
28–31 EDQDPA	Extended doorbell queue dequeue pointer address bits. These are the highest order address bits.

19.5.126 Inbound doorbell n queue dequeue Pointer address register (SRIO_IDQDPAR)

The inbound doorbell queue dequeue pointer address registers (EID QDPAR and ID QDPAR) contain the double-word address for the first doorbell in memory to be processed. Software must initialize these registers to the first doorbell entry location in memory. When the current doorbell has been processed, the processor sets ID MR[DI]. The doorbell hardware then increments the doorbell queue dequeue pointer address register to point to the next doorbell entry in memory and clears ID MR[DI]. When processing multiple doorbells, instead of setting ID MR[DI] for each doorbell, the processor can write these registers directly.

If the doorbell queue enqueue pointer and the doorbell queue dequeue pointer are not equal (indicating that the queue is not empty), the processor can read the next doorbell from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by the processor, the queue is empty and all outstanding doorbells have been processed.

Address: C 0000h base + 1 346Ch offset = D 346Ch



SRIO IDQDPAR field descriptions

Field	Description
0-28 DQDPA	Doorbell queue dequeue pointer address. Contains the double-word address of the first doorbell in memory to process. Note that this base address must be queue-size aligned.
29-31 -	This field is reserved. Reserved

19.5.127 Extended inbound doorbell n queue enqueue pointer address register (SRIO_EIDQEPA)

Used with ID QEPAR; see description in [Inbound doorbell n Queue enqueue pointer address register](#).

Address: C_0000h base + 1_3470h offset = D_3470h

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

SRIO_EIDQEPA field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EDQEPA	Doorbell queue enqueue pointer address bits. These are the highest order address bits. This field can only be written when the doorbell controller is disabled or undefined operation results.

19.5.128 Inbound doorbell n Queue enqueue pointer address register (SRIO_IDQEPA)

The doorbell queue enqueue pointer address registers (EID QEPAR and ID QEPAR) contain the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize ID QEPAR and EID QEPAR to match the doorbell queue dequeue pointer address. When a doorbell packet is received by the doorbell controller, it writes the doorbell information to the next location in the queue (indicated by the address in ID QEPAR and EID QEPAR) and then increments ID QEPAR and EID QEPAR to point to the next doorbell location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, then the queue is now full and the doorbell controller does not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If the ID MR[QFIE] bit is set, then the ID SR[QFI] is set and the interrupt is generated.
- If the enqueue and dequeue pointers were the same before receiving the doorbell, the queue has transitioned from empty to not empty. When the number of doorbells received matches the configured threshold, the ID SR[DIQ] bit is set. If the DRM[DIQIE] bit is set, then the ID SR[DIQI] bit is also set and the inbound doorbell interrupt is generated.

Address: C 0000h base + 1 3474h offset = D 3474h

SRIO IDQEPAR field descriptions

Field	Description
0-28 DQEPA	Doorbell queue enqueue pointer address. Contains the double-word address of the next doorbell location to be added to the queue. This field can only be written when the doorbell controller is disabled or undefined operation results. Note that this base address must be queue-size aligned.
29-31 -	This field is reserved. Reserved

19.5.129 Inbound doorbell n maximum interrupt report interval register (SRIO_IDMIRIR)

The maximum interrupt interval register contains a time-out timer value to define the maximum amount of time that a doorbell entry can be at the head of the doorbell queue before generating an interrupt (if enabled) if the DIQ_THRESH limit is not reached. The reset, or default, value of this counter is the maximum interrupt interval .

The resolution of this timer is $152 / (\text{platform frequency})$. For example, at a platform frequency of 600 MHz, the maximum interval value is $0xFF_FFFF * 152 / 600\text{MHz} = 4.25$ seconds.

Address: C 0000h base + 1 3478h offset ≡ D 3478h

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

SRIO IDMIRIR field descriptions

Field	Description
0–23 MIRI	Maximum interrupt report interval—Maximum doorbell-in-queue to interrupt generation time. A value of 0 disables the timer. This field can only be written when the doorbell controller is disabled or undefined operation results.
24–31 -	This field is reserved. Reserved

19.5.130 Inbound port-write n mode register (SRIO_IPWMR)

The port-write mode register allows software to enable the port-write controller and to control various port-write operation characteristics.

Address: C_0000h base + 1_34E0h offset = D_34E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R W	Reserved								SEN	Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R W	Reserved								QFIE	Reserved		EIE	Reserved			CQ	PWE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

SRIO_IPWMR field descriptions

Field	Description
0–10 -	This field is reserved. Reserved
11 SEN	Snoop enable. When set enables snooping the local processor when writing port-write data payload into memory. For proper operation, this field should only be modified when the port-write controller is not enabled.
12–22 -	This field is reserved. Reserved
23 QFIE	Queue full interrupt enable. When set, generates the error/port-write interrupt if the queue is full (that is, the controller has written the port-write data payload into memory). No error/port-write interrupt is generated if this bit is cleared. For proper operation, this field should only be modified when the port-write controller is not enabled.
24–25 -	This field is reserved. Reserved
26 EIE	Error Interrupt enable. When set, generates the port-write/error interrupt when a transfer error (IPW0SR[TE]) event occurs. No port-write/error interrupt is generated if this bit is cleared.
27–29 -	This field is reserved. Reserved
30 CQ	Clear queue. Software sets this bit after processing an inbound port-write operation. Hardware clears the queue full bit (IPWSR[QF]), clears this bit, and allows another port-write to be received. This bit is always read as a 0.
31 PWE	Port write enable. If this bit is set the port-write controller has been initialized and can service an incoming port-write operation.

19.5.131 Inbound port-write n status register (SRIO_IPWSR)

The port-write status register reports various port-write conditions.

Address: C_0000h base + 1_34E4h offset = D_34E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								QF		Reserved					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								TE	Reserved		QFI	PWD	PWB	Reserved	
W									w1c			w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_IPWSR field descriptions

Field	Description
0–10 -	This field is reserved. Reserved
11 QF	Queue full. If the queue becomes full, then this bit is set. This bit is cleared when the clear queue bit is set (IPWMR[QF]) and the queue is not full. This bit is also cleared if the port-write controller is disabled. (Read-only)
12–23 -	This field is reserved. Reserved
24 TE	Transaction error. This bit is set when there an internal error condition occurs during the port-write operation. (Bit reset, write 1 clear). For proper operation, this field should only be modified when the port-write controller is not enabled.
25–26 -	This field is reserved. Reserved
27 QFI	Queue full interrupt. If the queue becomes full and the QFIE bit in the Mode Register is set, then this bit is set and an interrupt generated. This bit is cleared by writing a 1.
28 PWD	Port-write discarded. This bit is set if a port-write is discarded while the port-write controller is enabled but busy. This bit is cleared by writing a 1.
29 PWB	Port-write busy (Read-only). <ul style="list-style-type: none"> 0 This bit is cleared after the port-write payload has been written to memory. 1 Indicates that a port-write has been received and the port-write payload is currently being written to memory. Disabling the port-write controller does not affect this bit.
30–31 -	This field is reserved. Reserved

19.5.132 Extended inbound port-write n queue base address register (SRIO_EIPWQBAR)

Used with IPW QBAR; see description in [Inbound port-write n queue base address register](#).

Address: C_0000h base + 1_34E8h offset = D_34E8h

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

SRIO_EIPWQBAR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EPWQBA	Extended port-write queue base address bits. These are the highest order address bits. This field can only be written when the port-write controller is disabled or undefined operation results.

19.5.133 Inbound port-write n queue base address register (SRIO_IPWQBAR)

The port-write queue base address registers (EIPW QBAR and IPW QBAR) contain the 64-byte cache line address for the port-write data payload. Software must initialize these registers to the desired location in memory.

Address: C_0000h base + 1_34ECh offset = D_34ECh

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

SRIO_IPWQBAR field descriptions

Field	Description
0–25 PWQBA	Port-write queue base address. Contains the cache line address of the port-write data payload. This field can only be written when the port-write controller is disabled or undefined operation results. When populating the address in this register it is not necessary to shift the address as the lower 6 bits are merely ignored to ensure cache line alignment.
26–31 -	This field is reserved. Reserved

19.6 Serial RapidIO functional description

19.6.1 Segmented Outbound Window Description

All outbound windows have the capability to have 2 or 4 segments, all of which are equal in size, numbered 0 and 1, or 0 through 3, respectively (the standard 8540 unsegmented window definition becomes segment 0). Each segment assigns attributes and the target deviceID for an outbound transaction. All segments of a window translate to the same translation address in the target.

Additionally, each segment can be set up with 2, 4, or 8 subsegments, all of which are equal in size. These subsegments allow a single segment to target a number of numerically adjacent target device IDs, and, again, they all translate to the same translation address in the targets. For example, a segment with 8 subsegments can be configured to generate a transaction with the same set of attributes to target deviceIDs 0, 1, 2, 3, 4, 5, 6, or 7, depending on which subsegment is addressed.

Note that subsegments are only supported when multiple segments are chosen.

This allows a window to be configured so that aliases can be created to the same offset within the target device so that a single window can be used to generate different transaction types. Without segmented windows, achieving the equivalent behavior would require multiple windows. The figure below shows an example of this capability. A window is defined to be 4kB in size, and is defined to have 4 segments and no subsegments. Each segment is assigned to target deviceID 0x05, and each segment is given a different write transaction type attribute - segment 0 is assigned NWRITE, segment 1 is assigned SWRITE, segment 2 is assigned NWRITE_R, and segment 3 is assigned FLUSH. Since all of the segments are assigned to target the same device, by writing to the same offset in each segment, a different write transaction can be generated to the target to the same offset in the target.

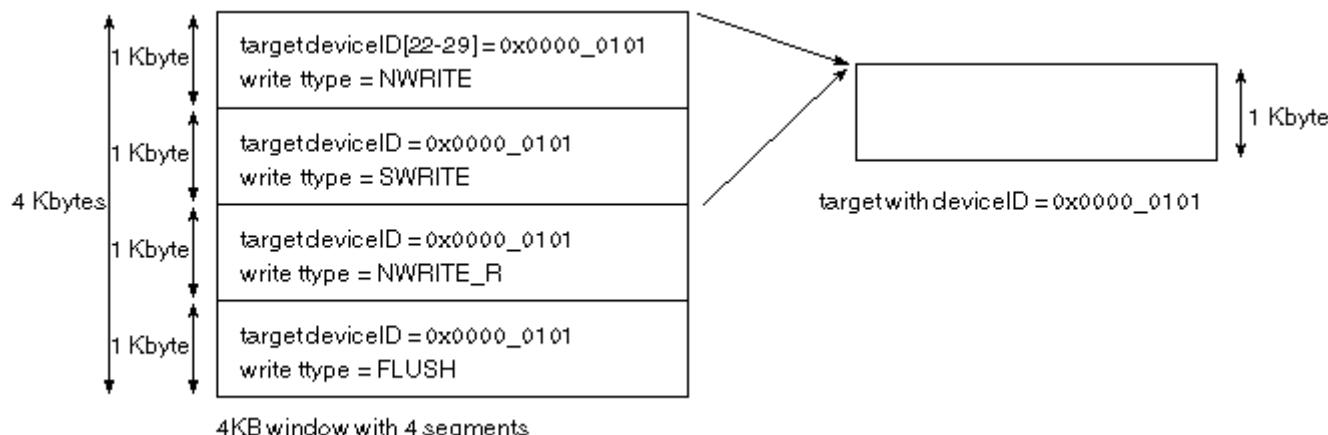


Figure 19-319. Example of Attribute Aliasing

So, writing to offset 0x0 in segment 0 is translated (as defined in the translation address registers) and generates a NWRITE transaction to offset 0x0 in a 1KB window in the target with deviceID = 0x05. A write to offset 0x0 in segment 2 is also translated, to the same offset in the target device as the write to segment 0, but this time a NWRITE_R transaction is generated.

Another use is that the same window can be used to target multiple devices with the same translation offset. Without segmented, (and subsegmented) windows, achieving the equivalent behavior would require multiple windows. The following figure shows an example of this multi-targeting. For example, a 4kB window is set up with 2 segments of 2 subsegments. Each segment is assigned a write type of NWRITE, but each segment and subsegment has a different target deviceID. Segments 0 and 1 are assigned target deviceIDs 4 and 5, and 8 and 9, respectively.

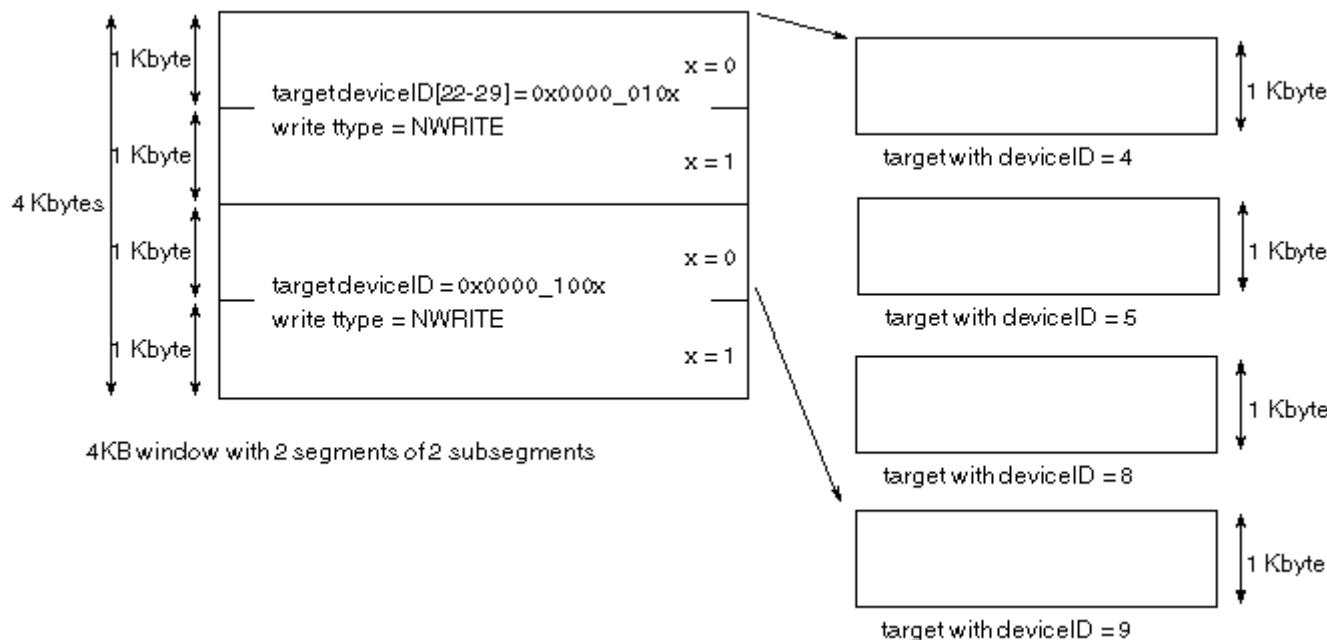


Figure 19-320. Example of Multi-Targeting

In this example, a write to offset 0x0 in segment 0 is translated as defined, and a NWRITE transaction is generated targeted to deviceID 4. A corresponding write to segment 1 to offset 0x400 is also translated but also using the assigned deviceID instead of the translation address bits [22-29]. The generated NWRITE transaction has the same target device offset as the write to segment 0, but is instead targeted to deviceID 9.

Combinations of aliasing and multi-targeting are also possible for a window.

19.6.2 RapidIO transaction summary

The RapidIO endpoint on this device supports all RapidIO I/O transactions, all RapidIO message passing transactions, and a few RapidIO GSM transactions.

Table 19-318. RapidIO I/O transactions

IO Transaction	ftype	ttype	Status	Description
NREAD	0010	0100	NA	Read
ATOMIC inc		1100		Read and post-increment with response
ATOMIC dec		1101		Read and post-decrement with response
ATOMIC set		1110		Read and set to all-1s with response
ATOMIC clr		1111		Read and set to all-0s with response
NWRITE	0101	0100		Write with no response
NWRITE_R		0101		Write with response
SWRITE	0110	N/A		Streaming-Write

Table continues on the next page...

Table 19-318. RapidIO I/O transactions (continued)

IO Transaction	ftype	ttype	Status	Description	
MAINT read	1000	0000		Maintenance read	
MAINT write		0001		Maintenance write	
MAINT read response		0010	0000	Done maintenance read response	
			0111	Error response	
MAINT write response		0011	0000	Done maintenance write response	
			0111	Error response	
MAINT port-write		0100	NA	Maintenance port-write ¹	
RESPONSE without data	1101	0000	0000	I/O done response	
			0111	I/O error response	
RESPONSE with data		1000	0000	I/O done response with data	

1. Limited to inbound RapidIO packets only

Table 19-319. RapidIO message passing transactions

MSG Transaction	ftype	ttype	status	Description
DOORBELL	1010	NA	NA	Doorbell
MESSAGE	1011			Message
RESPONSE without data	1101	0000	0000	Doorbell done response
			0011	Doorbell retry response
			0111	Doorbell error response
		0001	0000	Message done response
			0011	Message retry response
			0111	Message error response

Table 19-320. RapidIO GSM transactions

GSM Transaction	ftype	ttype	status	Description
IO_READ_HOME	0010	0010	NA	I/O Read Home ¹
FLUSH with data	0101	0001		GSM Flush with data ¹
RESPONSE without data	1101	0000	0000	GSM done response
			0011	GSM retry response
			0101	GSM done intervention response
			0111	GSM error response
	1000	0000	GSM done response	
		0001	GSM data only response	

1. Limited to RapidIO packet generation only

19.6.3 RapidIO packet format summary

[Table 19-321](#) summarizes the small transport field packet formats for supported RapidIO transaction types for LP-Serial operation.

Note that this RapidIO endpoint limits configuration read and write requests to 32-bit data accesses.

Table 19-321. RapidIO small transport field packet format

Transaction	Bits																		
	32								32						16			64	
	5	3	2	2	4	8	8	4	4	8	8	1	1	2					
NREAD, ATOMIC (inc/dec/inc/dec), IO_READ_HOME	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	rdsiz	src TID	addr	w d p tr	xam bs	NA					
NWRITE_R, FLUSH with data	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	wrsiz	src TID	addr	w d p tr	xam bs	dword 0 -> dword n					
NWRITE	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	wrsiz	don't care	addr	w d p tr	xam bs	dword 0 -> dword n					
SWRITE	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	addr(29), rsv(1) = 0, xambs(2)						dword 0 -> dword n					
MAINT read	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	rd/ wrsiz	src TID	hop cnt	cfg offs et	w d p tr	rsv = 0	NA				
MAINT write	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	rd/ wrsiz	src TID	hop cnt	cfg offs et	w d p tr	rsv = 0	dword 0 (32-bit)				
MAINT port-write	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	rd/ wrsiz	rsv = 0	hop cnt	rsv = 0	w d p tr	rsv = 0	dword 0 -> dword n				
MAINT response without data	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	status	tar TID	hop cnt	rsv = 0		NA					
MAINT response with data	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	status	tar TID	hop cnt	rsv = 0		dword 0 (32-bit)					
RESPONSE without data	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ftyp e	status	tar TID	NA								

Table continues on the next page...

Table 19-321. RapidIO small transport field packet format (continued)

Transaction	Bits															
	32								32							
	5	3	2	2	4	8	8	4	4	8	8	8	1 3	1	2	64
RESPONSE without data for message	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ttyp e	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	ttyp e	status	tar TID	dword 0 -> dword n					
DOORBELL	ackl D	rsv = 0	prio	tt	ftyp e	des tID	src ID	rsv = 0		src TID	Info- msb	I n f o - ls b				
MESSAGE	ackl D	rsv = 0	prio	tt	ftyp e	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					

The large transport field packet formats extends the destination and source IDs to 16-bits each.

19.6.4 RapidIO control symbol summary

Table 19-322 summarizes the 1x/4x LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part IV: Physical Layer 1x/4x LP-Serial Specifications, Chapter4, PCS and PMA Layers*, for 8B/10B data and special (/PD/, /SC/, idle, sync, skip, align) characters. The 32-bit LP-Serial control symbol is comprised of the eight bit special character and the 24-bit control symbol format.

Table 19-322. 1x/4x LP-serial control symbol format

Bits						Description	
24							
stype0	param0	param1	stype1	cmd	CRC		
3	5	5	3	3	5		
000	pkt_ackID	buf_stat	-		crc	Packet accepted	
001	pkt_ackID	buf_stat	-		crc	Packet retry	
010	pkt_ackID	cause	-		crc	Packet not accepted cause: 00001: Received unexpected ackID on packet 00010: Received a control symbol with bad CRC	

Table continues on the next page...

Table 19-322. 1x/4x LP-serial control symbol format (continued)

Bits						Description	
24							
stype0	param0	param1	stype1	cmd	CRC		
3	5	5	3	3	5		
						00011: Non-maintenance packet reception is stopped 00100: Received packet with bad CRC 00101: Received invalid character or a valid but illegal character 11111: General error	
100	ackID_stat	buf_stat	-	crc		Status ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7	
110	ackID_stat	port_stat	-	crc		Link-response ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7 port_stat: 00010: Error; unrecoverable 00100: Retry stopped 00101: Error stopped 10000: OK	
-			000	000	crc	Start of packet	
-			001	000	crc	Stomp	
-			010	000	crc	End of packet	
-			011	000	crc	Restart from retry	

Table continues on the next page...

Table 19-322. 1x/4x LP-serial control symbol format (continued)

Bits						Description	
24							
stype0	param0	param1	stype1	cmd	CRC		
3	5	5	3	3	5		
-			100	cmd	crc	Link request cmd: 011: Reset the receiving device 100: Return input port status; functions as a restart-from-error control symbol under error conditions	
-			101	000	crc	Multicast-event	
-			111	000	crc	NOP (ignore)	

19.6.5 Accessing configuration registers using rapidIO packets

There are two suggested methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

The first method is to use RapidIO NREAD and NWRITE_R requests to access the registers through a window defined by the local configuration space base address command and status register (LCSBA1CSR). See [Local configuration space base address 1 command and status register \(SRIOLCSBA1CSR\)](#), for more information. Subject to the following conditions, inbound requests whose RapidIO address hits the window defined by LCSBA1CSR are translated to the local address range indicated by the CCSRBAR:

- Access to the registers are restricted to 32-bit requests.
- Only NREAD and NWRITE_R requests are supported. If an NWRITE, SWRITE, or ATOMIC request hits the LCSBA1CSR window, an illegal transaction decode error is generated and logged.
- If external configuration accesses are disabled (LLCR[ECRAB] = 1), any configuration access through the LCSBA1CSR window is denied. In this case, writes are ignored and a 32-bit data payload of all zeros is returned for a non-maintenance configuration read.

The second method is based on a RapidIO MAINT requests. This method allows an external device limited access to local RapidIO configuration register space only. Any maintenance access beyond the first 64 Kbytes (the lower 64 Kbytes contain the RapidIO

architecture registers; the upper 64 Kbytes contain the RapidIO implementation registers) of RapidIO configuration register space, is denied. A 32-bit data payload of all zeros is returned if for a read response.

A third method, though not suggested, would use an inbound ATMU window to translate RapidIO NREAD and NWRITE_R requests to configuration accesses. This method does not support the configuration access protection features offered by the LCSBACSR window and RapidIO MAINT requests.

19.6.5.1 Guidelines

The RapidIO endpoint limits requests to configuration register space to 32-bit data accesses.

If the order of completion is important, inbound configuration accesses should be assumed incomplete until an appropriate response has been received. It is suggested that there be only one outstanding configuration request at a time to ensure that requests are completed in the order they were intended.

19.6.5.2 Outbound maintenance accesses

Outbound OCeaN NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the OCeaN address falls within the bounds of an outbound ATMU window that is setup for generating a maintenance request.

The ATMU window specifies the configuration offset, hop count, source and destination ID, critical request flow, and priority for the outbound RapidIO packet.

19.6.6 RapidIO outbound ATMU

19.6.6.1 Valid hits to multiple ATMU windows

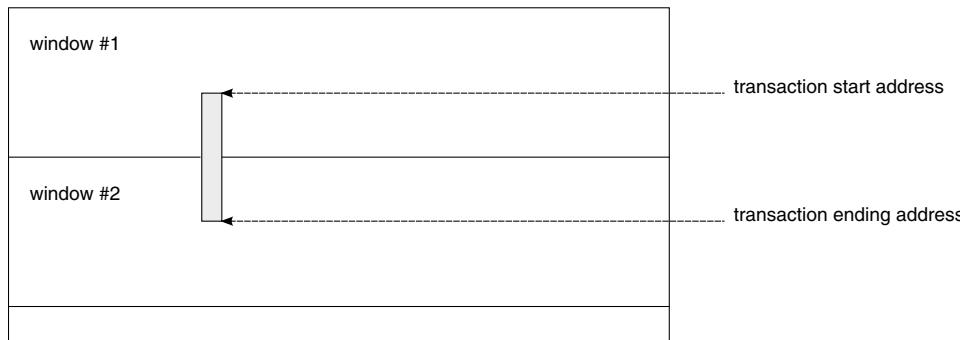
If a request hits multiple ATMU windows, window 1 has the highest priority of the nine outbound ATMU windows (windows 1-8, default).

Window 2 is given the next higher priority and is followed by windows 3 through 8. The default window has the lowest priority.

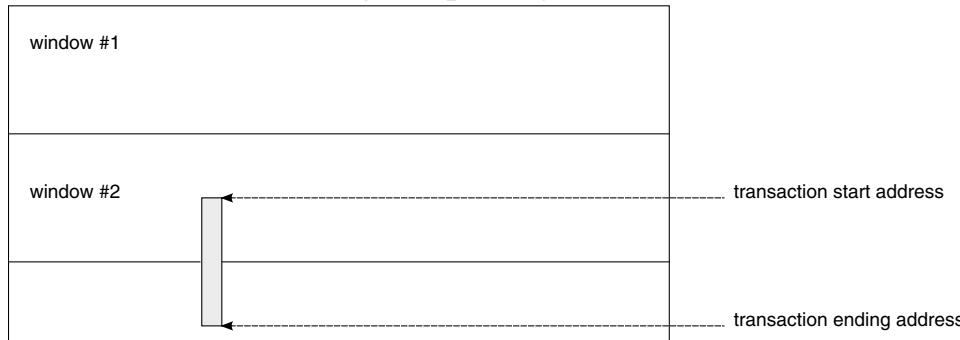
If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-8) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.



2. If a request hits (base address match) multiple ATMU windows (1-8) and the transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.



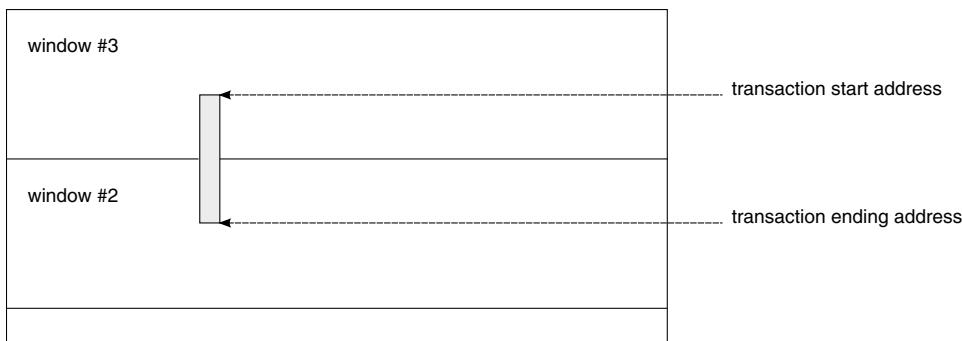
19.6.6.2 Outbound window boundary crossing errors

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries.

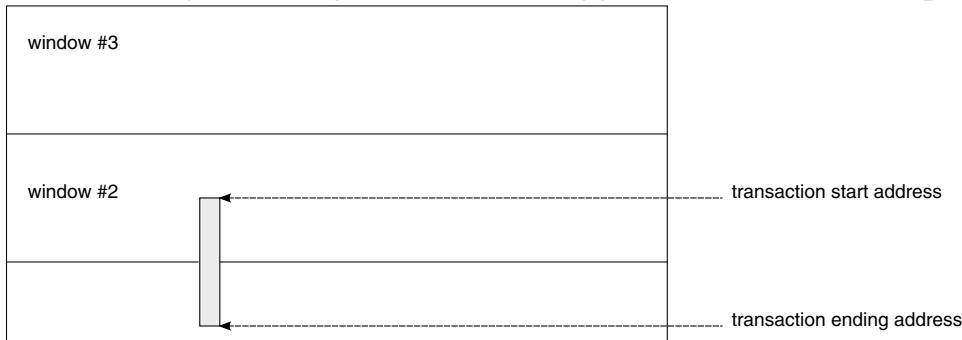
The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-8, default) and the transaction's end address extends into another ATMU window with higher priority,

an ATMU crossed boundary error is generated and logged. The outbound request is discarded.

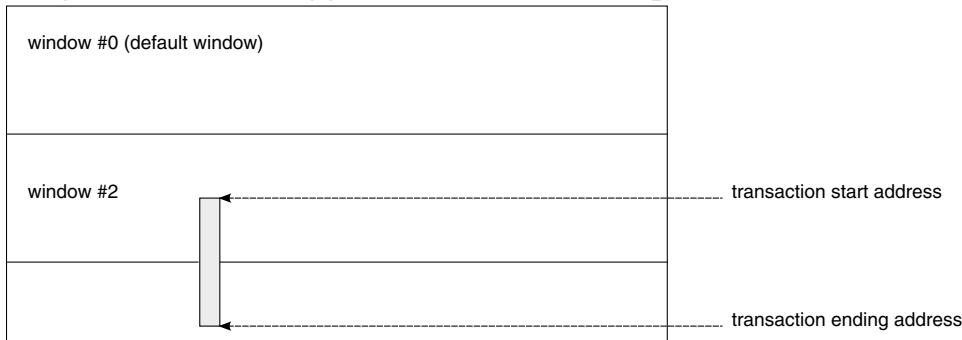


2. If a request hits multiple ATMU windows (1-8, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Other window boundary crossing errors are:

1. If a request hits (base address match) an ATMU window (1-8) and the transaction's end address exceeds the size of the window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Boundary crossing errors (outbound ATMU boundary crossing, segment boundary crossing, and subsegment boundary crossing) are logged in the LTLEDCSR[OACB] configuration register field.

If a request misses all ATMU windows (1-8) and the transaction's end address exceeds the maximum size of the default window, an outbound ATMU crossed boundary error is not generated. The outbound request is forwarded to the RapidIO target device.

19.6.7 RapidIO inbound ATMU

19.6.7.1 Hits to multiple ATMU windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the five inbound ATMU windows (windows 1-4, default).

Window 2 is given the next higher priority and is followed by windows 3 and 4. The default window has the lowest priority.

If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-4) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.
2. If a request hits (base address match) multiple ATMU windows (1-4) and transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.

19.6.7.2 Inbound window boundary crossing errors

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries.

The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-4, default) and the transaction's end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged.

2. If a request hits multiple ATMU windows (1-4, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an inbound ATMU crossed boundary error is generated and logged.

Other window boundary crossing errors are:

1. If a request hits (base address match) an ATMU window (1-4) and the transaction's end address exceeds the size of the window, an inbound ATMU crossed boundary error is generated and logged.
2. If a NREAD/NWRITE_R/NWRITE/SWRITE request hits (base address match) an ATMU window (1-4, default) and the transaction's end address extends into the region defined as the local configuration space window, an inbound ATMU crossed boundary error is generated and logged.

A RapidIO error response is generated for RapidIO requests that require a response. RapidIO requests that do not require a response are dropped.

Inbound ATMU boundary crossing errors are logged in the LTLEDCSR[IACB] configuration register.

If a request misses all ATMU windows (1-4) and the transaction's end address exceeds the maximum size of the default window, an inbound ATMU crossed boundary error is not generated.

19.6.8 Generating link-request/reset-device

In LP-Serial mode the link partner cannot be reliably reset using the link-request/reset-device control symbols since the input port receiver cannot be disabled independent of the output port driver. The input port driver needs to be disabled to prevent non idle control symbols from being transmitted between the four link-request/reset-device control symbols. For example, if a packet was received on the inbound side after one of the four link-request/reset-device control symbols was sent outbound, the required sequence of four link-request/reset-device symbols would be interrupted with the packet acknowledgement (packet accept, packet retry or packet not accept).

19.6.9 Outbound drain mode

- The RapidIO port is placed into Drain mode when one of the following occurs:
 - PmPCR[OBDEN] is set
 - the Failed Threshold has been encountered and the PmCCSR[SPF] and PmCCSR[DPE] are both set
 - the packet time-to-live counter expires causing PmPCR[OBDEN] to be set

- When the RapidIO port is placed into Drain mode, the RapidIO port discards all packets in the outgoing data stream. Since the data stream is invalid, the RapidIO port also puts its Outbound port back to normal state. Any received acknowledgements and link-responses are considered invalid during this period (since the RapidIO port has cleared out all acknowledgement history).
- The RapidIO port's outbound and outstanding ackID shows that all outstanding packets at the time Drain mode was entered were accepted, whether they truly were accepted or not. If the outbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of the Drain mode. Also, if the link-partner needs to be put back into inbound OK state, then software should send a link-request/input-status. The recommended sequence for recovering from Drain mode is given in [Software assisted error recovery register support](#).
- PmPCR[OBDEN] also causes any queued up packet acknowledgements to be discarded if the port is uninitialized (the RapidIO port lets them be transferred if the port is initialized). Drain mode due to failed threshold does not cause any packet acknowledgements to be dropped.
- If a discarded packet in the outgoing data stream requires a logical response, a packet response time-out occurs as long as the packet response timer is enabled (PRTOCCSR is non 0).

19.6.10 Input port disable mode

- The RapidIO port is placed into Input Port Disable mode when PmCCSR[PD] is set.
- When the RapidIO port is placed in Input Port Disable mode, the RapidIO port discards all incoming data stream (obviously). Since the incoming stream is invalid, the RapidIO port also puts its inbound port back to normal state.
- When the RapidIO port is placed in input port disable mode, the RapidIO port also:
 - ends any packet capture that was in progress.
 - clears the link-request/reset-device count.
- The RapidIO port's inbound ackID shows that all packets successfully received by the RapidIO port at the time input port disable mode was entered were accepted. If output port disable mode was entered at the same time, some packet acknowledgements may be discarded by the RapidIO port. If the inbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of Input Port Disable mode.

19.6.11 Software assisted error recovery register support

- PmLMREQCSR is only supported for recovery from drain mode, including hot-swap support. Consistent with this statement, software should only write this register when the port is in Drain mode.
- The proper sequence for recovering from Drain mode is:
 - a. Software ensures that link activity is stopped. This should include:
 1. software polls for Port OK bit to be set
 2. software waits longer than the link time-out value
 - b. Software generates a link-request/input-status to obtain the link-partner's inbound ackID value.
 - c. Software changes the RapidIO port's outbound ackID to this value (if necessary).
 - d. If the link-partner was hot-inserted, software changes the RapidIO port's inbound ackID to zero. (Note that software needs to know when the link-partner was hot-inserted.)
 - e. NOTE: If software can guarantee that the link-partner does not attempt to forward any packets to this RapidIO port, then software may write the outbound ackID of the link-partner to match the inbound ackID of this RapidIO port. However, if the link-partner attempts to forward another packet while this write is still outstanding, and the ackIDs already happened to line-up, then the write actually causes the ackIDs to not match, and the link can not be recovered.
 - f. Software should cause the link-partner to send a link-request/input-status just to ensure that the RapidIO port's inbound port is in Normal operation.
 - g. Software clears the Failed Encountered bit or the Output Buffer Drain Enable bit; whichever one caused the Drain mode (thus clearing Drain mode).
- Software is responsible for timing software generated link-requests. If the response valid bit is not set in some reasonable period of time, the software should write another request in the register.
- When software writes PmLMREQCSR, software should make sure to successfully read PmLMRESPCSR set, otherwise, software may read a "stale" ackID status/link status later.
- Note that when the RapidIO port's outbound ackID is written by software using PmLASCSR, the inbound ackID is also written. Care must be taken to ensure that the inbound ackID is not written to an incorrect value. For example, PmCCSR[PL] could be set to prevent the inbound ackID from changing before PmLASCSR is written.

19.6.12 Hot-swap support

The two basic hot-insertion approaches described in the RapidIO Error Management Extensions are supported although the second approach is only partially supported.

Method one has a host bringing a field replacement unit (FRU) into the system. Method two has the FRU bringing itself into the system. Note that this RapidIO port is most likely the device being hot-inserted/extracted since typically this device would be connected to a switch but it could be the link partner of a device being hot-inserted/extracted.

19.6.12.1 Method 1

One possible sequence for when the host brings the FRU into the system is given.

This RapidIO port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

19.6.12.1.1 Extraction-method 1

- The host determines that the FRU has failed by getting a port response time-out when polling the port OK bit (PO bit in port m error and status CSR) of the FRU. Note that there are other ways to determine that the FRU has failed. This is one possible method to detect FRU failure.
- The host must perform the following steps to the FRU link partner before hot insertion of the FRU occurs.
 - set the port lockout bit (PL = 1 in the port m control CSR) preventing packet reception and transmission
 - prevent any new packets from arriving to the FRU link partner RapidIO port by all other RapidIO devices and discard all pending packets destined for the FRU so that...
 - congestion to this RapidIO port does not occur and cause other system problems
 - the FRU is not immediately flooded with old packets after insertion causing other errors like unsolicited responses
 - note that this RapidIO endpoint has a specific bit to enable the discard of pending packets (OBDEN in PmPCR). Also note that setting OBDEN in PmPCR forces the output state from error or retry to normal. In a switch, the time-to-live feature may be used to discard packets.
 - force the input state to normal. In this RapidIO endpoint, this occurs by disabling the input port receiver.
 - leave the input port receivers and output port drivers enabled so that initialization can complete when the FRU is inserted
 - clear all errors pertaining to the extracted RapidIO port in the FRU link partner and any other RapidIO port that was affected by the FRU failing (port response time-outs)

- if RapidIO endpoint, clear OBDEN in PmPCR for normal operation
- set the outbound and inbound ackID to 0x00 in the port m local ackID status CSR so that the ackID is correct and packets can be transmitted and received when the FRU is inserted
- The host indicates that the FRU should be removed.
- The FRU is removed from the system.

19.6.12.1.2 Insertion-method 1

- The FRU is inserted into the system
- Link initialization occurs and initialization complete status is indicated (port OK bit in the port m irror and status CSR) in both RapidIO ports
- The host determines that this link partner has been inserted by periodically polling the initialization complete status in the FRU link partner (PO bit = 1 in the port m irror and status command and status register)
- The host clears the output and input port enable bits and clears the port lockout bit in the port m control CSR in the FRU link partner allowing only maintenance transactions
- The host sets the master enabled and discovered bits in the FRU ($M = 1$ and $D = 1$ in the port general control CSR). Note that the master enable bit does not prevent the RapidIO endpoint from transmitting packets.
- The host completes configuration of the FRU
- The host sets the output and input port enable bits in the port m control CSR in the FRU link partner allowing transmission and reception of all packet types
- The host re-enables packets to be sent to this RapidIO port by other RapidIO devices
- System operation resumes

19.6.12.2 Method 2 with RapidIO port hot-swapped

One possible sequence for when the FRU brings itself into the system is given.

This RapidIO Port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

19.6.12.2.1 Extraction-method 2

- The FRU fails.
- New packets are prevented from arriving to the FRU link partner by all other RapidIO devices and all pending packets destined for the FRU are discarded so that the following occur:

- Congestion to this RapidIO port does not occur and cause other system problems.
- The FRU is not immediately flooded with old packets after insertion causing other errors such as unsolicited responses.
- This FRU link partner continues to have its drivers enabled.
- The FRU link partner continues to allow the transmission and reception of all packet types since its output and input port enable bits are set and the port lockout bit is cleared in the port *m* control CSR.
- The FRU is removed from the system.

19.6.12.2.2 Insertion-method 2

- The FRU is inserted.
- Link initialization occurs, initialization complete status is indicated in both RapidIO ports (PO bit = 1 in the port *m* error and status command and status register).
- After initialization is complete, both devices set the port OK bit = 1 in the port *m* control CSR.
- The FRU sets its port lockout bit in the port *m* control CSR.
- The FRU generates a link-request/input-status to its link partner using the port *m* link maintenance request register. The FRU determines the link partner's inbound ackID by reading the port *m* link maintenance response register. The FRU then sets its outbound ackID in the port *m* local ackID status CSR.
- The FRU only enables maintenance transactions by clearing its output and input port enable bits in the port *m* control CSR and by clearing its port lockout bit in the port *m* control CSR.
- The FRU generates a maintenance write to its link partner's port *m* local ackID status CSR to set the link partner's outbound ackID value to 0. Upon receipt of the maintenance write, the link partner sets its outbound ackID value and generates the maintenance response using the new value. Note that if all packets intended for the link partner have not been discarded, or if any new packets intended for the link partner arrive, this step may fail (the RapidIO endpoint has no way to protect against this).
- The FRU completes configuration.
- The FRU enables all packets to be transmitted and received by setting its output and input port enable bits in the port *m* control CSR.
- System operation resumes.

19.6.13 Software re-training

A user can issue a software controlled re-training of the link in LP-Serial mode in order to update certain configurations by following the steps below.

- Software on the host must ensure that all RapidIO transactions have completed and link activity has stopped.
- Set the PnCCSR[PD] bit on the host to disable the port and force the initialization state machines to reset.
- Set the PnPCR[OBDEN] bit on the host to enable the discarding of any pending packets. (There should be none if proper steps were taken to ensure there was no link activity.)
- Clear the PnPCR[OBDEN] bit on the host.
- Configure new operating width (via PnCCSR[PWO]) or any other new configurations.
- Clear the PnCCSR[PD] bit on the host to re-enable the drivers.
- Poll the PnESCSR[PO] bit on the host until it is set, which indicates the link has attained port and link initialization.
- Poll the PnESCSR[PO] bit on the agent until it is set, which indicates the link has attained port and link initialization.
- Poll the PnESCSR[OES] bit on the agent until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
- Poll the PnESCSR[OES] bit on the host until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
- Clear the agent's error and status registers.
- Clear the host's error and status registers.
- Begin normal packet transfer.

19.6.14 Errors and error handling

This section describes how the logical and physical layers detect and react to RapidIO errors.

The action of the core on notification of any of these errors is described minimally here. See *RapidIO Interconnect Specification, Revision 1.2 Part VII (Error Management Extensions Specifications)* for more details on specific errors described below.

19.6.14.1 RapidIO error description

RapidIO errors are classified under three categories: recoverable errors, notification errors, and fatal errors.

Recoverable errors are non-fatal transmission errors (such as corrupt packet or control symbols, and general protocol errors) that RapidIO supports hardware detection of and a recovery mechanism for, as described in the *RapidIO Interconnect Specification*,

Revision 1.2. In these cases, the appropriate bit is set in the port *merror* detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by port *merror* enable CSR) is captured in the port *merror* capture CSRs. No interrupt is generated or actions required for a recoverable error. Recoverable errors are detected in the physical layer only.

Notification errors are non-recoverable non-fatal errors detected by RapidIO (such as degraded threshold, port-write received, and all logical/transport layer (LTL) errors captured). Because they are non-recoverable (and in some cases have caused a packet to be dropped), notification by interrupt is available. However, because they are non-fatal, response to the interrupt is not crucial to port performance; that is, the port is still functional. When a notification error is detected, the appropriate bit is set in the error-specific register, an interrupt is generated, and in some cases, the error is captured. In all cases, the RapidIO port continues operating. Notification errors are detected in both the physical and logical layer.

NOTE

To prevent processor stalls, the PmLOPTTLCR registers must be initialized to a non-zero value. See [Port 1 Logical Outbound Packet time-to-live configuration register \(SRIO_P1LOPTTLCR\)](#).

The RapidIO controller detects two fatal errors: exceeded failed threshold and exceeded consecutive retry threshold. In these cases, the port has failed because its recoverable error rate has exceeded a predefined failed threshold or because it has received too many packet retries in a row. In the first case, the controller sets the output failed-encountered bit in the port *merror* and status CSR; the RapidIO output hardware may or may not stop (based on stop-on-port-failed-encounter-enable and drop-packet-enable bits). In the second case, the controller sets the retry counter threshold trigger exceeded bit in the port *mimplementation* error CSR; the RapidIO hardware continues to operate. In both cases, an interrupt is generated, and while the port continues operating at least partially, a system-level fix (such as reset) is recommended to clean up the controller's internal queues and resume normal operation. Fatal errors are detected in the physical layer only.

19.6.14.2 Physical layer RapidIO errors detected

[Table 19-323](#) lists all the RapidIO link errors detected by the RapidIO endpoint physical layer and the actions taken by the RapidIO endpoint. The error enable column lists the control bits that may disable the error checking associated with a particular error (if blank, error checking cannot be disabled). The cause field column indicates what cause field is used with the associated packet-not-accept control symbol for input error recovery. The EME error enable/detect column indicates which bit of the PmERECSR

allows the error to increment the error rate counter and lock the port *mirror* capture registers, and likewise which bit of the PmEDCSR is set when the error has been detected.

[Table 19-324](#) lists the RapidIO endpoint behavior after exceeding certain preset limits (degraded threshold, failed threshold, retry threshold).

Table 19-323. Physical RapidIO errors detected

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
Recoverable Errors						
1a	Received character had a disparity error. (serial)	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character	Delineation Error	DE
1a	Received an invalid character, or valid but illegal character (serial)	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	The four control character bits associated with the received symbol do not make sense (not 0000, 1000, 1111) (serial)	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	Control symbol does not begin with an /SC/ or /PD/ control character. (serial)	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1c	Received packet with embedded idles. (serial)	-	Enter Input Error Stopped.	5: Received invalid/illegal character		
1d	Received a control symbol with a bad CRC	PmPCR[CC] enables detect.	Enter Input Error Stopped. Enter Output Error Stopped.	2: Received a control symbol with bad CRC	Received corrupt control symbol	CCS
1d	Missing start: Packet data received without previous SOP control symbol	-	Enter Input Error Stopped.	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE
1e	Received packet that is < 64 bits	-	Enter Input Error Stopped.	7/31: General error		
1e	Received an EOP control symbol when there is no packet being received.	-	Enter Input Error Stopped.	7/31: General error		
1e	Received a stomp control symbol when there is no packet being received.	-	Enter Input Error Stopped.	7/31: General error		

Table continues on the next page...

Table 19-323. Physical RapidIO errors detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2a	Received a Restart-from-retry control symbol when in the "OK" state	-	Enter Input Error Stopped	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE
2a	Received packet with a bad CRC value.	PmPCR[C CP] enables detect.	Enter Input Error Stopped.	4: bad CRC on packet.	Received packet with bad CRC	CRC
2a	Received packet which exceeds the maximum allowed size by the RapidIO spec.	-	Enter Input Error Stopped.	7/31: General error	Received packet exceeds 276 Bytes	EM
2b	Received packet with unexpected ackID value (out-of-sequence ACKID)	-	Enter Input Error Stopped.	1: Received unexpected ACKID on packet	Received packet with unexpected ackID	UA
2c	Received a non-maintenance packet when non-maintenance packet reception is stopped	Non-maint. packet reception is stopped when 'Input Port Enable' = 0.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	-
2d	Any packet received while Port Lockout bit is set	All packet reception is stopped when Port Lockout bit is set.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	-
-	Received a Link request control symbol before servicing previous link request.	Not detected.				
2a	Received packet-not-accepted ACK control symbol.	-	Enter Output Error Stopped.	-	Received packet-not-accepted symbol	PNA
2b	Received an ACK (accepted, or retry) control symbol when there are no outstanding packets	-	Enter Output Error Stopped.	-	Unsolicited ACK symbol	UCS
2b	Received packet ACK (accepted) for a packet whose transmission has not finished	-	Enter Output Error Stopped.	-		

Table continues on the next page...

Table 19-323. Physical RapidIO errors detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2b	Received a Link response control symbol when no outstanding request.	-	Enter Output Error Stopped.	-		
2c	Received an ACK (accepted or retry) control symbol with an unexpected ACKID.	-	Enter Output Error Stopped.	-	Received ack. control symbol with unexpected ackID	AUA
2c	Link_response received with an ackID that is not outstanding	-	Re-enter Output Error Stopped.	-	Non-outstanding ackID	NOA
2d	An ACK control symbol is not received within the specified time-out interval.	PLTOCCS R[TV] > 0 enables detect.	Enter Output Error Stopped.	-	Link time-out	LTO
2d	A Link response is not received within the specified time-out interval	PLTOCCS R[TV] > 0 enables detect.	(re-) Enter Output Error Stopped.	-		

Table 19-324. Physical RapidIO threshold response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt clear ¹
Notification Errors					
Error Rate Counter has met or exceeded the Degraded Threshold.	PmERTCSR[ERDTT] > 0 & any bit in PmERCSR enables detect and interrupt generation.	Generate Interrupt. Continue to operate normally.	Degraded Threshold	PmESCS R[ODE]	Write 1 to PmESCS R[ODE]
Fatal Errors					
Consecutive Retry Counter has met or exceeded the Retry Counter Threshold Trigger	PRETCR[RET] > 0 enables detect and interrupt generation	Generate Interrupt. Port is in priority order.	Consecutive Retry Threshold	PmIECSR[RETE]	Write 1 to PmIECSR[RETE]
Error Rate Counter has met or exceeded the Failed Threshold.	PmERTCSR[ERFTT] > 0 & any bit in PmERCSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on PmCCSR[SPF] and PmCCSR[DPE] -- port can continue transmitting packets or can stop sending output packets, keeping or dropping them.	Failed Threshold	PmESCS R[OFE]	Write 1 to PmESCS R[OFE].

- Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt.

19.6.14.3 Logical layer errors and error handling

This section describes how the logical layer detects and reacts to RapidIO errors.

The action of the core on notification of any of these errors is described minimally here. Reference *RapidIO Interconnect Specification, Revision 1.2 Part VII (Error Management Extensions Specifications)*.

19.6.14.3.1 Logical layer RapidIO errors detected

Table 19-325 through **Table 19-338** lists all the errors detected by the RapidIO endpoint logical layer and the actions taken by the RapidIO endpoint. Note that when the RapidIO endpoint action includes sending an error response to either OCN or RapidIO, an error response is only sent if the original transaction was a request that required a response. Otherwise, no error response is sent. When dealing with multiple errors, discard of packet has higher priority than error response.

For misaligned transactions, the error management extension registers are updated with each child.

All packet field positions are assumed to be in the mode (small or large transport) configured. For example, when configured for small transport mode and a large transport mode NREAD packet is received, the transaction type field bit positions checked correspond to a small transport type NREAD packet.

Table 19-325. Hardware errors for NRead transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of Read transaction is	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packet, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet	RapidIO packet is dropped.

Table continues on the next page...

Table 19-325. Hardware errors for NRead transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
				bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLCACCSR[XA] gets packet bits 94-95, LTLCACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when pass_through is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (pass_through or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Received RapidIO packet with reserved TType for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
RdSize	-	-	-	-	-

Table continues on the next page...

Table 19-325. Hardware errors for NRead transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Not Checked for error.					
SrcTID	-	-	-	-	-
Not Checked for error.					
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to Inbound window boundary crossing errors	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	OCN error response is generated to self
Address:WdPtr:Xambs Request hits a protected ATMU window	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
Address:WdPtr:Xambs Beginning address matches LCSBA1CSR with non 32 bit read request. Performed only when ttype == 4'b0100	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
Header Size Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Large Transport packet has 14 valid bytes and two bytes of padding of 0's. Padding of 0's is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
PayloadSize Not Applicable	-	-	-	-	-

Table 19-326. Hardware errors for maintenance read/write req transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of maintenance read or write request transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits	RapidIO packet is dropped

Table continues on the next page...

Table 19-326. Hardware errors for maintenance read/write req transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
				24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLCACCSR[XA] gets packet bits 94-95, LTLCACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCACCSR[FT] gets packet bits 12-15, LTLCACCSR[TT] gets packet bits 48-51, LTLCACCSR[MI] gets 0's	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Reserved Transaction Type for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	RapidIO packet is dropped

Table continues on the next page...

Table 19-326. Hardware errors for maintenance read/write req transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
RdSize Read/Write request size is not for 4 bytes	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
SrcTID Not Checked for error.	-	-	-	-	-
HopCount Not Checked for error.	-	-	-	-	-
Config Offset Not Checked for error.	-	-	-	-	-
Header Size Maintenance Read request - Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Maintenance Write request - total header size is not 12 Bytes for Small Transport packet or not 16 Bytes for Large Transport packet. Padding of 0's in last two bytes of Large Transport packet is not checked.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self
PayloadSize Write request with payload not equal to 8 bytes. Read request with payload not 0 bytes	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self

Table 19-327. Hardware errors for atomic (inc, dec, set, or clr) read transaction

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of read transaction is 3	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet	RapidIO packet is dropped

Table continues on the next page...

Table 19-327. Hardware errors for atomic (inc, dec, set, or clr) read transaction (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
				bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	OCN error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 19-327. Hardware errors for atomic (inc, dec, set, or clr) read transaction (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransactionType Non-Atomic type is tested with Nread	-	-	-	-	-
TransactionType Received Atomic Increment request with DOCAR[AI] disabled. Received Atomic decrement request with DOCAR[AD] disabled. Received Atomic Set request with DOCAR[AS] disabled. Received Atomic Clear request with DOCAR[AC] disabled.	Yes if LTLEECS R[UT] is set	LTLEDCSR[UT]	Yes	Same as second entry	Error response is generated to self
RdSize Not unsupported RdSize request is not for contiguous one, two or four bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
SrcTID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Not unsupported Request hits a protected ATMU window or the LCSBA1CSR Refer to Outbound window boundary crossing errors	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to Outbound window boundary crossing errors	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	Error response is generated to self
Header Size Not unsupported Header size is not 12 Bytes for small Transport packet and not 16 Bytes for Large Transport packet Padding of 0's in last two bytes of Large Transport packet is not checked	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self

Table 19-328. Hardware errors for NWrite, NWrite_r, and unsupported atomic test-and-swap transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments		
Priority Transaction priority is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTACCSR[XA] gets packet bits 78-79, LTACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 32-35, LTLCCSR[MI] gets 0's For Large Transport type packets. LTACCSR[XA] gets packet bits 94-95, LTACCSR[A] gets packet bits 64-92, LTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 48-51, LTLCCSR[MI] gets 0's			RapidIO packet is dropped.
Critical Request Flow Not Checked for error.	-	-	-	-	-		
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped		
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped		
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite		

Table continues on the next page...

Table 19-328. Hardware errors for NWrite, NWrite_r, and unsupported atomic test-and-swap transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Not applicable	-	-	-	-	-
TransactionType Received RapidIO packet for Atomic test-and-swap transaction	Yes if LTLEECS R[UT] is set	LTLEDCSR[UT]	Yes	Same as first entry	OCN error response is generated to self
TransactionType Received RapidIO packet with reserved TType for this ftype Packet is treated as Nwrite Transaction	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped
WrSize Not unsupported transaction WrSize request is for one of reserved sizes	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Nwrite request hits LCSBA1CSR	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No for Nwrite.	Same as first entry	RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Write request hits overlapping ATMU windows Refer to Outbound window boundary crossing errors	Yes if LTLEECS R[IACB] is set	LTLEDCSR[IACB]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite

Table continues on the next page...

Table 19-328. Hardware errors for NWrite, NWrite_r, and unsupported atomic test-and-swap transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
SrcTID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Nwrite_r address matches LCSBA1CSR with non 32 bit read request. Performed only for Nwrite_r packet	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r.	Same as first entry	OCN error response is generated to self
Header Size Not unsupported transaction Header size is less than 12 bytes for small Transport packet or less than 16 bytes for Large Transport packet - that is, no payload present. Large Transport packet has 14 valid bytes and 2 bytes of padding of 0s. Padding of 0s is not checked.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
PayloadSize Not unsupported transaction Payload is greater than that indicated by {wdptr:wrsiz} field, payload is not double word aligned or does not have any payload	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	OCN error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite

Table 19-329. Hardware errors for SWrite transactions

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Swrite transaction priority is 3	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
Critical Request Flow Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 19-329. Hardware errors for SWrite transactions (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTACCSR[XA] gets packet bits 62-63, LTACCSR[A] gets packet bits 32-60, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 32-35, LTLCCSR[MI] gets 0's For Large Transport type packets. LTACCSR[XA] gets packet bits 78-79, LTACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 48-51, LTLCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to Outbound window boundary crossing errors	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as third entry	RapidIO packet is dropped

Table continues on the next page...

Table 19-329. Hardware errors for SWrite transactions (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Address:WdPtr:Xambs Request hits a protected ATMU window or the LCSBA1CSR	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
PayloadSize Payload size is not in DWs, has exceeded 256 bytes or has no payload.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped

Table 19-330. Hardware errors for maintenance response transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments	
Priority Not UR Response priority is not higher than RapidIO maintenance request priority	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's		RapidIO packet is dropped and ignored
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored	
Received TT which is not enabled. - Error valid when passthrough	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored	

Table continues on the next page...

Table 19-330. Hardware errors for maintenance response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
is disabled and accept_all is disabled Or when accept_all is enabled.					
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECS R[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR Received RapidIO packet with reserved TType for this ftype	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR Maintenance read/write response does not correspond to an outstanding valid message read/write request.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
HopCount Not Checked for error.	-	-	-	-	-
Status Not UR Is not "Done" or "Error" Not "Done" status for "read_response" transaction type with payload "Error" status with payload.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR Error Response	Yes if LTLEECS R[IER] is set	LTLEDCSR[IER]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.

Table continues on the next page...

Table 19-330. Hardware errors for maintenance response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
Header Size Not UR Maintenance Read response - total payload size with done status is not greater than 4 Bytes. Maintenance Write response - total header size is less than 12 Bytes for Small Transport packet or is less than 16 Bytes for Large Transport packet. Padding of 0's for Small or Large Transport packets is not verified.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
PayloadSize Not UR Maintenance write response has payload. Maintenance read response with done status and payload not matching valid request size, request size for the response is invalid or payload size is not dword aligned.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Packet response time-out Response is not received by configured time	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.

Table 19-331. Hardware errors for IO/GSM response transactions (not maintenance)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO request priority	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTACCSR[XA] gets packet bits 78-79 (if available), LTACCSR[A] gets packet bits 48-76 (if available), LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 32-35, LTLCCSR[MI] gets 0's For Large Transport type packets. LTACCSR[XA] gets packet bits 94-95 (if available), LTACCSR[A] gets packet bits 64-92 (if available), LTTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 48-51, LTLCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
TransportType Received reserved TT for this ftype	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 19-331. Hardware errors for IO/GSM response transactions (not maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Does not match the request's DestID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR Received RapidIO packet with reserved TType	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR IO read response does not correspond to an outstanding valid IO/GSM read request. IO write response does not correspond to an outstanding valid IO/GSM write request.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR IO transaction - Is not "Done" or "Error" GSM transaction IO_Read_Home Is not "Done - Data-Only", "Done - Done-Intervention", "Done", "Retry" or "Error". Flush_w_Data response is not "Done", "Retry" or "Error" Transaction type of "Response_with_data" and status is not done	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Status Not UR GSM Error response	Yes if LTLEECS R[GER] is set	LTLEDCSR[GER]	Yes if data is not received for this request.	Same as first entry except error capture is done from original request packet.	OCN error response is generated to requestor if data is not forwarded to it. Else the RapidIO packet is dropped.

Table continues on the next page...

Table 19-331. Hardware errors for IO/GSM response transactions (not maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not UR IO Error Response	Yes if LTLEECS R[IER] is set	LTLEDCSR[IER]	Yes	Same as first entry except error capture is done from original request packet.	OCN error response is generated to requestor.
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR (All non-maintenance and non-message) Write response - Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet GSM - "Done" response packet size to "Flush" is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet. "Done-Intervention" is not 8 Bytes for Small Transport and 12 Bytes for Large Transport field. Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Payload Size Not UR IO - Read Response - total payload is not of the size requested. "Done" or "Done-Data_Only" response to IO_Read_Home with incorrect payload size. Response with transaction type "response_with_no_data" has payload	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table continues on the next page...

Table 19-331. Hardware errors for IO/GSM response transactions (not maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Retry Not UR GSM request has had one more than configured number of retries for non misaligned request. The misaligned GSM request has had one to four (cumulative for the corresponding child requests) more than configured number of retries.	Yes if LTLEECS R[RETE] is set	LTLEDCSR[RETE]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. "GSM - IO_Read_Home" - Done_With_Data is not received in configured time when Done_Intervention is received for non misaligned request or last child of misaligned request. Done response is not received in configured time for non misaligned request or last child of misaligned request. EME capture occurs for each child packet response time-out.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request.	OCN error response is generated to requestor.
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. "GSM - IO_Read_Home" - Done_Intervention is not received in configured time when Done_With_Data is received. This is true for both non misaligned or misaligned requests.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	No	Same as first entry except error capture is done from original request.	An OCN done response is generated when the Done_With_Data is received for non misaligned requests or the last child of a misaligned request.

Table continues on the next page...

Table 19-331. Hardware errors for IO/GSM response transactions (not maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
					Therefore, an error response cannot be sent when the packet response time-out occurs.
GSM - IO_Read_Home Not UR Done response, Retry response, or Error response is after Done_Intervention response or Data_only is received.	Yes if LTLEECS R[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Not UR Response for OCN packets not requiring response, but converted to "response" type packet by ATMU receives Error response. For example, NWrite converted to NWrite_r received "Error" response	Yes if LTLEECSR[IER]/ [GER]/ [RETE] is set	LTLEDCSR[IER]/ [GER]/ [RETE]	No	Same as first entry except error capture is done from original request.	RapidIO packet is dropped and ignored.
Response for OCN packets not requiring response, but converted to "response" type packet by ATMU is not received by configured time.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	No	Same as first entry except error capture is done from original request.	No error response is generated.

Table 19-332. Hardware errors for DMA message response transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO request priority	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31,	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 19-332. Hardware errors for DMA message response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
				LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets packet bits 40-47 For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets packet bits 56-63	
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECS R[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not checked. To be a message response TType has to be 0x1	-	-	-	-	-

Table continues on the next page...

Table 19-332. Hardware errors for DMA message response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not UR DMA Message Error response	Yes if LTLEECS R[DMAME R] is set	LTLEDCS R[DMAME R]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.
Status Not UR Received status of reserved type	Yes if LTLEECS R[ITD] is set	Yes if LTLEDCS R[ITD] is set	No	Same as first entry	RapidIO packet is dropped and ignored
No outstanding transaction for this letter, mailbox and message segment	Yes if LTLEECS R[UR] is set	LTLEDCS R[UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR (All non-maintenance and non-message) DMA response - Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECS R[ITD] is set	LTLEDCS R[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Payload Size Not UR Payload size is not zero	Yes if LTLEECS R[ITD] is set	LTLEDCS R[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Retry Not UR DMA request - has had more than configured number of retries	Yes if LTLEECS R[RETE] is set	LTLEDCS R[RETE]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.
Packet response time-out Response is not received by configured time.	Yes if LTLEECS R[PRT] is set	LTLEDCS R[PRT]	Yes	Same as first entry except error capture is done from original request	OCN error response is generated to requestor.

Table 19-333. Hardware errors for message request transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	-	-	-	-	-
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	<p>Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 32-35, LTLCCSR[MI] gets packet bits 40-47.</p> <p>For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 48-51, LTLCCSR[MI] gets packet bits 56-63.</p>	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 19-333. Hardware errors for message request transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
MsgLen,Ssize,Ltr,Mbox,MsgSeg Not Checked for error.	-	-	-	-	-
PayloadSize Message payload size is larger than the specified ssize, or is of size 0 when seg_len == msg_len. Or Message payload size is not equal to specified ssize when seg_len != msg_len.	Yes if LTLEECSR[MFE] is set	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped	Same as third entry.	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
Reserved ssize field	Yes if LTLEECSR[MFE] is set	LTLEDCSR[MFE]	Yes if priority is not 3. Else packet is dropped	Same as third entry.	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
Other Received Message request with SOCAR[M] disabled	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.

Table 19-334. Hardware errors for message response transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Checked for error.	-	-	-	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits	-

Table continues on the next page...

Table 19-334. Hardware errors for message response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
				12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets packet bits 40-47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLLTLDIDCCSR[DIDMSB] gets bits 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets packet bits 56-63.	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
DestID (All non-maintenance) DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECS R[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
SourceID Not Checked for error.	-	-	-	-	-
Status Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 19-334. Hardware errors for message response transactions (continued)

Error	Interrupt Generate d	Status Bit Set	OCN Error Response Generate d	Logical/Transport Layer Capture Register	Comments
Other Received Message response with SOCAR[M] disabled.	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored

Table 19-335. Hardware errors for doorbell request transaction

Error	Interrupt Generate d	Status Bit Set	Error Response Generate d	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	-	-	-	-	-
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped

Table continues on the next page...

Table 19-335. Hardware errors for doorbell request transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	-	-	-	-	-
SrcTID Not Checked for error.	-	-	-	-	-
Other Received Doorbell request with DOCAR[D] disabled.	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	Yes if priority is not 3. Else packet is dropped	Same as third entry	OCN error response is sent to self if request priority is not 3. Else packet is dropped.

Table 19-336. Hardware errors for doorbell response transactions

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR or UT Response priority is not higher than RapidIO request priority	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92,	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 19-336. Hardware errors for doorbell response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
				LTLTLTDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECS R[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR or UT Not one of Done/Error/Retry	Yes if LTLEECS R[ITD] is se	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
TransactionType Not UR or UT Anything other than Done_No_Data	Yes if LTLEECS R[ITD] is se	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 19-336. Hardware errors for doorbell response transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECS R[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR or UT Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet Note: Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet response time-out Response is not received by configured time	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except capture registers are loaded from original request RapidIO packet.	OCN doorbell PRT response is generated to requestor.

Table 19-337. Hardware errors for PortWrite transaction

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	-	-	-	-	-
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 32-35, LTLCCSR[MI] gets 0's	RapidIO packet is dropped

Table continues on the next page...

Table 19-337. Hardware errors for PortWrite transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
				Large Transport type packets. LTACCSR[XA] gets packet bits 94-95, LTACCSR[A] gets packet bits 64-92, LTTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCSR[FT] gets packet bits 12-15, LTLCCSR[TT] gets packet bits 48-51, LTLCCSR[MI] gets 0's	
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Not Checked for error.	-	-	-	-	-
WrSize Not unsupported transaction Is one of reserved sizes or less than 4 bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
SrcTID Not Checked for error.	-	-	-	-	-
HopCount Not Checked for error.	-	-	-	-	-
ConfigOffset Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 19-337. Hardware errors for PortWrite transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
PayloadSize Not unsupported transaction An incorrect port-write wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes). Payload size is greater than the value defined by wr_size. Payload size is not dword aligned when the wr_size is not 4 bytes.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
Other Received PortWrite transaction with DOCAR[PW] disabled.	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	No	Same as third entry	RapidIO packet is dropped

Table 19-338. Hardware errors for reserved Ftype

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Ftype Ftype is not IO Read, IO Write, SWrite, Maintenance request, Maintenance Response, Response (Ftype 13), Doorbell or Message class and it is not a passthrough transaction. (passthrough is not enabled or accept_all is enabled or transaction is addressed to this port)	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped

Table continues on the next page...

Table 19-338. Hardware errors for reserved Ftype (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECS R[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped
Address:WdPtr:Xambs Request hits overlapping ATMU windows. Refer to Outbound window boundary crossing errors Packet checked as non Swrite packet	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as first entry	RapidIO packet is dropped
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window or the LCSBA1CSR Packet checked as non Swrite packet	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped

Table 19-339. Hardware errors for outbound transaction crossed ATMU boundary

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
OCN Address and payload size OCN address range for Outbound RapidIO transaction hits multiple ATMU windows. Refer to Outbound window boundary crossing errors	Yes if LTLEECSR[OACB] is set and/or LTLEECSR[PRT] is set	LTLEDCSR[OACB], LTLEDCSR[PRT]		Using the original request RapidIO packet send out by OB, for small Transport type, LTLCACCSR[XA] gets packet bits 78-79, LTLCACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCACCSR[FT] gets packet bits 12-15, LTLCACCSR[TT] gets packet bits 32-35, LTLCACCSR[MI] gets 0's For Large Transport type packets, LTLCACCSR[XA] gets packet bits 94-95, LTLCACCSR[A] gets packet bits 64-92, LTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCACCSR[FT] gets packet bits 12-15, LTLCACCSR[TT] gets packet bits 48-51, LTLCACCSR[MI] gets 0's	-

Table 19-340. Hardware errors for outbound packet time-to-live errors

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
Packet time-to-live error	Yes if LTLEECSR[PTTL] is set	LTLEDCSR[PTTL]		Using the RapidIO packet attempted to send outbound, if configured in small transport mode, LTLCACCSR[XA] gets packet bits 78-79, LTLCACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCACCSR[FT] gets packet bits 12-15, LTLCACCSR[TT] gets packet bits 32-35, LTLCACCSR[MI] gets 0's For large transport mode, LTLCACCSR[XA] gets packet bits 94-95, LTLCACCSR[A] gets packet bits 64-92, LTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47,	-

Table 19-340. Hardware errors for outbound packet time-to-live errors

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
				LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	

19.7 RapidIO message unit

This message unit supports multicast and non-multicast single segment messages and non-multicast multiple segment messages.

19.7.1 Message unit overview

The RapidIO message unit supports a message passing programming model for inter-processor and inter-device communication.

This model enables a producer to send a message across the interconnect fabric to a consumer's message hardware, called a mailbox. The receiving mailbox hardware places the message in a queue located in local memory. A message may consist of one to sixteen segments. When a configured number of messages have been received, an interrupt (if enabled) is generated to the interrupt controller for the processor to process the messages. Messages can be queued for transmission in the producer's memory, and the message hardware will process them sequentially. Messages can also be queued in the consumer's memory while software processes them sequentially. The depths of the queues in the producer and consumer are configurable by software. A multicast function allows single segment messages to be sent to multiple consumers.

The message unit is compliant with the message passing logical specification contained in the *RapidIO Interconnect Specification, Revision 1.2* (but assumes the msgseg field is redefined for more mailboxes when single segment messages are being sent). The most common use of the message passing model is in systems where a processing element is only allowed to access memory that is local to itself, and communication between processing elements is achieved through message passing and communication is address independent.

Each inbound message controller has a dedicated interrupt that can be used to notify software that a configured number of messages have been received. Similarly, each outbound message controller has a dedicated interrupt that can be used to notify software that there are no more messages for the outbound mailbox controller to process.

The message controller is managed through a set of run-time registers.

19.7.2 Message unit features

- Support for two outbound message controllers with the following features
 - chaining and direct modes
 - extended mailboxes (XMBOX) for single segment messages
 - multicast up to 32 RapidIO destinations for single segment messages
 - transmitting to any mailbox and extended mailbox for a single segment message (letter 3 is reserved for an alternative message source like the DMA but can be used if the DMA does not generate messages and the RapidIO endpoint is configured appropriately)
 - transmitting to any mailbox for multi segment message (letter 3 is reserved for an alternative message source like the DMA, but can be used if the DMA does not generate messages and the RapidIO endpoint is configured appropriately)
 - segment size up to 256 bytes
 - up to 16 segment messages with a total payload of up to 4 Kbytes
 - one entire message (up to a 16-message segments) can be transmitted before receiving any response
 - one entire single segment message to all multicast destinations (up to 32) can be transmitted before receiving any response
 - all message segment transfers for a message transaction must complete before the next message transaction begins
 - pipelined transmission of a full message in each message controller but all responses must be received before the next message can be transmitted
 - in chaining mode the next descriptor can be fetched before the current message completes (descriptor prefetching)
- Support for two inbound message controllers with the following features
 - reception of any mailbox and letter for a single or multi segment message
 - segment size up to 256 bytes
 - up to 16 segment messages with a total payload of up to 4 Kbytes
 - full inbound line rate performance
 - back-to-back message reception of the same or lower priority
 - out-of-order message segment reception
 - concurrent inbound message controller operation

19.7.3 Outbound modes of operation

- Direct mode: Software is expected to program all the necessary registers for sending an outbound message.
- Chaining mode: A descriptor that describes the message information is fetched from local memory before a message is sent.
- Multicast mode: A single segment message (256 bytes or less) is sent to multiple destinations. This mode is supported in direct or chaining mode.

The following sections describe the structure and operation of the outbound message controller and inbound message controller hardware in the data message controller.

19.7.4 Outbound message controller operation

The outbound message controller is responsible for sending messages stored in local memory.

The outbound message controller supports three modes of operation, direct, chaining, and multicast mode. In direct mode, software programs the necessary registers to point to the beginning of the message in memory. In chaining mode, software programs the necessary registers to point to the beginning of the first valid descriptor in memory. The descriptor provides all the necessary registers to start the message transfer. In multicast mode, a single segment message can be sent to multiple destinations. Multicast mode is supported in direct or chaining mode.

Each outbound message controller uses a unique letter number. For example, if there are two outbound message controllers, message controller 0 uses letter 0 and message controller 1 uses letter 1. The mailbox number is programmable through the destination port register, OM_nDPR. Each message controller can only operate on one message at a time.

19.7.4.1 Direct mode operation

In direct mode (OM_nMR[MUTM] is set in the outbound message mode register) the outbound message controller does not read descriptors from memory, but instead uses the current parameters programmed in the outbound message controller registers to start the transfer.

In direct mode, software is responsible for initializing all the parameters in all the necessary registers to start the message transmission. The message transfer is started when the outbound message controller start bit, OM_nMR[MUS], in the outbound message mode register transitions from a 0 to 1 and the outbound message controller is not already busy. If the outbound message controller is already busy, OM_nMR[MUS] transitioning from 0 to 1 is ignored. Software is expected to program all the appropriate registers before setting OM_nMR[MUS].

There are many ways in which software can interact with the message controller. One sequence of events to start and complete a transfer in direct mode is as follows:

- Poll the status register message unit busy bit, OM_nSR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the status bits (OM_nSR[MER], OM_nSR[RETE], OM_nSR[PRT], OM_nSR[TE], OM_nSR[QOI], OM_nSR[QFI], OM_nSR[EOMI], and OM_nSR[QEI])
- Initialize the source address (EOM_nSAR, OM_nSAR), destination port (OM_nDPR), destination attributes (OM_nDATR), retry error threshold (OM_nRETCR), and double-word count (OM_nDCR) registers. If multicast mode is enabled (OM_nDATR[MM]) initialize the multicast group and list (OM_nMGR, OM_nMLR).
- Initialize the outbound message mode register message unit transfer mode bit, OM_nMR[MUTM] = 1, to indicate direct mode. Other control parameters must also be initialized in the mode register.
- Clear, then set the mode register message unit start bit, OM_nMR[MUS], to start the message transfer.
- OM_nSR[MUB] bit is set by the outbound message controller to indicate the message transfer is in progress.
- The outbound message controller reads a message segment from local memory using the source address register (EOM_nSAR, OM_nSAR).
- If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
- After the message read to local memory completes, the message is sent.
- The OM_nSR[MUB] is cleared by the outbound message controller after the message operation completes. A non multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
 - done response received
 - error response
 - a packet response time-out received

- retry error threshold exceeded
- an internal error occurs during the local memory access
- After the outbound message operation completes, the outbound message interrupt is generated if the end of message outbound message interrupt event is enabled (OMnDATR[EOMIE]).

19.7.4.1.1 Direct mode interrupts

The outbound message controller interrupt can be generated for one reason in direct mode.

- End-Of-Message - generates an interrupt after the completion of a message if this interrupt event is enabled (OMnDATR[EOMIE] is a 1). OMnSR[EOMI] provides the event that caused this interrupt. The interrupt is held until the OMnSR[EOMI] bit is cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons in direct mode.

- message error response - an interrupt is generated after a message error response is received and this interrupt event is enabled (OMnMR[EIE])
- packet response time-out - an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (OMnMR[EIE])
- retry error threshold exceeded - an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (OMnMR[EIE])
- transaction error - an interrupt is generated after an OCN error response is received and this interrupt event is enabled (OMnMR[EIE])

19.7.4.1.2 Direct mode message error response errors

When a message error response is received by the message controller the following occurs.

- The message controller sets the message error response status bit (OMnSR[MER])
- If OMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by OMnSR[MUB]) the message controller stops

19.7.4.1.3 Direct mode packet response time-out errors

When a packet response time-out occurs for a message segment the following occurs.

- The message controller sets the packet response time-out status bit (OMnSR[PRT])

- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops

19.7.4.1.4 Direct mode retry error threshold exceeded errors

When a retry error threshold exceeded error occurs for a message segment the following occurs.

- The message controller sets the retry threshold exceed status bit (OM_nSR[RETE])
- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops

19.7.4.1.5 Direct mode transaction errors

When an internal error occurs during a local memory read by the message controller the following occurs.

- The message controller sets the transaction error bit (OM_nSR[TE])
- Message segments that have an internal error are not sent because the message data is not available
- Memory reads that already were generated before the internal error occurred are also not transferred
- Additional memory reads for the same message operation are generated but not transferred
- All subsequent message segments for the same message operation are not transferred. This includes retried message segments.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops
- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated

19.7.4.1.6 Direct mode error handling

When an error occurs and the SRIO error/port-write interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling OM_nSR[MUB]
- Software disables the message controller by clearing OM_nMR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM_nSR[MER], OM_nSR[PRT], OM_nSR[RETE], or OM_nSR[TE])

When an error occurs and the SRIO error/port-write interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OMnSR[MER], OMnSR[PRT], OMnSR[RETE], or OMnSR[TE])
- Software verifies the message controller has stopped operation by polling OMnSR[MUB]
- Software disables the message controller by clearing OMnMR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OMnSR[MER], OMnSR[PRT], OMnSR[RETE], or OMnSR[TE])

19.7.4.1.7 Disabling and enabling the message controller

Once the message controller is started, it cannot be stopped.

19.7.4.1.8 Direct mode hardware error handling

The following list possible error conditions and what occurs when they are encountered.

The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port given in [Errors and error handling](#).

Table 19-341. Outbound message direct mode hardware errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Request	An internal error occurred during a read of the message segment from local memory	1	SRIO error/port-write if OMnMR[EI] set	Transaction error in the outbound message status register (OMnSR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]).	No	-	Message controller stops after the current message operation completes. The descriptor dequeue pointer is not incremented in chaining mode.
Message Request	An internal error occurred for an earlier message	2	No	None	Yes	-	Message controller stops after the current

Table continues on the next page...

Table 19-341. Outbound message direct mode hardware errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
	segment local memory read. An internal error for a subsequent message segment local memory read for the same message may or may not occur.						message operation completes.
Undefined Packet	Reserved ftype encoding ¹	3	SRIO error/port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	Reserved tt encoding ¹	3	SRIO error/port-write if LTLEECSR[TSE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE] Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	Large transport size when operating in small transport size or small transport size when operating in large transport size ¹	3	SRIO error/port-write if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	Yes	Updated with the packet ²	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Message Response	Illegal Destination ID ¹	3	SRIO error/port-write if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	ttype (transaction field) is not message response ¹	3	SRIO error/port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.

Table continues on the next page...

Table 19-341. Outbound message direct mode hardware errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Response	Message response received and no outbound mailboxes are supported ¹	3	SRIO error/port-write if LTLEECSR[UR] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	No	Updated with the packet ²	Packet is ignored and discarded.
Message Response	reserved response status (not done, retry, or error)	4a	SRIO error/port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	message response packet size is incorrect	4a	SRIO error/port-write if LTLEECSR[ITD] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	Incorrect Source ID	4b	SRIO error/port-write if LTLEECSR[UR] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	letter, mbox and msgseg not outstanding or letter, mbox and xmbox not outstanding	4b	SRIO error/port-write if LTLEECSR[UR] set	Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	RapidIO priority is less than or equal to message request	4c	SRIO error/port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Message Response	error response	5	SRIO error/port-write if LTLEECSR[MER] set. SRIO error/port-write if OMnMR[EI E].	Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. OMnSR[MER] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet ²	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

Table continues on the next page...

Table 19-341. Outbound message direct mode hardware errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Response	number of retries exceeds limit	5	SRIO error/port-write if LTLEECS R[RETE] set. SRIO error/port-write if OMnMR[EIE].	Retry error threshold exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] OMnSR[RETE] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet ²	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.
Message Response	packet response time-out	unrelated	SRIO error/port-write if LTLEECS R[PRT] set. SRIO error/port-write if OMnMR[EIE].	Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT] . OMnSR[PRT] bit set if in direct mode or if in chaining mode.	Yes	Updated with the corresponding message request packet ² . The LTLDIDCCSR[SIDMSB] and LTLDIDCCSR[SID] field is 0.	Message segment transfer complete. The descriptor dequeue pointer is not incremented in chaining mode.

1. These error types are actually detected in the RapidIO port, not in the message controller
2. If operating in small transport size configuration using the packet LTACCSR[XA] gets the extended address (packet bits 78-79), LTACCSR[A] gets the address (packet bits 48-76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24-31), LTLCCSR[FT] gets the ftype (packet bits 12-15), LTLCCSR[TT] gets the ttype (packet bits 32-35), LTLCCSR[MI] gets the msg info (packet bits 40-47). If operating in large transport size configuration using the packet LTACCSR[XA] gets the extended address (packet bits 94-95), LTACCSR[A] gets the address (packet bits 64-92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24-31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32-39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40-47), LTLCCSR[FT] gets the ftype (packet bits 12-15), LTLCCSR[TT] gets the ttype (packet bits 48-51) if the message request packet is captured or 0 if the message response packet is captured, LTLCCSR[MI] gets the msg info (packet bits 56-63).

19.7.4.1.9 Direct mode programming errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 19-342. Outbound message direct mode programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Double-word count greater than 256 bytes when multicast mode selected	No	None	Undefined operation results

Table continues on the next page...

Table 19-342. Outbound message direct mode programming errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Double-word count set to a reserved value	No	None	Undefined operation results
Transaction flow level set to 3	No	None	Undefined operation results
Target interface set to an invalid RapidIO port	No	None	Undefined operation results
Source address for message read is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Address for error enqueue address pointer is invalid	No	No	Local memory captures the transaction and generates an interrupt.
Register values changed during operation	No	No	Undefined operation results

19.7.4.2 Chaining mode

In chaining mode, OM_nMR[MUTM] = 0 in the outbound message mode register, message descriptors are built in local memory in a circular queue.

There are several options available to the programmer in chaining mode. Software can build one or more descriptors in memory before initializing the outbound message controller registers, or it can initialize the outbound message controller registers and then build the descriptors. The enqueue pointer (OM_nDQEPR, EOM_nDQEPR) is maintained by software. The outbound message controller dequeues descriptors, processes them and increments the dequeue pointer (OM_nDQDPAR, EOM_nDQDPAR) to point to the next descriptor in the queue.

[Figure 19-321](#) depicts a sample structure of the outbound portion of the message controller, and a descriptor queue, with each valid descriptor queue entry pointing to a valid message. In this example, the descriptor queue has eight entries, four of which are currently valid. The local processor enqueues descriptors and the outbound message controller dequeues the descriptors.

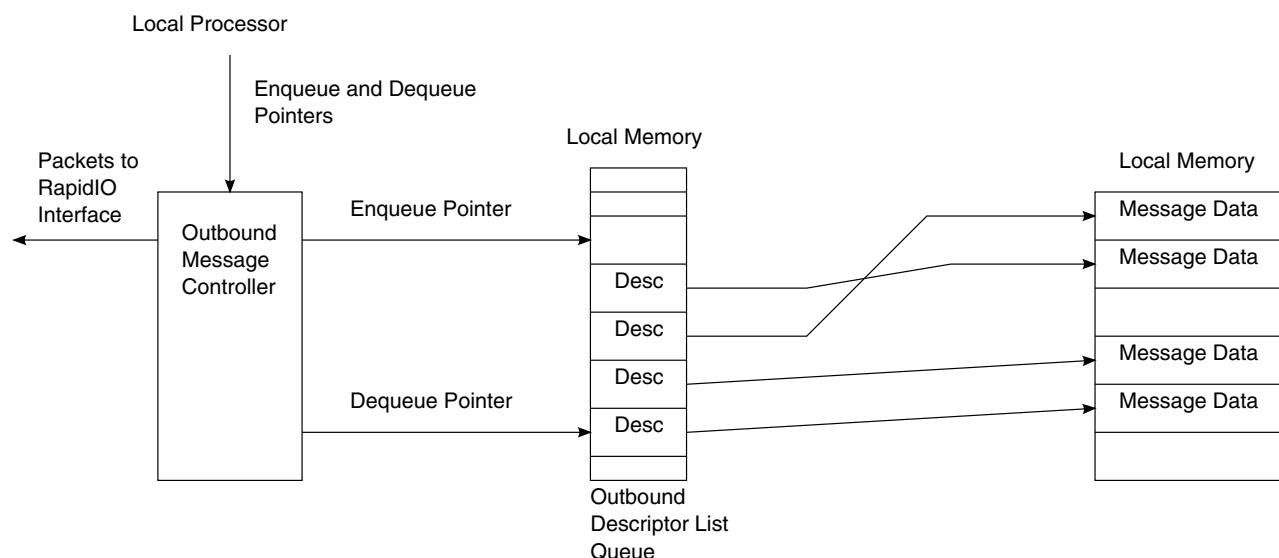


Figure 19-321. Outbound frame queue structure

19.7.4.2.1 Message controller initialization

There are many ways in which software can interact with the message controller.

One method to initialize the message controller is as follows:

- Poll the status register message unit busy bit, OM_nSR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the message unit start bit (OM_nMR[MUS]).
- Initialize the descriptor queue dequeue pointer address registers (OM_nDQDPAR, EOM_nDQDPAR) and the descriptor queue enqueue pointer address registers (OM_nDQEPR, EOM_nDQEPR). These need to be initialized to the same value for proper operation.
- These pairs of registers must also be queue size aligned (that is, the queue must be aligned on a boundary equal to number of queue entries x 32 bytes (size of each queue descriptor)). For example, if there are 16 entries in the queue, the queue must be 512-byte aligned.

The number of queue entries is set in OM_nMR[CIRQ_SIZ]; see [Outbound message n mode register \(SRIO_OM_nMR\)](#).

- Initialize the retry error threshold in the outbound message retry error threshold configuration register (OM_nRETCR).
- Clear OM_nMR[MUTM] for chaining mode.
- If using single segment multicast mode, set OM_nDATR[MM].
- Configure the other control parameters in the mode register (OM_nMR).
- Clear OM_nSR[MER, PRT, RETE, TE, QOI, QFI, EOMI and QEI]. If OM_nSR[MER, PRT, RETE, TE or QOI] are not cleared, the message controller can not start a new

message operation. Incorrect status will be indicated if the other status bits are not cleared.

- Set the message unit start ($OMnMR[MUS]$). This enables the outbound message controller and causes the descriptor dequeue pointer ($OMnDQDPAR$, $EOMnDQDPAR$) to be saved off as the base address of the descriptor queue.

19.7.4.2.2 Chaining mode operation

The method to start and complete transfers by adding descriptors after initializing the message unit is as follows:

- Create one or more descriptors in local memory starting at the address pointed to by the descriptor queue enqueue pointer address register ($OMnDQEPR$, $EOMnDQEPR$)
- Either increment the enqueue pointer address registers ($OMnDQEPR$, $EOMnDQEPR$) by setting $OMnMR[MUI]$ for each descriptor entry added or directly change the enqueue pointer address register ($OMnDQEPR$). If $OMnMR[MUI]$ is set by software, the message controller clears this bit after successfully incrementing the enqueue pointer.
- When the descriptor queue is not empty, the message controller reads the descriptor from local memory using the address pointed to by the dequeue pointer ($OMnDQDPAR$, $EOMnDQDPAR$) and sets the busy bit ($OMnSR[MUB]$).
- If another descriptor is available, the message controller reads the next descriptor from local memory using the address pointed to by the dequeue pointer ($OMnDQDPAR$, $EOMnDQDPAR$). The message controller can not prefetch more than one descriptor.
- The message controller sets $OMnSR[MUB]$ to indicate that the message transfer is in progress. $OMnSR[MUB]$ remains set until the descriptor queue is empty or a transaction error occurs.
- Additional descriptors can be created and enqueued by software while the message controller is busy ($OMnSR[MUB]$). Software can continue adding descriptors as long as the descriptor queue is not full. If software is adding descriptors using the $OMnMR[MUI]$ bit, overflowing the queue can be prevented by polling the queue full bit ($OMnSR[QF]$) before creating and enqueueing the next descriptor.
- After the descriptor memory read completes, the corresponding message segment is read from local memory.
- If a message has multiple segments, the outbound message controller reads the other message segments from local memory.
- After the message read to local memory completes, the message is sent.
- If multicast is enabled, all of the indicated targets are sent the same message.

- A non multicast message transfer completes after all message segments complete. A multicast message transfer completes after all message segments complete for each destination. A message segment completes when one of the following occurs:
 - done response received
 - error response received
 - a packet response time-out occurred
 - retry error threshold exceeded
 - an internal error occurred during the descriptor (all message segments complete) or message read of local memory
- When processing for the current descriptor completes, if another descriptor is available the above steps are repeated.
- If a RapidIO error response is received, the message error response bit is set (OMnSR[MER]) and outbound message controller operation stops after all message segments complete. If OMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- If a packet response time-out occurs, the packet response time-out bit is set (OMnSR[PRT]). If OMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- If the retry error threshold value is exceeded for a specific segment, the retry error threshold exceeded bit is set (OMnSR[RETE]) and outbound message controller operation stops after all message segments complete. If OMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- If an internal error occurs while reading local memory, the transaction error bit is set (OMnSR[TE]) and outbound message controller operation stops after all message segments complete. If OMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- The above process continues until the descriptor queue is empty (dequeue pointer equals the enqueue pointer).
- The message unit clears OMnSR[MUB] after completing the processing of the last descriptor or a transaction error occurs.
- If an error occurs the message unit must be disabled, re-initialized and re-enabled before another message can be sent.

19.7.4.2.3 Changing descriptor queues in chaining mode

When software wants to switch to another descriptor queue in local memory, it must wait for the processing of the current queue to complete as indicated by the busy bit (OMnSR[MUB]).

Software then disables the message controller by clearing OM_nMR[MUS], changes the enqueue and dequeue descriptor pointers (EOM_nDQEPAR, OM_nDQEPAR and EOM_nDQDPAR, OM_nDQDPAR) and re-enables the message unit by setting OM_nMR[MUS].

19.7.4.2.4 Preventing queue overflow in chaining mode

Software must guarantee that descriptors are not added to an already full queue.

When the increment bit is used (OM_nMR[MUI]) software can poll the queue full bit (OM_nSR[QF]) before enqueueing another descriptor. When software sets the enqueue pointer directly, software is responsible for not overflowing the descriptor queue.

19.7.4.2.5 Switching between direct and chaining modes

The message unit architecture allows switching from direct mode to chaining mode and vice-versa once all the required parameters have been initialized in the appropriate registers and when the message unit is not busy with a current transfer as indicated by OM_nSR[MUB] being cleared.

When switching from direct mode to chaining mode, if OM_nMR[MUS] is cleared and set, the message unit is re-initialized in chaining mode, and the outbound message descriptor queue dequeue pointer address is saved off as the new base address of the circular queue in memory. When switching from chaining mode to direct mode, OM_nMR[MUS] must also be cleared and set.

19.7.4.2.6 Chaining mode descriptor format

The descriptor contains information for the message unit controller to transfer data.

Software must ensure that each descriptor is aligned on a 32-byte boundary and that the descriptor queue is on a queue boundary (in other words, on a boundary equal to number of queue entries x 32 bytes (size of each queue descriptor)). For each descriptor in the queue, the message unit controller starts a new message operation with the control parameters specified by the descriptor.

Table 19-343. Outbound message unit descriptor summary

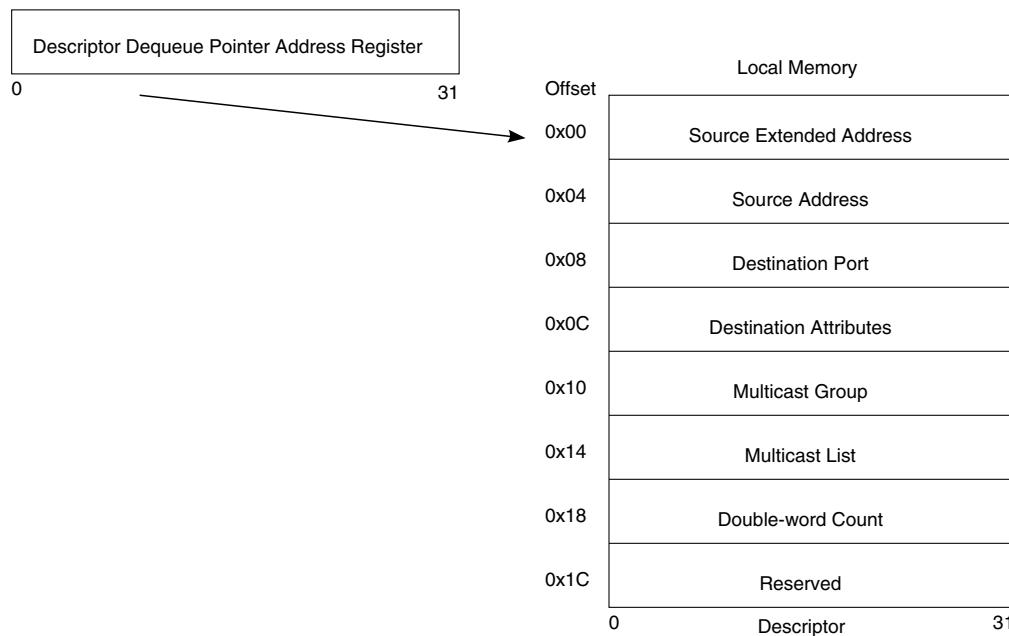
Descriptor Field	Description
Source Extended Address	Contains the source address of the message operation for local addresses of greater than 32 bits. After the message controller reads the descriptor from memory, this field is loaded into the source extended address register.
Source Address	Contains the source address of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the source address register.

Table continues on the next page...

Table 19-343. Outbound message unit descriptor summary (continued)

Descriptor Field	Description
Destination Port	Contains the destination port of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination port register.
Destination Attributes	Contains transaction attributes of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination attributes register.
Multicast Group	Contains the logical multicast group. Groups are defined a list of 32 numerically consecutive destinations by deviceID.
Multicast List	Contains a bit vector list of consecutive destinations by deviceID.
Double-word Count	Contains the number of doublewords for the message operation. After the message controller reads the descriptor from memory, this field is loaded into the double-word count register.
Reserved	-

Figure 19-322 depicts the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

**Figure 19-322. Descriptor dequeue pointer and descriptor**

19.7.4.2.7 Chaining mode controller interrupts

The outbound message interrupt can be generated for several reasons.

- Queue empty—an interrupt is generated to the interrupt controller if the queue goes empty and the interrupt event is enabled ($OMnMR[QEIE]$ is a 1). The event that caused the outbound message interrupt is indicated by $OMnSR[QEI]$. The interrupt is held until the queue is no longer empty and the $OMnSR[QEI]$ bit is cleared by writing a 1.

- Queue full-an interrupt is generated to the interrupt controller if the queue is full and the interrupt event is enabled ($OMnMR[QFIE]$ is a 1). The event that caused the outbound message interrupt is indicated by $OMnSR[QFI]$. The interrupt is held until the queue is no longer full and the $OMnSR[QFI]$ bit is cleared by writing a 1.
- Queue overflow-an interrupt is generated to the interrupt controller if the queue is full, the increment bit is set ($OMnMR[MUI]$) and the interrupt event is enabled ($OMnMR[QOIE]$ is a 1). The event that caused the outbound message interrupt is indicated by $OMnSR[QOI]$. The message unit must be re-initialized. The interrupt is held until the $OMnSR[QOI]$ bit has been cleared by writing a 1. This interrupt is also generated if the enqueue pointer is directly written and causes an overflow.
- End-of-message-an interrupt is generated after the completion of the message if this interrupt event is enabled ($OMnDATR[EOMIE]$ is a 1). The event that caused this interrupt is indicated by $OMnSR[EOMI]$. The interrupt is held until the $OMnSR[EOMI]$ bit is cleared by writing a 1. This allows an interrupt to be generated after a particular descriptor has been processed.

The error/port-write interrupt can be generated for the following reasons in direct mode.

- Message error response-an interrupt is generated after a message error response is received and this interrupt event is enabled ($OMnMR[EIE]$)
- Packet response time-out-an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled ($OMnMR[EIE]$)
- Retry error threshold exceeded-an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled ($OMnMR[EIE]$)
- Transaction error-an interrupt is generated after a message error response is received and this interrupt event is enabled ($OMnMR[EIE]$)

19.7.4.2.8 Chaining mode message error response errors

When a message error response is received by the message controller the following occurs.

- The message controller sets the message error response status bit ($OMnSR[MER]$)
- If $OMnMR[EIE]$ is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by $OMnSR[MUB]$) the message controller stops

19.7.4.2.9 Chaining mode packet response time-out errors

When a packet response time-out occurs for a message segment the following occurs.

- The message controller sets the packet response time-out status bit ($OMnSR[PRT]$)

- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops

19.7.4.2.10 Chaining mode retry error threshold exceeded errors

When a retry error threshold exceeded error occurs for a message segment the following occurs.

- The message controller sets the retry threshold exceed status bit (OM_nSR[RETE])
- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops

19.7.4.2.11 Chaining mode transaction errors

When an internal error occurs during a local memory read by the message controller the following occurs.

- The message controller sets the transaction error bit (OM_nSR[TE])
- If the internal error occurs during the descriptor memory read by the message controller, no message segment memory reads are generated and no message segments are sent
- Message segments that have an internal error are not sent because the message data is not available
- Memory reads that already were generated before the internal error occurred are also not transferred.
- Additional memory reads for the same message operation are generated but not transferred.
- All subsequent message segments for the same message operation are not transferred. This includes retried message segments.
- After the message operation completes (indicated by OM_nSR[MUB]) the message controller stops
- If OM_nMR[EIE] is set, the interrupt SRIO error/port-write is generated.

19.7.4.2.12 Chaining mode error handling

When an error occurs and the SRIO error/port-write interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling OM_nSR[MUB]

- Software disables the message controller by clearing OM_nMR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM_nSR[MER], OM_nSR[PRT], OM_nSR[RETE], or OM_nSR[TE])

When an error occurs and the SRIO error/port-write interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OM_nSR[MER], OM_nSR[PRT], OM_nSR[RETE], or OM_nSR[TE])
- Software verifies the message controller has stopped operation by polling OM_nSR[MUB]
- Software disables the message controller by clearing OM_nMR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM_nSR[MER], OM_nSR[PRT], OM_nSR[RETE], or OM_nSR[TE])

19.7.4.2.13 Chaining mode hardware error handling

The following list additional error conditions beyond the errors listed for direct mode.

All the errors listed in the direct mode section can also occur. The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port given in [Errors and error handling](#).

Table 19-344. Outbound message chaining mode hardware errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Message Segment Sent	Logical/Transport Layer Capture Register	Comments
Message Request	An internal error occurred during a read of the descriptor from local memory	0	SRIO error/port-write if OM _n MR[EI E] set	Transaction error in the outbound message status register (OM _n SR[TE]). Message Failed in the Mailbox CSR (MCSR[FA]).	No	-	Message controller stops. Note that the descriptor dequeue pointer is not incremented.

19.7.4.2.14 Chaining mode programming errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

These errors are in addition to the programming errors listed for direct mode.

Table 19-345. Outbound message chaining mode programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Enqueued descriptor address is invalid	No	No	Local memory captures the transaction and generate an interrupt.
Address for descriptor enqueue address pointer is invalid	No	No	Local memory captures the transaction and generate an interrupt.
Descriptor queue enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
Descriptor queue size set to a reserved value	No	No	Undefined operation results
Address of descriptor enqueue pointer set to a value outside of queue	No	No	Undefined operation results
Enqueueing of descriptors causes descriptor queue overflow	Outbound message interrupt enable set (OMnMR[QOIE])	Queue overflow (OMnSR[QOI])	Message controller stops.
Queue misaligned	No	No	May result in duplicate messages being sent

19.7.4.3 Message controller arbitration

Service control defines the order in which each message controller sends messages from its message queues when there are more than one outbound message units.

There are two options:

- Fixed priority-the lowest numbered message unit has the highest priority. Message unit 0 has the highest priority.
- Rotating priority- in this mode the order in which the message units take turns sending messages is round robin (message units 0, 1, 2, 3, 0, and so on). The number of messages a message unit can send is from 1 to 64.

OMnMR[SCNTL] configures the message units for these modes and defines operation in direct or chaining mode.

In fixed priority mode, all message segments for message unit 0 must complete before another lower priority message unit (message units 1, 2, 3, and so on) can start. However, in rotating priority mode, another message unit can start processing a message as soon as all message segments have been transmitted. Also, if in fixed priority mode and a message unit other than message unit 0 is processing a message, message unit 0 can start processing a message as soon as all of the message segments have been transmitted.

19.7.5 Inbound message controller operation

The inbound message controller is responsible for receiving messages and placing them in a circular frame queue in local memory.

The inbound message controller can receive segments of a message in any order. The address of where to write the message is computed as: Base address + (msgseg x ssize in double-words). Unlike the outbound message controller where the enqueue pointer is controlled by software and the dequeue pointer is controlled by hardware, the inbound message controller controls the enqueue pointer and the software controls the dequeue pointer.

[Figure 19-323](#) depicts a sample structure of the inbound message frames and the frame pointers. In this example, the frame queue has eight entries, three of which are currently valid. The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of messages have been received, an interrupt is generated to the processor. After processing a received message, the local processor can either write the inbound message mode register mailbox increment bit (IMnMR[MI]) causing the dequeue pointer to point to the next message frame in the queue or wait until all the received messages have been processed and write the dequeue pointer.

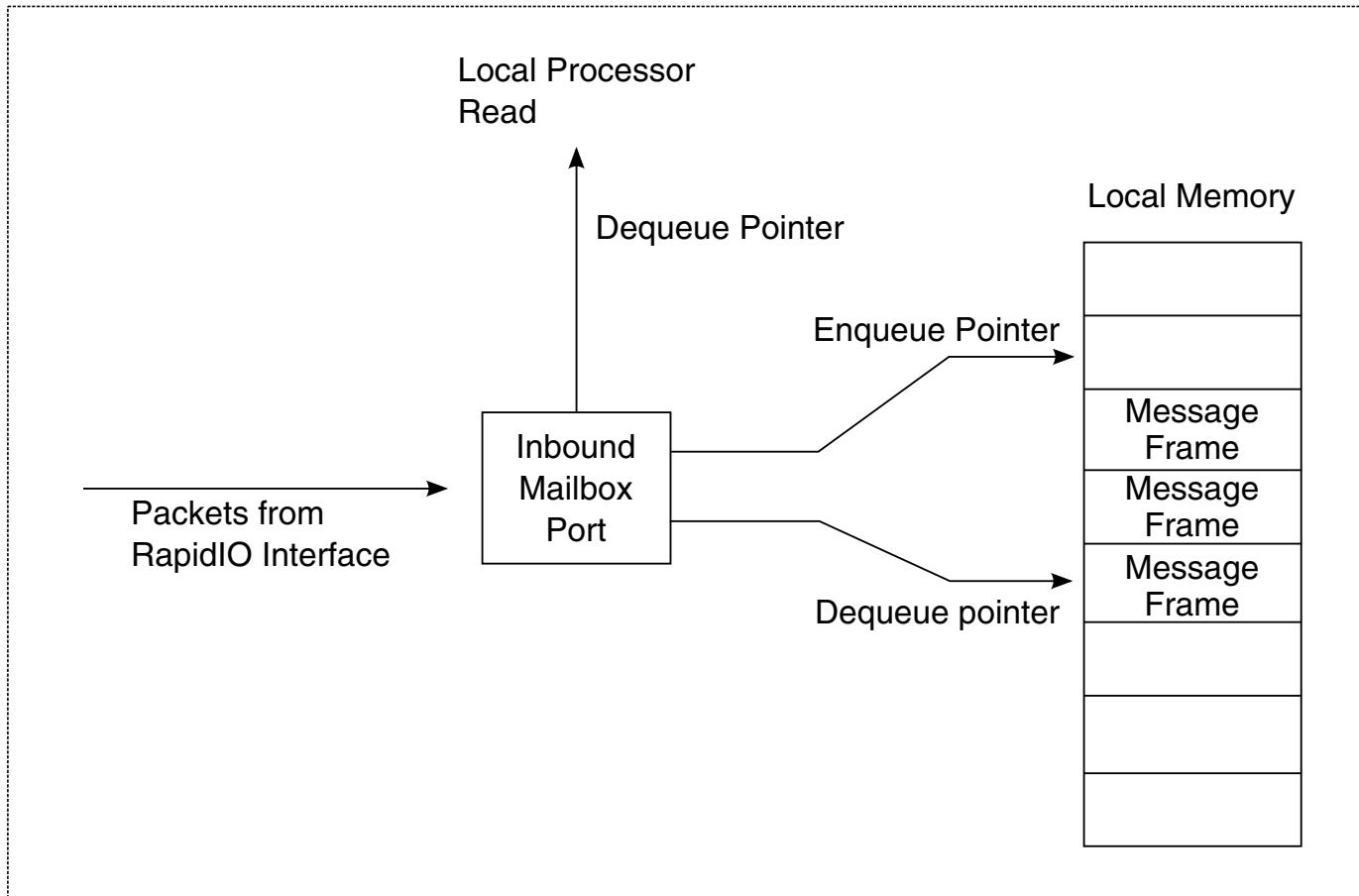


Figure 19-323. Inbound message structure

19.7.5.1 Inbound message controller initialization

The sequence of events to initialize the message controller is as follows:

- Initialize the frame queue dequeue pointer address registers (IM_nFQDPAR, EIM_nFQDPAR) and the frame queue enqueue pointer address registers (EIM_nFQEPR, IM_nFQEPR). These need to be initialized to the same value for proper operation.

These also must be queue size aligned (in other words, the queue to which they point must be aligned on a boundary equal to number of queue entries x frame size). For example, if there are eight entries in the queue and the frame size is 128 bytes, the queue must be 1024-byte aligned.

- Clear the status register (IM_nSR).
- Set the mailbox enable bit (IM_nMR[ME]) along with the other control parameters (frame queue size, message-in-queue threshold, frame size, snoop enable, and the various interrupt enables) in the inbound message mode register (IM_nMR).

19.7.5.2 Inbound controller operation

There are many ways in which software can interact with the message controller. One method is as follows:

- The inbound message controller receives a message segment request from the RapidIO port. If the inbound message controller is enabled ($IMnMR[ME] = 1$), the inbound message controller has received all the segments for the previous message, and the frame queue is not full then the message segment is accepted.
- The inbound message controller computes the address for each segment of the message (up to 16 segments per message) using the value of the inbound message frame queue enqueue pointer address registers and the segment number.
- The inbound message controller writes each segment to the circular queue in local memory at the computed address.
- Once the entire message is received and the write of all message segments have completed, the enqueue pointer is incremented to point to the next message frame in local memory by the inbound message controller. A message operation completes after all message segments for the message complete. A message segment completes when one of the following occurs:
 - the memory write completes (either successfully or an internal error occurred)
 - a message request time-out occurs (all message segments that have not yet been received are now complete)
- The message segments for a message can immediately be followed by the message segments for another message if certain rules are followed. If a message segment arrives for a new message before all of the previous message memory writes complete for the previous message and the RapidIO priority (the prio field value in the message packet) of the message is equal to or lower than the RapidIO priority of all of the previous messages, the message is processed by the message controller and a memory write is generated to the appropriate frame queue entry. If the RapidIO priority of the message is higher than any of the previous message memory writes that have not completed, the message controller generates a retry. Also, if a message segment arrives before all of the previous message memory writes for the same message complete and the new message segment is higher priority than the previous message segments, then the message controller generates a retry. The $IMnSR[MB]$ is cleared by the message controller when all message operations complete.
- An inbound message interrupt is generated to the local processor if the number of messages in the queue is greater than or equal to the configured message-in-queue threshold ($IMnMR[MIQ_THRESH]$) and this event is enabled to generate the interrupt ($IMnMR[MIQIE]$).

- Software determines that the message-in-queue event caused the interrupt by detecting that the message-in-queue interrupt bit is set when reading the inbound message status register (IMnSR[MIQI]).
- Software processes the frame queue entry pointed to by the frame dequeue pointer address registers (IMnFQDPAR, EIMnFQDPAR).
- Software increments the dequeue pointer address registers (IMnFQDPAR, EIMnFQDPAR) by setting the message increment bit (IMnMR[MI]). Software determines if there are any more messages to process by reading the queue empty bit (IMnSR[QE]). If the queue is not empty, the previous two steps are repeated.
- Optionally, software reads the enqueue pointer address registers (IMnFQEPR, EIMnFQEPR) and processes all the received messages. After the message processing is complete the dequeue pointer address registers (IMnFQDPAR, EIMnFQDPAR) are written.
- Software clears the message-in-queue interrupt bit (IMnSR[MIQI]) by writing a 1 to the IMnSR[MIQI] bit.

19.7.5.3 Message steering

Messages are forwarded to the inbound message controllers as follows:

- Messages directed to mailbox 0, regardless of letter number, are forwarded to message controller 0.
- Messages directed to mailbox 1, 2 or 3, regardless of letter number, are forwarded to message controller 1.

A message controller which has received part of a multi-segment message can not process another message until all segments for that message have arrived. Subsequent messages will be retried if targeted for the same message controller. A maximum of two multi-segment messages can be reassembled at one time if handled by different message controllers.

19.7.5.4 Inbound message controller retry response conditions

The conditions to generate a logical layer retry (response retry) are:

- Local memory circular queue is full and a message is received.
- The inbound message controller has already received at least one segment of a multi-segment message but has not received all message segments (determined by a different RapidIO source ID, RapidIO destination ID or mailbox)
- Message received with a higher priority than all of the previous messages that are being written to memory but have not completed.

Note that if all inbound messages are the same RapidIO priority, the third condition for generating a retry can not occur.

19.7.5.5 Inbound message controller interrupts

The inbound message controller generates the inbound message interrupt for several different events.

Each event can be individually enabled.

- Message-in-queue-an interrupt is generated whenever
 - The circular queue has accumulated the specified number of messages and this interrupt event is enabled ($IMnMR[MIQIE]$). The event that caused this interrupt is indicated by $IMnSR[MIQI]$. The interrupt is held until the dequeue pointer and enqueue pointer indicate that the specified number of messages is not in the frame queue and the $IMnSR[MIQI]$ bit has been cleared by writing a 1.
 - The circular queue has one or message in it, the specified number of message has not accumulated, a message has not been dequeued for the maximum interrupt report interval and this interrupt event is enabled ($IMnSR[MIQIE]$). The event that caused this interrupt is indicated by $IMnSR[MIQI]$. The interrupt is held until the $IMnSR[MIQI]$ bit has been cleared by writing a 1.
- Queue full-an interrupt is generated whenever the circular queue becomes full and this interrupt event is enabled ($IMnSR[QFIE]$). The event that caused this interrupt is indicated by $IMnSR[QFI]$. The interrupt is held until the queue is not full and the $IMnSR[QFI]$ bit has been cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons.

- Message request time-out-an interrupt is generated after a message request time-out occurs and this interrupt event is enabled ($IMnMR[EIE]$)
- Transaction error- an interrupt is generated after a OCN error response is received and this interrupt event is enabled ($IMnMR[EIE]$)

19.7.5.5.1 Message request time-out errors

The message request time-out counter starts after the first valid segment of a multisegment message is received and the time-out counter is enabled.

When a message request time-out occurs the following occurs.

- The message controller sets the message request time-out status bit ($IMnSR[MRT]$)
- All message segments that have not yet been received are considered complete

- If IMnMR[EIE] is set, the interrupt SRIO error/port-write is generated
- Once all message segments complete (indicated by IMnSR[MB]), the message controller stops

19.7.5.2 Inbound message controller transaction errors

When an internal error occurs during a local memory write by the message controller the following occurs.

- The message controller sets the transaction error bit (IMnSR[TE]) and enters the error state
- The message controller returns an error response
- Memory writes that already were generated before the internal error occurred also return an error response
- After the message operation completes (indicated by IMnSR[MB]) the message controller stops
- If IMnMR[EIE] is set, the interrupt SRIO error/port-write is generated.

19.7.5.3 Inbound message controller error handling

When an error occurs and the SRIO error/port-write interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the message controller has stopped operation by polling IMnSR[MB]
- Software clears the error by writing a 1 to the corresponding status bit (IMnSR[MRT], and/or IMnSR[TE])
- The message unit must be disabled, re-initialized and re-enabled before another message can be received.

When an error occurs and the SRIO error/port-write interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (IMnSR[MRT] and/or IMnSR[TE])
- Software verifies the message controller has stopped operation by polling IMnSR[MB]
- Software disables the message controller by clearing IMnMR[ME]
- Software clears the error by writing a 1 to the corresponding status bit (IMnSR[MRT] and/or IMnSR[TE])

19.7.5.4 Inbound message controller hardware error handling

The following lists possible error conditions and what occurs when they are encountered.

The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that messages are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in [Table 19-346](#).

Table 19-346. Inbound message hardware errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
A reserved ftype ¹	1	SRIO error/port-write if LTLEECS R[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Reserved tt encoding ¹	1	SRIO error/port-write if LTLEECS R[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size ¹	1	SRIO error/port-write if LTLEECS R[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE]	No	No Response	Updated with the packet ²	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID ¹	1	SRIO error/port-write if LTLEECS R[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
An incorrect message packet size (payload is not than the specified ssize except for the last segment. The last	1	SRIO error/port-write if LTLEECS R[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.

Table continues on the next page...

Table 19-346. Inbound message hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
segment's payload can be less than or equal to the segment size but not 0. Note that payload sizes are dword multiples.) ¹							
Reserved ssize field ¹	1	SRIO error/port-write if LTLEECS R[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Message received and no mailboxes are supported as indicated by DOCAR[M] ¹	1	SRIO error/port-write if LTLEECS R[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Message packet size larger than the configured frame size and message controller is enabled	2	SRIO error/port-write if LTLEECS R[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
An inbound message packet with a RapidIO priority of 3	2	SRIO error/port-write if LTLEECS R[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Not an error. The RapidIO priority is not consistent for all message segments of a message	2	-	-	Yes	Done or Retry Response	-	Retry response occurs if the higher priority message segment is received while the memory write for a corresponding lower priority message segment is outstanding

Table continues on the next page...

Table 19-346. Inbound message hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Message segment number is larger than the number of message segments in the message	2	SRIO error/port-write if LTLEECS R[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Duplicate message segment is received (Note that all segments of a multi segment message must be received before the next message begins).	2	SRIO error/port-write if LTLEECS R[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
msglen (number of segments in a message) is not consistent in all segments of a multi segment message	2	SRIO error/port-write if LTLEECS R[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
ssize (segment size) is not consistent in all segments of a multi segment message	2	SRIO error/port-write if LTLEECS R[MFE] set	Message format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Message received for an unsupported mailbox but at least one mailbox is supported	2	SRIO error/port-write if LTLEECS R[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	-	Packet is ignored and discarded. This error only applies if a mailbox is not supported. This error is not currently supported since all mailboxes are supported.

Table continues on the next page...

Table 19-346. Inbound message hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Message Controller Disabled and Message Received	2	No	None	No	Error	-	Packet is ignored and discarded.
Message Controller Enabled but in the Error State and Message Received	2	No	None	No	Error	-	Packet is ignored and discarded.
Internal error occurred during the write of the frame queue entry to memory	3	SRIO error/port-write if IMnMRI[EIE] set	Transaction error in the Message status register (IMnSR[TE]). Message Failed in the Message CSR (MCSR[FA])	No	Error	-	Message controller stops after the current message operation completes. The enqueue pointer is not incremented.
An internal error occurred for an earlier posted frame queue entry memory write and a subsequent frame queue entry memory write is posted before the internal error is detected. An internal error may or may not occur during the subsequent frame queue entry memory write. The frame queue could be for the same message or a different message.	4	No	None	Yes	Error	-	-
Message request to request time-out	unrelated	SRIO error/port-write if LTLEECS R[MRT]	Message request time-out in the Logical/Transport Layer Error Detect	No	No	Updated with the previous message request packet	All message segments received before the

Table 19-346. Inbound message hardware errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
for multi segment messages		set. SRIO error/port-write if OMnMR[EE].	CSR LTLEDCSR[MRT]. IMnSR[MRT] bit set.			except that the message segment field (bits 4 to 7 of LTLCCCSR[MI]) is updated with the lowest message segment number that has not yet been received. ²	time-out update memory. The enqueue pointer is not incremented. The message operation completes.

1. These error types are actually detected in the RapidIO port, not in the message controller.
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78-79), LTLACCSR[A] gets the address (packet bits 48-76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24-31), LTLCCCSR[FT] gets the ftype (packet bits 12-15), LTLCCCSR[TT] gets the ttype (packet bits 32-35), LTLCCCSR[MI] gets the msg info (packet bits 40-47). If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94-95), LTLACCSR[A] gets the address (packet bits 64-92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24-31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32-39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40-47), LTLCCCSR[FT] gets the ftype (packet bits 12-15), LTLCCCSR[TT] gets the ttype (packet bits 48-51), LTLCCCSR[MI] gets the msg info (packet bits 56-63).

19.7.5.5.5 Programming errors

The following is a partial list of programming errors that result in undefined or undesired hardware operation.

Table 19-347. Inbound message programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the message in queue threshold (IMnMR[MIQ_THRESH]) or reserved value of the circular frame queue size (IMnMR[CIRQ_SIZE])	No	No	Undefined operation results
The message in-queue threshold is equal to the frame queue size	No	No	Message in queue interrupt occurs when queue is full
The message in-queue threshold is greater than the frame queue size	No	No	Message in queue interrupt never occurs
Frame queue entry written to non-existent memory	No	No	Memory controller causes the interrupt and updates capture registers. IMnSR[TE] is set due to the internal error.

Table continues on the next page...

Table 19-347. Inbound message programming errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Message enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue frame pointer register is set incorrectly.	No	No	Undefined operation results
Queue misaligned	No	No	May cause unpredictable behavior

19.7.5.6 Disabling and enabling the inbound message controller

When the message controller is disabled by clearing IMnMR[ME] the following occurs.

- Queue full clears (IMnSR[QF])
- Message-in-queue clears (IMnSR[MIQ])
- Queue empty is set (IMnSR[QE])

Once the message controller is disabled, an error response is generated for all new message packets. If the message controller is disabled before all of the message segments for a multisegment message are received, a message request time-out must occur and all pending frame queue writes must complete before message busy clears (IMnSR[MB]).

For proper message controller operation, before the message controller is re-enabled, the following register writes must be performed: clear (IMnSR[MB]) and (IMnMR[ME]), and initialize the frame queue dequeue pointer address registers (IMnFQDPAR, EIMnFQDPAR) and the frame queue enqueue pointer address registers (IMnFQEPR, EIMnFQEPR) to the same value.

19.7.6 Message unit message passing logical specification registers

The mailbox command and status register (MCSR) provides the status for the four inbound and the four outbound controllers.

These read-only status bits indicate the state of each of the message controllers.

- Available (MCSR[A])-Indicates that the inbound message controller is enabled (IMnMR[ME]), the inbound message controller is not in the internal error state (IMnSR[TE] = 0) and the inbound message controller did not detect a message request time-out (IMnSR[MRT] = 0)
- Full (MCSR[FU])-This bit reflects the inbound message controller queue full status

- Empty (MCSR[EM])-This bit reflects the state of the outbound message controller message empty status
- Busy (MCSR[B])-This bit reflects the state of the inbound message controller busy bit IMnSR[MB]
- Failed (MCSR[FA])-This bit is set if any of the following bits are set:
 - The inbound message controller transaction error status bit IMnSR[TE]
 - The inbound message controller message request time-out status bit IMnSR[MRT]
 - The outbound message controller transaction error status bit OMnSR[TE] is set
 - The outbound message controller packet response time-out bit OMnSR[PRT] is set
 - The outbound message controller message error response received status bit OMnSR[MER] is set
 - The outbound message controller retry error threshold exceeded status bit OMnSR[RETE] is set
- Error (MCSR[ERR])-This bit is always 0

19.8 RapidIO doorbell and port-write unit

This section describes the operation of the doorbell and port-write controllers, which are part of the RapidIO message unit.

The doorbell and port-write controllers are compliant with the message passing logical specification contained in the *RapidIO Interconnect Specification, Revision 1.2*. The doorbell controller generates and receives doorbells. The port-write controller receives but does not generate port-writes.

The doorbell and port-write controllers are controlled through a set of run-time registers.

19.8.1 Doorbell and port-write unit features

- Support for one outbound doorbell controller
- Support for one inbound doorbell controller
 - The doorbell controller can sustain back-to-back inbound doorbells
- Support for one inbound port-write controller with the following features
 - Up to 64 bytes of payload
 - Only one inbound port-write queue entry

19.8.2 Doorbell controller

RapidIO supports a doorbell type that contains no data payload.

The RapidIO architecture references outbound and inbound doorbell controllers. This doorbell unit supports both inbound and outbound doorbells. Inbound doorbells are handled by the doorbell controller similar to how the message controller handles inbound data messages. The doorbell controller receives the doorbells and places it in a circular queue located in local memory. Additional doorbells can be received and forwarded to memory before the previous doorbell memory write completes as long as the RapidIO priority is the same or lower than the previous doorbell. The doorbell is retried if the RapidIO priority is higher than the previous doorbell. Outbound doorbells are generated through a memory-mapped write port rather than a queue. An outbound doorbell must complete before another outbound doorbell can be generated.

[Figure 19-324](#) depicts an example of the structure of the inbound doorbell queue and pointers. The doorbell queue has eight entries, three of which are currently valid. The doorbell controller enqueues doorbells and the local processor dequeues doorbells.

The doorbell entry size is fixed at 64 bits because doorbell packets only pass a small amount of information, making the enqueue and dequeue pointers double-word addresses.

The outbound doorbell controller has a dedicated interrupt that can be used to notify software that a doorbell has been sent. Each inbound doorbell controller has a dedicated interrupt that can be used for notification of incoming doorbells.

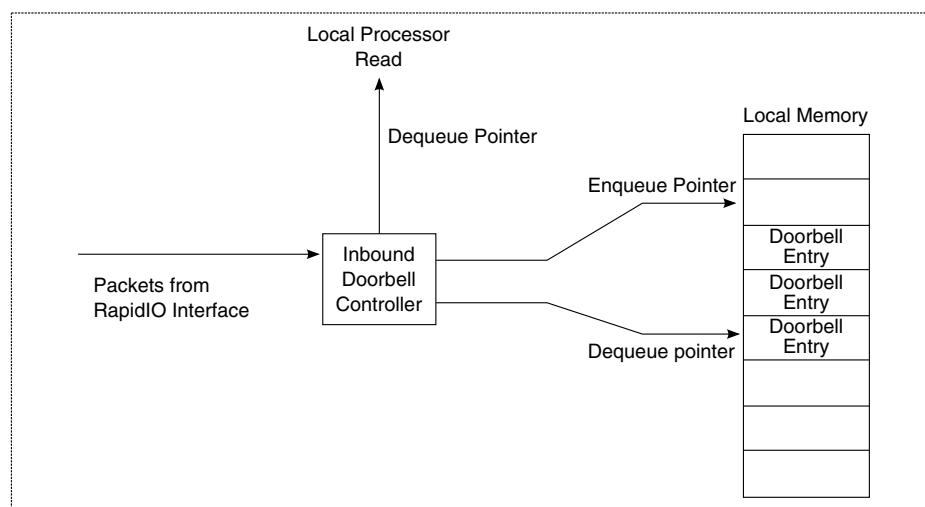


Figure 19-324. Inbound doorbell queue and pointer structure

19.8.2.1 Outbound doorbell controller

The outbound doorbell controller is used to generate doorbells.

Software is responsible for initializing all the parameters in all the necessary registers to start the doorbell transmission. The doorbell transfer is started when the doorbell start bit, OD MR[DUS], in the outbound doorbell mode register transitions from a 0 to 1 and the doorbell controller is not already busy. Software is expected to program all the appropriate registers before setting ODMR[DUS].

There are many ways in which software can interact with the doorbell controller. One method to generate a doorbell is as follows:

- Poll the status register doorbell unit busy bit, ODSR[DUB], to make sure the outbox is not busy with a previously initiated doorbell.
- Clear the status bits (ODSR[MER], ODSR[RETE], ODSR[PRT], and ODSR[EODI])
- Initialize the destination port (OD DPR), destination attributes (OD DATR) and retry error threshold (OD RETCR) registers
- Clear, then set the doorbell start bit, ODMR[DUS], to start the doorbell transfer.
- ODSR[DUB] is set when ODMR[DUS] transitions from a 0 to a 1 to indicate the doorbell transfer is in progress.
- The outbound doorbell controller sends the doorbell
- The ODSR[DUB] is cleared by the outbound doorbell controller after the doorbell operation completes. A doorbell completes when one of the following occurs:
 - done response received
 - error response received
 - a packet response time-out occurred
 - the retry limit was exceeded
- The outbound doorbell interrupt is generated if the end of doorbell outbound doorbell interrupt event is enabled (ODDATR[EODIE]).

19.8.2.1.1 Outbound doorbell controller interrupts

The "SRIO outbound doorbell" controller interrupt is generated after the completion of a doorbell (done, error, packet response time-out or retry limit exceeded) if this interrupt event is enabled (ODDATR[EODIE] is a 1).

The event that caused this interrupt is indicated by ODSR[EODI]. The interrupt is held until the ODSR[EODI] bit has been cleared by writing a 1.

The "SRIO error/port-write" interrupt can be generated for the following reasons:

- RapidIO error response. An interrupt is generated after a RapidIO error response is received and this interrupt event is enabled (LTLEECSR[MER])

- Packet response time-out. An interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (LTLEECSR[PRT])
- Retry error threshold exceeded. An interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (LTLEECSR[RETE])

19.8.2.1.2 Error response errors

When a RapidIO error response is received by the doorbell controller the following occurs:

- The doorbell controller sets the message error response status bits (ODSR[MER] and LTLEDCSR[MER])
- If LTLEECSR[MER] is set, the interrupt "SRIO error/port-write" is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops.

19.8.2.1.3 Outbound doorbell controller packet response time-out errors

When a packet response time-out occurs for a doorbell the following occurs:

- The doorbell controller sets the packet response time-out status bits (ODSR[PRT] and LTLEDCSR[PRT])
- If LTLEECSR[PRT] is set, the interrupt "SRIO error/port-write" is generated and EPWISR[PINT] is set.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops.

19.8.2.1.4 Outbound doorbell controller retry error threshold exceeded errors

When a retry error threshold exceeded error occurs for a doorbell the following occurs:

- The doorbell controller sets the retry threshold exceed status bits (ODSR[RETE] and LTLEDCSR[RETE])
- If LTLEECSR[RETE] is set, the interrupt "SRIO error/port-write" is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops.

19.8.2.1.5 Error handling

When an error occurs and the "SRIO error/port-write" interrupt is generated, the following occurs:

- Software determines the cause of the interrupt and processes the error
 - LTLEDCSR and ODSR capture the error condition for outbound doorbells
 - EPWISR[PINT] is set for PRT error since it is detected by the SRIO controller
 - Note that LTLEDCSR is a capture once register, so ODSR should be examined to make sure an outbound doorbell error did not occur immediately after another captured error
- Software verifies the doorbell controller has stopped operation by polling ODSR[DUB]
- Software disables the doorbell controller by clearing ODMR[DUS]
- Software clears the error by writing a 1 to the corresponding outbound doorbell status bits (ODSR[PRT], ODSR[PRT], and/or ODSR[RETE] as well as LTLEDCSR)

When an error occurs and the "SRIO error/port-write" interrupt is not enabled, the following occurs:

- Software determines that an error has occurred by polling the status bits (ODSR[MER], ODSR[PRT], and/or ODSR[RETE])
- Software verifies the doorbell controller has stopped operation by polling ODSR[DUB]
- Software disables the doorbell controller by clearing ODMR[DUS]
- Software clears the error by writing a 1 to the corresponding status bits (ODSR[MER], ODSR[PRT], and/or ODSR[RETE])

19.8.2.1.6 Disabling and enabling the doorbell controller-outbound

Once the doorbell controller is started, it cannot be stopped.

19.8.2.1.7 Outbound doorbell controller hardware error handling

The following is a list possible error conditions and what occurs when they are encountered.

The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected, no additional error checking beyond the current level is performed. Note that outbound doorbell responses are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These checks are in addition to the error condition checks provided by the RapidIO port given in [Errors and error handling](#).

Table 19-348. Outbound doorbell hardware errors

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Undefined Packet	Reserved ftype encoding ¹	1	SRIO error/ port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	Reserved tt encoding ¹	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[TSE].	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	Large transport size when operating in small transport size or small transport size when operating in large transport size ¹	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[TSE].	Yes	Updated with the packet ²	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Doorbell Response	Illegal Destination ID ¹	1	SRIO error/ port-write if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[ITTE]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	doorbell not outstanding ¹	1	SRIO error/ port-write if LTLEECSR[UR] set	Unsolicited response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UR]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	ttype (transaction field) is not doorbell response ¹	1	SRIO error/ port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	RapidIO priority is less than or equal to outbound request ¹	2	SRIO error/ port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	Incorrect Source ID ¹	2	SRIO error/ port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport	Yes	Updated with the packet ²	Packet is ignored and discarded.

Table continues on the next page...

Table 19-348. Outbound doorbell hardware errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbel I Sent	Logical/Transport Layer Capture Register	Comments
				Layer Error Detect CSR LTLEDCSR[ITD]			
Doorbell Response	reserved response status ¹	2	SRIO error/ port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	doorbell response packet size is incorrect ¹	2	SRIO error/ port-write if LTLEECSR[MFE] set	Message Format error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[MFE]	Yes	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Response	error response	3	SRIO error/ port-write if LTLEECSR[MER] set.	Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. ODSR[MER] bit set	Yes	Updated with the corresponding doorbell request packet ²	doorbell transfer complete
Doorbell Response	number of retries exceeds limit	3	SRIO error/ port-write if LTLEECSR[RETE] set.	Retry limit exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] ODSR[RETE] bit set.	Yes	Updated with the corresponding doorbell request packet ²	doorbell transfer complete
Doorbell Response	packet response time-out ¹	unrelated	SRIO error/ port-write if LTLEECSR[PRT] set.	Packet response time-out in the logical/transport layer error detect CSR LTLEDCSR[PRT] in RapidIO endpoint. ODSR[PRT] bit set.	Yes	Updated with the doorbell request packet in the RapidIO endpoint ²	doorbell transfer complete. Note that the RapidIO endpoint sends special priority 3 pkt indicating doorbell time-out.

1. These error types are actually detected in the RapidIO port, not in the doorbell controller
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78-79), LTLACCSR[A] gets the address (packet bits 48-76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24-31), LTLCCCSR[FT] gets the ftype (packet bits 12-15), LTLCCCSR[TT] gets the ttype (packet bits 32-35), LTLCCCSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94-95), LTLACCSR[A] gets the address (packet bits 64-92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24-31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32-39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40-47), LTLCCCSR[FT] gets the ftype (packet bits 12-15), LTLCCCSR[TT] gets the ttype (packet bits 48-51), LTLCCCSR[MI] gets 0.

19.8.2.1.8 Outbound doorbell controller programming errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 19-349. Outbound doorbell programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Transaction flow level set to reserved (2'b11)	No	None	Unit hangs. If the transaction flow level is then changed to a value other than reserved, the doorbell operation starts using this new transaction flow level.
Target interface set to an invalid RapidIO port	No	None	Undefined operation results
Register values changed during operation	No	No	Undefined operation results

19.8.2.2 Inbound doorbell controller

The inbound doorbell controller is responsible for receiving doorbells and placing them in a circular doorbell queue in local memory.

The inbound controller controls the enqueue pointer and the software controls the dequeue pointer. After a configured number of doorbells have been received, an interrupt is generated to the processor. After processing a received doorbell, the local processor can either write the doorbell mode register increment bit (IDMR[DI]) causing the dequeue pointer to point to the next doorbell in the queue or wait until all the received doorbells have been processed and write the dequeue pointer.

19.8.2.2.1 Inbound doorbell controller initialization

There are many ways in which software can interact with the doorbell controller.

One method to initialize the doorbell controller is as follows:

- Initialize the doorbell queue dequeue pointer address registers (DQDPAR, EDQDPAR) and the doorbell queue enqueue pointer address registers (DQEPEPAR, EDQEPEPAR). These need to be initialized to the same value for proper operation. These also must be queue size aligned (in other words, the queue to which they point must be aligned on a boundary equal to number of queue entries x 8 bytes). For example, if there are eight entries in the queue, the queue must be 64-byte aligned.

- Clear the status register (IDSR).
- Set the doorbell enable bit (IDMR[DE]) along with the other control parameters (doorbell queue size, doorbell-in-queue threshold, snoop enable, and the various interrupt enables) in the doorbell mode register (IDMR).

19.8.2.2 Inbound doorbell controller operation

There are many ways in which software can interact with the doorbell controller.

One method is as follows:

- The doorbell controller receives a doorbell. If the inbound doorbell controller is enabled (IDMR[DE] = 1) and the doorbell queue is not full, then the doorbell is accepted.
- The 16-bit information field along with the RapidIO source ID and RapidIO destination ID are stored in local memory by the doorbell controller using the value of the doorbell queue enqueue pointer address registers (DQEPR, EDQEPR).
- Once the memory write completes the enqueue pointer is incremented to point to the next doorbell queue entry in local memory.
- If another doorbell arrives before all of the previous doorbell memory writes complete and the RapidIO priority of the doorbell is equal to or lower than the RapidIO priority of all of the previous doorbells, the doorbell is processed by the doorbell controller and a memory write is generated to the appropriate doorbell queue entry. If the RapidIO priority of the doorbell is higher than any of the previous doorbell memory writes that have not completed, the doorbell controller generates a retry.
- An inbound doorbell interrupt is generated to the local processor because the number of doorbells in the queue is greater than or equal to the configured doorbell-in-queue threshold (IDMR[DIQ_THRESH]) and this event is enabled to generate the interrupt (IDMR[DIQIE]).
- Software determines that the doorbell-in-queue event caused the interrupt by detecting that the doorbell-in-queue interrupt bit is set when reading the doorbell status register (IDSR[DIQI]).
- Software processes the doorbell queue entry pointed to by the doorbell dequeue pointer address registers (DQDPAR, EDQDPAR).
- Software increments the dequeue pointer address registers (DQDPAR, EDQDPAR) by setting the doorbell increment bit (IDMR[DI]).
- Software determines if there are any more doorbells to process by reading the queue empty bit (IDSR[QE]). If the queue is not empty, the previous two steps are repeated.
- Software clears the doorbell-in-queue interrupt bit (IDSR[DIQI]) by writing a 1 to the IDSR[DIQI] bit.

19.8.2.2.3 Inbound doorbell queue entry format

This section defines the format of the doorbell information written to memory by the doorbell controller.

Each doorbell entry in the queue has 2 offsets of 32 bits, one for target information and one for source information. The target information is stored because a RapidIO port can be configured to accept packets from any destination. In addition, when there are multiple RapidIO ports on one device each port may be configured with a different destination ID.

Table 19-350. Inbound doorbell target info definition

Bits	Name	Description
0-15	-	Reserved
16-23	ETID	Extended target ID when in large transport mode, reserved when in small transport mode
24-31	TID	Target ID field from the received doorbell packet

Table 19-351. Source info definition

Bits	Name	Description
0-7	ESID	Extended Source ID when in large transport mode, reserved when in small transport mode
8-15	SID	Source ID field from the received doorbell packet
16-23	INFO MSB	Most significant byte of the info field from the received doorbell packet
24-31	INFO LSB	Least significant byte of the info field from the received doorbell packet

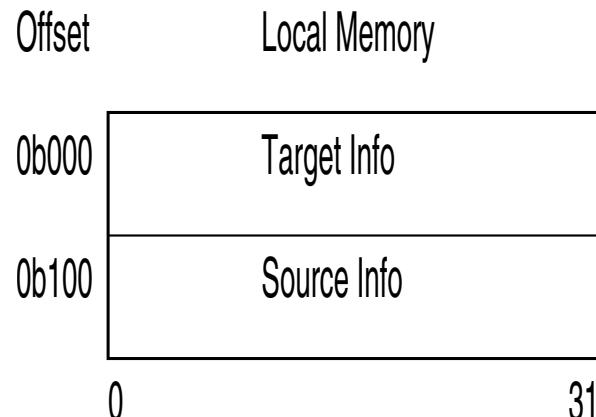


Figure 19-325. Doorbell entry format

19.8.2.2.4 Inbound doorbell controller retry response conditions

There are two conditions in which a doorbell is retried at the logical layer (response retry).

- Doorbell received and there are no entries available in the doorbell queue.
- Doorbell received with a higher priority than all of the previous doorbells that are being written to memory but have not completed.

If all inbound doorbells are the same RapidIO priority, the second condition for generating a retry can not occur.

19.8.2.2.5 Doorbell controller interrupts

There is one doorbell controller interrupt per inbound doorbell controller.

The following events can generate this interrupt.

- Doorbell-in-queue-this event generates an interrupt under the following conditions:
 - the circular queue has accumulated the configured number of doorbells specified by the doorbell-in-queue threshold (IDMR[DIQ_THRESH]) and this interrupt event is enabled (IDMR[DIQIE]). The event that caused this interrupt is indicated by IDSR[DIQI]. The interrupt is held until the dequeue pointer and enqueue pointer indicate that the specified number of doorbells is not in the doorbell queue and the IDSR[DIQI] bit has been cleared by writing a 1.
 - the circular queue has one or more doorbells in it, the specified number of doorbells has not accumulated, a doorbell has not been dequeued for the maximum interrupt report interval and this interrupt event is enabled (IDMR[DIQIE]). The event that caused this interrupt is indicated by IDSR[DIQI]. The interrupt is held until either IDMR[DI] has been set or DQDPAR[DQDPA] has been written followed by clearing IDSR[DIQI].
- Queue full-an interrupt is generated each time the circular queue becomes full and this interrupt event is enabled (IDSR[QFIE]). The event that caused this interrupt is indicated by IDSR[QFI]. The interrupt is held until the queue is not full and the IDSR[QFI] bit has been cleared by writing a 1.

The error/port-write interrupt can be generated for the following reasons.

- Transaction error-an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IDMR[EIE])

19.8.2.2.6 Doorbell controller transaction errors

When an internal error occurs during a local memory write by the doorbell controller the following occurs.

- The doorbell controller sets the transaction error bit (IDSR[TE]) and enters the error state

- The doorbell controller returns an error response
- If IDMR[EIE] is set, the interrupt SRIO error/port-write is generated.

19.8.2.2.7 Doorbell controller error handling

When an error occurs and the SRIO error/port-write interrupt is generated, the following occurs.

- Software determines the cause of the interrupt and processes the error
- Software verifies the doorbell controller has stopped operation by polling IDSR[DB]
- Software clears the error by writing a 1 to the corresponding status bit (IDSR[TE])
- The doorbell unit must be disabled, re-initialized and re-enabled before another doorbell can be received.

19.8.2.2.8 Inbound doorbell controller hardware error handling

The following list possible error conditions and what occurs when they are encountered. The error checking level indicates the order in which errors are checked.

Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that doorbells are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit. These check are in addition to the error condition checks provided by the RapidIO port given in [Errors and error handling](#).

Table 19-352. Inbound doorbell hardware errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
Reserved ftype ¹	1	SRIO error/ port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Reserved tt encoding ¹	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Large transport size when operating in	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet ²	Packet is ignored and discarded. An error or illegal

Table continues on the next page...

Table 19-352. Inbound doorbell hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
small transport size or small transport size when operating in large transport size ¹							transaction target error response is not generated.
Illegal Destination ID ¹	1	SRIO error/ port-write if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Inbound doorbell received and inbound doorbells are not supported as indicated by DOCAR[D] ¹	1	SRIO error/ port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
An inbound doorbell packet with a RapidIO priority of 3	2	SRIO error/ port-write if LTLEECSR[ITD] set	Illegal transaction decode in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
An incorrect doorbell packet size (not one datum in small transport mode or not two datums in large transport mode)	2	SRIO error/ port-write if LTLEECSR[MFE] set	Message Format error in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MFE]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Doorbell Controller Disabled and Doorbell Received	3	No	None	No	Error	-	Packet is ignored and discarded.
Doorbell Controller Enabled but in the Error State and Doorbell Received	3	No	None	No	Error	-	Packet is ignored and discarded.
Internal error occurred during the write of the	4	SRIO error/ port-write if OMMR[EIE] set	Transaction error in the Doorbell status register (ISSR[TE]). Doorbell	No	Error	-	Doorbell controller stops after the current

Table continues on the next page...

Table 19-352. Inbound doorbell hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue Entry Written in Local Memory	Response Status	Logical/Transport Layer Capture Register	Comments
doorbell queue entry to memory			Failed in the Port-write and Doorbell CSR (PWDCSR[FA])				doorbell operation completes. The enqueue pointer is not incremented.
An internal error occurred for an earlier posted write of a doorbell queue entry to memory and a subsequent write of a doorbell queue entry to memory is posted before the internal error is detected. An internal error may or may not occur during the subsequent write of a doorbell queue entry to memory.	5	No	None	Yes	Error	-	-

1. These error types are actually detected in the RapidIO port, not in the doorbell controller.
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78-79), LTLACCSR[A] gets the address (packet bits 48-76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24-31), LTLCSSR[FT] gets the ftype (packet bits 12-15), LTLCSSR[TT] gets the ttype (packet bits 32-35), LTLCSSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94-95), LTLACCSR[A] gets the address (packet bits 64-92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24-31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32-39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40-47), LTLCSSR[FT] gets the ftype (packet bits 12-15), LTLCSSR[TT] gets the ttype (packet bits 48-51), LTLCSSR[MI] gets 0.

19.8.2.9 Inbound doorbell controller programming errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 19-353. Inbound doorbell programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the doorbell in queue threshold (IDMR[DIQ_THTRES]) or reserved value of the circular doorbell queue size (IDMR[CIRQ_SIZE])	No	No	Undefined operation results
The doorbell in-queue threshold is equal to the doorbell queue size	No	No	Doorbell in queue interrupt occurs when queue is full
The doorbell in-queue threshold is greater than the doorbell queue size	No	No	Doorbell in queue interrupt never occurs
Doorbell queue entry written to non-existent memory	No	No	Memory controller causes the interrupt and update capture registers
Doorbell enqueue and dequeue pointers are not initialized to the same value	No	No	Undefined operation results
The dequeue pointer register is set incorrectly.	No	No	Undefined operation results

19.8.2.10 Disabling and enabling the doorbell controller-inbound

When the doorbell controller is disabled by clearing IDMR[DE] the following occurs.

- Queue full clears (IDSR[QF]).
- Doorbell-in-queue clears (IDSR[DIQ]).
- Queue empty is set (IDSR[QE]).
- Doorbell busy clears (IDSR[DUB]) after all pending doorbell queue entry writes to local memory complete.

Before the doorbell controller is re-enabled the doorbell busy bit must be cleared (IDSR[DB]) and the doorbell queue dequeue pointer address registers (DQDPAR, EDQDPAR) and the doorbell queue enqueue pointer address registers (DQEPPAR, EDQEPPAR) must be initialized to the same value for proper doorbell controller operation.

19.8.2.3 Doorbell message passing logical specification registers

The port-write and doorbell command and status register (PWDCSR) includes several doorbell controller status bits.

These read-only status bits indicate the state of doorbell controller 0.

- Available (PWDCSR[A]). Indicates that the inbound doorbell controller is enabled (IDMR[DE]) and the doorbell controller is not in the internal error state (IDSR[TE] = 0).
- Full (PWDCSR[FU]). This bit reflects the inbound doorbell controller queue full status bit (IDSR[QF]).
- Empty (PWDCSR[EM]). This bit reflects the inverted state of the outbound doorbell busy bit (ODSR[DUB] = 0).
- Busy (PWDCSR[B]). This bit reflects the state of the inbound doorbell controller busy bit IDSR[DUB].
- Failed (PWDCSR[FA]). This bit reflects the state of the transaction error status bit IDSR[TE].
- Error (PWDCSR[ERR]). This bit is always a 0.

19.8.3 Port-write controller

The implementation of the port-write controller is very similar to the inbound message and doorbell controllers except only one queue entry is supported.

The port write is intended as an error reporting mechanism from an end point-less device to a control processor or other system host.

[Figure 19-326](#) depicts an example of the structure of the inbound queue and pointer. The port-write queue only contains one entry with a fixed size of 64 bytes and aligned to a cache line boundary.

The port-write controller uses the error/port-write interrupt (SRIO error/port-write) for notification of incoming port-writes.

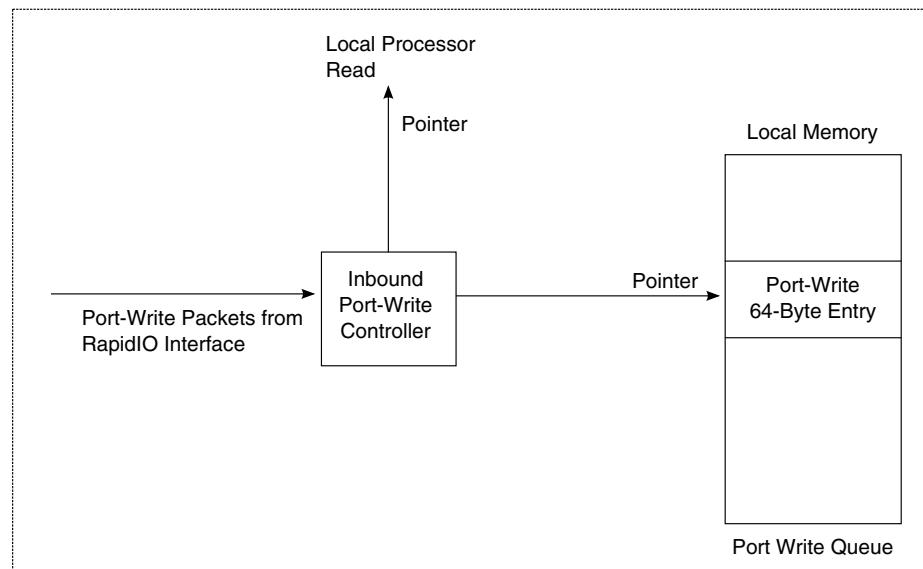


Figure 19-326. Inbound port-write structure

19.8.3.1 Port-write controller initialization

There are many ways in which software can interact with the port-write controller.

One method to initialize the port-write controller is as follows:

- Initialize the port-write queue base address registers (IPWQBAR, EIPWQBAR).
- Clear the status register (IPWSR).
- Set the port-write enable bit (IPWMR[PWE]) along with the other control parameters (snoop enable and the interrupt enable) in the inbound port-write mode register (IPWMR).

19.8.3.2 Port-write controller operation

There are several ways in which software can interact with the port-write controller.

One method is as follows:

- The port-write controller receives a port-write from the RapidIO port. If the inbound port-write controller is enabled (IPWMR[PWE] = 1) and the port-write queue is not full, then the port-write is accepted.
- 64 bytes of payload is stored by the port-write controller in local memory using the value of the port-write queue base address registers (IPWQBAR, EIPWQBAR). Valid payload sizes include 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes. Note that 64 bytes are always written to memory. If the actual payload size is less than 64 bytes, the non payload data written is undefined.

- If the queue full interrupt enable bit is set (IPWMR[QFIE]) after the memory write completes the error/port-write interrupt is generated by the port-write controller.
- An inbound error/port-write interrupt is generated to the local processor because a port-write was received and this event is enabled to generate the interrupt (IPWMR[QFIE]). Note that the RMU actually generates the SRIo error/port-write output and this is combined with the error interrupt to generate the error/port-write interrupt.
- Software determines that the queue full event caused the interrupt by detecting that the queue full interrupt bit is set when reading the inbound port-write status register (IPWSR[QFI]). Note there are many events that can generate this interrupt. Software must read several registers to determine that the interrupt was generated due to a port-write.
- Software processes the port-write queue entry pointed to by the port-write base address registers (IPWQBAR, EIPWQBAR).
- Software sets the clear queue bit (IPWMR[CQ]) re-enabling the hardware to receive another port-write.
- Software clears the queue full interrupt bit (IPWSR[QFI]) by setting the IPWSR[QFI].

19.8.3.3 Port-write controller interrupt

The error/port-write interrupt is used by the port-write controller.

This interrupt is used to notify the processor that some type of error event has occurred in a RapidIO port, message controller, doorbell controller or port-write controller. There are many events that can generate this interrupt. For example, the error management extensions use this interrupt to notify that error events have occurred. In the port-write controller the following event can generate this interrupt.

- Queue full-an interrupt is generated when this interrupt event is enabled (IPWMR[QFIE]) and a port-write is received and has been written to memory. The event that caused this interrupt is indicated by IPWSR[QFI]. The interrupt is held until the queue is no longer full and the IPWSR[QFI] bit has been cleared by writing a 1.
- Transaction error-an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IPWMR[EIE]).

19.8.3.4 Discarding port-writes

While the queue full bit is set or if a port-write is currently being written to memory but has not completed all received port-writes are discarded.

When a port-write is discarded for one of these reasons the controller sets the port write discarded bit (IPWSR[PWD]). Note that the port-write busy bit (IPWSR[PWB]) indicates that a port-write is currently being written to memory but has not completed.

19.8.3.5 Transaction errors

When an internal error occurs during a local memory write by the port-write controller the following occurs:

- The port-write controller sets the transaction error bit (IPWSR[TE]) and enters the error state.
- If IPWMR[EIE] is set, the interrupt SRIO error/port-write is generated.

19.8.3.5.1 Port-write controller error handling

When an error occurs and the SRIO error/port-write interrupt is generated, the following occurs:

- Software determines the cause of the interrupt and processes the error.
- Software verifies the port-write controller has stopped operation by polling IPWSR[PWB].
- Software disables the port-write controller by clearing IPWMR[PWE].
- Software clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).
- The port-write unit must be disabled, re-initialized and re-enabled before another maintenance port-write can be received.

When an error occurs and the SRIO error/port-write interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (IPWSR[TE]).
- Software verifies the port-write controller has stopped operation by polling IPWSR[PWB].
- Software disables the port-write controller by clearing IPWMR[PWE].
- Software clears the error by writing a 1 to the corresponding status bit (IPWSR[TE]).

19.8.3.6 Port-write controller hardware error handling

The following list shows the possible error conditions and what occurs when they are encountered.

The error checking level indicates the order in which errors are checked. Multiple errors can be checked at an error checking level. Once an error is detected no additional error checking beyond the current level is performed. Note that port-writes are processed in a pipeline fashion. The first error detected in the processing pipeline updates the error management extensions registers.

These error condition checks are provided by the messaging unit.

These check are in addition to the error condition checks provided by the RapidIO port given in [Errors and error handling](#).

Table 19-354. Inbound port-write hardware errors

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
reserved ftype ¹	1	SRIO error/ port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[UT]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Reserved tt encoding ¹	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Large transport size when operating in small transport size or small transport size when operating in large transport size ¹	1	SRIO error/ port-write if LTLEECSR[TSE] set	Transport size error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[TSE].	No	No Response	Updated with the packet ²	Packet is ignored and discarded. An error or illegal transaction target error response is not generated.
Illegal Destination ID ¹	1	SRIO error/ port-write if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/ Transport	No	No Response	Updated with the packet ²	Packet is ignored and discarded.

Table continues on the next page...

Table 19-354. Inbound port-write hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
			Layer Error Detect CSR LTLEDCSR[ITTE]				
An incorrect wr_size encoding (not 4, 8, 16, 24, 32, 40, 48, 56 or 64 bytes), payload size greater than the size defined by wr_size, or not dword aligned when the size is not 4 bytes. ¹	1	SRIO error/ port-write if LTLEECSR[ITD] set	Message Format error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
wr_size value reserved ¹	1	SRIO error/ port-write if LTLEECSR[ITD] set	Message Format error in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[ITD]	No	No Response	Updated with the packet ²	Packet is ignored and discarded.
Inbound maintenance port-write received and inbound maintenance port-writes are not supported as indicated by DOCAR[PW] ¹	1	SRIO error/ port-write if LTLEECSR[UT] set	Unsupported transaction in the Logical/ Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	Updated with the packet ²	Packet is ignored and discarded.
Not an error -	2	-	-	Yes	No Response	Inbound port write considered priority 2 by the inbound port write controller since response from memory is required at priority 3.	-

Table continues on the next page...

Table 19-354. Inbound port-write hardware errors (continued)

Error	Error Checking Level	Interrupt Generated	Status Bit Set	Queue entry written in local memory	Response status	Logical/Transport Layer Capture Register	Comments
An inbound port-write packet with a RapidIO priority of 3							
Port-write Controller Disabled and Port-write Received	2	No	None	No	No Response	-	Packet is ignored and discarded.
Port-write Controller Enabled but in the Error State and Port-write Received	2	No	None	No	No Response	-	Packet is ignored and discarded.
Internal error occurred during the write of the port-write queue entry to memory	3	SRIO error/ port-write if IPWMR[EIE] set	Transaction error in the Port-write status register (IPWSR[TE]). Port-write Failed in the Port-write and Doorbell CSR (PWDCSR[PFA])	No	No Response	-	Port-write controller stops after the current port-write operation completes.

1. These error types are actually detected in the RapidIO port, not in the port-write controller
2. If operating in small transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 78-79), LTLACCSR[A] gets the address (packet bits 48-76), LTLDIDCCSR[MDID] gets 0, LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[MSID] gets 0, LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 24-31), LTLCSSR[FT] gets the ftype (packet bits 12-15), LTLCSSR[TT] gets the ttype (packet bits 32-35), LTLCSSR[MI] gets 0. If operating in large transport size configuration using the packet LTLACCSR[XA] gets the extended address (packet bits 94-95), LTLACCSR[A] gets the address (packet bits 64-92), LTLDIDCCSR[MDID] gets the most significant byte of the destination ID (packet bits 16-23), LTLDIDCCSR[DID] gets the least significant byte of the destination ID (packet bits 24-31), LTLDIDCCSR[MSID] gets the most significant byte of the source ID (packet bits 32-39), LTLDIDCCSR[SID] gets the least significant byte of the source ID (packet bits 40-47), LTLCSSR[FT] gets the ftype (packet bits 12-15), LTLCSSR[TT] gets the ttype (packet bits 48-51), LTLCSSR[MI] gets 0.

19.8.3.6.1 Port-write controller programming errors

The following is the list of programming errors.

Table 19-355. Inbound port-write programming errors

Error	Interrupt Generated	Status Bit Set	Comments
Port-write queue entry written to non-existent memory	No	No	When a write to memory occurs, the memory controller causes its own interrupt and update its own capture registers. An internal error response will be returned. When the port-write controller receives the error response it will set the transaction error bit (IPWSR[TE]) and be in the error state.

19.8.3.7 Disabling and enabling the port-write controller

When the port-write controller is disabled by clearing IPWMR[PWE] the following occurs:

- Queue full clears (IPWSR[QF]).
- Port-write busy clears (IPWSR[PWB]) after a pending port-write queue entry write completes.

Before the port-write controller is re-enabled (IPWMR[PWE]) the port-write busy bit must be clear (IPWSR[PWB]).

19.8.3.8 Port-write controller message passing logical specification registers

The port-write and doorbell command and status register (PWDCSR) includes several port-write controller status bits.

These read-only status bits only indicate the state of the port-write controller.

- Available (PWDCSR[PA]). Indicates that the port-write controller is enabled (IPWMR[PWE]), the only port-write queue entry is available to be written (IPWSR[QF]) = 0 and the port-write controller is not in the internal error state (IPWSR[TE] = 0).
- Full (PWDCSR[PFU]). This bit reflects the state of the queue full status bit (IPWSR[QF]).
- Empty (PWDCSR[PEM]). This bit always a 1 since a port-write cannot be generated.
- Busy (PWDCSR[PB]). This bit reflects the state of the busy bit IPWSR[PWB].

- Failed (PWDCSR[PFA]). This bit reflects the state of the transaction error status bit IPWSR[TE].
- Error (PWDCSR[PE]). This bit is always a 0.

Chapter 20

DMA Controller

20.1 DMA overview

This chapter describes the DMA controller offered on this chip.

It describes a single DMA controller, which is instantiated twice on this chip. As such, all functionality is identical between the two controllers with the exception of the register offsets, as noted in the DMA register summary table , and the external signals, as noted in DMA signals table.

The DMA controller transfers blocks of data between the many interface and functional modules of this chip, independent of the cores or external hosts. It has four high-speed DMA channels. Both the cores and external devices can initiate DMA transfers. The channels are capable of complex data movement and advanced transaction chaining. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

This figure shows the block diagram of the DMA controller.

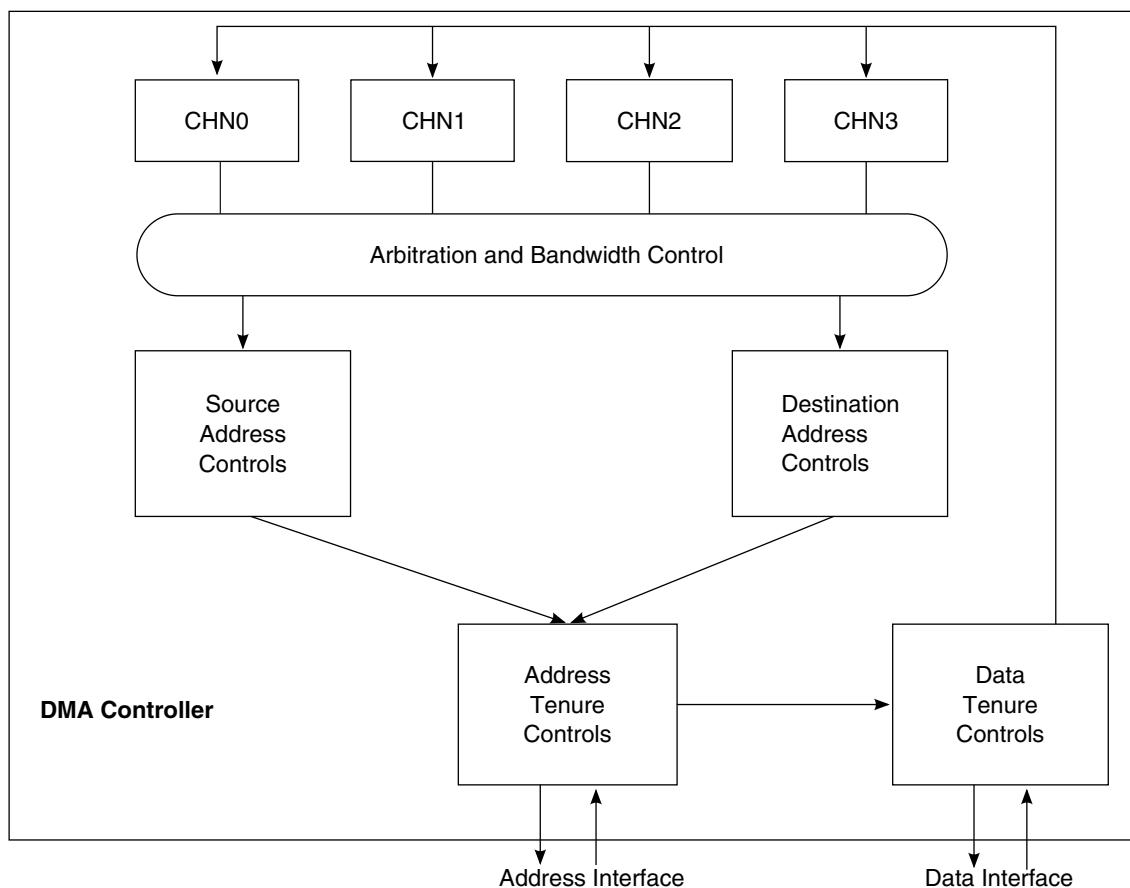


Figure 20-1. DMA block diagram

20.1.1 DMA features summary

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Up to 256 bytes for DMA sub-block transfers to maximize performance
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using DMA_DREQ_B, DMA_DACK_B, and DMA_DDONE_B.

20.1.2 DMA modes of operation

The DMA module has two modes of operation: basic and extended.

Basic mode is the DMA legacy mode, which does not support advanced features. Extended mode supports advanced features such as striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways.

Table 20-1 summarizes the relationship between the modes and the following features:

- Direct mode-No descriptors are involved. Software must initialize the required fields as described in [Table 20-2](#) before starting a transfer.
- Chaining mode-Software must initialize descriptors in memory and the required fields as described in [Table 20-2](#) before starting a transfer.
- Single-write start mode-The DMA process can be started using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- External control capability-This allows an external agent to start, pause, and check the status of a DMA transfer that has already been initialized.
- Channel continue capability-The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability-The software can abort a previously initiated transfer by setting the bit MR_n[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 20-1. Relationship of modes and features

Mode	Mode with one additional feature	Mode with two additional features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (extended chaining)	ExtCE (ExtC external control)
		ExtCS (ExtC single-write start)

The following table describes bit settings required for each DMA mode of operation.

Table 20-2. DMA mode bit settings

Modes with features	MRn[XFE]	MRn[CTM]	MRn[SRW]	MRn[CDSM_SWSM]	MRn[EMS_EN]
Basic direct modes					
Basic direct	0	1	0	0	0
Basic direct external control	0	1	0	0	1
Basic direct single-write start	0	1	1	1 or 0	0
Basic chaining modes					
Basic chaining	0	0	Reserved	0	0
Basic chaining external control	0	0	Reserved	0	1
Basic chaining single-write start	0	0	Reserved	1	0
Extended direct modes					
Extended direct	1	1	0	0	0
Extended direct external control	1	1	0	0	1
Extended direct single-write start	1	1	1	1 or 0	0
Extended chaining modes					
Extended chaining	1	0	Reserved	0	0
Extended chaining external control	1	0	Reserved	0	1
Extended chaining single-write start	1	0	Reserved	1	0

See [DMA functional description](#) for details on these modes.

This figure shows the general DMA operational flow chart.

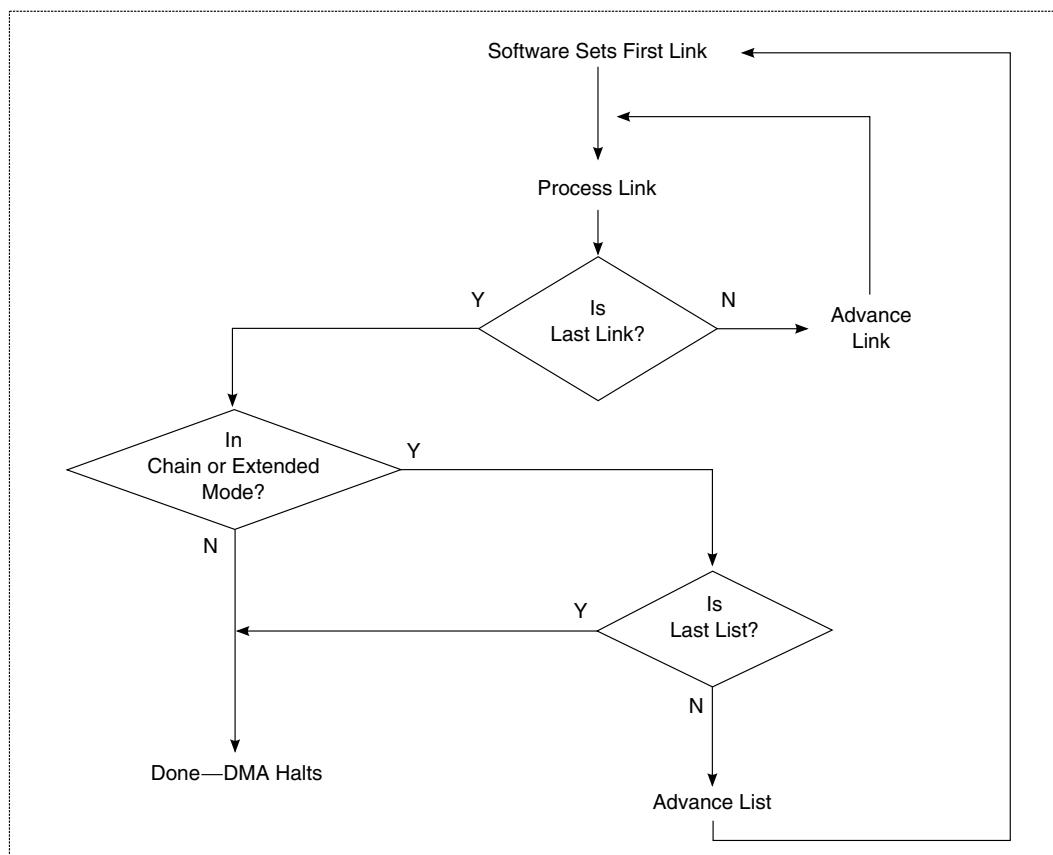


Figure 20-2. DMA operational flow chart

20.2 DMA external signal description

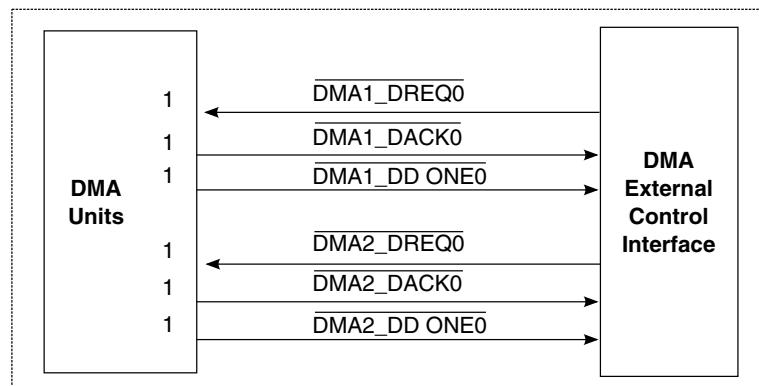
This section describes the external DMA signals.

20.2.1 Signal overview

This figure summarizes the DMA controller signals.

NOTE

External Control is supported for DMA channel 0 only.

**Figure 20-3. DMA signal summary**

Note the DMA signals are multiplexed with a variety of other functionality. See [Signal Multiplexing Details](#).

20.2.2 DMA signal descriptions

This table describes the external DMA control signals. These signals are only utilized when operating in external master mode.

See [External control mode transfer](#).

NOTE

External Control is supported on channel 0 only

Table 20-3. DMA signals-detailed signal descriptions

Signal	I/O	Description
DMA_DREQ_B	I	DMA request. Indicates the start of a DMA transfer or a restart from a paused request. Assertion of DMA_DREQ_B causes MRn[CS] to be set, thereby activating the corresponding DMA channel.
		State Meaning Asserted-Assertion of DMA_DREQ_B while DMA_DACK_B is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while DMA_DACK_B is asserted results in an illegal condition. Negated-Negation while DMA_DACK_B is asserted has no effect. Negation before the assertion of DMA_DACK_B results in an illegal condition.
		Timing Assertion-Can be asserted asynchronously Negation- Must remain asserted at least until the assertion of the corresponding DMA_DACK_B
DMA_DACK_B	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress
		State Meaning Asserted-Indicates that a DMA transfer is currently in progress. Asserted after the assertion of DMA_DREQ_B to indicate the start of a transfer Negated-Negated after finishing a complete transfer or after entering a paused state if MRn[EMP_EN] is set
		Timing Assertion-Asynchronous assertion; asserted for more than three system clocks

Table continues on the next page...

Table 20-3. DMA signals-detailed signal descriptions (continued)

Signal	I/O	Description
		Negation-Asynchronous negation; negated for more than three system clocks
DMA_DDONE_B	O	DMA done. Indicates that a DMA transfer is complete
		State Meaning Asserted-Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.
		Negated-Indicates that the current transfer is in process
		Timing Assertion-Always asserts asynchronously after the negation of the final DMA_DACK_B to indicate completion of a transfer. For a paused transfer, DMA_DDONE_B is asserted asynchronously after the negation of the final DMA_DACK_B.
		Negation-Negated asynchronously after the assertion of DMA_DREQ_B for the next transfer

20.3 DMA controller memory map

This section provides a detailed description of all accessible DMA memory and registers . The descriptions include individual, bit-level descriptions and reset states of each register. Undefined 4-byte address spaces within the map are reserved.

This table lists the DMA registers and their offsets. Note that the full register address is comprised of the programmable CCSRBAR together with the fixed DMA block base address and offset listed in table below.

DMA memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_0100	DMA mode register (DMA1_MR0)	32	R/W	0800_0000h	20.3.1/1589
10_0104	DMA status register (DMA1_SR0)	32	R/W	0000_0000h	20.3.2/1593
10_0108	DMA current link descriptor extended address register (DMA1_ECLNDAR0)	32	R/W	0000_0000h	20.3.3/1594
10_010C	DMA current link descriptor address register (DMA1_CLNDAR0)	32	R/W	0000_0000h	20.3.4/1595
10_0110	DMA source attributes register (DMA1_SATR0)	32	R/W	0000_0000h	20.3.5/1597
10_0114	DMA source address register (DMA1_SAR0)	32	R/W	0000_0000h	20.3.6/1598
10_0118	DMA destination attributes register (DMA1_DATR0)	32	R/W	0000_0000h	20.3.7/1599
10_011C	DMA destination address register (DMA1_DAR0)	32	R/W	0000_0000h	20.3.8/1600
10_0120	DMA byte count register (DMA1_BCR0)	32	R/W	0000_0000h	20.3.9/1601

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
10_0124	DMA extended next link descriptor address register (DMA1_ENLNDAR0)	32	R/W	0000_0000h	20.3.10/ 1601
10_0128	DMA next link descriptor address register (DMA1_NLNNDAR0)	32	R/W	0000_0000h	20.3.11/ 1602
10_0130	DMA extended current list descriptor address register (DMA1_ECLSDAR0)	32	R/W	0000_0000h	20.3.12/ 1603
10_0134	DMA current list descriptor address register (DMA1_CLSDAR0)	32	R/W	0000_0000h	20.3.13/ 1604
10_0138	DMA extended next list descriptor address register (DMA1_ENLSDAR0)	32	R/W	0000_0000h	20.3.14/ 1605
10_013C	DMA next list descriptor address register (DMA1_NLSDAR0)	32	R/W	0000_0000h	20.3.15/ 1605
10_0140	DMA source stride register (DMA1_SSR0)	32	R/W	0000_0000h	20.3.16/ 1606
10_0144	DMA destination stride register (DMA1_DSR0)	32	R/W	0000_0000h	20.3.17/ 1607
10_0180	DMA mode register (DMA1_MR1)	32	R/W	0800_0000h	20.3.1/1589
10_0184	DMA status register (DMA1_SR1)	32	R/W	0000_0000h	20.3.2/1593
10_0188	DMA current link descriptor extended address register (DMA1_ECLNDAR1)	32	R/W	0000_0000h	20.3.3/1594
10_018C	DMA current link descriptor address register (DMA1_CLNDAR1)	32	R/W	0000_0000h	20.3.4/1595
10_0190	DMA source attributes register (DMA1_SATR1)	32	R/W	0000_0000h	20.3.5/1597
10_0194	DMA source address register (DMA1_SAR1)	32	R/W	0000_0000h	20.3.6/1598
10_0198	DMA destination attributes register (DMA1_DATR1)	32	R/W	0000_0000h	20.3.7/1599
10_019C	DMA destination address register (DMA1_DAR1)	32	R/W	0000_0000h	20.3.8/1600
10_01A0	DMA byte count register (DMA1_BCR1)	32	R/W	0000_0000h	20.3.9/1601
10_01A4	DMA extended next link descriptor address register (DMA1_ENLNDAR1)	32	R/W	0000_0000h	20.3.10/ 1601
10_01A8	DMA next link descriptor address register (DMA1_NLNNDAR1)	32	R/W	0000_0000h	20.3.11/ 1602
10_01B0	DMA extended current list descriptor address register (DMA1_ECLSDAR1)	32	R/W	0000_0000h	20.3.12/ 1603
10_01B4	DMA current list descriptor address register (DMA1_CLSDAR1)	32	R/W	0000_0000h	20.3.13/ 1604
10_01B8	DMA extended next list descriptor address register (DMA1_ENLSDAR1)	32	R/W	0000_0000h	20.3.14/ 1605
10_01BC	DMA next list descriptor address register (DMA1_NLSDAR1)	32	R/W	0000_0000h	20.3.15/ 1605
10_01C0	DMA source stride register (DMA1_SSR1)	32	R/W	0000_0000h	20.3.16/ 1606
10_01C4	DMA destination stride register (DMA1_DSR1)	32	R/W	0000_0000h	20.3.17/ 1607

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_0200	DMA mode register (DMA1_MR2)	32	R/W	0800_0000h	20.3.1/1589
10_0204	DMA status register (DMA1_SR2)	32	R/W	0000_0000h	20.3.2/1593
10_0208	DMA current link descriptor extended address register (DMA1_ECLNDAR2)	32	R/W	0000_0000h	20.3.3/1594
10_020C	DMA current link descriptor address register (DMA1_CLNDAR2)	32	R/W	0000_0000h	20.3.4/1595
10_0210	DMA source attributes register (DMA1_SATR2)	32	R/W	0000_0000h	20.3.5/1597
10_0214	DMA source address register (DMA1_SAR2)	32	R/W	0000_0000h	20.3.6/1598
10_0218	DMA destination attributes register (DMA1_DATR2)	32	R/W	0000_0000h	20.3.7/1599
10_021C	DMA destination address register (DMA1_DAR2)	32	R/W	0000_0000h	20.3.8/1600
10_0220	DMA byte count register (DMA1_BCR2)	32	R/W	0000_0000h	20.3.9/1601
10_0224	DMA extended next link descriptor address register (DMA1_ENLNDAR2)	32	R/W	0000_0000h	20.3.10/1601
10_0228	DMA next link descriptor address register (DMA1_NLNDAR2)	32	R/W	0000_0000h	20.3.11/1602
10_0230	DMA extended current list descriptor address register (DMA1_ECLSDAR2)	32	R/W	0000_0000h	20.3.12/1603
10_0234	DMA current list descriptor address register (DMA1_CLSDAR2)	32	R/W	0000_0000h	20.3.13/1604
10_0238	DMA extended next list descriptor address register (DMA1_ENLSDAR2)	32	R/W	0000_0000h	20.3.14/1605
10_023C	DMA next list descriptor address register (DMA1_NLSDAR2)	32	R/W	0000_0000h	20.3.15/1605
10_0240	DMA source stride register (DMA1_SSR2)	32	R/W	0000_0000h	20.3.16/1606
10_0244	DMA destination stride register (DMA1_DSR2)	32	R/W	0000_0000h	20.3.17/1607
10_0280	DMA mode register (DMA1_MR3)	32	R/W	0800_0000h	20.3.1/1589
10_0284	DMA status register (DMA1_SR3)	32	R/W	0000_0000h	20.3.2/1593
10_0288	DMA current link descriptor extended address register (DMA1_ECLNDAR3)	32	R/W	0000_0000h	20.3.3/1594
10_028C	DMA current link descriptor address register (DMA1_CLNDAR3)	32	R/W	0000_0000h	20.3.4/1595
10_0290	DMA source attributes register (DMA1_SATR3)	32	R/W	0000_0000h	20.3.5/1597
10_0294	DMA source address register (DMA1_SAR3)	32	R/W	0000_0000h	20.3.6/1598
10_0298	DMA destination attributes register (DMA1_DATR3)	32	R/W	0000_0000h	20.3.7/1599
10_029C	DMA destination address register (DMA1_DAR3)	32	R/W	0000_0000h	20.3.8/1600
10_02A0	DMA byte count register (DMA1_BCR3)	32	R/W	0000_0000h	20.3.9/1601
10_02A4	DMA extended next link descriptor address register (DMA1_ENLNDAR3)	32	R/W	0000_0000h	20.3.10/1601
10_02A8	DMA next link descriptor address register (DMA1_NLNDAR3)	32	R/W	0000_0000h	20.3.11/1602

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
10_02B0	DMA extended current list descriptor address register (DMA1_ECLSDAR3)	32	R/W	0000_0000h	20.3.12/ 1603
10_02B4	DMA current list descriptor address register (DMA1_CLSDAR3)	32	R/W	0000_0000h	20.3.13/ 1604
10_02B8	DMA extended next list descriptor address register (DMA1_ENLSDAR3)	32	R/W	0000_0000h	20.3.14/ 1605
10_02BC	DMA next list descriptor address register (DMA1_NLSDAR3)	32	R/W	0000_0000h	20.3.15/ 1605
10_02C0	DMA source stride register (DMA1_SSR3)	32	R/W	0000_0000h	20.3.16/ 1606
10_02C4	DMA destination stride register (DMA1_DSR3)	32	R/W	0000_0000h	20.3.17/ 1607
10_0300	DMA general status register (DMA1_DGSR)	32	R	0000_0000h	20.3.18/ 1608
10_1100	DMA mode register (DMA2_MR0)	32	R/W	0800_0000h	20.3.1/1589
10_1104	DMA status register (DMA2_SR0)	32	R/W	0000_0000h	20.3.2/1593
10_1108	DMA current link descriptor extended address register (DMA2_ECLNDAR0)	32	R/W	0000_0000h	20.3.3/1594
10_110C	DMA current link descriptor address register (DMA2_CLNDAR0)	32	R/W	0000_0000h	20.3.4/1595
10_1110	DMA source attributes register (DMA2_SATR0)	32	R/W	0000_0000h	20.3.5/1597
10_1114	DMA source address register (DMA2_SAR0)	32	R/W	0000_0000h	20.3.6/1598
10_1118	DMA destination attributes register (DMA2_DATR0)	32	R/W	0000_0000h	20.3.7/1599
10_111C	DMA destination address register (DMA2_DAR0)	32	R/W	0000_0000h	20.3.8/1600
10_1120	DMA byte count register (DMA2_BCR0)	32	R/W	0000_0000h	20.3.9/1601
10_1124	DMA extended next link descriptor address register (DMA2_ENLNDAR0)	32	R/W	0000_0000h	20.3.10/ 1601
10_1128	DMA next link descriptor address register (DMA2_NLNDAR0)	32	R/W	0000_0000h	20.3.11/ 1602
10_1130	DMA extended current list descriptor address register (DMA2_ECLSDAR0)	32	R/W	0000_0000h	20.3.12/ 1603
10_1134	DMA current list descriptor address register (DMA2_CLSDAR0)	32	R/W	0000_0000h	20.3.13/ 1604
10_1138	DMA extended next list descriptor address register (DMA2_ENLSDAR0)	32	R/W	0000_0000h	20.3.14/ 1605
10_113C	DMA next list descriptor address register (DMA2_NLSDAR0)	32	R/W	0000_0000h	20.3.15/ 1605
10_1140	DMA source stride register (DMA2_SSR0)	32	R/W	0000_0000h	20.3.16/ 1606
10_1144	DMA destination stride register (DMA2_DSR0)	32	R/W	0000_0000h	20.3.17/ 1607
10_1180	DMA mode register (DMA2_MR1)	32	R/W	0800_0000h	20.3.1/1589
10_1184	DMA status register (DMA2_SR1)	32	R/W	0000_0000h	20.3.2/1593

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_1188	DMA current link descriptor extended address register (DMA2_ECLNDAR1)	32	R/W	0000_0000h	20.3.3/1594
10_118C	DMA current link descriptor address register (DMA2_CLNDAR1)	32	R/W	0000_0000h	20.3.4/1595
10_1190	DMA source attributes register (DMA2_SATR1)	32	R/W	0000_0000h	20.3.5/1597
10_1194	DMA source address register (DMA2_SAR1)	32	R/W	0000_0000h	20.3.6/1598
10_1198	DMA destination attributes register (DMA2_DATR1)	32	R/W	0000_0000h	20.3.7/1599
10_119C	DMA destination address register (DMA2_DAR1)	32	R/W	0000_0000h	20.3.8/1600
10_11A0	DMA byte count register (DMA2_BCR1)	32	R/W	0000_0000h	20.3.9/1601
10_11A4	DMA extended next link descriptor address register (DMA2_ENLNDAR1)	32	R/W	0000_0000h	20.3.10/1601
10_11A8	DMA next link descriptor address register (DMA2_NLNDAR1)	32	R/W	0000_0000h	20.3.11/1602
10_11B0	DMA extended current list descriptor address register (DMA2_ECLSDAR1)	32	R/W	0000_0000h	20.3.12/1603
10_11B4	DMA current list descriptor address register (DMA2_CLSDAR1)	32	R/W	0000_0000h	20.3.13/1604
10_11B8	DMA extended next list descriptor address register (DMA2_ENLSDAR1)	32	R/W	0000_0000h	20.3.14/1605
10_11BC	DMA next list descriptor address register (DMA2_NLSDAR1)	32	R/W	0000_0000h	20.3.15/1605
10_11C0	DMA source stride register (DMA2_SSR1)	32	R/W	0000_0000h	20.3.16/1606
10_11C4	DMA destination stride register (DMA2_DSR1)	32	R/W	0000_0000h	20.3.17/1607
10_1200	DMA mode register (DMA2_MR2)	32	R/W	0800_0000h	20.3.1/1589
10_1204	DMA status register (DMA2_SR2)	32	R/W	0000_0000h	20.3.2/1593
10_1208	DMA current link descriptor extended address register (DMA2_ECLNDAR2)	32	R/W	0000_0000h	20.3.3/1594
10_120C	DMA current link descriptor address register (DMA2_CLNDAR2)	32	R/W	0000_0000h	20.3.4/1595
10_1210	DMA source attributes register (DMA2_SATR2)	32	R/W	0000_0000h	20.3.5/1597
10_1214	DMA source address register (DMA2_SAR2)	32	R/W	0000_0000h	20.3.6/1598
10_1218	DMA destination attributes register (DMA2_DATR2)	32	R/W	0000_0000h	20.3.7/1599
10_121C	DMA destination address register (DMA2_DAR2)	32	R/W	0000_0000h	20.3.8/1600
10_1220	DMA byte count register (DMA2_BCR2)	32	R/W	0000_0000h	20.3.9/1601
10_1224	DMA extended next link descriptor address register (DMA2_ENLNDAR2)	32	R/W	0000_0000h	20.3.10/1601
10_1228	DMA next link descriptor address register (DMA2_NLNDAR2)	32	R/W	0000_0000h	20.3.11/1602
10_1230	DMA extended current list descriptor address register (DMA2_ECLSDAR2)	32	R/W	0000_0000h	20.3.12/1603

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_1234	DMA current list descriptor address register (DMA2_CLSDAR2)	32	R/W	0000_0000h	20.3.13/ 1604
10_1238	DMA extended next list descriptor address register (DMA2_ENLSDAR2)	32	R/W	0000_0000h	20.3.14/ 1605
10_123C	DMA next list descriptor address register (DMA2_NLSDAR2)	32	R/W	0000_0000h	20.3.15/ 1605
10_1240	DMA source stride register (DMA2_SSR2)	32	R/W	0000_0000h	20.3.16/ 1606
10_1244	DMA destination stride register (DMA2_DSR2)	32	R/W	0000_0000h	20.3.17/ 1607
10_1280	DMA mode register (DMA2_MR3)	32	R/W	0800_0000h	20.3.1/1589
10_1284	DMA status register (DMA2_SR3)	32	R/W	0000_0000h	20.3.2/1593
10_1288	DMA current link descriptor extended address register (DMA2_ECLNDAR3)	32	R/W	0000_0000h	20.3.3/1594
10_128C	DMA current link descriptor address register (DMA2_CLNDAR3)	32	R/W	0000_0000h	20.3.4/1595
10_1290	DMA source attributes register (DMA2_SATR3)	32	R/W	0000_0000h	20.3.5/1597
10_1294	DMA source address register (DMA2_SAR3)	32	R/W	0000_0000h	20.3.6/1598
10_1298	DMA destination attributes register (DMA2_DATR3)	32	R/W	0000_0000h	20.3.7/1599
10_129C	DMA destination address register (DMA2_DAR3)	32	R/W	0000_0000h	20.3.8/1600
10_12A0	DMA byte count register (DMA2_BCR3)	32	R/W	0000_0000h	20.3.9/1601
10_12A4	DMA extended next link descriptor address register (DMA2_ENLNDAR3)	32	R/W	0000_0000h	20.3.10/ 1601
10_12A8	DMA next link descriptor address register (DMA2_NLNDAR3)	32	R/W	0000_0000h	20.3.11/ 1602
10_12B0	DMA extended current list descriptor address register (DMA2_ECLSDAR3)	32	R/W	0000_0000h	20.3.12/ 1603
10_12B4	DMA current list descriptor address register (DMA2_CLSDAR3)	32	R/W	0000_0000h	20.3.13/ 1604
10_12B8	DMA extended next list descriptor address register (DMA2_ENLSDAR3)	32	R/W	0000_0000h	20.3.14/ 1605
10_12BC	DMA next list descriptor address register (DMA2_NLSDAR3)	32	R/W	0000_0000h	20.3.15/ 1605
10_12C0	DMA source stride register (DMA2_SSR3)	32	R/W	0000_0000h	20.3.16/ 1606
10_12C4	DMA destination stride register (DMA2_DSR3)	32	R/W	0000_0000h	20.3.17/ 1607
10_1300	DMA general status register (DMA2_DGSR)	32	R	0000_0000h	20.3.18/ 1608

20.3.1 DMA mode register (DMAx_MRn)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics.

Address: Base address + 100h offset + (128d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved				BWC				Reserved		EMP_EN	Reserved		EMS_EN	DAHTS	
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
	SAHTS	DAHE	SAHE	Reserved	SRW	EOSIE	EOLNIE	EOISIE	EIE	XFE	CDSM_SWSM	CA	CTM	CC	CS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAx_MRn field descriptions

Field	Description								
0–3 -	This field is reserved.								
4–7 BWC	<p>Bandwidth/pause control. Defined when single and multiple channels are executing or if MRn[EMP_EN] is set in external transfer mode.</p> <p>The value of MRn[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and re-arbitrates (switches to the next channel).</p> <p>In external pause mode, the value of MRn[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of DREQ_B resumes channel operation.</p> <table> <tbody> <tr> <td>0000</td> <td>1 byte</td> </tr> <tr> <td>0001</td> <td>2 bytes</td> </tr> <tr> <td>0010</td> <td>4 bytes</td> </tr> <tr> <td>0011</td> <td>8 bytes</td> </tr> </tbody> </table>	0000	1 byte	0001	2 bytes	0010	4 bytes	0011	8 bytes
0000	1 byte								
0001	2 bytes								
0010	4 bytes								
0011	8 bytes								

Table continues on the next page...

DMAx_MRn field descriptions (continued)

Field	Description
	<p>0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011-1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel. Bandwidth is unlimited with respect to other active channels that do not have unlimited bandwidth setting, that is, the other channels are using a setting not equal to 1111b. In case multiple channels specify unlimited setting, they arbitrate at a 1024 byte boundary.</p>
8–9 -	This field is reserved.
10 EMP_EN	<p>External master pause enable. Valid only if MRn[EMS_EN] is set.</p> <p>0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MRn[BWC].</p>
11–12 -	This field is reserved.
13 EMS_EN	<p>External master start enable. This bit does not apply to single-write start modes (direct or chaining).</p> <p>0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MRn[CS].</p>
14–15 DAHTS	<p>Destination address hold transfer size. Indicates the transfer size used for each transaction while MRn[DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MRn[DAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior.</p> <p>00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes</p>
16–17 SAHTS	<p>Source address hold transfer size. Indicates the transfer size used for each transaction while MRn[SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MRn[SAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior.</p> <p>00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes</p>
18 DAHE	<p>Destination address hold enable</p> <p>0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MRn[DAHTS]. Hardware only supports aligned transfers for this feature.</p>
19 SAHE	Source address hold enable

Table continues on the next page...

DMA_n_MR_n field descriptions (continued)

Field	Description
	<p>0 Disable source address hold 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MR_n[SAHTS]. Hardware only supports aligned transfers for this feature.</p>
20 -	This field is reserved.
21 SRW	<p>Single register write (Direct mode only; reserved for chaining mode.)</p> <p>0 Normal operation 1 Enable a write to the source address register to simultaneously set MR_n[CS], starting a DMA transfer, when MR_n[CDSM_SWSM] is also set. Setting this bit and clearing CDSM_SWSM causes a write to the destination address register to simultaneously set MR_n[CS], starting a DMA transfer.</p>
22 EOSIE	<p>End-of-segments interrupt enable</p> <p>0 Do not generate an interrupt at the completion of a data transfer. CLNDARn[EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SR_n[EOSI] is set). This bit overrides the CLNDARn[EOSIE].</p>
23 EOLNIE	<p>End-of-links interrupt enable</p> <p>0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDARn[EOLND] is set).</p>
24 EOLSIE	<p>End-of-lists interrupt enable</p> <p>0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDARn[EOLND] and NLSDARn[EOLSD] are set).</p>
25 EIE	<p>Error interrupt enable</p> <p>0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.</p>
26 XFE	<p>Extended features enable</p> <p>0 Disable striding feature in direct mode or disable list chaining feature in chaining mode. 1 Enable striding feature in direct mode or enable list chaining feature in chaining mode.</p>
27 CDSM_SWSM	<p>In chaining mode, current descriptor start mode/single-write start mode is as follows:</p> <ul style="list-style-type: none"> In basic mode (MR_n[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MR_n[CS], starting a DMA transfer. In extended chaining mode (MR_n[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MR_n[CS], starting a DMA transfer. <p>In direct mode, setting this bit and MR_n[SRW] causes a write to the source address register to simultaneously set MR_n[CS], starting a DMA transfer. Clearing this bit and setting MR_n[SRW] causes a write to the destination address register to simultaneously set MR_n[CS], starting a DMA transfer. This bit must be cleared when MR_n[SRW] is cleared.</p>
28 CA	<p>Channel abort</p> <p>0 No effect 1 Cause the current transfer to be aborted and SR_n[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.</p>

Table continues on the next page...

DMA_x_MR_n field descriptions (continued)

Field	Description
29 CTM	<p>Channel transfer mode</p> <p>0 Configure the channel in chaining mode.</p> <p>1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.</p>
30 CC	<p>Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode.</p> <p>0 No effect</p> <p>1 The DMA transfer restarts the transferring process starting at the current descriptor address.</p>
31 CS	<p>Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. Note that in external control mode, deasserting DMA_DREQ_B does NOT clear this bit.</p> <p>0 Halt the DMA process if channel is busy (SRn[CB] is set). No effect if the channel is not busy.</p> <p>1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.</p>

20.3.2 DMA status register (DMAx_SRn)

The status registers report various DMA conditions during and after a DMA transfer.

Address: Base address + 104h offset + (128d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R																	
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R									TE			CH	PE	EOLNI	CB	EOSI	EOLSI
W																	
Reset	0	0	0	0	0	0	0	0	w1c			w1c	w1c	0	0	0	

DMAx_SRn field descriptions

Field	Description
0–23 -	This field is reserved.
24 TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See DMA errors , for additional information.
25 -	This field is reserved.
26 CH	Channel halted. Cleared automatically by hardware if MR n [CS] is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel (SRn[CB] is cleared), this bit remains 0. 1 DMA transfer was successfully halted by software and can be resumed.
27 PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28 EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MRn[EOLNIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29 CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 DMA transfer is currently in progress.
30 EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MRn[EOSIE] is set or if CLNDARn[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31 EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MRn[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

20.3.3 DMA current link descriptor extended address register (DMAx_ECLNDARn)

These registers contain the upper 4 bits of the address of the current link descriptor. See [DMA current link descriptor address register \(DMA_CLNDARn\)](#) for a description of basic chaining mode.

Address: Base address + 108h offset + (128d × i), where i=0d to 3d

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

DMA_x_ECLNDAR_n field descriptions

Field	Description
0–27 -	This field is reserved.
28–31 ECLNDA	Current link descriptor extended address (upper 4 bits of 36-bit address)

20.3.4 DMA current link descriptor address register (DMA_x_CLNDAR_n)

Current link descriptor address registers contain the address of the current link descriptor. For devices with 36-bit addressing, the upper 4 bits of the address are described in [DMA current link descriptor extended address register \(DMA_ECLNDAR_n\)](#).

In basic chaining mode, shown in the figure below, software must initialize these registers to point to the first link descriptors in memory.

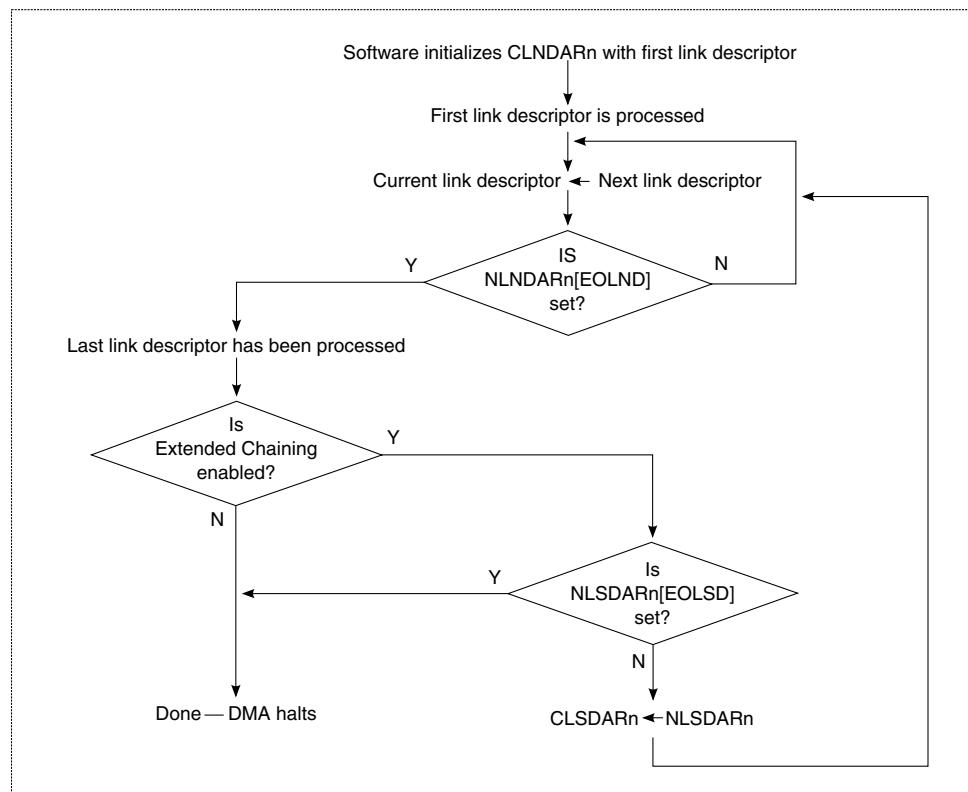


Figure 20-104. Basic Chaining Mode Flow Chart

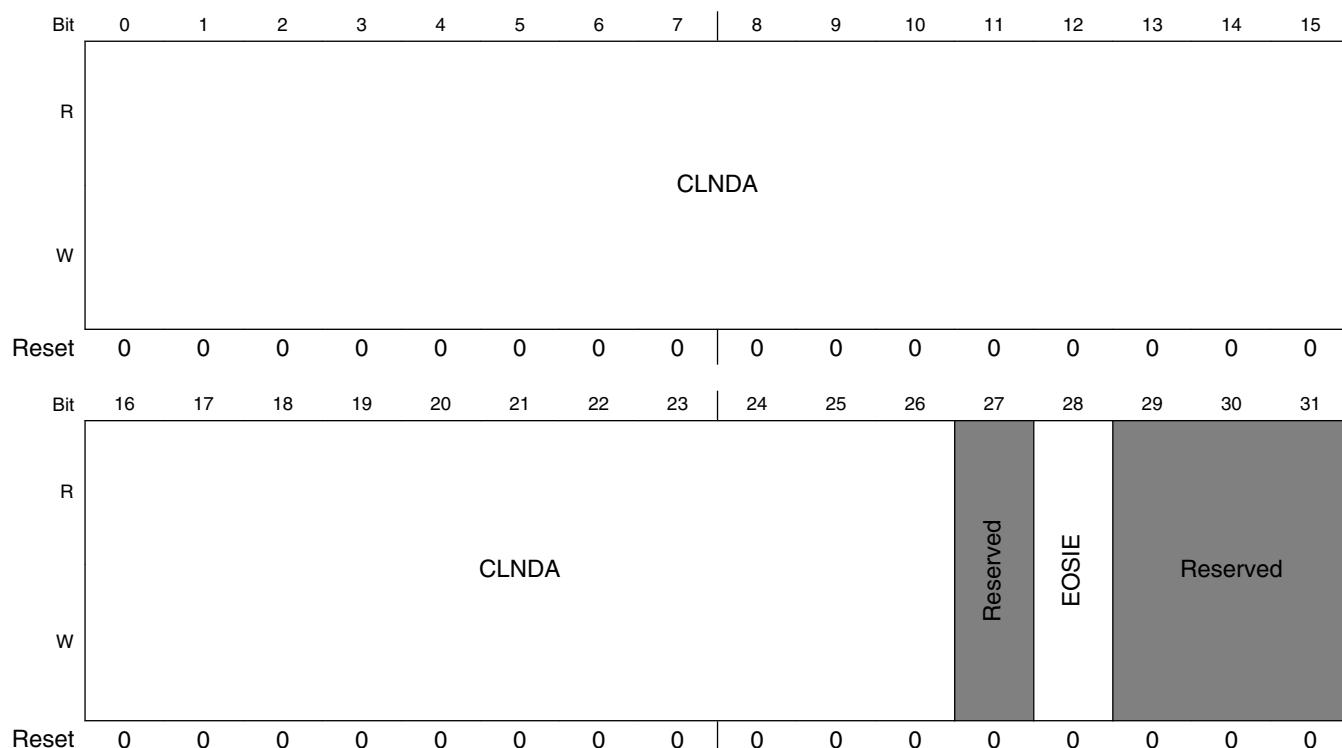
After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address registers and NLNDAR_n [EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller

DMA controller memory map

reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSNDAR_n [EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Address: Base address + 10Ch offset + (128d × i), where i=0d to 3d

**DMAx_CLNDARn field descriptions**

Field	Description
0–26 CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary. (This is the lower portion of the 36-bit address formed by CLNDARn[CLNDA] and ECLNDARn[ECLNDA].)
27 -	This field is reserved.
28 EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31 -	This field is reserved.

20.3.5 DMA source attributes register (DMAx_SATRn)

The source attributes registers contain the transaction attributes to be used for the DMA operation.

Address: Base address + 110h offset + (128d × i), where i=0d to 3d

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

DMAx_SATRn field descriptions

Field	Description
0–6 -	This field is reserved.
7 SSME	Source stride mode enable. Ignored in basic mode (MRn[XFE] is cleared). Striding on the source address can be accomplished by enabling SATRn[SSME] and setting the desired stride size and distance in the SSRn. 0 Stride mode disabled 1 Stride mode enabled
8–11 -	This field is reserved. Reserved
12–15 SREADTYPE	DMA source transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space . Patterns not shown are reserved. 0100 Read, do not snoop local processor 0101 Read, snoop local processor
16–27 -	This field is reserved.
28–31 ESAD	Extended source address. ESAD represents the four high-order bits of the 36-bit source address.

20.3.6 DMA source address register (DMAx_SARn)

The source address registers contain the address from which the DMA controller reads data. In direct mode, if MRn[CDSM_SWSM] and MRn[SRW] are set, a write to this register simultaneously sets MRn[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 114h offset + (128d × i), where i=0d to 3d

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

Reset 0

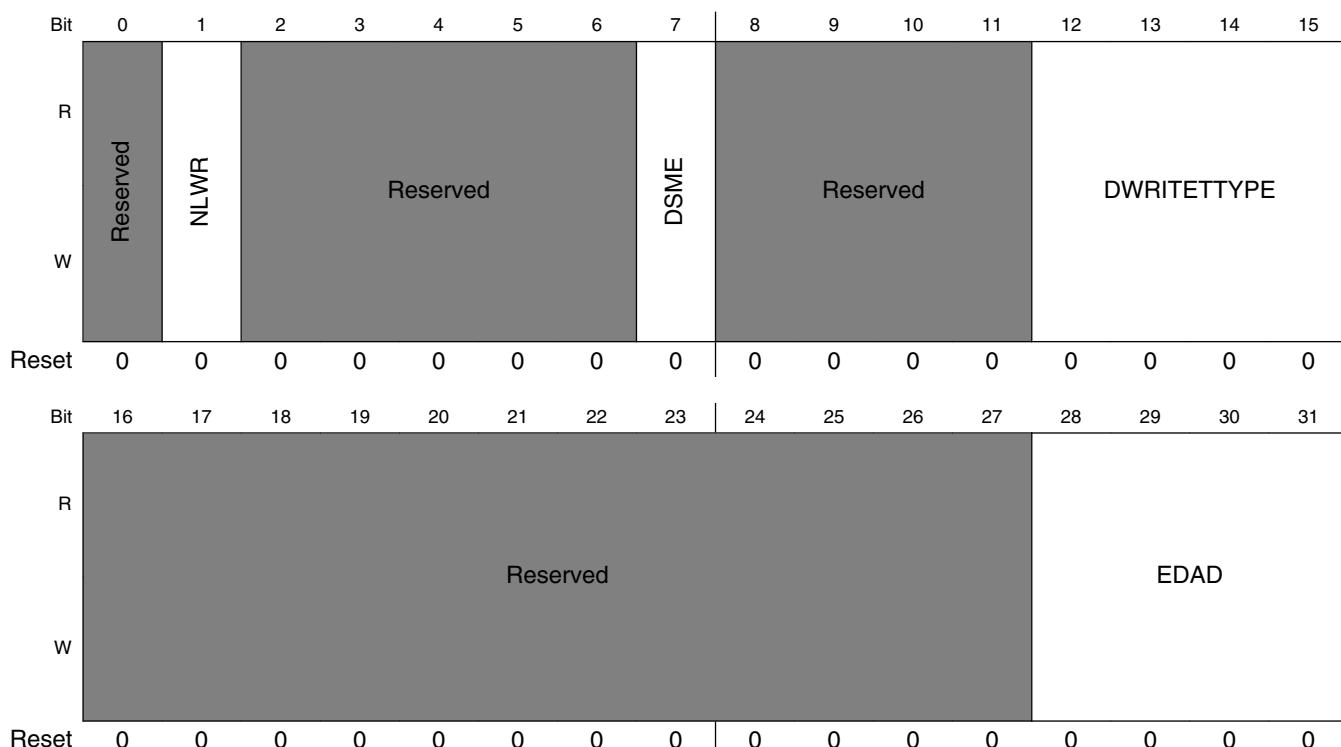
DMAx_SARn field descriptions

Field	Description
0–31 SAD	Source address. This register contains the low-order bits of the 36-bit source address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

20.3.7 DMA destination attributes register (DMAx_DATRn)

The destination attributes registers contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATRn[DSME]. The target interface is derived from the local access window and outbound ATMU mappings and the transaction is obtained from the value specified in DATRn[DWRITETTYPE].

Address: Base address + 118h offset + (128d × i), where i=0d to 3d



DMAx_DATRn field descriptions

Field	Description
0	This field is reserved.
1 NLWR	<p>No last write with response.</p> <p>Performance impact:</p> <ul style="list-style-type: none"> When NLWR is cleared only one "write-with-response" transaction can be processed/in-flight at a time, which prevents pipelining of "write-with-response" transactions. When NLWR is set "write-without-response" transactions can be issued in a pipelined manner not being stalled by a write response. <p>Error impact:</p>

Table continues on the next page...

DMAx_DATRn field descriptions (continued)

Field	Description
	<ul style="list-style-type: none"> When NLWR is cleared the returned response includes error status which is used to update the SR[TE] field. When NLWR is set, since no response is returned, the SR[TE] field is not updated with write transaction error status. <p>0 "Write-with-response" transactions are generated for the last transaction in a multi-transaction transfer. Single writes will be NWRITE_R. A response is returned from the module targeted by the write transfer (for example, DDR controller, eLBC, PCI Express) informing the DMA of write completion and error status.</p> <p>1 "Write-without-response" transactions are generated for the last transaction in a multi-transaction transfer. The DMA considers a transfer to be complete when the last write transaction is issued (not necessarily when the transaction has reached the target).</p>
2–6 -	This field is reserved.
7 DSME	<p>Destination stride mode enable</p> <p>Ignored in basic mode (MRn[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSRn.</p> <p>0 Stride mode disabled</p> <p>1 Stride mode enabled</p>
8–11 -	<p>This field is reserved.</p> <p>Reserved</p>
12–15 DWRITETTYPE	<p>DMA destination transaction type. Reserved values result in a programming error being detected and logged in SR[PE].</p> <p>Transaction type to run on local address space</p> <p>0000-0011 Reserved</p> <p>0100 Write, do not snoop local processor</p> <p>0101 Write, snoop local processor</p> <p>0110 Reserved</p> <p>0111 Reserved</p> <p>1000-1111 Reserved</p>
16–27 -	This field is reserved.
28–31 EDAD	Extended destination address. EDAD represents the four high-order bits of the 36-bit destination address.

20.3.8 DMA destination address register (DMAx_DARn)

The destination address registers contain the addresses to which the DMA controller writes data.

In direct mode, if MRn[SRW] is set and MRn[CDSM_SWSM] is cleared, a write to this register simultaneously sets MRn[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 11Ch offset + (128d × i), where i=0d to 3d

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

Reset 0

DMAx_DARn field descriptions

Field	Description
0–31 DAD	Destination address. This register contains the low-order bits of the 36-bit destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

20.3.9 DMA byte count register (DMAx_BCRn)

The byte count register contains the number of bytes to transfer.

Address: Base address + 120h offset + (128d × i), where i=0d to 3d

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

Reset 0

DMAx_BCRn field descriptions

Field	Description
0–5 -	This field is reserved.
6–31 BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is $(2^{26}) - 1$ bytes.

20.3.10 DMA extended next link descriptor address register (DMAx_ENLNDARn)

These registers contain the upper 4 bits of the address of the next link descriptor in memory. See [DMA next link descriptor address register \(DMA_NLNNDARn\)](#).

Address: Base address + 124h offset + (128d × i), where i=0d to 3d

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

Reset 0

DMAx_ENLNDArn field descriptions

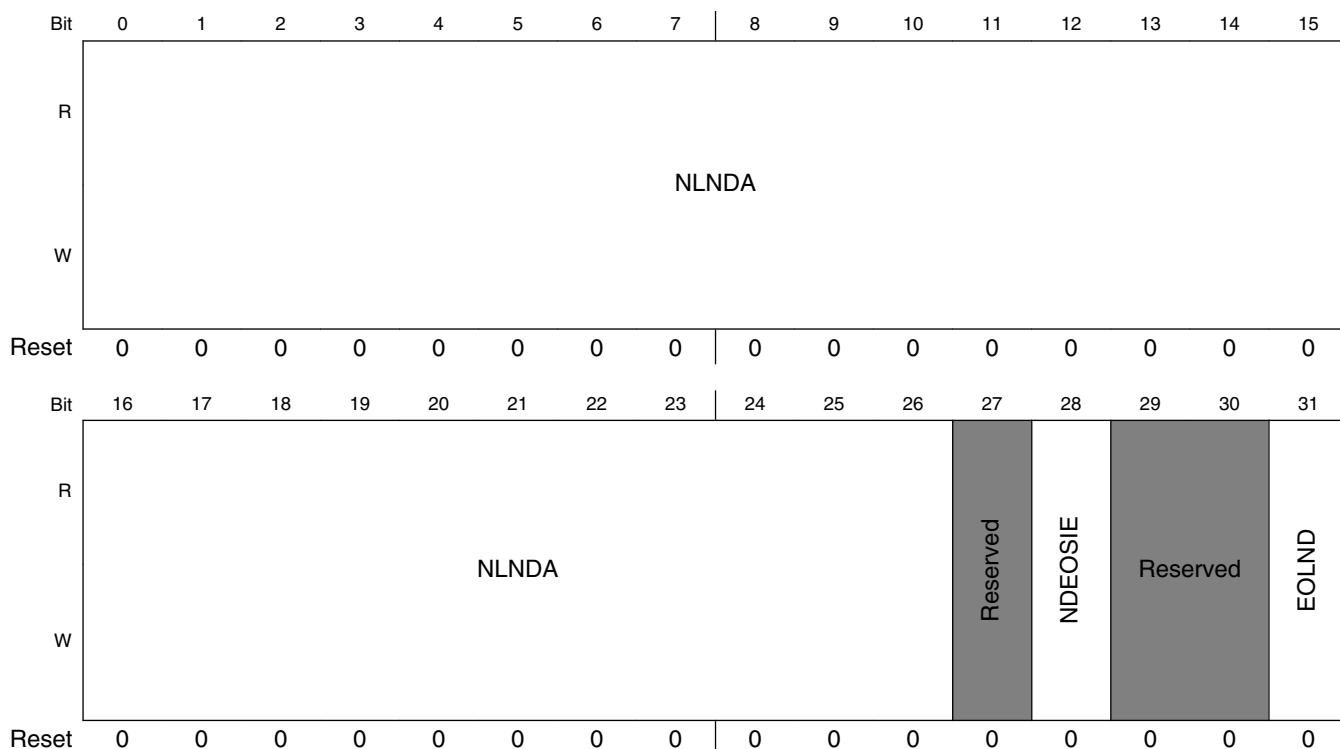
Field	Description
0–27 -	This field is reserved.
28–31 ENLNDA	Next link descriptor extended address bits (upper 4 bits of 36-bit address)

20.3.11 DMA next link descriptor address register (DMAx_NLNDArn)

Current link descriptor address registers contain the address for the next link descriptor in memory. For devices with 36-bit addressing, the upper 4 bits of the address are described in [DMA extended next link descriptor address register \(DMA_ENLNDArn\)](#).

Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

Address: Base address + 128h offset + (128d × i), where i=0d to 3d

**DMAx_NLNDArn field descriptions**

Field	Description
0–26 NLNDA	Next link descriptor address. Contains the next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.

Table continues on the next page...

DMax_NLNDARn field descriptions (continued)

Field	Description
27 -	This field is reserved.
28 NDEOSIE	<p>Next descriptor end-of-segment interrupt enable</p> <p>0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished.</p> <p>1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.</p>
29-30 -	This field is reserved.
31 EOLND	<p>End-of-links descriptor. This bit is ignored in direct mode.</p> <p>0 This descriptor is not the last link descriptor in memory for this list.</p> <p>1 This descriptor is the last link descriptor in memory for this list. If this bit is set, the DMA controller advances to the next list descriptor in memory if NLSDARn[EOLSD] is also set in extended mode.</p>

20.3.12 DMA extended current list descriptor address register (DMAx_ECLSDARn)

These registers contain the upper 4 bits of the current address of the list descriptor in memory in extended chaining mode. See [DMA current list descriptor address register \(DMA_CLSDAR \$n\$ \)](#).

Address: Base address + 130h offset + (128d × i), where i=0d to 3d

DMAx ECLSDARn field descriptions

Field	Description
0-27 -	This field is reserved.
28-31 ECLSDA	Current list descriptor extended address bits (upper 4 bits of 36-bit address)

20.3.13 DMA current list descriptor address register (DMA_x_CLSDAR_n)

The current list descriptor address registers contain the current address of the list descriptor in memory in extended chaining mode. For chips with 36-bit addressing, the upper 4 bits of the address are described in [DMA extended current list descriptor address register \(DMA_ECLSDAR_n\)](#).

In extended chaining mode, software must initialize CLS DAR n and ECLSDAR n to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR n [EOLSD] in the next list descriptor address register is clear, the DMA controller reads the new current list descriptor from memory to process that list. If EOLSD in the next list descriptor address register is set and the last link in the current list is finished, all DMA transfers are complete.

Address: Base address + 134h offset + (128d × i), where i=0d to 3d

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

Reset 0

DMA_x_CLSDAR_n field descriptions

Field	Description
0–26 CLSDA	Current list descriptor address. Contains the low-order bits of the 36-bit current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31 -	This field is reserved.

20.3.14 DMA extended next list descriptor address register (DMAx_ENLSDARn)

These registers contain the upper 4 bits of the address of the next list descriptor in memory. See [DMA next list descriptor address register \(DMA_NLSDARn\)](#).

Address: Base address + 138h offset + (128d × i), where i=0d to 3d

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

Reset 0

DMAx_ENLSDARn field descriptions

Field	Description
0–27 -	This field is reserved.
28–31 ENLSDA	Next list descriptor extended address bits (upper 4 bits of 36-bit address)

20.3.15 DMA next list descriptor address register (DMAx_NLSDARn)

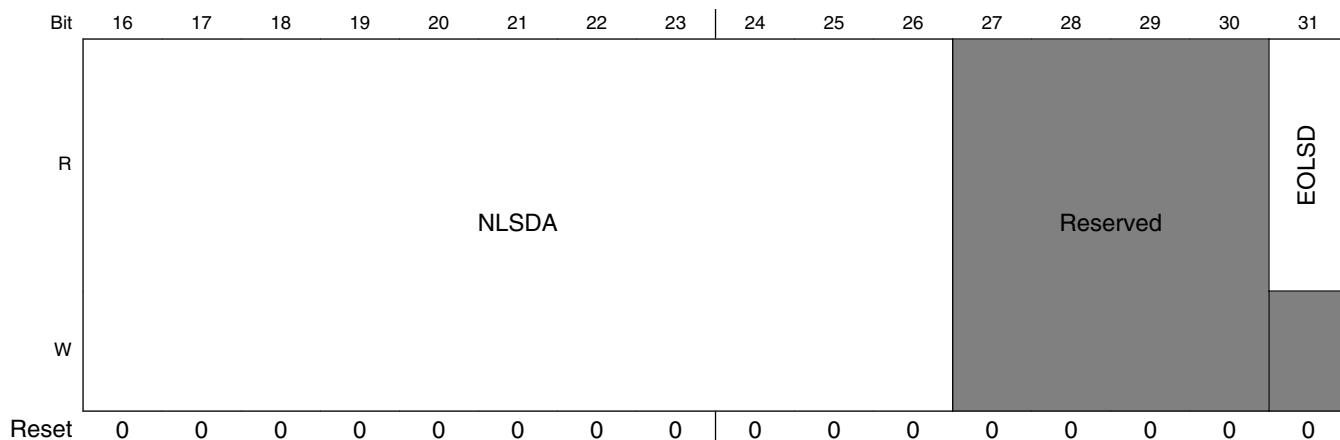
The next list descriptor address registers contain the address for the next list descriptor in memory. For devices with 36-bit addressing, the upper 4 bits of the address are described in [DMA extended next list descriptor address register \(DMA_ENLSDARn\)](#).

If the contents are transferred to the current list descriptor address register, they become effective for the current transfer in extended chaining mode.

Address: Base address + 13Ch offset + (128d × i), where i=0d to 3d

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



DMAx_NLSDARn field descriptions

Field	Description
0–26 NLSDA	Next list descriptor address. Contains the low-order bits of the 36-bit next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30 -	This field is reserved.
31 EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

20.3.16 DMA source stride register (DMAx_SSRRn)

The source stride register contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

Address: Base address + 140h offset + (128d × i), where i=0d to 3d



DMAx_SSRRn field descriptions

Field	Description
0–7 -	This field is reserved.

Table continues on the next page...

DMAx_SSFn field descriptions (continued)

Field	Description
8–19 SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31 SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

20.3.17 DMA destination stride register (DMAx_DSRn)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated.

Address: Base address + 144h offset + (128d × i), where i=0d to 3d

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

Reset 0

DMAx_DSRn field descriptions

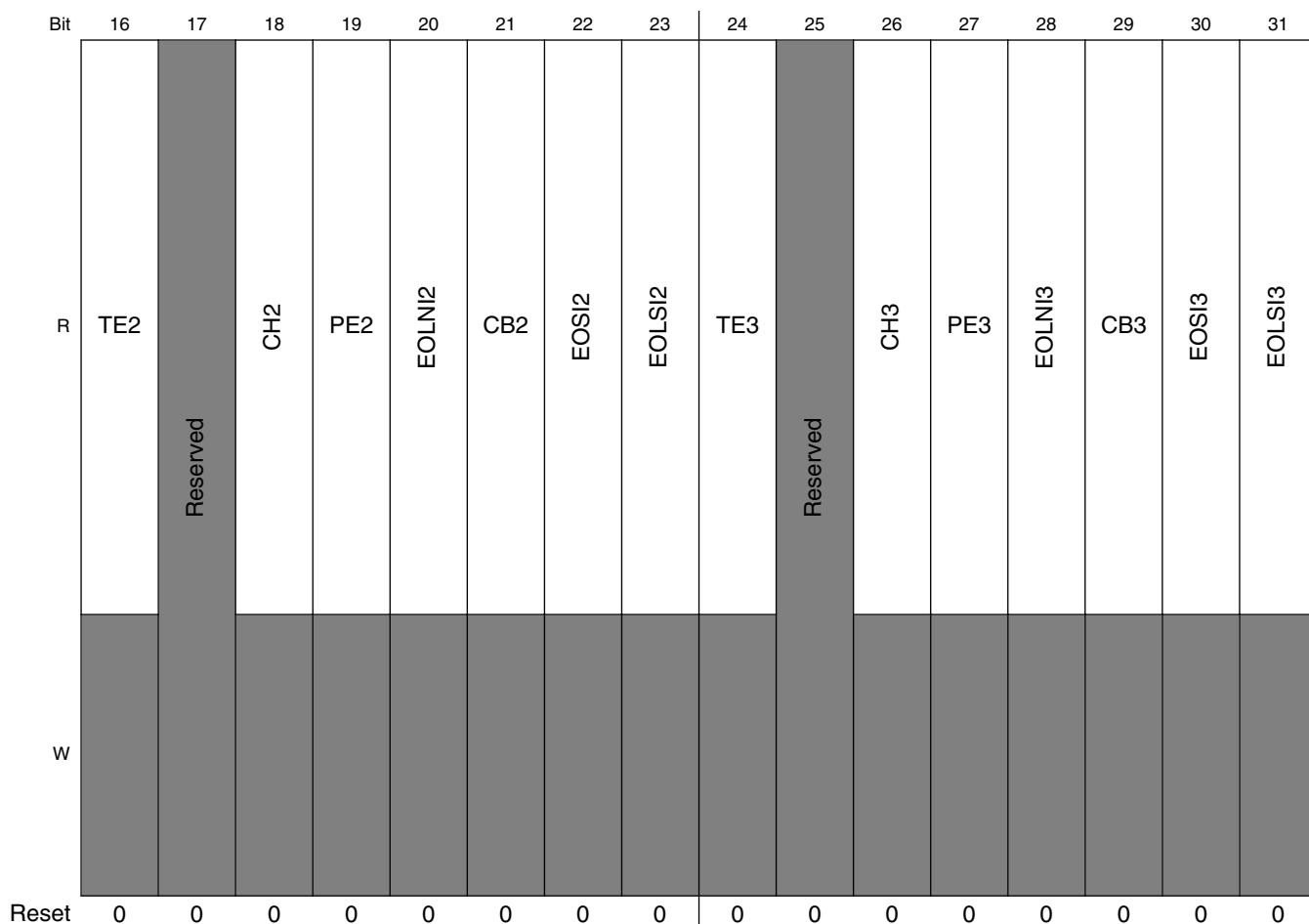
Field	Description
0–7 -	This field is reserved.
8–19 DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31 DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

20.3.18 DMA general status register (DMAx_DGSR)

The DMA general status register combines all of the status bits each channel into one register. This register is read-only.

Address: Base address + 300h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE0	Reserved	CH0	PE0	EOLN10	CB0	EOS10	EOLSI0	TE1	Reserved	CH1	PE1	EOLN11	CB1	EOS11	EOLSI1
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DMAx_DGSR field descriptions

Field	Description
0 TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1 -	This field is reserved.
2 CH0	Channel halted, channel 0
3 PE0	Programming error, channel 0
4 EOLNI0	End-of-links interrupt, channel 0
5 CB0	Channel busy, channel 0
6 EOSI0	End-of-segment interrupt, channel 0
7 EOLSI0	End-of-lists/direct interrupt, channel 0

Table continues on the next page...

DMAx_DGSR field descriptions (continued)

Field	Description
8 TE1	Transfer error, channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9 -	This field is reserved.
10 CH1	Channel halted, channel 1
11 PE1	Programming error, channel 1
12 EOLNI1	End-of-links interrupt, channel 1
13 CB1	Channel busy, channel 1
14 EOSI1	End-of-segment interrupt, channel 1
15 EOLSI1	End-of-lists/direct interrupt, channel 1
16 TE2	Transfer error, channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17 -	This field is reserved.
18 CH2	Channel halted, channel 2
19 PE2	Programming error, channel 2
20 EOLNI2	End-of-links interrupt, channel 2
21 CB2	Channel busy, channel 2
22 EOSI2	End-of-segment interrupt, channel 2
23 EOLSI2	End-of-lists/direct interrupt, channel 2
24 TE3	Transfer error, channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25 -	This field is reserved.
26 CH3	Channel halted, channel 3
27 PE3	Programming error, channel 3

Table continues on the next page...

DMAx_DGSR field descriptions (continued)

Field	Description
28 EOLNI3	End-of-links interrupt, channel 3
29 CB3	Channel busy, channel 3
30 EOSI3	End-of-segment interrupt, channel 3
31 EOLSI3	End-of-lists/direct interrupt, channel 3

20.4 DMA functional description

This section describes the function of the DMA controller.

20.4.1 DMA channel operation

All DMA channels support two different modes of operation: a basic mode ($MRn[XFE]$ is cleared) and an extended mode ($MRn[XFE]$ is set). In both modes, a channel can be activated by clearing and setting $MRn[CS]$, or through the single-write start mode using $MRn[CDSM_SWSM]$ and $MRn[SRW]$, or through an external control mode using $MRn[EMS_EN]$ and the external DMA_DREQ signal.

In basic mode, the channel can be programmed in basic direct mode or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode or extended chaining mode. Extended mode provides more capabilities, such as extended descriptor chaining, striding capabilities, and a more flexible descriptor structure.

The DMA controller supports misaligned transfers for both the source and destination addresses. In order to maximize performance, the source and destination engines issue one or more transactions to reach the desired alignment based on the rules described in [Source/destination transaction size calculations](#). The DMA always reads/writes the maximum number of bytes for a given transfer as described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input. Using 256 bytes over the RapidIO interface reduces packet overhead which translates to increased bandwidth utilization through the interface.

The DMA controller supports bandwidth control, which prevents a channel from consuming all the data bandwidth in the controller. Each channel is allowed to consume the bandwidth of the shared resources as specified by the bandwidth control value. After

the channel uses its allotted bandwidth, the arbiter grants the next channel access to the shared resources. The arbitration is round robin between the channels. This feature is also used to implement the external control pause feature. If the external control start and pause are enabled in the MR n , the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit, MR n [CC]. See [Table 20-262](#) for more complete descriptions of the channel states and state transitions.

20.4.1.1 Source/destination transaction size calculations

The DMA controller may issue smaller transactions from the source and destination address engines in an effort to reach alignment for improved performance.

The flow chart below shows the decision points made in determining the transaction size. The starting *Txfer_Size* is determined by min(BCR[BC],MR[BWC]), *Stride_En* is determined by SATR n [SSME] or DATR n [DSME], and *Stride_Size* is determined by SSR n [SSS] or DSR n [DSS].

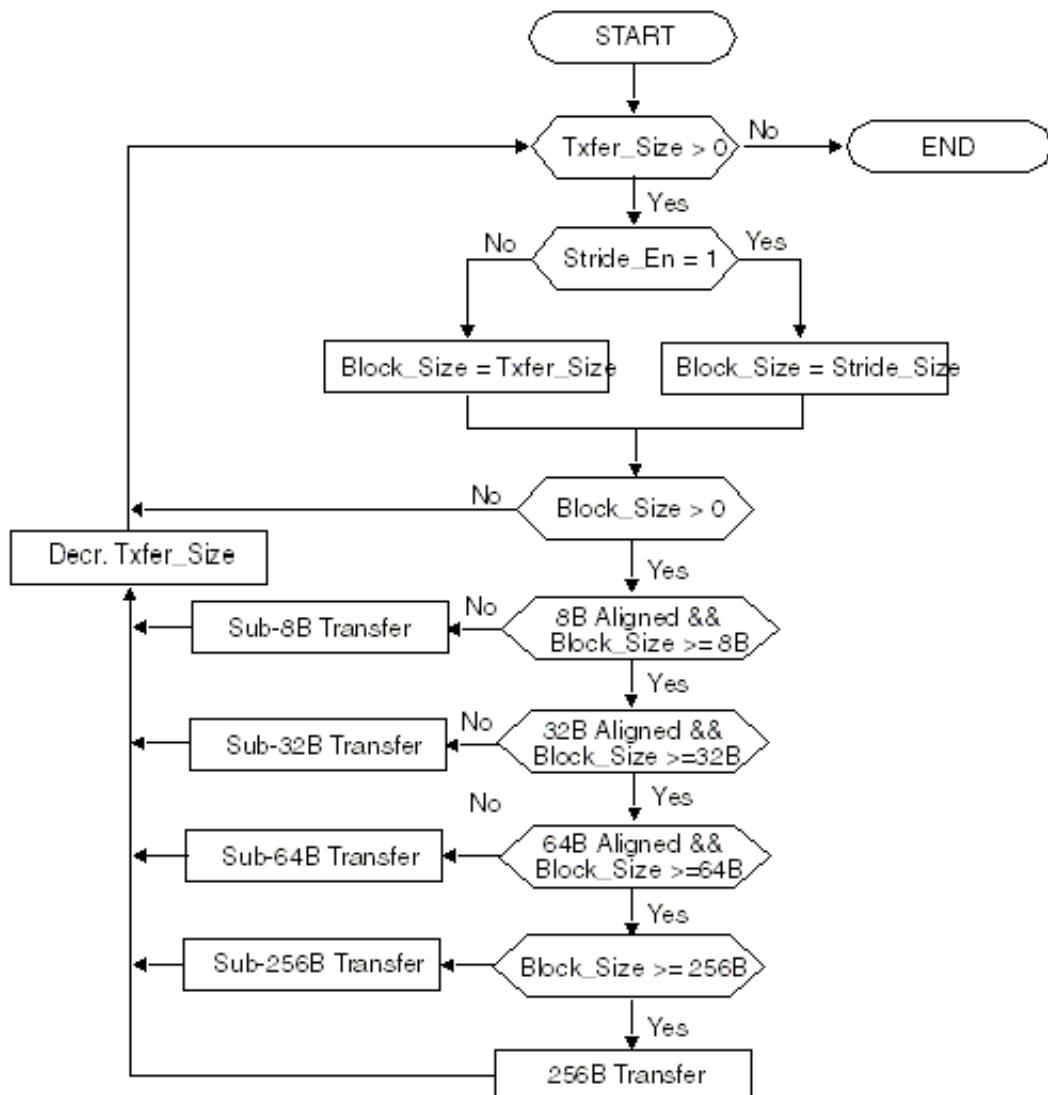


Figure 20-277. Source/destination engine transaction size flow chart

For example, if BCR[BC]=512 bytes and MR[BWC]=256 bytes, reading from starting address 0x5D0 will result in the following transaction sizes:

```

-- Channel arbitration --
0x5D0 - 16 bytes
0x5E0 - 32 bytes
0x600 - 128 bytes
0x680 - 64 bytes
0x6C0 - 16 bytes
-- Channel arbitration --
0x6D0 - 16 bytes
0x6E0 - 32 bytes
0x700 - 128 bytes
0x780 - 64 bytes
0x7C0 - 16 bytes

```

In the example above, the bandwidth control limits the channel from ever reaching the maximum transaction size of 256 bytes. Software should align addresses or increase the available bandwidth for best performance.

20.4.1.2 Basic DMA mode transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset.

The different modes of operation under the basic mode are explained in the following sections.

20.4.1.2.1 Basic direct mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer.

Software is responsible for initializing SAR n , SATR n , DAR n , DATR n , and BCR n registers. The DMA transfer is started when MR n [CS] is set. Software is expected to program all the appropriate registers before setting MR n [CS] to a 1. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
2. Initialize SAR n , SATR n , DAR n , DATR n and BCR n .
3. Set the mode register channel transfer mode bit, MR n [CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear, then set the mode register channel start bit, MR n [CS], to start the DMA transfer.
5. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR n [CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR n [EOSIE] is set.

20.4.1.2.2 Basic, direct, single-write start mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer.

Software is responsible for initializing the SATR n , DATR n , and BCR n registers. Setting MR n [SRW] configures the DMA controller to begin the DMA transfer either when SAR n is written or when DAR n is written, determined by the state of MR n [CDSM_SWSM]. Writing to SAR n initiates the DMA transfer if MR n [CDSM_SWSM] is set. Writing to DAR n initiates the DMA transfer if MR n [CDSM_SWSM] is cleared. The DMA controller automatically sets the channel start bit, MR n [CS]. Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR n), DATR n , and BCR n registers.
3. Set the mode register channel transfer mode bit, MR n [CTM], and the single-write start direct mode bit, MR n [SRW]. Other control parameters may also be initialized in the mode register. Set MR n [CDSM_SWSM] for transfers started using SAR n . Clear MR n [CDSM_SWSM] for transfers started using the DAR n .
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR n [CS].
5. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR n [CB] is automatically cleared by the DMA controller after the transfer is finished, if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR n [EOSIE] is set.

20.4.1.2.3 Basic chaining mode

In basic chaining mode, software must first build link descriptor segments in memory.

Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer.

The transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory.
2. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
3. Initialize CLNDAR n and ECLNDAR n to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR n [CTM], as well as MR n [XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear and then set the mode register channel start bit, MR n [CS], to start the DMA transfer.
6. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR n [CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

20.4.1.2.4 Basic chaining, single-write start mode

Basic chaining, single-write start mode allows a chain to be started by writing the current link descriptor address register (CLNDAR n).

(Note that ECLNDAR n must be written *first* so that the full 36-bit descriptor address is present when the chain starts.) Setting MR n [CDSM_SWSM] in the mode register causes MR n [CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR n [CDSM_SWSM], and clear the extended features enable bit MR n [XFE]. Also, clear the channel transfer mode bit, MR n [CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR n and ECLNDAR n to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR n [CS].
5. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR n [CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

20.4.1.3 Extended DMA mode transfer

The extended DMA mode also operates in chaining and direct mode.

It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting $MRn[XFE]$.

20.4.1.3.1 Extended direct mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities.

The bit settings are the same as in direct mode with the exception of the $MRn[XFE]$ being set. Striding on the source address can be accomplished by setting $SATRn[SSME]$ and setting the desired stride size and distance in $SSRn$. Striding on the destination address can be accomplished by setting $DATRn[DSME]$ and setting the desired stride size and distance in $DSRn$.

20.4.1.3.2 Extended direct, single-write start mode

Extended direct, single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities.

The bit settings are also the same with the exception of $MRn[XFE]$ being set. Striding on the source address can be accomplished by setting $SATRn[SSME]$ and setting the desired stride size and distance in $SSRn$. Striding on the destination address can be accomplished by setting $DATRn[DSME]$ and setting the desired stride size and distance in $DSRn$.

20.4.1.3.3 Extended chaining mode

In extended chaining mode, the software must first build list and link descriptor segments in memory.

Then $CLSDARn$ and $ECLSDARn$ must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list

descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR n and ECLSDAR n to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR n [CTM], to indicate chaining mode. MR n [XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear and then set the mode register channel start bit, MR n [CS], to start the DMA transfer.
6. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR n [CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

20.4.1.3.4 Extended chaining, single-write start mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer.

Setting MR n [CDSM_SWSM] causes MR n [CS] to be set automatically when CLSDAR n is written. (Note that ECLSDAR n must be written *first* so that the full 36-bit descriptor address is present when the chain starts.) The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR n [CDSM_SWSM] and MR n [XFE] and clear MR n [CTM] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see [Table 20-262](#)) to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR n [CS].
5. SR n [CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR n [CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR n [CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

20.4.1.4 External control mode transfer

An external control can be used to control DMA channels that support external control signals by setting $MR_n[EMS_EN]$.

The external control can direct the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or restart a paused DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion. Note that external control cannot cause a channel to enter a paused state.

The pause feature can be enabled by setting $MR_n[EMP_EN]$. $MR_n[BWC]$ specifies how much data to allow a specific channel to transfer before entering a paused state by clearing $MR_n[CS]$. Note however, that write data for a paused transfer may not have reached the target interface when so indicated. The channel can be restarted from a paused state by the asserted edge of $DREQ_B$ as driven by an external master. In chaining modes, the channel does not pause for descriptor fetch transfer; it only pauses during the actual data transfer.

Note that when operating the DMA in chaining mode the register byte count field, $BCR[BC]$, must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

The following signals are defined for the external control interface:

- **DMA_DREQ_B**-Asserting edge triggers a DMA transfer start or restart from a pause request. Sets $MR_n[CS]$. (Note that negating DMA_DREQ_B does NOT clear $MR_n[CS]$.)
- **DMA_DACK_B**-Indicates a DMA transfer currently in progress. $SR_n[CB]$ is set.
- **DMA_DDONE_B**-Indicates the completion of the DMA controller's involvement in the transfer and the readiness to accept a new DMA command. $SR_n[CB]$ is clear. Note however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in [Table 20-3](#). The timing diagram of the external control interface is shown in this figure.

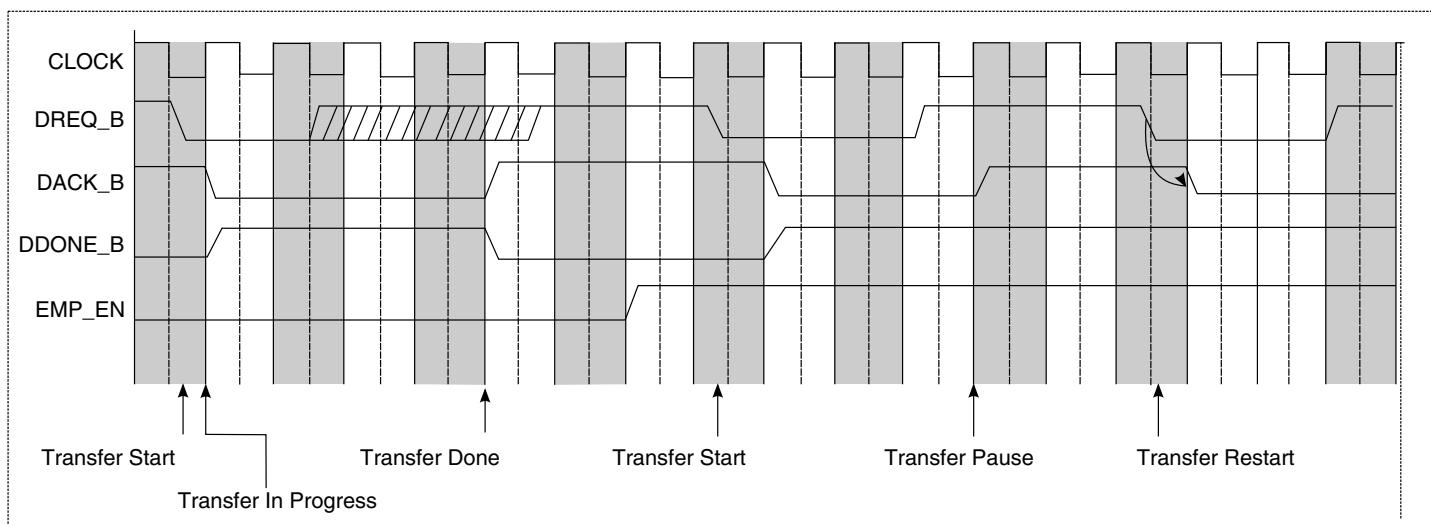


Figure 20-278. External control interface timing

20.4.1.5 Channel continue mode for cascading transfer chains

The channel continue mode (enabled when $MRn[CC]$ is set) offers software the flexibility of having the DMA controller start on descriptors that have already been programmed while software continues to build more descriptors in memory.

Software can set the end-of-links descriptor (EOLND) in basic mode, or end-of-lists descriptor (EOLSD) in extended mode, to cause the channel to go into a halted state while software continues to build other descriptors in memory. Software can then set CC to force hardware to continue where it left off. Channel continue is only meaningful for chaining modes, not direct mode.

If CC is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor. The channel busy ($SRn[CB]$) bit is cleared when the DMA controller reaches EOLND/EOLSD and is set again when it initiates the refetch of the link or list descriptor.

If CC is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their

respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

20.4.1.5.1 Basic mode

On a channel continue, the descriptor at the current link descriptor address registers (CLNDAR n and ECLNDAR n) is refetched to get the next link descriptor address field as updated by software.

The channel halts if NLNDAR n [EOLND] is still set. If EOLND is zero, the next link descriptor address is copied into CLNDAR n and ECLNDAR n and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

20.4.1.5.2 Extended mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR n and ECLSDAR n) address register is refetched to get the next list descriptor address field as updated by software.

The channel halts if NLSDAR n [EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR n and ECLSDAR n registers and the channel continues with another descriptor fetch of the current list descriptor address. As a result, two list descriptor fetches always exist after channel continue before the first link descriptor fetch and the first transfer.

20.4.1.6 Channel abort

Software can abort a previously initiated transfer by setting MR n [CA].

Once the DMA channel controller detects a zero-to-one transition of MR n [CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR n [CB]. Successful completion of a software initiated abort request can be recognized by MR n [CA] being set and SR n [CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR n [TE] is set), or the channel has completed all transfers, then SR n [CB] being cleared may not signify that the controller entered a halt state due to the abort request.

20.4.1.7 Bandwidth control

$MRn[BWC]$ specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware.

This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 Kbyte at a time when no other channel is active.

The maximum performance can be achieved on a single channel by disabling bandwidth control. This is recommended especially if only a single channel is utilized.

20.4.1.8 Channel state

This table defines the state of a channel based on the values of the channel start ($MRn[CS]$), channel busy ($SRn[CB]$), transfer error ($SRn[TE]$), and channel continue ($MRn[CC]$) bits.

Table 20-262. Channel state table

$MRn[CS]$	$SRn[CB]$	$SRn[TE]$	$MRn[CC]$	Channel state
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared $MRn[CS]$.
0	1	0	1	Channel remains in halt state.
-	1	1	-	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state
1	1	0	0	Transfer in progress
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue

20.4.1.9 Illustration of stride size and stride distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance.

The stride distance is added to the current base address to point to the next quantity of data to be transferred. This figure illustrates the stride size and distance parameters.

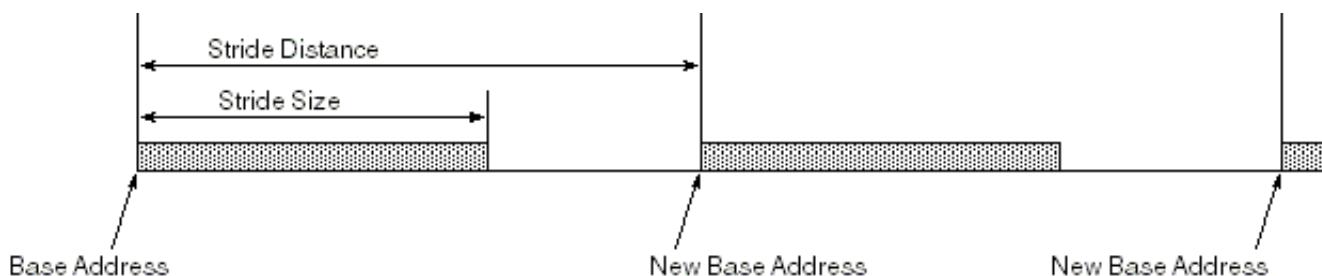


Figure 20-279. Stride size and stride distance

As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

20.4.2 DMA transfer interfaces

The DMA can be used to achieve data transfers across the entire memory map.

20.4.3 DMA errors

On a transfer error (for example, non-correctable ECC errors on memory accesses, parity errors on local bus flash controller, address mapping errors, and so on), the DMA halts by setting SR n [TE] and generates an interrupt if MR n [EIE] is set.

On a programming error, the DMA sets SR n [PE] and generates an interrupt if MR n [EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero
- Stride transfer started with a stride size of zero
- Illegal type, defined by SATR n [SREADTTYPE] and DATR n [DWRITETTYPE], used for the transfer.

20.4.4 DMA descriptors

The DMA engine recognizes the following types of descriptors:

- List descriptors connect lists of link descriptors.
- Link descriptors describe the DMA activity that is to take place.

DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the NLNDAR n [EOLND] bit in the next link descriptor and NLSDAR n [EOLSD] in the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

This table summarizes the DMA list descriptors.

Table 20-263. List DMA descriptor summary

Descriptor field	Description
Next list descriptor extended address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor extended address registers.
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor extended address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor extended address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

This table summarizes the DMA link descriptors.

Table 20-264. Link DMA descriptor summary

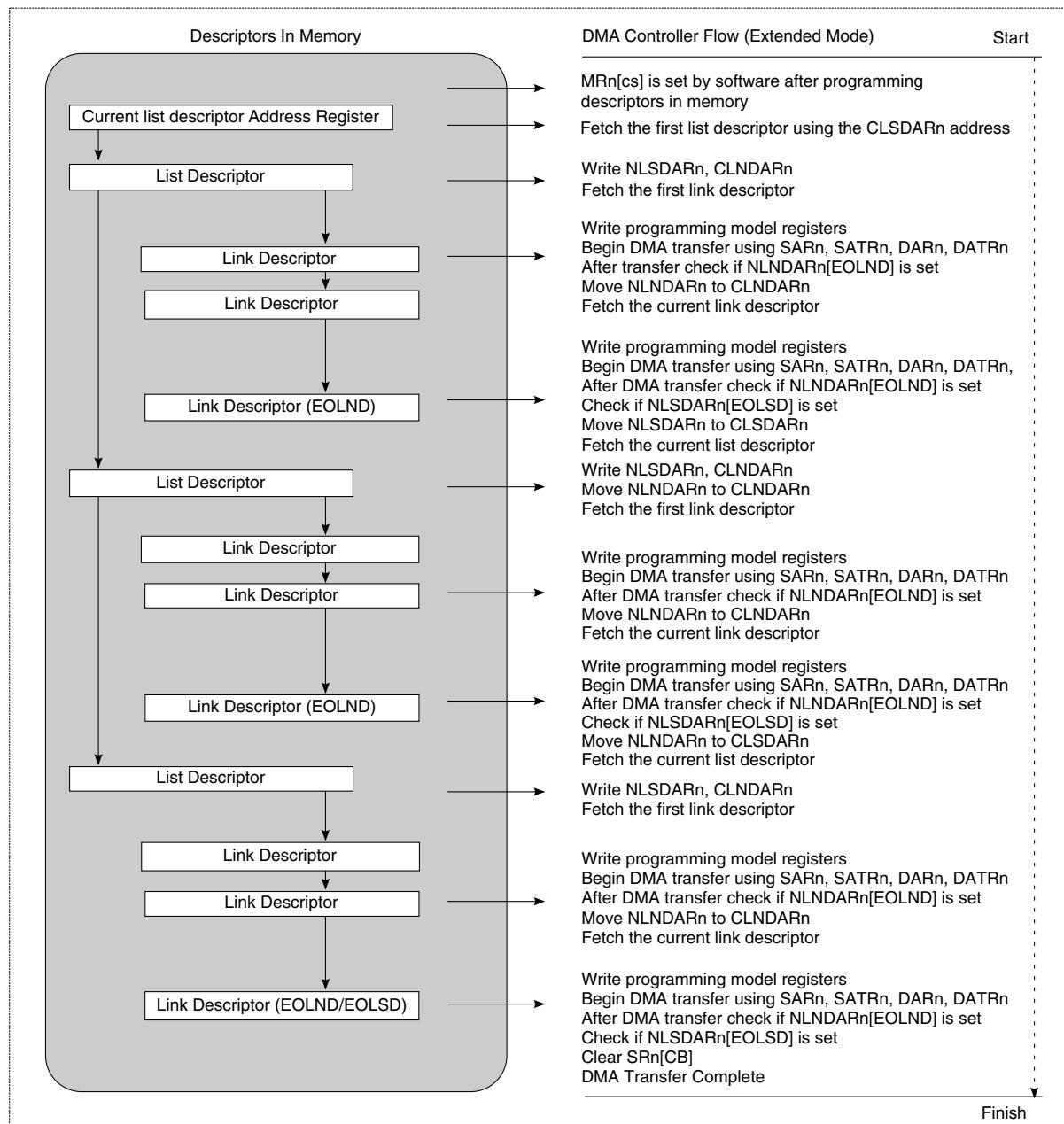
Descriptor field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor extended address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the extended next link descriptor address registers

Table continues on the next page...

Table 20-264. Link DMA descriptor summary (continued)

Descriptor field	Description
Next link descriptor address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the next link descriptor address registers.
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the byte count register.

This figure describes the DMA transaction flow.

**Figure 20-280. DMA transaction flow with DMA descriptors**

This figure describes the format of the list descriptors.

Offset	
0x00	Next List Descriptor Extended Address
0x04	Next List Descriptor Address
0x08	First Link Descriptor Extended Address
0x0C	First Link Descriptor Address
0x10	Source Stride
0x14	Destination Stride
0x18	Reserved
0x1C	Reserved

Figure 20-281. List descriptor format

This figure describes the format of the link descriptors.

Offset	
0x00	Source Attributes
0x04	Source Address
0x08	Destination Attributes
0x0C	Destination Address
0x10	Next Link Descriptor Extended Address
0x14	Next Link Descriptor Address
0x18	Byte Count
0x1C	Reserved

Figure 20-282. Link descriptor format

20.4.5 DMA controller limitations and restrictions

This section is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, avoid stride sizes less than 64 bytes. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are as follows:

- All interface capabilities from where descriptors are being fetched must support read sizes of 32 bytes or greater.
- If MR_n[SAHE] is set, the source interface transfer size capability must be greater than or equal to MR_n[SAHTS]. The source address must be aligned to a size specified by SAHTS.
- If MR_n[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR_n[DAHTS]. The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if MR_n[DAHE] is set and source striding is not supported if MR_n[SAHE] is set.

20.5 DMA system considerations

This section provides information about how to make most effective use of the DMA channels.

20.5.1 Unusual DMA scenarios

The following is a description of unusual DMA paths including explanations of why some functional modules cannot serve as DMA targets.

The following topics are addressed:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

20.5.1.1 DMA to core

The L1 cache cannot be a direct DMA target because it cannot be directly addressed by software.

However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

20.5.1.2 DMA to configuration, control, and status registers (CCSR)

Because any internal register can be addressed with the four-channel DMA controller, configuration, control, and status registers throughout the device are valid DMA targets.

However, the primary purpose of DMA, to reduce processor load by moving large blocks of data, is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I²C controller or programmable interrupt controller (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

20.5.1.3 DMA to I²C

The I²C controller is not transparent to DMA transfers.

Observe the caveats listed in [DMA to configuration, control, and status registers \(CCSR\)](#) when accessing any I²C register, including the data register (I2CDR).

20.5.1.4 DMA to DUART

The DUART provides complete and sophisticated DMA support, which is described in [FIFO Mode](#).

20.5.1.5 DMA in multi-partition systems

The DMA controller uses a privileged copy of the LAWs to resolve source and destination targets.

This means that when a logical address is passed to the DMA controller, it compares this logical address with true physical addresses in the LAWs. Because of this behavior, the DMA controller has the capability to resolve its target without the need for a PAMU, providing optimization when targeting outbound I/O transactions. Because the PAMU only authorizes inbound transactions, it does not authorize transactions that resolve to outbound I/O. Note that this behavior only applies to outbound I/O transactions from the DMA controller; only the DMA controller has access to the privileged copy of the LAWs.

In a multi-partitioned system (such as a system partitioned and controlled under a Hypervisor), where logical addresses are not mapped one-to-one to physical addresses, one of the following three techniques must be implemented to help ensure absolute partitioning:

1. The programming of the DMA controller from a guest OS must be virtualized in software so that the guest OS must make Hypervisor calls (hcalls) in order to program the DMA. The Hypervisor is aware of the global system memory map and can block any OS from trying to target an address it deems invalid for that OS.
2. The Hypervisor must enforce the following logical address restrictions on all partitions:
 - a. All I/O logical addresses must be mapped one-to-one using their true physical addresses.
 - b. All guest OSs must map their I/O logical address space in the same globally unique system-wide range, which does not overlap other non-I/O addresses. All OSs use this space only for I/O.

For example, the following represents a valid setup:

- OS-A is aware of logical addresses A-C, that are mapped one-to-one with physical addresses A-C.
- OS-B is aware of logical addresses D-F, that are mapped one-to-one with physical addresses D-F.
- A law entry for DDR maps the entire SDRAM space to physical addresses A-F.
- The PCI express controller 1 outbound window is mapped to physical addresses 2-3.

In this example all logical addresses are mapped one-to-one with true physical addresses, and the outbound PCI express space does not overlap either OS's view of logical space.

As a counter-example, the following represents an invalid setup:

- OS-A is aware of logical addresses 0-2, that are mapped to physical addresses A-C.
- OS-B is aware of logical addresses D-F, that are mapped one-to-one with physical addresses D-F.
- A LAW entry for DDR maps the entire SDRAM space to physical addresses A-F.
- The PCI express controller 1 outbound window is mapped from 1-2.

In this example, the setup is invalid because the logical addresses of OS-A overlap the outbound PCI express physical I/O window. If the DMA targets logical address 1, it will use its privileged copy of the LAWs to resolve the target, and because address 1 overlaps with the outbound PCI express window, it will target PCI express instead of DDR, which may not be intended.

3. Do not grant direct DMA controller access to a guest OS which cannot meet one of the previous two constraints.

Chapter 21

Data Path Acceleration Architecture (DPAA) Overview and P4080 DPAA Implementation

21.1 DPAA Introduction and Terms

The data path processing subsystem is an implementation of the QorIQ Data Path Acceleration Architecture (DPAA). The QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual (DPAARM) describes the superset of DPAA functionality. The chip implements a unique subset of this functionality, as described in the [P4080-Specific DPAA Implementation Details](#).

As shown in the following figure, the DPAA consists of a parse, classify, and distribute module (PCD) and an accelerator module (FMan), along with related network interfaces (dTSEC and 10GEC), hardware offload accelerators (SEC, and PME) and the infrastructure blocks that support these accelerators (QMan and BMan). Each of these accelerators provides its own unique set of functionality and thus each has its own unique requirements for how software must program and interact with that accelerator.

The DPAA's common infrastructure modules and the desire to be able to pass data received from one accelerator to another led to some common requirements, restrictions, and conventions. This chapter outlines requirements and conventions that are common to multiple accelerators within the DPAA.

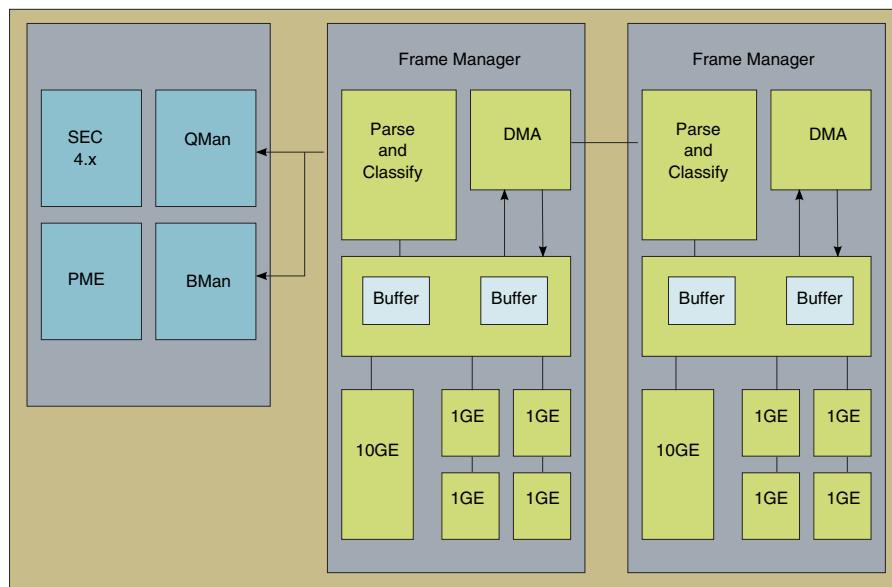


Figure 21-1. QorIQ Data Path Acceleration Architecture

The following list describes common DPAA terms and definitions.

Buffer

Region of contiguous memory, allocated by software, may be managed by the DPAA Buffer Manager (BMan).



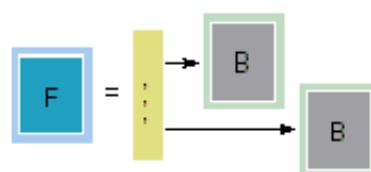
Buffer pool

Set of buffers with common characteristics (mainly size, alignment, access control)



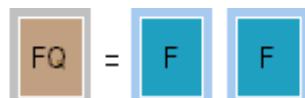
Frame

Contents of a single buffer or multiple buffers referenced by a table that hold data. For example, packet payload, header, and other control information.



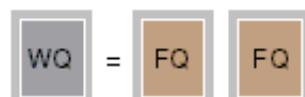
Frame queue (FQ)

FIFO of frames



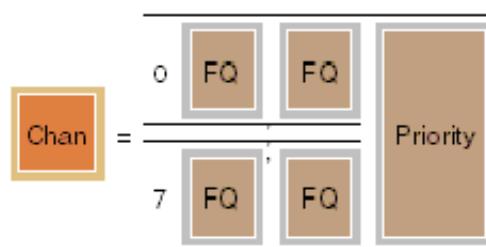
Work queue (WQ)

FIFO of frame queues (FQs)



Channel

Set of eight work queues (WQs), with hardware provided prioritized access



Dedicated channel

Channel statically assigned to a particular end point from which that end point can dequeue frames. End-point may be a CPU, FMan, PME, or SEC.

Pool channel

A channel statically assigned to a group of end points from which any of the end points may dequeue frames.

21.2 Data Formats Used in the DPAA

Data-to-be-processed is passed within the DPAA in distinct units known as “frames.” The actual data is held in buffers in memory and a description of the frame is what is passed within the DPAA.

21.2.1 Frame Descriptor (FD)

A frame descriptor (FD) is the basic element queued by the QMan. See Chapter 8, "Frame Manager" in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* for a description of the QMan's usage of the FD. The use and interpretation of some fields is specific to the producer or consumer of the FD. Such usage is described in the chapter/section covering that module.

21.2.1.1 FD Format

Table 21-1. Frame Descriptor (FD)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						
DD	LIODN_OFFSET						BPID										ELIODN_OFFSET			Reserved			ADDR														
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63						
ADDR																																					
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95						
FORMAT	OFFSET										LENGTH																										
	LENGTH or CONGESTION WEIGHT																																				
96	97	98	99	10	10	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11	11	11	11	12	12	12	12	12	12	12	12					
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7
STATUS/CMD																																					

Table 21-2. FD Field Descriptions

FD Bits	FD Name	FD Field Description
0-1	DD	<p>Dynamic Debug marking code point</p> <p>A frame with a non-zero debug code point can generate a trace event at various enqueue and/or dequeue points within the QMan. Individual QMan users have different mechanisms and criteria for setting non-zero DD values.</p>
2-7	LIODN_OFFSET	<p>Frame LIODN Offset (6 lsbs of complete LIODN_OFFSET)</p> <p>The LIODN_OFFSET is set when the FD is enqueued. It is used by a hardware module that dequeues the FD as part of the memory access control mechanism supported by the PAMU (see Role of the PAMU in the DPAA)</p> <p>See Accessing Memory Using Logical IO Device Numbers (LIODNs) for more information on the use of this field.</p>
8-15	BPID	<p>Buffer Pool ID</p> <p>The Buffer Pool ID is the number of the BMan buffer pool to which the buffer at ADDR should be released, if it is to be released to BMan.</p>
16-19	ELIODN_OFFSET	<p>Frame Extended LIODN offset (4 msbs of complete LIODN_OFFSET)</p> <p>The complete 10-bit LIODN_OFFSET value is a concatenation of {ELIODN_OFFSET, LIODN_OFFSET}.</p> <p>See Accessing Memory Using Logical IO Device Numbers (LIODNs) for more information on the use of this field.</p>

Table continues on the next page...

Table 21-2. FD Field Descriptions (continued)

FD Bits	FD Name	FD Field Description
20-23	-	Reserved
24-63	ADDR	<p>Address</p> <p>The memory address of the start of the buffer holding the frame data or the buffer containing the scatter/gather list describing the frame</p>
64-66	FORMAT	A coded value that defines the format of the OFFSET and LENGTH fields, and of the frame referenced by this FD. See Frame Format Codes for a description of these codes.
O p t i o n a l i f i e l d s	OFFSET	<p>This field is valid only when FORMAT bits 65-66 are 00; otherwise, a frame offset value of 0 is implied.</p> <p>Frame offset is the number of bytes from the frame address (ADDR) at which actual data begins. Note that this field is not present in all FDs</p>
76-95	LENGTH	Total number of valid bytes of data in the frame. Note that this field is not present in all FDs.
67-95	CONGESTION WEIGHT	A value used by the QMan for certain congestion management/avoidance calculations. Note that this field is not present in all FDs.
96-127	STATUS/CMD	<p>Status or Command</p> <p>This field allows the sender of a frame to communicate some out-of-band information to the receiver of the frame. For example, this field can be used to communicate a command describing what processing is to be performed to a hardware accelerator or the error/output status after a hardware accelerator has finished processing a frame.</p>

21.2.1.2 FD Considerations

To support the diverse needs of the various accelerators in the DPAA, multiple frame formats are defined. FD[FORMAT] designates the frame format described by the FD. Depending on the situation, one of the following must be stored:

- Data beginning at an offset from the actual start of the buffer, saving room at the beginning of the buffer (for adding new headers, for example)
- Large amount of data in contiguous memory

Because these cases tend to be mutually exclusive, formats are defined that trade off between the size of FD[LENGTH] and FD[OFFSET]. Depending on the FD[FORMAT], one of the following is present in the FD:

- FD[LENGTH] and FD[OFFSET]
- Just FD[LENGTH]
- Just FD[CONGESTION WEIGHT]

These fields occupy the same bits within the FD.

21.2.2 Multi-Buffer Frames

The actual data in a frame is held in one or more memory buffers. If multiple buffers are required, a scatter/gather table (also known as a block vector or IO vector table) is used to describe the buffers in a multi-buffer frame. See [Scatter/Gather Entry Format](#) for a description of this data structure.

21.2.2.1 Scatter/Gather Entry Format

Table 21-3. Scatter/Gather Table Entry Format

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Reserved (must be 0)																													ADDR		
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
1	ADDR																															
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
2	E	F	LENGTH																													
	96	97	98	99	10	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11	11	11	11	11	12	12	12	12	12	12	12
3	Reserved (must be 0)										BPID										Reserved (Must be 0)	OFFSET										

Table 21-4. Scatter Gather Table Entry Field Description

Bits	Name	Description
0-23	-	Reserved (must be 0)
24-63	ADDR	Address of the buffer referenced by this table entry. This buffer may contain data or it may contain another scatter/gather table.
64	E	Extension bit. If set, this table entry points to a buffer that contains another scatter/gather table.
65	F	Final bit. If set, this is the last table entry for this frame. Note that the E (extension) bit takes precedence over the F (final) bit. If both are set, the F bit is ignored and processing continues with the entries in the scatter/gather table in referenced buffer.
66-95	LENGTH	Depending on the context, this field either specifies the number of valid bytes of data in the referenced buffer or the size of the referenced buffer (in the case that empty buffers are being provided to an accelerator). Note that if the E (extension) bit is set, LENGTH is ignored.
96-103	-	Reserved (must be 0)
104-111	BPID	Buffer Pool ID: The number of the BMan buffer pool to which this buffer should be released, if it is released to BMan.
112-114	-	Reserved (must be 0)
115-127	OFFSET	An offset in bytes from the ADDR at which valid data starts Note that if the E (extension) bit is set, "valid data" is a scatter/gather table.

21.2.2.2 Multi-Buffer Frame Considerations

Important multi-buffer frame considerations are as follows:

- The first scatter/gather table in a multi-buffer frame is held in the buffer referenced by the FD. In the case of a short, multi-buffer frame, the table starts at OFFSET bytes from the beginning of the buffer.
- “Trees” or hierarchy are not supported, because for simple, multi-buffer frames, processing never returns to the table in the original buffer.
- If the E bit is set, it indicates that the table entry points to another buffer containing more table entries. When a scatter/gather table entry with the E bit set is encountered, processing proceeds from the first entry in the new table found in the new buffer. This new table may begin at some offset from the start of the buffer as defined by the OFFSET field in the entry with the E bit set.
- In simple, multi-buffer frames, the LENGTH field in a table entry with its E bit set can be ignored.
- The E bit takes precedence over the F bit (that is, if both are set in an entry, the F bit is ignored).

This figure shows a simplified representation of a "long" simple frame using a scatter/gather table. Note that this diagram does not show the use of the Extension bit.

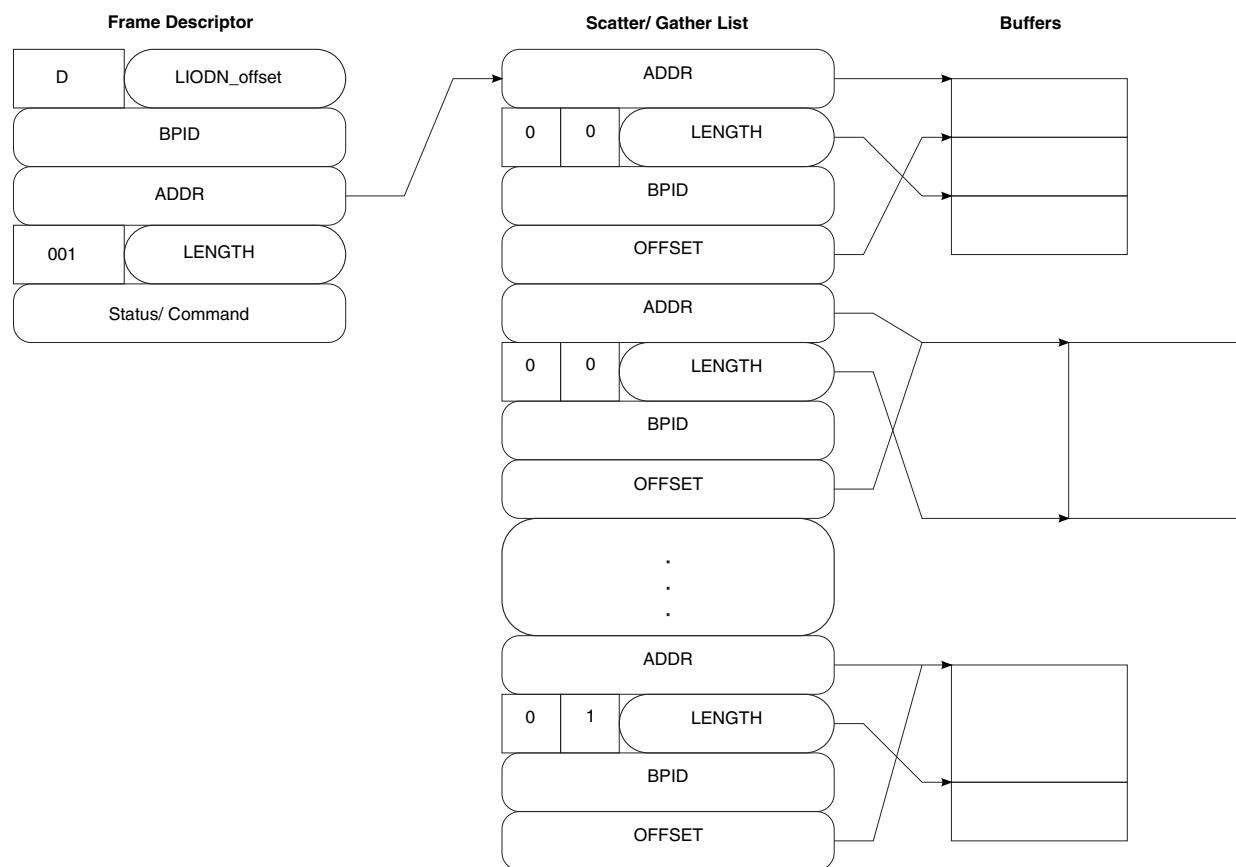


Figure 21-2. Simplified Representation of a Long, Multi-Buffer, Simple Frame

21.2.2.3 Situations Where Multi-Frame Buffering Stops

Multi-buffer frame processing stops in the following situations:

- When the data referenced by an entry with the F bit is processed regardless of the LENGTH indicated in the FD for the frame. In some cases, a mismatch (less data found in the frame compared to the FD length) may be an error condition for the accelerator module that is performing the processing, and it is reported as such.
- Processing also stops when the number of bytes specified by the overall length in the frame's FD have been processed. In this case, it is not necessary to have encountered an entry with the F bit set and it is not considered an error when this occurs.

21.2.3 Single-Buffer Frames

Most frame formats consist of a single delineated unit of data such as a single packet or Protocol Data Unit (PDU).

For a single buffer, the FD[LENGTH] field represents the total amount of valid data in the frame. The OFFSET represents the offset to the start of valid data.

For multiple buffers (used for scatter/gather), the FD[LENGTH] field still represents the total amount of valid data in the frame, but the OFFSET represents the offset to the start of the scatter/gather table in the first buffer.

21.2.4 Compound Frames

Frames may also consist of multiple, related, distinct units such as the encrypted form of a packet along with the decrypted form of the same packet. These are known as compound frames, which consist of two or more simple frames. Each simple frame in a compound frame can in turn be stored in a single buffer or in multiple buffers. Compound frames exist so that all of the related data can be passed in a single unit within the DPAA. If an FD has a FORMAT code that identifies it as a compound frame then the FD[CONGESTION WEIGHT] is used to by the QMan for active queue management calculations such as WRED.

21.2.4.1 When to Use Compound Frames

Compound frames are used for the following purposes:

- To pass multiple, distinct pieces of data either to or from a hardware accelerator. For instance, it may be required to pass frames containing both encrypted and decrypted forms of a packet.
- To supply empty buffers to a hardware accelerator into which it may place its output. In this case, it is not desirable to use the BMan. Any LENGTH and OFFSET fields that refer to a specific buffer have special meaning; OFFSET specifies the byte offset from ADDR where the accelerator should start storing data and LENGTH gives the number of bytes of data which can be stored in the remainder of the buffer.

Consider the following when using compound frames:

- When multiple input or output frames are described by a compound frame, their order is consumer-dependent.
- If a compound frame is used to pass empty buffers to a consumer for its output, those buffers are in the first frame of the compound frame.
- If a module uses a compound frame to return the input frame as well as its output frame, the output frame is the first frame of the compound frame and the input frame is the second frame.

21.2.4.2 Compound Frame Considerations

Important compound frame considerations are as follows:

- Compound frames use the same scatter/gather table format as simple frames, but with slightly different semantics for the fields in the table entries.
- The buffer referenced by FD[ADDR] of a compound frame must contain a scatter/gather table (the compound scatter/gather table). A compound frame contains FD[CONGESTION WEIGHT] but does not contain FD[LENGTH] or FD[OFFSET]. The compound scatter/gather table must start at FD[ADDR].
- Each entry in the compound scatter/gather table references a simple frame. ADDR, LENGTH, BPID, and OFFSET fields in the compound scatter/gather table entry replace the corresponding fields in an FD. A compound frame contains FD[CONGESTION WEIGHT] but does not contain FD[LENGTH] or FD[OFFSET]. The compound scatter/gather table must start at FD[ADDR].
- The E bit is set if the simple frame occupies multiple buffers. The E bits in multiple entries in the compound scatter/gather table may be set, because each simple frame in the compound frame may occupy multiple buffers. In this case, the buffer at ADDR contains a scatter/gather table.
- The scatter/gather table(s) that make up a simple frame that is part of a compound frame follow the same semantics as described in [Scatter/Gather Entry Format](#).
- There is no equivalent in the compound scatter/gather table for the FORMAT field in the FD. The entries in a compound scatter/gather table cannot themselves be compound frames; the E bit is used to indicate a multi-buffer frame, and the LENGTH and OFFSET fields supported in the scatter/gather entry are a superset of the fields in an FD.
- The F bit must be set in the last entry of a compound frame scatter/gather table.

This figure shows a representation of a compound frame.

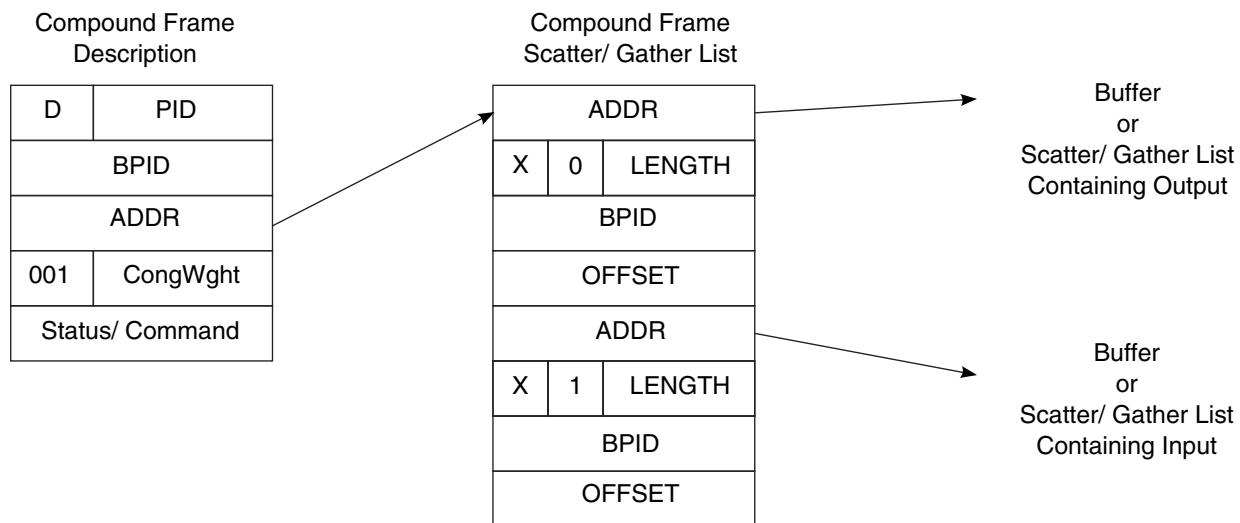


Figure 21-3. Simplified Representation of a Compound Frame

21.2.5 Simple Frames

Simple frames can be either short or long. Regardless of the length of the frame, data is stored in a single-buffer frame, whereas scatter/gather tables are stored in a multi-buffer frame.

A short frame can be no more than (1 Mbyte - 1 byte)-long but the data (in the case of a single buffer frame) or scatter/gather table (in the case of a multi-buffer frame) can be stored starting at an offset from the beginning of the buffer referenced by the FD. Long frames can be as much as (512 Mbytes - 1 byte)-long, but the data or scatter/gather table must be stored starting at the beginning of the buffer referenced by the FD.

Because network data (packets) are typically much less than 64 Kbytes in size, the short format is useful for most network processing needs.

21.2.6 Frame Format Codes

This table defines the frame format codes and provides a brief description of the frame format.

Table 21-5. Frame Format Codes

Value	Mnemonic	Definition	Size of LENGTH, OFFSET, and CONGESTION WEIGHT Fields
'000'	Short single buffer simple frame	Simple frame Single buffer, Offset and "small" length	9b OFFSET, 20b LENGTH
'010'	Long single buffer simple frame	Simple frame, single buffer, "big" length	29b LENGTH No OFFSET
'100'	Short multi-buffer simple frame	Simple frame, Scatter Gather table, Offset and "small" length	9b OFFSET, 20b LENGTH
'110'	Long multi-buffer simple frame	Simple frame, Scatter Gather table, "big" length	29b LENGTH No OFFSET
'001'	Compound frame	Compound Frame	29b <CONGESTION WEIGHT> No LENGTH or OFFSET
'011'	NA	Reserved	Not defined
'101'	NA	Reserved	Not defined
'111'	NA	Reserved	Not defined

21.2.7 Frame Formats Supported by Accelerators

Hardware accelerators (FMan, PME, and SEC) support a subset of the DPAA frame formats. This table summarizes the frame formats supported by various accelerators.

Table 21-6. Frame Format Support Matrix

	Short, Single Buffer		Long, Single Buffer		Short Multi-Buffer		Long Multi-Buffer		Compound Frames	
	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
FMan	Yes	Yes	No	No	16-entry table E bit not supported	16-entry table E bit not supported	No	No	No	No
PME	Yes	Return input Frame only	Yes	Return input frame only	Yes	Return input frame only	Yes	Return input frame only	Yes	Yes
SEC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

21.2.8 Special Values and Exceptions

The ADDR, BPID, and LENGTH fields can have special meanings under certain conditions, as follows:

- FDs
- Compound frame scatter/gather tables
- Multi-buffer scatter/gather tables

Exceptions and special behaviors are as follows:

- When the LENGTH field that describes a specific buffer is zero, this indicates that the buffer contains no data. If an accelerator finds a LENGTH of zero it should not access (read or write) the memory at ADDR.
- The LENGTH field in a scatter/gather table entry of a simple multi-buffer frame with its E (extension) bit set is an exception, because the value of this field is ignored.
-  If an ADDR field that describes a specific buffer is zero, it indicates that there is no buffer. Regardless of the value of LENGTH, the “buffer” at ADDR=0 should not be accessed (read or written). Note that this means that buffers starting at address 0 are not valid buffers and this memory should not be used for buffers.

21.2.9 Releasing Buffers to the BMan

 Accelerators only release buffers to the BMan that they have processed or, in the case of a buffer with a zero LENGTH, that they have passed over during processing. ~~The exact configuration is dependent on the specific accelerator.~~

- ~~Buffers with a non-zero ADDR are released.~~
- ~~Buffers with a zero ADDR, BPID, and LENGTH are not released.~~

REQUIREMENT: Because processing stops when the frame’s total length of data is processed or after the data referenced by a scatter/gather entry with its F (final) bit set is processed, ensure that all buffers in a frame are released. For example, software must ensure that there are no entries in a scatter/gather table for a multi-buffer frame that are beyond the frame’s total length or described in entries after the entry with the F bit set.

21.3 Accessing Memory Using Logical IO Device Numbers (LIODNs)

This topic describes use of LIODNs and peripheral access management units (PAMUs).

21.3.1 Role of the PAMU in the DPAA

QorIQ multicore SoCs include peripheral access management units (PAMUs), which provide a number of functions including memory access control. When the PAMU is enabled, any hardware module in the SoC other than the cores access memory through PAMU. The PAMU can authorize the access, remap the memory address, and remap the transaction type (for example, a simple "write" to a "write with allocate to cache").

The PAMU determines what action to take and whether to authorize the action on the basis of the memory address, a logical IO device number (LIODN), and a configured table (logically) indexed by a LIODN and address. Hardware devices that need to access memory must provide a LIODN in addition to the memory address.

21.3.2 Using Multiple LIODNs

This topic describes the benefits and requirements of using multiple LIODNs.

21.3.2.1 Benefit of Using Multiple LIODNs

By using different LIODNs, a single hardware module can perform memory accesses on behalf of different requestors with the mapping and access controls appropriate to that requestor.

21.3.2.2 LIODN Requirements

Some important LIODN requirements are as follows:

- The requestor must communicate the LIODN to the hardware blockmodule. In the DPAA, this is done using the “<LIODN OFFSET>”. <LIODN OFFSET> is formed by concatenating LIODN_OFFSET and ELIODN_OFFSET from the FD with the latter used as the 4 msbs.
- Hardware module determine the LIODN to use in accessing memory by adding <LIODN OFFSET> to a configured <LIODN BASE>. Because <LIODN OFFSET> is in the FD, the module can use a different LIODN for each frame.
- Some hardware module may make different types of memory accesses as a result of dequeuing a frame. For instance, they may access per queue context/state descriptors as well as reading and writing frame data. These modules may have different <LIODN BASE> values for these different types of access. These modules compute

different LIODNs using the <LIODN OFFSET> from the FD and the appropriate <LIODN BASE> configured for the type of access.

- The <LIODN OFFSET> must be set by the producer (enqueuer) of the FD. For software portals, this is done by the QMan from values configured for the portal to ensure that accesses by hardware module on behalf of software are controlled. In other words, software running on a core cannot get a hardware blockmodule to make accesses to memory, because it is not permitted to make these accesses by setting the <LIODN OFFSET> value in an FD.

21.4 Packet Walk-Through Example

The following example walks a packet through a DPAA-enabled, QorIQ device to illustrate how the DPAA offloads and accelerates packet forwarding while leaving key decisions under software control:

1. Datapath processing begins when Ethernet frames arrive at a network interface, assumed to be one of the Ethernet MACs (dTSEC and 10GEC) within a Frame Manager (FMan). Alternatively, packets can arrive across a peripheral bus from an external network interface, in which case, the same packet walk-through stages can be applied with the help of a CPU acting as an I/O processor.
2. After FMan receives the Ethernet frame, it requests one or more buffers from the hardware Buffer Manager (BMan) to store the frame. BMan maintains pools of buffers, each with software-defined characteristics, and FMan is initialized to request a buffer from the most appropriate pool. If a sufficiently large buffer cannot be found for the incoming frame, FMan stores the frame across several smaller buffers and create a scatter/gather list for these buffers.
3. FMan's configurable parsing and filing capabilities can perform initial classification, sufficient to steer a packet toward a control processor, or toward one of the datapath processors for flow-specific processing (or additional classification by means of software). The steering of a packet towards a processor or a group of processors could be also based on differentiating flows by means of the Quality of Service attributes of the flow (for example, DSCP, IP precedence, or by user-defined proprietary headers).
4. Steering is accomplished by FMan issuing an enqueue command to QMan with the designated Frame Queue ID, along with frame parameters such as Frame Queue ID and Color Marking. In this example FMan's initial classification causes it to select a Frame Queue ID which the Queue Manager uses to steer the Frame Queue to the dedicated channel of logical CPU#3, a CPU operating in a datapath role. Potentially, the Frame Queue could be selected based on the QoS requirements of the flow with a policing profile associated with the selection.

5. Continuing the example, QMan places the Frame Queue onto Work Queue#5 of CPU#3's dedicated channel. The Frame Queue was queued to CPU#3 due to user configuration decisions that all packets belonging to this specific flow should be processed by CPU#3, possibly to take advantage of special processing instructions locked in CPU#3's private cache, or due to user-defined load balancing. The Frame Queue was placed in Work Queue#5 for QoS reasons, as the amount of traffic processed from each Work Queue relative to other work queues is also user-configurable. All Frame Queues on a Work Queue have equal priority, but the amount of traffic that the CPU draws from each is user-configurable to allow fairness and appropriate bandwidth allocation.
6. Once configured, QMan can take care of the packet/frame level scheduling requirements of the CPU, that is, QMan would appropriately schedule the Work Queue and Frame Queue within the Work Queue. QMan can be configured to perform a stashing operation in response to a CPU (or co-processor) accessing a Frame Queue.
7. CPU#3 performs protocol processing on the packet, including modification of the packet data in the buffer. Any modifications which change the length of the packet would be noted by means of changes to the frame. The CPU determines that the packet requires IPsec ESP processing, and enqueues the frame back to the QMan using the FQ ID associated with the packet's specific ESP tunnel.
8. The QMan uses this Frame Queue ID to determine that the next consumer of the Frame Queue is the SEC, and places the Frame Queue onto Work Queue#5 of the SEC's dedicated channel. The SEC pulls Frame Queues from the Work Queues in its dedicated channel according to user-defined weights in a WFQ model.
9. The SEC dequeues and processes data from the Frame Queue, adding the tunnel IP header, ESP header, IV, trailer, and HMAC, and writing encrypted data to either the original buffers, or new buffers which the SEC requests from BMan. The SEC updates the frame description (length change due to the addition of the headers, trailers, and HMAC) and enqueues the FQ back to QMan using a configured FQ ID. In this case, the FQ ID causes the QMan to enqueue the FQ to Work Queue #5 of logical CPU#4's dedicated channel. CPU#4 dequeues the FQ, determines the outbound interface for the newly encrypted packet, updates the tunnel IP header, and enqueues the frame back to QMan on a new FQ ID. This FQ ID causes the QMan to enqueue the FQ to a channel serviced by FMan #2, which dequeues the FQ, transmits one or more packets from that FQ out the appropriate dTSEC, and releases the buffers back to BMan.

The processing pipeline used in this example is not required by the QorIQ DPAA. The initial classification could have caused the packet to be steered toward a CPU dedicated to fine-grained classification, or to a pool channel of CPUs, any of which could have performed the operations described. CPU#3 could have added the ESP header and trailer to the packet and sent it to the SEC for crypto-only processing. Following SEC

processing, the flow was steered to CPU#4, however it could have just as easily been steered back to CPU#3, or to a pool channel. At any FQ ID transition, the relative priority of the flow could have been elevated or reduced by enqueueing it to a different work queue.

21.5 P4080-Specific DPAA Implementation Details

The *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* describes the superset of DPAA functionality. The P4080 implements a unique subset of this functionality, as described in the following sections.

21.5.1 P4080 Queue Manager (QMan) Implementation

The QMan of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- QMan block base address: 0x31_8000
- 2-Kbyte frame queue (FQ) cache
- 2-Kbyte SFDRs
- 256 congestion groups

21.5.2 P4080 Buffer Manager (BMan) Implementation

The BMan of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- BMan block base address: 0x31_A000
- 64 buffer pools

21.5.3 P4080 Frame Manager (FMan) Implementation

The FMan of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- FMan1 block base address: 0x40_0000
- FMan2 block base address: 0x50_0000

- Four dTSEC Ethernet MACs per Frame Manager
 - Up to eight SGMII interfaces and two parallel interfaces. See signals description and configuration chapter for valid interface combinations and corresponding signalling.
 - Block base addresses are as follows:
 - FM1 dTSEC1: 0x4E_0000
 - FM1 dTSEC2: 0x4E_2000
 - FM1 dTSEC3: 0x4E_4000
 - FM1 dTSEC4: 0x4E_6000
 - FM2 dTSEC1: 0x5E_0000
 - FM2 dTSEC2: 0x5E_2000
 - FM2 dTSEC3: 0x5E_4000
 - FM2 dTSEC4: 0x5E_6000
 - dTSEC PortIDs (RX/TX):
 - dTSEC1: 0x08/0x28
 - dTSEC2: 0x09/0x29
 - dTSEC3: 0x0A/0x2A
 - dTSEC4: 0x0B/0x2B
- One 10G Ethernet MAC (10GEC) per Frame Manager
 - Block base addresses are as follows:
 - FM1 10GEC: 0x4F_0000
 - FM2 10GEC: 0x5F_0000
 - 10GEC PortID (RX/TX):
 - 10GEC: 0x10/0x30
- Ethernet management interface 1 (EMI1)
 - Block base address: 0x4E_1000
 - Interface signals correspond to registers in FMan1 dTSEC1 memory space
- Ethernet management interface 2 (EMI2)
 - Block base address: 0x4F_1000
 - Interface signals correspond to registers in FMan1 10GEC memory space
- Seven Offline/Host Command ports (PortIDs: 0x01..0x07)
- Two FMan Controller complexes
- 160-Kbyte internal FMan memory
- 64-Kbyte FMan Controller memory
- Up to 32 Keygen schemes
- Up to 256 Policer profiles
- Up to 32 entries in FMan DMA command queue
- Up to 128 TNUMs
- Up to 3 FMan debug flows

21.5.4 P4080 Datapath Three Speed Ethernet Controller (dTSEC) Implementation

The datapath three-speed Ethernet controller of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation caveat::

- The dTSEC programming model supports pre-pending the preamble bytes (before SFD) to the received frame in memory. However, P4080 does not support RX preamble extraction. RX preamble extraction is disabled by default (MACCFG2[PreAmRxEN]=0) and should not be enabled.

21.5.5 P4080 Security and Encryption Engine (SEC) Implementation

The Security and Encryption Engine of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- SEC block base address: 0x30_0000

21.5.6 P4080 Pattern Matching Engine (PME) Implementation

The Pattern Matching Engine of the P4080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- PME block base address: 0x31_6000
- 32-Kbyte possible patterns

Chapter 22

Multicore Programmable Interrupt Controller (MPIC)

22.1 MPIC overview

This chapter describes how the multicore programmable interrupt controller (MPIC) manages the various types of interrupt sources managed by or originating from within the MPIC. It describes how to configure the MPIC to manage arbitration of those interrupt sources to ensure that the highest-priority interrupts are handled as quickly as possible.

The MPIC conforms to the OpenPIC architecture, but with many enhancements. The interrupt controller provides multiprocessor interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them in the context of interrupts that are generated from within the MPIC (such as messaging, timer, and interprocessor interrupts), and delivering them to the appropriate destination for servicing.

22.1.1 MPIC features summary

The MPIC supports the following interrupt sources:



- External-Off-chip signals, IRQ[0:1]; see [Table 22-3](#)
- Internal-Up to 112 internal interrupt sources. It is implementation specific which of these sources are used and how they are assigned. These internal interrupt sources originate from peripheral logic within the integrated device; they signal error conditions that need to be addressed by software.
- Shared message signaled registers-Defined within the MPIC, used for cross-program communication, triggered on a register write, and cleared on a register read.
 - Four interprocessor interrupt channels.
 - Two groups of four 32-bit message interrupt channels.

- Three blocks of eight shared message signaled interrupt sources with up to 32 sharers per shared interrupt register intended to support up to three PCI Express host ports on a device.
- Interrupts generated from within the MPIC itself, which are as follows:
 - Two groups (A and B) of four global 32-bit timers clocked with the MPIC input clock or the RTC input. Timers within each group can be concatenated to time longer durations.
 - Interprocessor interrupts (IPI)-Intended for communication between different processor cores on the same device. (Can be used for self-interrupt.)
 - Message registers-From within the MPIC. Triggered on register write, cleared on read. Used for interprocessor communication.

The MPIC has the following features:

- Three different modes of operation:
 - Supports the external proxy facility of the e500mc core
 - Legacy *intn* interrupt delivery to the processor core(s)
 - Pass-through mode (MPIC disabled) in which the MPIC directs interrupts off-chip for external servicing.
- The following types of programmable interrupt outputs:
 - The *intn* outputs. Any of the MPIC interrupt sources can be programmed to direct interrupt requests to *intn*. Handling of such interrupt requests follows the OpenPIC specification, which guarantees that the highest priority interrupts can supersede lower priority ones. [Interrupts routed to int](#) describes how the MPIC logic handles these interrupts.
 - The *cintn* outputs. These are connected to the critical interrupt pin on the processor core(s).
 - The *mcpn* outputs. These are connected to the machine check pin on the processor core(s).
 - Interrupt output signal, IRQ_OUT_B. [Interrupts Routed to cint, mcp, sie, or IRQ_OUT_B](#) describes how the MPIC logic supports these interrupts.
 - SoC Interrupt Event outputs (*sien*). These are used to signal events to the SoC.
- Programming model compatible with the OpenPIC architecture.
 - Message, interprocessor and global timer interrupts. (Note that the interprocessor and global timer interrupts can only be routed to *intn*.)
 - The following OpenPIC-defined features support only interrupts routed to the *int* signals:
 - Fully-nested interrupt delivery, guaranteeing that the interrupt source with the highest priority is given precedence over lower priority interrupts, including any that are in service.
 - 16 programmable interrupt priority levels
 - Support for identifying and handling spurious interrupts

- Support for up to 8 processors.
 - Interrupts can be routed to any processor core, depending on the configuration
 - Multicast delivery mode for interprocessor and global timer interrupts allowing these interrupts to be routed to any or all of the cores
 - Multicast delivery mode for edge-triggered external interrupts
- Processor core initialization control
- Processor core reset control
- Processor non-maskable interrupt control
- Programmable resetting of the MPIC unit through the global configuration register
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller. In 8259 mode, an interrupt causes assertion of a local (that is, internal to the integrated device) interrupt output signal (IRQ_OUT_B).

22.1.2 MPIC block diagram

This figure shows a block diagram of the MPIC.

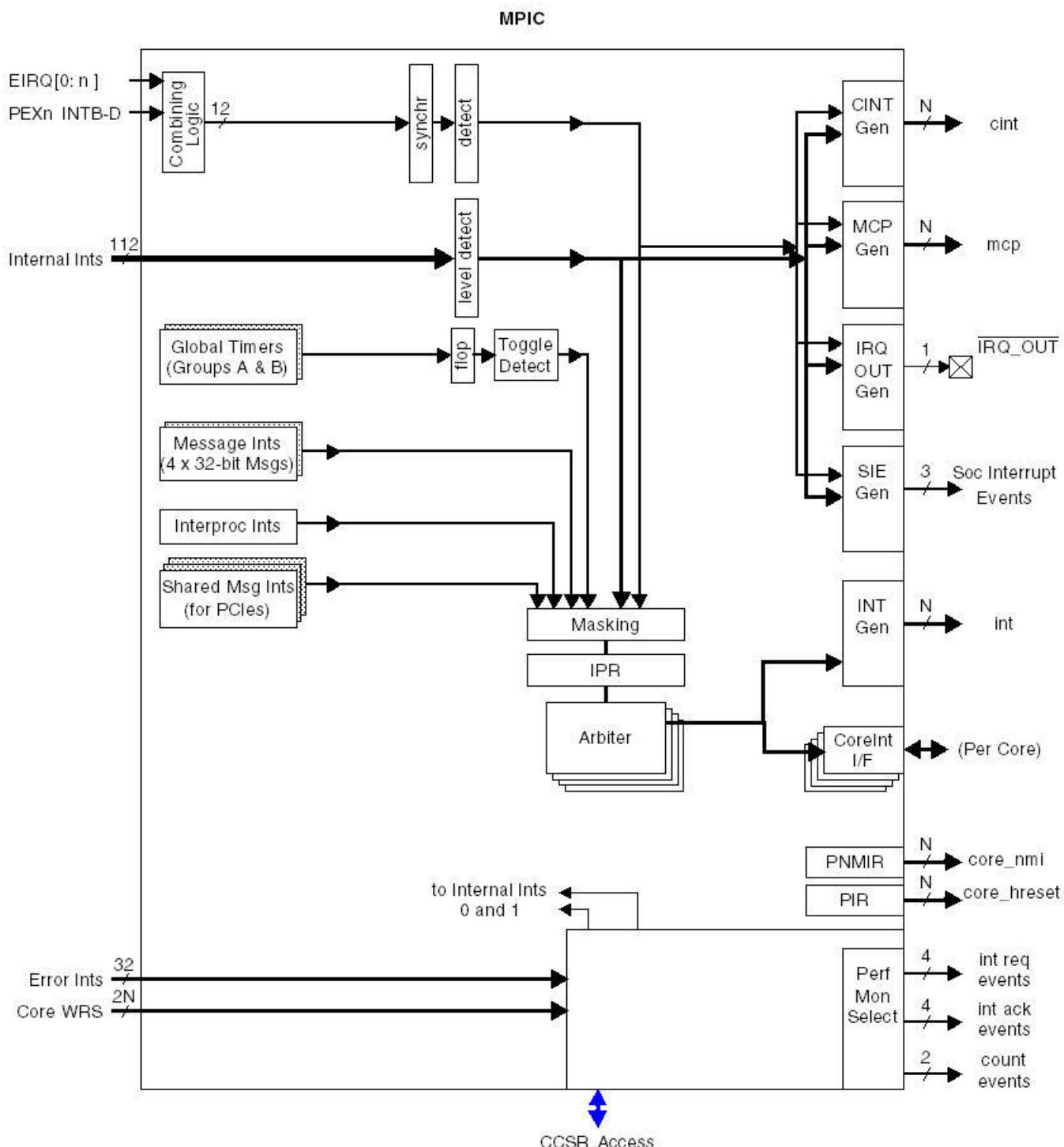


Figure 22-1. MPIC block diagram

22.1.3 The MPIC in Multiple- and Single-Core Implementations

In a multiprocessor system, it is possible for the MPIC to direct interrupts to another processor. This functionality is supported by certain multiprocessor aspects of the programming model.

Note that while they are part of the programming model, such resources are reserved in a single-processor device.

22.1.4 MPIC modes of operation

The MPIC operates in one of the following modes:

- External Proxy Facility Mode (GCR[M] = 11)
- Mixed Mode (GCR[M] = 01)
- Pass-Through Mode (GCR[M] = 00)

22.1.4.1 External Proxy Facility Mode (GCR[M] = 11)

In external proxy facility mode, the vector is directly forwarded to the processor without the need for a software fetch. This feature is supported to all the interrupts. Detailed description of the external proxy facility is described in the core reference manual .

22.1.4.2 Mixed Mode (GCR[M] = 01)

In mixed mode, external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in [Flow of interrupt control](#).

22.1.4.3 Pass-Through Mode (GCR[M] = 00)

The MPIC provides a mechanism to support alternate external interrupt controllers such as the PC/AT-compatible 8259 interrupt controller architecture. After a hard reset, the MPIC defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to core 0 as shown in [Figure 22-2](#); all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor core 0. The MPIC does not perform a vector fetch from an 8259 interrupt controller.

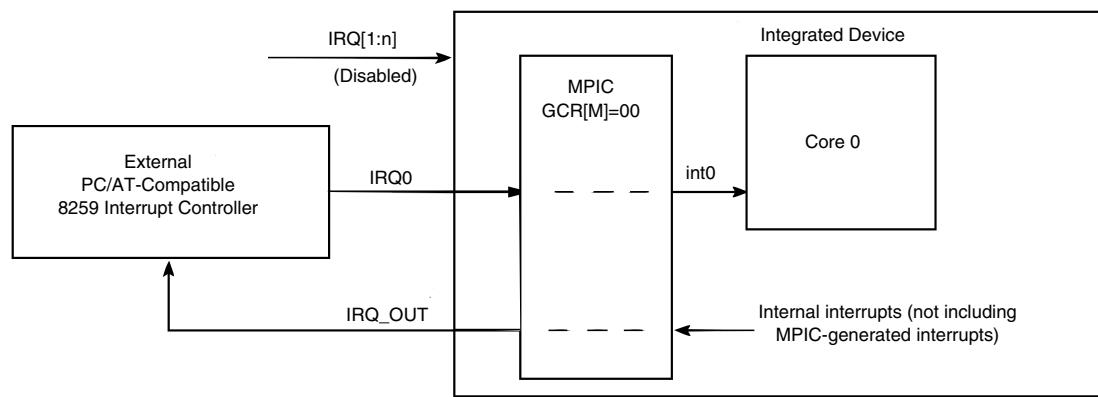


Figure 22-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts shown in [Internal Interrupt Sources](#) are not forwarded to core 0. Instead, the MPIC passes the raw interrupts from the internal sources to the interrupt output signal, [IRQ_OUT_B](#).

NOTE

In pass-through mode, interrupts generated within the MPIC (global timers, interprocessor, and message register interrupts) are disabled. If internal or MPIC-generated interrupts must be reported internally to the processor, mixed mode must be used. In pass-through mode, the target for each interrupt must be set to *int* ($xILRn[INTTGT] = 0x00$). This is the default setting of each register; therefore, no programming should be necessary to comply with this requirement.

22.1.5 Interrupts to the Processor Core

The *intn* signals, which signal an interrupt to the respective cores, are the main interrupt outputs from the MPIC unit to the processor cores. The on-chip power management module monitors the *intn* signals and wakes the corresponding core when one of these asserts and the core is in a low-power state (see [MPIC block diagram](#)).

The interrupt sources can also specify the critical interrupt (*cintn*), machine check (*mcpn*), or the external interrupt ([IRQ_OUT_B](#)) outputs if the corresponding destination bits are set. For internal and external interrupts, the interrupt destination is decoded based on the Interrupt Level Registers (see [External interrupt n \(IRQn\) level register \(MPIC_EILR\)](#) and [Internal interrupt n level register \(MPIC_IILRn\)](#)) and $xIDRn[Pn]$. These interrupt sources must be level sensitive.

The MPIC also defines the PIR, described in [Processor initialization register \(MPIC_PIR\)](#), which can be used to assert *hresetn* and trigger a hard reset. It also defines the PNMIR, described in [Processor NMI register \(MPIC_PNMIR\)](#), which can be used to assert *coren_nmi* and to trigger a non-maskable interrupt. Processor core interrupts generated by the MPIC are described in [Table 22-1](#).

Table 22-1. Processor Core Interrupts Generated by the MPIC-Types and Sources

Source	Internal, External, Message, Shared MSI, Global Timer, Interprocessor	Internal, External	Internal, External	Internal, External	PIR	PNMIR
MPIC output	<i>intn</i>	<i>cintn</i>	<i>mcpn</i>	IRQ_OUT_B, <i>sien</i>	<i>core0_hreset</i> through <i>core7_hreset</i>	<i>core0_nmi</i> through <i>core7_nmi</i>
Description	MPIC signals an interrupt in corresponding core.	Causes critical interrupt condition in corresponding core.	Causes machine check condition in corresponding core.	IRQ_OUT_B, <i>sien</i> assertion	One of the following: <ul style="list-style-type: none"> External HRESET_B assertion (and negation) <i>hresetn</i> output from MPIC due to PIR update. 	<i>core0_nmi</i> through <i>core7_nmi</i> outputs due to PNMIR update.

22.1.6 Interrupt Sources

The MPIC can receive separate interrupts from the following sources:



- External-Off-chip signals, IRQ[0:1]
- Internal-On-chip sources from peripheral logic within the integrated device. See [Interrupt Assignments](#).
- Shared message signaled registers-Supports PCI Express Message Signalled Interrupts. Triggered on register write, cleared on read. Used for cross-program communication.
- Sources inside the MPIC:
 - Global timers A and B internal to the MPIC
 - Interprocessor interrupts (IPI)-Intended for communication between different processor cores on the same device. (Can be used for self-interrupt.)
 - Message registers-From within the MPIC. Triggered on register write, cleared on read. Used for interprocessor communication.

- Internal Error-Internal errors from datapath steered to processor interrupt, critical, or machine check of any core. It has a shared vector and destination (Internal Interrupt 0) supporting one bank for 32 error types.
- Watchdog timers-Watchdog timer events that is steered to processor interrupt, critical, or machine check of any core. It has a shared vector and destination (Internal Interrupt 1).

22.1.6.1 Interrupt Destinations

When changed from Pass-Through mode to any of the other modes (external proxy or mixed mode), the MPIC directs all global timer, shared message signaled, and interrupts from external and internal sources to the *int0* output (which is connected to the *int* signal of processor core 0). An interrupt can be routed to another core by explicitly setting the destination register for that source.

The interprocessor and global timers interrupts can be programmed to be routed to any core's *int* signal (single destination) or to the *int* signals of multiple cores (multicasting).

Internal and external interrupts have more destination options, but only one destination can be chosen for a single non-multicasting interrupt. Instead of being routed to the *int* input of one of the processor cores, these interrupts can be alternatively routed to *IRQ_OUT_B*, *sie*, *mcp*, or *cint* with respect to the MPIC and routed to any core. In addition, some edge-triggered external interrupts (*IRQ[0:3]*) can be routed to the *int* signals of multiple cores (multicast); however, multicasting level-sensitive external interrupts is not supported. These options are selected by writing to the *xILRn[INTTGT]* field in the appropriate interrupt level register and *xIDRn[Pn]* in the appropriate destination register.

Message and Shared MSI interrupts can only be routed to one of *intn* by writing to *xIDRn[Pn]* fields in the appropriate destination register.

[Table 22-2](#) summarizes the valid interrupt destinations for the various interrupt sources handled by the MPIC. It also indicates whether multicasting (MC) is supported or if only single destination (SD) is allowed for the interrupt. Single destination is used in the *IRQ_OUT_B* interrupt output column because there is only one *IRQ_OUT_B* signal—multicasting would not make sense for this selection.

Table 22-2. Destinations for MPIC Interrupt, NMI, and Reset Sources

Category	Interrupt Source	MPIC Interrupt Outputs ¹ (Only one output type per source)					NMI & Core Reset		Debug ²	Comments
		<i>intn</i>	<i>cintn</i>	<i>mcpn</i>	<i>IRQ_OUT_B</i>	<i>sien</i>	<i>coren_nmi</i>	<i>coren_hreset</i>		
Standard Interrupt Sources										
External Sources	EIRQ[0:m]	MC	MC ¹	MC ³	SD ¹	SD ¹	-	-	SD	-
SoC Internal Sources ⁴	IRQ[0:n]	SD	MC	MC	SD	SD	-	-	SD	-
Shared Message Signalled Interrupts	Initiated by Inbound PCIe write transactions	SD	-	-	-	-	-	-	SD	This set has no level registers.
MPIC Sources	Global Timers	MC	-	-	-	-	-	-	SD	This set has no level registers.
	Interprocessor Interrupts (IPI)	MC	-	-	-	-	-	-	SD	This set has no level registers.
	Message Interrupts	SD	-	-	-	-	-	-	SD	This set has no level registers.
	Shared Message Signalled Interrupts	SD	-	-	-	-	-	-	SD	This set has no level registers.
Register-Driven NMI and Core Resets										
MPIC Register Driven	PNMIR	-				MC	-	-	-	-
	PIR	-				-	MC	-	-	-

1. Interrupt outputs are mutually exclusive and selected as follows: *intn*: *xILR[INTTGT]* = 0x00 *cintn*: *xILR[INTTGT]* = 0x01 *mcpn*: *xILR[INTTGT]* = 0x02 *sien*: *xILR[INTTGT]* = 0xF0 through 0xF2 *IRQ_OUT_B*: *xILR[INTTGT]* = 0xFF
2. Perfmon events can operate in parallel with selected MPIC interrupt outputs.
3. EIRQ source must be programmed as level-sensitive in *EIVPRn[S]* when used for this interrupt output. Note that all other sources are inherently level-sensitive.
4. Includes the following: ORed SoC error interrupt as IRQ[0] and ORed watchdog timer (WDT) interrupt as IRQ[1].

For each interrupt source, only one MPIC interrupt output type can be used as programmed in its corresponding *xILRn[INTTGT]*. For example, it is not possible to generate both an *intn* and *IRQ_OUT_B* in response to EIRQ[3] but it is possible to generate a *cint0* and *cint3* to cores 0 and 1 in response to EIRQ[3] since these share the same MPIC interrupt output type. Performance monitor events can be programmed to operate in parallel with a selected MPIC interrupt output.

22.1.6.2 Interrupt Routing-Mixed Mode

When an interrupt request is delivered to the MPIC, the corresponding interrupt destination register is checked to determine where the request should be routed, as follows:

- If Interrupt Level Register $xILRn[INTTGT] = 0xFF$, the interrupt is routed off-chip to the interrupt output signal, `IRQ_OUT_B`.
- If Interrupt Level Register $xILRn[INTTGT] = 0xF0$ to $0xF2$, the interrupt is routed to the corresponding SoC interrupt event output signals, *sien*. See [Interrupt Assignments](#) for *sien* routing assignments.
- If Interrupt Level Register $xILRn[INTTGT] = 0x01$, and one, but not multiple, $xIDRn[Pn]$ is set, the interrupt is routed to the appropriate *cintn*.
- If Interrupt Level Register $xILRn[INTTGT] = 0x02$, and one, but not multiple, $xIDRn[Pn]$ is set, the interrupt is routed to the appropriate *mcpn*.
- If Interrupt Level Register $xILRn[INTTGT] = 0x00$, and one, but not multiple, $xIDRn[Pn]$ is set, the interrupt is routed to the appropriate *intrn signal*. In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow is as described in [Flow of interrupt control](#). Note that multicasting interrupts (global timer, interprocessor, and some edge-triggered external interrupts) can be set to any or all of P0 through P7; other interrupt sources cannot.

22.2 MPIC external signal descriptions

The following sections describe the MPIC signals.

22.2.1 MPIC signal descriptions

 The MPIC device external interface signals are described in this table. There are [distinct](#) external interrupt request input signals and 1 interrupt request output pin `IRQ_OUT_B`.

This table provides detailed descriptions of the external MPIC signals.

Table 22-3. External interrupt signals-detailed signal descriptions

Signal	I / O	Description
<code>IRQ[0:1]</code>	I	<p>External Interrupt request 0-. The polarity and sense of each of these signals are programmable. All of these inputs can be driven asynchronously.</p> <p>NOTE: See PCI Express INTx/External Interrupt IRQn Sharingfor information on how external interrupt request signals and PCI virtual INTx signals are shared.</p>

Table continues on the next page...

Table 22-3. External interrupt signals-detailed signal descriptions (continued)

Signal	I / O	Description
	State Meaning	Asserted-When an external interrupt signal is asserted (according to the programmed polarity), the MPIC checks its priority and the interrupt is conditionally passed to the processor designated in the corresponding destination register. In pass-through mode, only interrupts detected on IRQ0 are passed directly to core 0. Negated-There is no incoming interrupt from that source.
	Timing	Assertion-All of these inputs can be asserted asynchronously. Negation-Interrupts programmed as level-sensitive must remain asserted until serviced. Timing requirements for edge-sensitive interrupts can be found in the .
IRQ_OUT	O	Interrupt request out. Active-high out of the IP block. When the MPIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See MPIC modes of operation .
	State Meaning	Asserted-At least one interrupt is currently being signalled to the external system. Negated-Indicates no interrupt source currently routed to IRQ_OUT.
	Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of IRQ_OUT occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion-Internal interrupt source: 2 clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Negation-Follows interrupt source negation with the following delay: Internal interrupt: 2 clock cycles External interrupt: 4 cycles.

22.3 MPIC Memory Map

MPIC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_0000	Block revision register 1 (MPIC_BRR1)	32	R	0040_0000h	22.3.1/1689
4_0010	Block revision register 2 (MPIC_BRR2)	32	R	0000_0000h	22.3.2/1689
4_0040	Interprocessor interrupt dispatch register n (MPIC_IPIDR0)	32	W	0000_0000h	22.3.3/1690
4_0050	Interprocessor interrupt dispatch register n (MPIC_IPIDR1)	32	W	0000_0000h	22.3.3/1690
4_0060	Interprocessor interrupt dispatch register n (MPIC_IPIDR2)	32	W	0000_0000h	22.3.3/1690
4_0070	Interprocessor interrupt dispatch register n (MPIC_IPIDR3)	32	W	0000_0000h	22.3.3/1690
4_0080	Current task priority register (MPIC_CTPR)	32	R/W	0000_000Fh	22.3.4/1691
4_0090	Who am I register (MPIC_WHOAMI)	32	R	See section	22.3.5/1692
4_00A0	Interrupt acknowledge register (MPIC_IACK)	32	R	0000_0000h	22.3.6/1693
4_00B0	End of interrupt register (MPIC_EOI)	32	W	0000_0000h	22.3.7/1694
4_1000	Feature reporting register (MPIC_FRR)	32	R	See section	22.3.8/1695
4_1020	Global configuration register (MPIC_GCR)	32	R/W	0000_0000h	22.3.9/1696

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_1080	Vendor identification register (MPIC_VIR)	32	R	0000_0000h	22.3.10/ 1696
4_1090	Processor initialization register (MPIC_PIR)	32	R/W	0000_0000h	22.3.11/ 1697
4_1098	Processor NMI register (MPIC_PNMIR)	32	R/W	0000_0000h	22.3.12/ 1698
4_10A0	IPI vector/priority register n (MPIC_IPIVPR0)	32	R/W	8000_0000h	22.3.13/ 1699
4_10B0	IPI vector/priority register n (MPIC_IPIVPR1)	32	R/W	8000_0000h	22.3.13/ 1699
4_10C0	IPI vector/priority register n (MPIC_IPIVPR2)	32	R/W	8000_0000h	22.3.13/ 1699
4_10D0	IPI vector/priority register n (MPIC_IPIVPR3)	32	R/W	8000_0000h	22.3.13/ 1699
4_10E0	Spurious vector register (MPIC_SVR)	32	R/W	0000_FFFFh	22.3.14/ 1700
4_10F0	Timer frequency reporting register [Group A] (MPIC_TFRRA)	32	R/W	0000_0000h	22.3.15/ 1701
4_1100	Global timer current count register [Group A] n (MPIC_GTCCRA0)	32	R	0000_0000h	22.3.16/ 1701
4_1110	Global timer base count register [Group A] n (MPIC_GTB CRA0)	32	R/W	8000_0000h	22.3.17/ 1702
4_1120	Global timer vector/priority register [Group A] n (MPIC_GTVPRA0)	32	R/W	8000_0000h	22.3.18/ 1703
4_1130	Global timer destination register [Group A] n (MPIC_GTDRA0)	32	R/W	0000_0001h	22.3.19/ 1704
4_1140	Global timer current count register [Group A] n (MPIC_GTCCRA1)	32	R	0000_0000h	22.3.16/ 1701
4_1150	Global timer base count register [Group A] n (MPIC_GTB CRA1)	32	R/W	8000_0000h	22.3.17/ 1702
4_1160	Global timer vector/priority register [Group A] n (MPIC_GTVPRA1)	32	R/W	8000_0000h	22.3.18/ 1703
4_1170	Global timer destination register [Group A] n (MPIC_GTDRA1)	32	R/W	0000_0001h	22.3.19/ 1704
4_1180	Global timer current count register [Group A] n (MPIC_GTCCRA2)	32	R	0000_0000h	22.3.16/ 1701
4_1190	Global timer base count register [Group A] n (MPIC_GTB CRA2)	32	R/W	8000_0000h	22.3.17/ 1702
4_11A0	Global timer vector/priority register [Group A] n (MPIC_GTVPRA2)	32	R/W	8000_0000h	22.3.18/ 1703
4_11B0	Global timer destination register [Group A] n (MPIC_GTDRA2)	32	R/W	0000_0001h	22.3.19/ 1704
4_11C0	Global timer current count register [Group A] n (MPIC_GTCCRA3)	32	R	0000_0000h	22.3.16/ 1701

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_11D0	Global timer base count register [Group A] n (MPIC_GTBCRA3)	32	R/W	8000_0000h	22.3.17/ 1702
4_11E0	Global timer vector/priority register [Group A] n (MPIC_GTPRA3)	32	R/W	8000_0000h	22.3.18/ 1703
4_11F0	Global timer destination register [Group A] n (MPIC_GTDRA3)	32	R/W	0000_0001h	22.3.19/ 1704
4_1300	Timer control register [Group A] (MPIC_TCRA)	32	R/W	0000_0000h	22.3.20/ 1705
4_1400	Message register [Group A] n (MPIC_MSGRA0)	32	R/W	0000_0000h	22.3.21/ 1707
4_1410	Message register [Group A] n (MPIC_MSGRA1)	32	R/W	0000_0000h	22.3.21/ 1707
4_1420	Message register [Group A] n (MPIC_MSGRA2)	32	R/W	0000_0000h	22.3.21/ 1707
4_1430	Message register [Group A] n (MPIC_MSGRA3)	32	R/W	0000_0000h	22.3.21/ 1707
4_1500	Message enable register [Group A] (MPIC_MERA)	32	R/W	0000_0000h	22.3.22/ 1707
4_1510	Message status register [Group A] (MPIC_MSRA)	32	R/W	0000_0000h	22.3.23/ 1708
4_1600	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA0)	32	R	0000_0000h	22.3.24/ 1709
4_1610	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA1)	32	R	0000_0000h	22.3.24/ 1709
4_1620	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA2)	32	R	0000_0000h	22.3.24/ 1709
4_1630	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA3)	32	R	0000_0000h	22.3.24/ 1709
4_1640	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA4)	32	R	0000_0000h	22.3.24/ 1709
4_1650	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA5)	32	R	0000_0000h	22.3.24/ 1709
4_1660	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA6)	32	R	0000_0000h	22.3.24/ 1709
4_1670	Shared message signaled interrupt register [Bank A] n (MPIC_MSIRA7)	32	R	0000_0000h	22.3.24/ 1709
4_1720	Shared message signaled interrupt status register [Bank A] (MPIC_MSISRA)	32	R	0000_0000h	22.3.25/ 1711
4_1740	Shared message signaled interrupt index register [Bank A] (MPIC_MSIIRA)	32	W	0000_0000h	22.3.26/ 1711
4_1800	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB0)	32	R	0000_0000h	22.3.27/ 1712
4_1810	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB1)	32	R	0000_0000h	22.3.27/ 1712

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_1820	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB2)	32	R	0000_0000h	22.3.27/ 1712
4_1830	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB3)	32	R	0000_0000h	22.3.27/ 1712
4_1840	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB4)	32	R	0000_0000h	22.3.27/ 1712
4_1850	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB5)	32	R	0000_0000h	22.3.27/ 1712
4_1860	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB6)	32	R	0000_0000h	22.3.27/ 1712
4_1870	Shared message signaled interrupt register [Bank B] n (MPIC_MSIRB7)	32	R	0000_0000h	22.3.27/ 1712
4_1920	Shared message signaled interrupt status register [Bank B] (MPIC_MSISRB)	32	R	0000_0000h	22.3.28/ 1714
4_1940	Shared message signaled interrupt index register [Bank B] (MPIC_MSIIRB)	32	W	0000_0000h	22.3.29/ 1715
4_1A00	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC0)	32	R	0000_0000h	22.3.30/ 1716
4_1A10	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC1)	32	R	0000_0000h	22.3.30/ 1716
4_1A20	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC2)	32	R	0000_0000h	22.3.30/ 1716
4_1A30	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC3)	32	R	0000_0000h	22.3.30/ 1716
4_1A40	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC4)	32	R	0000_0000h	22.3.30/ 1716
4_1A50	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC5)	32	R	0000_0000h	22.3.30/ 1716
4_1A60	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC6)	32	R	0000_0000h	22.3.30/ 1716
4_1A70	Shared message signaled interrupt register [Bank C] n (MPIC_MSIRC7)	32	R	0000_0000h	22.3.30/ 1716
4_1B20	Shared message signaled interrupt status register [Bank C] (MPIC_MSISRC)	32	R	0000_0000h	22.3.31/ 1718
4_1B40	Shared message signaled interrupt index register [Bank C] (MPIC_MSIIRC)	32	W	0000_0000h	22.3.32/ 1718
4_20F0	Timer frequency reporting register [Group B] (MPIC_TFRRB)	32	R/W	0000_0000h	22.3.33/ 1719
4_2100	Global timer current count register [Group B] n (MPIC_GTCCRB0)	32	R	0000_0000h	22.3.34/ 1720
4_2110	Global timer base count register [Group B] n (MPIC_GTBCRB0)	32	R/W	0000_0000h	22.3.35/ 1721
4_2120	Global timer vector/priority register [Group B] n (MPIC_GTVPRB0)	32	R/W	8000_0000h	22.3.36/ 1722

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_2130	Global timer destination register [Group B] n (MPIC_GTDRB0)	32	R/W	0000_0001h	22.3.37/ 1723
4_2140	Global timer current count register [Group B] n (MPIC_GTCCRB1)	32	R	0000_0000h	22.3.34/ 1720
4_2150	Global timer base count register [Group B] n (MPIC_GTBCRB1)	32	R/W	0000_0000h	22.3.35/ 1721
4_2160	Global timer vector/priority register [Group B] n (MPIC_GTVPRB1)	32	R/W	8000_0000h	22.3.36/ 1722
4_2170	Global timer destination register [Group B] n (MPIC_GTDRB1)	32	R/W	0000_0001h	22.3.37/ 1723
4_2180	Global timer current count register [Group B] n (MPIC_GTCCRB2)	32	R	0000_0000h	22.3.34/ 1720
4_2190	Global timer base count register [Group B] n (MPIC_GTBCRB2)	32	R/W	0000_0000h	22.3.35/ 1721
4_21A0	Global timer vector/priority register [Group B] n (MPIC_GTVPRB2)	32	R/W	8000_0000h	22.3.36/ 1722
4_21B0	Global timer destination register [Group B] n (MPIC_GTDRB2)	32	R/W	0000_0001h	22.3.37/ 1723
4_21C0	Global timer current count register [Group B] n (MPIC_GTCCRB3)	32	R	0000_0000h	22.3.34/ 1720
4_21D0	Global timer base count register [Group B] n (MPIC_GTBCRB3)	32	R/W	0000_0000h	22.3.35/ 1721
4_21E0	Global timer vector/priority register [Group B] n (MPIC_GTVPRB3)	32	R/W	8000_0000h	22.3.36/ 1722
4_21F0	Global timer destination register [Group B] n (MPIC_GTDRB3)	32	R/W	0000_0001h	22.3.37/ 1723
4_2300	Timer control register [Group B] (MPIC_TCRB)	32	R/W	0000_0000h	22.3.38/ 1724
4_2400	Message register [Group B] n (MPIC_MSGRB0)	32	R/W	0000_0000h	22.3.39/ 1726
4_2410	Message register [Group B] n (MPIC_MSGRB1)	32	R/W	0000_0000h	22.3.39/ 1726
4_2420	Message register [Group B] n (MPIC_MSGRB2)	32	R/W	0000_0000h	22.3.39/ 1726
4_2430	Message register [Group B] n (MPIC_MSGRB3)	32	R/W	0000_0000h	22.3.39/ 1726
4_2500	Message enable register [Group B] (MPIC_MERB)	32	R/W	0000_0000h	22.3.40/ 1726
4_2510	Message status register [Group B] (MPIC_MSRB)	32	R/W	0000_0000h	22.3.41/ 1727
4_3000	Performance monitor n mask register 0 (MPIC_PM0MR0)	32	R/W	00FF_FFFFh	22.3.42/ 1727
4_3020	Performance monitor n mask register 1 (MPIC_PM0MR1)	32	R/W	00FF_FFFFh	22.3.43/ 1728

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4_3040	Performance monitor n mask register 2 (MPIC_PM0MR2)	32	R/W	FFFF_FFFFh	22.3.44/1729
4_3050	Performance monitor n mask register 3 (MPIC_PM0MR3)	32	R/W	FFFF_FFFFh	22.3.45/1729
4_3060	Performance monitor n mask register 4 (MPIC_PM0MR4)	32	R/W	FFFF_FFFFh	22.3.46/1729
4_3070	Performance monitor n mask register 5 (MPIC_PM0MR5)	32	R/W	FFFF_FFFFh	22.3.47/1730
4_3100	Performance monitor n mask register 0 (MPIC_PM1MR0)	32	R/W	00FF_FFFFh	22.3.42/1727
4_3120	Performance monitor n mask register 1 (MPIC_PM1MR1)	32	R/W	00FF_FFFFh	22.3.43/1728
4_3140	Performance monitor n mask register 2 (MPIC_PM1MR2)	32	R/W	FFFF_FFFFh	22.3.44/1729
4_3150	Performance monitor n mask register 3 (MPIC_PM1MR3)	32	R/W	FFFF_FFFFh	22.3.45/1729
4_3160	Performance monitor n mask register 4 (MPIC_PM1MR4)	32	R/W	FFFF_FFFFh	22.3.46/1729
4_3170	Performance monitor n mask register 5 (MPIC_PM1MR5)	32	R/W	FFFF_FFFFh	22.3.47/1730
4_3200	Performance monitor n mask register 0 (MPIC_PM2MR0)	32	R/W	00FF_FFFFh	22.3.42/1727
4_3220	Performance monitor n mask register 1 (MPIC_PM2MR1)	32	R/W	00FF_FFFFh	22.3.43/1728
4_3240	Performance monitor n mask register 2 (MPIC_PM2MR2)	32	R/W	FFFF_FFFFh	22.3.44/1729
4_3250	Performance monitor n mask register 3 (MPIC_PM2MR3)	32	R/W	FFFF_FFFFh	22.3.45/1729
4_3260	Performance monitor n mask register 4 (MPIC_PM2MR4)	32	R/W	FFFF_FFFFh	22.3.46/1729
4_3270	Performance monitor n mask register 5 (MPIC_PM2MR5)	32	R/W	FFFF_FFFFh	22.3.47/1730
4_3300	Performance monitor n mask register 0 (MPIC_PM3MR0)	32	R/W	00FF_FFFFh	22.3.42/1727
4_3320	Performance monitor n mask register 1 (MPIC_PM3MR1)	32	R/W	00FF_FFFFh	22.3.43/1728
4_3340	Performance monitor n mask register 2 (MPIC_PM3MR2)	32	R/W	FFFF_FFFFh	22.3.44/1729
4_3350	Performance monitor n mask register 3 (MPIC_PM3MR3)	32	R/W	FFFF_FFFFh	22.3.45/1729
4_3360	Performance monitor n mask register 4 (MPIC_PM3MR4)	32	R/W	FFFF_FFFFh	22.3.46/1729
4_3370	Performance monitor n mask register 5 (MPIC_PM3MR5)	32	R/W	FFFF_FFFFh	22.3.47/1730

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4_3800	External interrupt summary register (MPIC_ERQSR)	32	R	0000_0000h	22.3.48/ 1730
4_3900	Error interrupt summary register 0 (MPIC_EISR0)	32	R	0000_0000h	22.3.49/ 1731
4_3910	Error interrupt mask register 0 (MPIC_EIMR0)	32	R/W	0000_0000h	22.3.50/ 1731
4_3A00	Watchdog status register summary register 0 (MPIC_WSRSR0)	32	w1c	0000_0000h	22.3.51/ 1732
4_3B00	Critical interrupt summary register 0 (MPIC_CISR0)	32	R	0000_0000h	22.3.52/ 1733
4_3B40	Critical interrupt summary register 1 (MPIC_CISR1)	32	R	0000_0000h	22.3.53/ 1733
4_3B50	Critical interrupt summary register 2 (MPIC_CISR2)	32	R	0000_0000h	22.3.54/ 1734
4_3B60	Critical interrupt summary register 3 (MPIC_CISR3)	32	R	0000_0000h	22.3.55/ 1734
4_3B70	Critical interrupt summary register 4 (MPIC_CISR4)	32	R	0000_0000h	22.3.56/ 1734
4_3C00	Machine check summary register 0 (MPIC_MCSR0)	32	R	0000_0000h	22.3.57/ 1735
4_3C40	Machine check summary register 1 (MPIC_MCSR1)	32	R	0000_0000h	22.3.58/ 1735
4_3C50	Machine check summary register 2 (MPIC_MCSR2)	32	R	0000_0000h	22.3.59/ 1736
4_3C60	Machine check summary register 3 (MPIC_MCSR3)	32	R	0000_0000h	22.3.60/ 1736
4_3C70	Machine check summary register 4 (MPIC_MCSR4)	32	R	0000_0000h	22.3.61/ 1737
4_3D00	IRQ_OUT/Soc Interrupt Event summary register 0 (MPIC IRQSIESR0)	32	R	0000_0000h	22.3.62/ 1737
4_3D40	IRQ_OUT/Soc Interrupt Event summary register 1 (MPIC IRQSIESR1)	32	R	0000_0000h	22.3.63/ 1738
4_3D50	IRQ_OUT/Soc Interrupt Event summary register 2 (MPIC IRQSIESR2)	32	R	0000_0000h	22.3.64/ 1738
4_3D60	IRQ_OUT/Soc Interrupt Event summary register 3 (MPIC IRQSIESR3)	32	R	0000_0000h	22.3.65/ 1739
4_3D70	IRQ_OUT/Soc Interrupt Event summary register 4 (MPIC IRQSIESR4)	32	R	0000_0000h	22.3.66/ 1739
4_4140	Shared message signaled interrupt index register [Bank A] (alias) (MPIC_MSIIRA_alias)	32	W	0000_0000h	22.3.67/ 1740
4_5140	Shared message signaled interrupt index register [Bank B] (alias) (MPIC_MSIIRB_alias)	32	W	0000_0000h	22.3.68/ 1740
4_6140	Shared message signaled interrupt index register [Bank C] (alias) (MPIC_MSIIIRC_alias)	32	W	0000_0000h	22.3.69/ 1741

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0000	External interrupt n (IRQn) vector/priority register (MPIC_EIVPR)	32	R/W	8000_0000h	22.3.70/ 0
5_0010	External interrupt n (IRQn) destination register (MPIC_EIDR)	32	R/W	0000_0001h	22.3.71/ 0
5_0018	External interrupt n (IRQn) level register (MPIC_EILP)	32	R/W	0000_0000h	22.3.72/ 0
5_0200	Internal interrupt n vector/priority register (MPIC_IIVPRO)	32	R/W	8080_0000h	22.3.73/ 1742
5_0210	Internal interrupt n destination register (MPIC_IIDR0)	32	R/W	0000_0001h	22.3.74/ 1743
5_0218	Internal interrupt n level register (MPIC_IILR0)	32	R/W	0000_0000h	22.3.75/ 1744
5_0220	Internal interrupt n vector/priority register (MPIC_IIVPR1)	32	R/W	8080_0000h	22.3.73/ 1742
5_0230	Internal interrupt n destination register (MPIC_IIDR1)	32	R/W	0000_0001h	22.3.74/ 1743
5_0238	Internal interrupt n level register (MPIC_IILR1)	32	R/W	0000_0000h	22.3.75/ 1744
5_0240	Internal interrupt n vector/priority register (MPIC_IIVPR2)	32	R/W	8080_0000h	22.3.73/ 1742
5_0250	Internal interrupt n destination register (MPIC_IIDR2)	32	R/W	0000_0001h	22.3.74/ 1743
5_0258	Internal interrupt n level register (MPIC_IILR2)	32	R/W	0000_0000h	22.3.75/ 1744
5_0260	Internal interrupt n vector/priority register (MPIC_IIVPR3)	32	R/W	8080_0000h	22.3.73/ 1742
5_0270	Internal interrupt n destination register (MPIC_IIDR3)	32	R/W	0000_0001h	22.3.74/ 1743
5_0278	Internal interrupt n level register (MPIC_IILR3)	32	R/W	0000_0000h	22.3.75/ 1744
5_0280	Internal interrupt n vector/priority register (MPIC_IIVPR4)	32	R/W	8080_0000h	22.3.73/ 1742
5_0290	Internal interrupt n destination register (MPIC_IIDR4)	32	R/W	0000_0001h	22.3.74/ 1743
5_0298	Internal interrupt n level register (MPIC_IILR4)	32	R/W	0000_0000h	22.3.75/ 1744
5_02A0	Internal interrupt n vector/priority register (MPIC_IIVPR5)	32	R/W	8080_0000h	22.3.73/ 1742
5_02B0	Internal interrupt n destination register (MPIC_IIDR5)	32	R/W	0000_0001h	22.3.74/ 1743
5_02B8	Internal interrupt n level register (MPIC_IILR5)	32	R/W	0000_0000h	22.3.75/ 1744
5_02C0	Internal interrupt n vector/priority register (MPIC_IIVPR6)	32	R/W	8080_0000h	22.3.73/ 1742

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_02D0	Internal interrupt n destination register (MPIC_IIDR6)	32	R/W	0000_0001h	22.3.74/ 1743
5_02D8	Internal interrupt n level register (MPIC_IILR6)	32	R/W	0000_0000h	22.3.75/ 1744
5_02E0	Internal interrupt n vector/priority register (MPIC_IIVPR7)	32	R/W	8080_0000h	22.3.73/ 1742
5_02F0	Internal interrupt n destination register (MPIC_IIDR7)	32	R/W	0000_0001h	22.3.74/ 1743
5_02F8	Internal interrupt n level register (MPIC_IILR7)	32	R/W	0000_0000h	22.3.75/ 1744
5_0300	Internal interrupt n vector/priority register (MPIC_IIVPR8)	32	R/W	8080_0000h	22.3.73/ 1742
5_0310	Internal interrupt n destination register (MPIC_IIDR8)	32	R/W	0000_0001h	22.3.74/ 1743
5_0318	Internal interrupt n level register (MPIC_IILR8)	32	R/W	0000_0000h	22.3.75/ 1744
5_0320	Internal interrupt n vector/priority register (MPIC_IIVPR9)	32	R/W	8080_0000h	22.3.73/ 1742
5_0330	Internal interrupt n destination register (MPIC_IIDR9)	32	R/W	0000_0001h	22.3.74/ 1743
5_0338	Internal interrupt n level register (MPIC_IILR9)	32	R/W	0000_0000h	22.3.75/ 1744
5_0340	Internal interrupt n vector/priority register (MPIC_IIVPR10)	32	R/W	8080_0000h	22.3.73/ 1742
5_0350	Internal interrupt n destination register (MPIC_IIDR10)	32	R/W	0000_0001h	22.3.74/ 1743
5_0358	Internal interrupt n level register (MPIC_IILR10)	32	R/W	0000_0000h	22.3.75/ 1744
5_0360	Internal interrupt n vector/priority register (MPIC_IIVPR11)	32	R/W	8080_0000h	22.3.73/ 1742
5_0370	Internal interrupt n destination register (MPIC_IIDR11)	32	R/W	0000_0001h	22.3.74/ 1743
5_0378	Internal interrupt n level register (MPIC_IILR11)	32	R/W	0000_0000h	22.3.75/ 1744
5_0380	Internal interrupt n vector/priority register (MPIC_IIVPR12)	32	R/W	8080_0000h	22.3.73/ 1742
5_0390	Internal interrupt n destination register (MPIC_IIDR12)	32	R/W	0000_0001h	22.3.74/ 1743
5_0398	Internal interrupt n level register (MPIC_IILR12)	32	R/W	0000_0000h	22.3.75/ 1744
5_03A0	Internal interrupt n vector/priority register (MPIC_IIVPR13)	32	R/W	8080_0000h	22.3.73/ 1742
5_03B0	Internal interrupt n destination register (MPIC_IIDR13)	32	R/W	0000_0001h	22.3.74/ 1743

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_03B8	Internal interrupt n level register (MPIC_IILR13)	32	R/W	0000_0000h	22.3.75/ 1744
5_03C0	Internal interrupt n vector/priority register (MPIC_IIVPR14)	32	R/W	8080_0000h	22.3.73/ 1742
5_03D0	Internal interrupt n destination register (MPIC_IIDR14)	32	R/W	0000_0001h	22.3.74/ 1743
5_03D8	Internal interrupt n level register (MPIC_IILR14)	32	R/W	0000_0000h	22.3.75/ 1744
5_03E0	Internal interrupt n vector/priority register (MPIC_IIVPR15)	32	R/W	8080_0000h	22.3.73/ 1742
5_03F0	Internal interrupt n destination register (MPIC_IIDR15)	32	R/W	0000_0001h	22.3.74/ 1743
5_03F8	Internal interrupt n level register (MPIC_IILR15)	32	R/W	0000_0000h	22.3.75/ 1744
5_0400	Internal interrupt n vector/priority register (MPIC_IIVPR16)	32	R/W	8080_0000h	22.3.73/ 1742
5_0410	Internal interrupt n destination register (MPIC_IIDR16)	32	R/W	0000_0001h	22.3.74/ 1743
5_0418	Internal interrupt n level register (MPIC_IILR16)	32	R/W	0000_0000h	22.3.75/ 1744
5_0420	Internal interrupt n vector/priority register (MPIC_IIVPR17)	32	R/W	8080_0000h	22.3.73/ 1742
5_0430	Internal interrupt n destination register (MPIC_IIDR17)	32	R/W	0000_0001h	22.3.74/ 1743
5_0438	Internal interrupt n level register (MPIC_IILR17)	32	R/W	0000_0000h	22.3.75/ 1744
5_0440	Internal interrupt n vector/priority register (MPIC_IIVPR18)	32	R/W	8080_0000h	22.3.73/ 1742
5_0450	Internal interrupt n destination register (MPIC_IIDR18)	32	R/W	0000_0001h	22.3.74/ 1743
5_0458	Internal interrupt n level register (MPIC_IILR18)	32	R/W	0000_0000h	22.3.75/ 1744
5_0460	Internal interrupt n vector/priority register (MPIC_IIVPR19)	32	R/W	8080_0000h	22.3.73/ 1742
5_0470	Internal interrupt n destination register (MPIC_IIDR19)	32	R/W	0000_0001h	22.3.74/ 1743
5_0478	Internal interrupt n level register (MPIC_IILR19)	32	R/W	0000_0000h	22.3.75/ 1744
5_0480	Internal interrupt n vector/priority register (MPIC_IIVPR20)	32	R/W	8080_0000h	22.3.73/ 1742
5_0490	Internal interrupt n destination register (MPIC_IIDR20)	32	R/W	0000_0001h	22.3.74/ 1743
5_0498	Internal interrupt n level register (MPIC_IILR20)	32	R/W	0000_0000h	22.3.75/ 1744

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_04A0	Internal interrupt n vector/priority register (MPIC_IIVPR21)	32	R/W	8080_0000h	22.3.73/ 1742
5_04B0	Internal interrupt n destination register (MPIC_IIDR21)	32	R/W	0000_0001h	22.3.74/ 1743
5_04B8	Internal interrupt n level register (MPIC_IILR21)	32	R/W	0000_0000h	22.3.75/ 1744
5_04C0	Internal interrupt n vector/priority register (MPIC_IIVPR22)	32	R/W	8080_0000h	22.3.73/ 1742
5_04D0	Internal interrupt n destination register (MPIC_IIDR22)	32	R/W	0000_0001h	22.3.74/ 1743
5_04D8	Internal interrupt n level register (MPIC_IILR22)	32	R/W	0000_0000h	22.3.75/ 1744
5_04E0	Internal interrupt n vector/priority register (MPIC_IIVPR23)	32	R/W	8080_0000h	22.3.73/ 1742
5_04F0	Internal interrupt n destination register (MPIC_IIDR23)	32	R/W	0000_0001h	22.3.74/ 1743
5_04F8	Internal interrupt n level register (MPIC_IILR23)	32	R/W	0000_0000h	22.3.75/ 1744
5_0500	Internal interrupt n vector/priority register (MPIC_IIVPR24)	32	R/W	8080_0000h	22.3.73/ 1742
5_0510	Internal interrupt n destination register (MPIC_IIDR24)	32	R/W	0000_0001h	22.3.74/ 1743
5_0518	Internal interrupt n level register (MPIC_IILR24)	32	R/W	0000_0000h	22.3.75/ 1744
5_0520	Internal interrupt n vector/priority register (MPIC_IIVPR25)	32	R/W	8080_0000h	22.3.73/ 1742
5_0530	Internal interrupt n destination register (MPIC_IIDR25)	32	R/W	0000_0001h	22.3.74/ 1743
5_0538	Internal interrupt n level register (MPIC_IILR25)	32	R/W	0000_0000h	22.3.75/ 1744
5_0540	Internal interrupt n vector/priority register (MPIC_IIVPR26)	32	R/W	8080_0000h	22.3.73/ 1742
5_0550	Internal interrupt n destination register (MPIC_IIDR26)	32	R/W	0000_0001h	22.3.74/ 1743
5_0558	Internal interrupt n level register (MPIC_IILR26)	32	R/W	0000_0000h	22.3.75/ 1744
5_0560	Internal interrupt n vector/priority register (MPIC_IIVPR27)	32	R/W	8080_0000h	22.3.73/ 1742
5_0570	Internal interrupt n destination register (MPIC_IIDR27)	32	R/W	0000_0001h	22.3.74/ 1743
5_0578	Internal interrupt n level register (MPIC_IILR27)	32	R/W	0000_0000h	22.3.75/ 1744
5_0580	Internal interrupt n vector/priority register (MPIC_IIVPR28)	32	R/W	8080_0000h	22.3.73/ 1742

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0590	Internal interrupt n destination register (MPIC_IIDR28)	32	R/W	0000_0001h	22.3.74/ 1743
5_0598	Internal interrupt n level register (MPIC_IILR28)	32	R/W	0000_0000h	22.3.75/ 1744
5_05A0	Internal interrupt n vector/priority register (MPIC_IIVPR29)	32	R/W	8080_0000h	22.3.73/ 1742
5_05B0	Internal interrupt n destination register (MPIC_IIDR29)	32	R/W	0000_0001h	22.3.74/ 1743
5_05B8	Internal interrupt n level register (MPIC_IILR29)	32	R/W	0000_0000h	22.3.75/ 1744
5_05C0	Internal interrupt n vector/priority register (MPIC_IIVPR30)	32	R/W	8080_0000h	22.3.73/ 1742
5_05D0	Internal interrupt n destination register (MPIC_IIDR30)	32	R/W	0000_0001h	22.3.74/ 1743
5_05D8	Internal interrupt n level register (MPIC_IILR30)	32	R/W	0000_0000h	22.3.75/ 1744
5_05E0	Internal interrupt n vector/priority register (MPIC_IIVPR31)	32	R/W	8080_0000h	22.3.73/ 1742
5_05F0	Internal interrupt n destination register (MPIC_IIDR31)	32	R/W	0000_0001h	22.3.74/ 1743
5_05F8	Internal interrupt n level register (MPIC_IILR31)	32	R/W	0000_0000h	22.3.75/ 1744
5_0600	Internal interrupt n vector/priority register (MPIC_IIVPR32)	32	R/W	8080_0000h	22.3.73/ 1742
5_0610	Internal interrupt n destination register (MPIC_IIDR32)	32	R/W	0000_0001h	22.3.74/ 1743
5_0618	Internal interrupt n level register (MPIC_IILR32)	32	R/W	0000_0000h	22.3.75/ 1744
5_0620	Internal interrupt n vector/priority register (MPIC_IIVPR33)	32	R/W	8080_0000h	22.3.73/ 1742
5_0630	Internal interrupt n destination register (MPIC_IIDR33)	32	R/W	0000_0001h	22.3.74/ 1743
5_0638	Internal interrupt n level register (MPIC_IILR33)	32	R/W	0000_0000h	22.3.75/ 1744
5_0640	Internal interrupt n vector/priority register (MPIC_IIVPR34)	32	R/W	8080_0000h	22.3.73/ 1742
5_0650	Internal interrupt n destination register (MPIC_IIDR34)	32	R/W	0000_0001h	22.3.74/ 1743
5_0658	Internal interrupt n level register (MPIC_IILR34)	32	R/W	0000_0000h	22.3.75/ 1744
5_0660	Internal interrupt n vector/priority register (MPIC_IIVPR35)	32	R/W	8080_0000h	22.3.73/ 1742
5_0670	Internal interrupt n destination register (MPIC_IIDR35)	32	R/W	0000_0001h	22.3.74/ 1743

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0678	Internal interrupt n level register (MPIC_IILR35)	32	R/W	0000_0000h	22.3.75/ 1744
5_0680	Internal interrupt n vector/priority register (MPIC_IIVPR36)	32	R/W	8080_0000h	22.3.73/ 1742
5_0690	Internal interrupt n destination register (MPIC_IIDR36)	32	R/W	0000_0001h	22.3.74/ 1743
5_0698	Internal interrupt n level register (MPIC_IILR36)	32	R/W	0000_0000h	22.3.75/ 1744
5_06A0	Internal interrupt n vector/priority register (MPIC_IIVPR37)	32	R/W	8080_0000h	22.3.73/ 1742
5_06B0	Internal interrupt n destination register (MPIC_IIDR37)	32	R/W	0000_0001h	22.3.74/ 1743
5_06B8	Internal interrupt n level register (MPIC_IILR37)	32	R/W	0000_0000h	22.3.75/ 1744
5_06C0	Internal interrupt n vector/priority register (MPIC_IIVPR38)	32	R/W	8080_0000h	22.3.73/ 1742
5_06D0	Internal interrupt n destination register (MPIC_IIDR38)	32	R/W	0000_0001h	22.3.74/ 1743
5_06D8	Internal interrupt n level register (MPIC_IILR38)	32	R/W	0000_0000h	22.3.75/ 1744
5_06E0	Internal interrupt n vector/priority register (MPIC_IIVPR39)	32	R/W	8080_0000h	22.3.73/ 1742
5_06F0	Internal interrupt n destination register (MPIC_IIDR39)	32	R/W	0000_0001h	22.3.74/ 1743
5_06F8	Internal interrupt n level register (MPIC_IILR39)	32	R/W	0000_0000h	22.3.75/ 1744
5_0700	Internal interrupt n vector/priority register (MPIC_IIVPR40)	32	R/W	8080_0000h	22.3.73/ 1742
5_0710	Internal interrupt n destination register (MPIC_IIDR40)	32	R/W	0000_0001h	22.3.74/ 1743
5_0718	Internal interrupt n level register (MPIC_IILR40)	32	R/W	0000_0000h	22.3.75/ 1744
5_0720	Internal interrupt n vector/priority register (MPIC_IIVPR41)	32	R/W	8080_0000h	22.3.73/ 1742
5_0730	Internal interrupt n destination register (MPIC_IIDR41)	32	R/W	0000_0001h	22.3.74/ 1743
5_0738	Internal interrupt n level register (MPIC_IILR41)	32	R/W	0000_0000h	22.3.75/ 1744
5_0740	Internal interrupt n vector/priority register (MPIC_IIVPR42)	32	R/W	8080_0000h	22.3.73/ 1742
5_0750	Internal interrupt n destination register (MPIC_IIDR42)	32	R/W	0000_0001h	22.3.74/ 1743
5_0758	Internal interrupt n level register (MPIC_IILR42)	32	R/W	0000_0000h	22.3.75/ 1744

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0760	Internal interrupt n vector/priority register (MPIC_IIVPR43)	32	R/W	8080_0000h	22.3.73/ 1742
5_0770	Internal interrupt n destination register (MPIC_IIDR43)	32	R/W	0000_0001h	22.3.74/ 1743
5_0778	Internal interrupt n level register (MPIC_IILR43)	32	R/W	0000_0000h	22.3.75/ 1744
5_0780	Internal interrupt n vector/priority register (MPIC_IIVPR44)	32	R/W	8080_0000h	22.3.73/ 1742
5_0790	Internal interrupt n destination register (MPIC_IIDR44)	32	R/W	0000_0001h	22.3.74/ 1743
5_0798	Internal interrupt n level register (MPIC_IILR44)	32	R/W	0000_0000h	22.3.75/ 1744
5_07A0	Internal interrupt n vector/priority register (MPIC_IIVPR45)	32	R/W	8080_0000h	22.3.73/ 1742
5_07B0	Internal interrupt n destination register (MPIC_IIDR45)	32	R/W	0000_0001h	22.3.74/ 1743
5_07B8	Internal interrupt n level register (MPIC_IILR45)	32	R/W	0000_0000h	22.3.75/ 1744
5_07C0	Internal interrupt n vector/priority register (MPIC_IIVPR46)	32	R/W	8080_0000h	22.3.73/ 1742
5_07D0	Internal interrupt n destination register (MPIC_IIDR46)	32	R/W	0000_0001h	22.3.74/ 1743
5_07D8	Internal interrupt n level register (MPIC_IILR46)	32	R/W	0000_0000h	22.3.75/ 1744
5_07E0	Internal interrupt n vector/priority register (MPIC_IIVPR47)	32	R/W	8080_0000h	22.3.73/ 1742
5_07F0	Internal interrupt n destination register (MPIC_IIDR47)	32	R/W	0000_0001h	22.3.74/ 1743
5_07F8	Internal interrupt n level register (MPIC_IILR47)	32	R/W	0000_0000h	22.3.75/ 1744
5_0800	Internal interrupt n vector/priority register (MPIC_IIVPR48)	32	R/W	8080_0000h	22.3.73/ 1742
5_0810	Internal interrupt n destination register (MPIC_IIDR48)	32	R/W	0000_0001h	22.3.74/ 1743
5_0818	Internal interrupt n level register (MPIC_IILR48)	32	R/W	0000_0000h	22.3.75/ 1744
5_0820	Internal interrupt n vector/priority register (MPIC_IIVPR49)	32	R/W	8080_0000h	22.3.73/ 1742
5_0830	Internal interrupt n destination register (MPIC_IIDR49)	32	R/W	0000_0001h	22.3.74/ 1743
5_0838	Internal interrupt n level register (MPIC_IILR49)	32	R/W	0000_0000h	22.3.75/ 1744
5_0840	Internal interrupt n vector/priority register (MPIC_IIVPR50)	32	R/W	8080_0000h	22.3.73/ 1742

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0850	Internal interrupt n destination register (MPIC_IIDR50)	32	R/W	0000_0001h	22.3.74/ 1743
5_0858	Internal interrupt n level register (MPIC_IILR50)	32	R/W	0000_0000h	22.3.75/ 1744
5_0860	Internal interrupt n vector/priority register (MPIC_IIVPR51)	32	R/W	8080_0000h	22.3.73/ 1742
5_0870	Internal interrupt n destination register (MPIC_IIDR51)	32	R/W	0000_0001h	22.3.74/ 1743
5_0878	Internal interrupt n level register (MPIC_IILR51)	32	R/W	0000_0000h	22.3.75/ 1744
5_0880	Internal interrupt n vector/priority register (MPIC_IIVPR52)	32	R/W	8080_0000h	22.3.73/ 1742
5_0890	Internal interrupt n destination register (MPIC_IIDR52)	32	R/W	0000_0001h	22.3.74/ 1743
5_0898	Internal interrupt n level register (MPIC_IILR52)	32	R/W	0000_0000h	22.3.75/ 1744
5_08A0	Internal interrupt n vector/priority register (MPIC_IIVPR53)	32	R/W	8080_0000h	22.3.73/ 1742
5_08B0	Internal interrupt n destination register (MPIC_IIDR53)	32	R/W	0000_0001h	22.3.74/ 1743
5_08B8	Internal interrupt n level register (MPIC_IILR53)	32	R/W	0000_0000h	22.3.75/ 1744
5_08C0	Internal interrupt n vector/priority register (MPIC_IIVPR54)	32	R/W	8080_0000h	22.3.73/ 1742
5_08D0	Internal interrupt n destination register (MPIC_IIDR54)	32	R/W	0000_0001h	22.3.74/ 1743
5_08D8	Internal interrupt n level register (MPIC_IILR54)	32	R/W	0000_0000h	22.3.75/ 1744
5_08E0	Internal interrupt n vector/priority register (MPIC_IIVPR55)	32	R/W	8080_0000h	22.3.73/ 1742
5_08F0	Internal interrupt n destination register (MPIC_IIDR55)	32	R/W	0000_0001h	22.3.74/ 1743
5_08F8	Internal interrupt n level register (MPIC_IILR55)	32	R/W	0000_0000h	22.3.75/ 1744
5_0900	Internal interrupt n vector/priority register (MPIC_IIVPR56)	32	R/W	8080_0000h	22.3.73/ 1742
5_0910	Internal interrupt n destination register (MPIC_IIDR56)	32	R/W	0000_0001h	22.3.74/ 1743
5_0918	Internal interrupt n level register (MPIC_IILR56)	32	R/W	0000_0000h	22.3.75/ 1744
5_0920	Internal interrupt n vector/priority register (MPIC_IIVPR57)	32	R/W	8080_0000h	22.3.73/ 1742
5_0930	Internal interrupt n destination register (MPIC_IIDR57)	32	R/W	0000_0001h	22.3.74/ 1743

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0938	Internal interrupt n level register (MPIC_IILR57)	32	R/W	0000_0000h	22.3.75/ 1744
5_0940	Internal interrupt n vector/priority register (MPIC_IIVPR58)	32	R/W	8080_0000h	22.3.73/ 1742
5_0950	Internal interrupt n destination register (MPIC_IIDR58)	32	R/W	0000_0001h	22.3.74/ 1743
5_0958	Internal interrupt n level register (MPIC_IILR58)	32	R/W	0000_0000h	22.3.75/ 1744
5_0960	Internal interrupt n vector/priority register (MPIC_IIVPR59)	32	R/W	8080_0000h	22.3.73/ 1742
5_0970	Internal interrupt n destination register (MPIC_IIDR59)	32	R/W	0000_0001h	22.3.74/ 1743
5_0978	Internal interrupt n level register (MPIC_IILR59)	32	R/W	0000_0000h	22.3.75/ 1744
5_0980	Internal interrupt n vector/priority register (MPIC_IIVPR60)	32	R/W	8080_0000h	22.3.73/ 1742
5_0990	Internal interrupt n destination register (MPIC_IIDR60)	32	R/W	0000_0001h	22.3.74/ 1743
5_0998	Internal interrupt n level register (MPIC_IILR60)	32	R/W	0000_0000h	22.3.75/ 1744
5_09A0	Internal interrupt n vector/priority register (MPIC_IIVPR61)	32	R/W	8080_0000h	22.3.73/ 1742
5_09B0	Internal interrupt n destination register (MPIC_IIDR61)	32	R/W	0000_0001h	22.3.74/ 1743
5_09B8	Internal interrupt n level register (MPIC_IILR61)	32	R/W	0000_0000h	22.3.75/ 1744
5_09C0	Internal interrupt n vector/priority register (MPIC_IIVPR62)	32	R/W	8080_0000h	22.3.73/ 1742
5_09D0	Internal interrupt n destination register (MPIC_IIDR62)	32	R/W	0000_0001h	22.3.74/ 1743
5_09D8	Internal interrupt n level register (MPIC_IILR62)	32	R/W	0000_0000h	22.3.75/ 1744
5_09E0	Internal interrupt n vector/priority register (MPIC_IIVPR63)	32	R/W	8080_0000h	22.3.73/ 1742
5_09F0	Internal interrupt n destination register (MPIC_IIDR63)	32	R/W	0000_0001h	22.3.74/ 1743
5_09F8	Internal interrupt n level register (MPIC_IILR63)	32	R/W	0000_0000h	22.3.75/ 1744
5_0A00	Internal interrupt n vector/priority register (MPIC_IIVPR64)	32	R/W	8080_0000h	22.3.73/ 1742
5_0A10	Internal interrupt n destination register (MPIC_IIDR64)	32	R/W	0000_0001h	22.3.74/ 1743
5_0A18	Internal interrupt n level register (MPIC_IILR64)	32	R/W	0000_0000h	22.3.75/ 1744

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0A20	Internal interrupt n vector/priority register (MPIC_IIVPR65)	32	R/W	8080_0000h	22.3.73/ 1742
5_0A30	Internal interrupt n destination register (MPIC_IIDR65)	32	R/W	0000_0001h	22.3.74/ 1743
5_0A38	Internal interrupt n level register (MPIC_IILR65)	32	R/W	0000_0000h	22.3.75/ 1744
5_0A40	Internal interrupt n vector/priority register (MPIC_IIVPR66)	32	R/W	8080_0000h	22.3.73/ 1742
5_0A50	Internal interrupt n destination register (MPIC_IIDR66)	32	R/W	0000_0001h	22.3.74/ 1743
5_0A58	Internal interrupt n level register (MPIC_IILR66)	32	R/W	0000_0000h	22.3.75/ 1744
5_0A60	Internal interrupt n vector/priority register (MPIC_IIVPR67)	32	R/W	8080_0000h	22.3.73/ 1742
5_0A70	Internal interrupt n destination register (MPIC_IIDR67)	32	R/W	0000_0001h	22.3.74/ 1743
5_0A78	Internal interrupt n level register (MPIC_IILR67)	32	R/W	0000_0000h	22.3.75/ 1744
5_0A80	Internal interrupt n vector/priority register (MPIC_IIVPR68)	32	R/W	8080_0000h	22.3.73/ 1742
5_0A90	Internal interrupt n destination register (MPIC_IIDR68)	32	R/W	0000_0001h	22.3.74/ 1743
5_0A98	Internal interrupt n level register (MPIC_IILR68)	32	R/W	0000_0000h	22.3.75/ 1744
5_0AA0	Internal interrupt n vector/priority register (MPIC_IIVPR69)	32	R/W	8080_0000h	22.3.73/ 1742
5_0AB0	Internal interrupt n destination register (MPIC_IIDR69)	32	R/W	0000_0001h	22.3.74/ 1743
5_0AB8	Internal interrupt n level register (MPIC_IILR69)	32	R/W	0000_0000h	22.3.75/ 1744
5_0AC0	Internal interrupt n vector/priority register (MPIC_IIVPR70)	32	R/W	8080_0000h	22.3.73/ 1742
5_0AD0	Internal interrupt n destination register (MPIC_IIDR70)	32	R/W	0000_0001h	22.3.74/ 1743
5_0AD8	Internal interrupt n level register (MPIC_IILR70)	32	R/W	0000_0000h	22.3.75/ 1744
5_0AE0	Internal interrupt n vector/priority register (MPIC_IIVPR71)	32	R/W	8080_0000h	22.3.73/ 1742
5_0AF0	Internal interrupt n destination register (MPIC_IIDR71)	32	R/W	0000_0001h	22.3.74/ 1743
5_0AF8	Internal interrupt n level register (MPIC_IILR71)	32	R/W	0000_0000h	22.3.75/ 1744
5_0B00	Internal interrupt n vector/priority register (MPIC_IIVPR72)	32	R/W	8080_0000h	22.3.73/ 1742

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0B10	Internal interrupt n destination register (MPIC_IIDR72)	32	R/W	0000_0001h	22.3.74/ 1743
5_0B18	Internal interrupt n level register (MPIC_IILR72)	32	R/W	0000_0000h	22.3.75/ 1744
5_0B20	Internal interrupt n vector/priority register (MPIC_IIVPR73)	32	R/W	8080_0000h	22.3.73/ 1742
5_0B30	Internal interrupt n destination register (MPIC_IIDR73)	32	R/W	0000_0001h	22.3.74/ 1743
5_0B38	Internal interrupt n level register (MPIC_IILR73)	32	R/W	0000_0000h	22.3.75/ 1744
5_0B40	Internal interrupt n vector/priority register (MPIC_IIVPR74)	32	R/W	8080_0000h	22.3.73/ 1742
5_0B50	Internal interrupt n destination register (MPIC_IIDR74)	32	R/W	0000_0001h	22.3.74/ 1743
5_0B58	Internal interrupt n level register (MPIC_IILR74)	32	R/W	0000_0000h	22.3.75/ 1744
5_0B60	Internal interrupt n vector/priority register (MPIC_IIVPR75)	32	R/W	8080_0000h	22.3.73/ 1742
5_0B70	Internal interrupt n destination register (MPIC_IIDR75)	32	R/W	0000_0001h	22.3.74/ 1743
5_0B78	Internal interrupt n level register (MPIC_IILR75)	32	R/W	0000_0000h	22.3.75/ 1744
5_0B80	Internal interrupt n vector/priority register (MPIC_IIVPR76)	32	R/W	8080_0000h	22.3.73/ 1742
5_0B90	Internal interrupt n destination register (MPIC_IIDR76)	32	R/W	0000_0001h	22.3.74/ 1743
5_0B98	Internal interrupt n level register (MPIC_IILR76)	32	R/W	0000_0000h	22.3.75/ 1744
5_0BA0	Internal interrupt n vector/priority register (MPIC_IIVPR77)	32	R/W	8080_0000h	22.3.73/ 1742
5_0BB0	Internal interrupt n destination register (MPIC_IIDR77)	32	R/W	0000_0001h	22.3.74/ 1743
5_0BB8	Internal interrupt n level register (MPIC_IILR77)	32	R/W	0000_0000h	22.3.75/ 1744
5_0BC0	Internal interrupt n vector/priority register (MPIC_IIVPR78)	32	R/W	8080_0000h	22.3.73/ 1742
5_0BD0	Internal interrupt n destination register (MPIC_IIDR78)	32	R/W	0000_0001h	22.3.74/ 1743
5_0BD8	Internal interrupt n level register (MPIC_IILR78)	32	R/W	0000_0000h	22.3.75/ 1744
5_0BE0	Internal interrupt n vector/priority register (MPIC_IIVPR79)	32	R/W	8080_0000h	22.3.73/ 1742
5_0BF0	Internal interrupt n destination register (MPIC_IIDR79)	32	R/W	0000_0001h	22.3.74/ 1743

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0BF8	Internal interrupt n level register (MPIC_IILR79)	32	R/W	0000_0000h	22.3.75/ 1744
5_0C00	Internal interrupt n vector/priority register (MPIC_IIVPR80)	32	R/W	8080_0000h	22.3.73/ 1742
5_0C10	Internal interrupt n destination register (MPIC_IIDR80)	32	R/W	0000_0001h	22.3.74/ 1743
5_0C18	Internal interrupt n level register (MPIC_IILR80)	32	R/W	0000_0000h	22.3.75/ 1744
5_0C20	Internal interrupt n vector/priority register (MPIC_IIVPR81)	32	R/W	8080_0000h	22.3.73/ 1742
5_0C30	Internal interrupt n destination register (MPIC_IIDR81)	32	R/W	0000_0001h	22.3.74/ 1743
5_0C38	Internal interrupt n level register (MPIC_IILR81)	32	R/W	0000_0000h	22.3.75/ 1744
5_0C40	Internal interrupt n vector/priority register (MPIC_IIVPR82)	32	R/W	8080_0000h	22.3.73/ 1742
5_0C50	Internal interrupt n destination register (MPIC_IIDR82)	32	R/W	0000_0001h	22.3.74/ 1743
5_0C58	Internal interrupt n level register (MPIC_IILR82)	32	R/W	0000_0000h	22.3.75/ 1744
5_0C60	Internal interrupt n vector/priority register (MPIC_IIVPR83)	32	R/W	8080_0000h	22.3.73/ 1742
5_0C70	Internal interrupt n destination register (MPIC_IIDR83)	32	R/W	0000_0001h	22.3.74/ 1743
5_0C78	Internal interrupt n level register (MPIC_IILR83)	32	R/W	0000_0000h	22.3.75/ 1744
5_0C80	Internal interrupt n vector/priority register (MPIC_IIVPR84)	32	R/W	8080_0000h	22.3.73/ 1742
5_0C90	Internal interrupt n destination register (MPIC_IIDR84)	32	R/W	0000_0001h	22.3.74/ 1743
5_0C98	Internal interrupt n level register (MPIC_IILR84)	32	R/W	0000_0000h	22.3.75/ 1744
5_0CA0	Internal interrupt n vector/priority register (MPIC_IIVPR85)	32	R/W	8080_0000h	22.3.73/ 1742
5_0CB0	Internal interrupt n destination register (MPIC_IIDR85)	32	R/W	0000_0001h	22.3.74/ 1743
5_0CB8	Internal interrupt n level register (MPIC_IILR85)	32	R/W	0000_0000h	22.3.75/ 1744
5_0CC0	Internal interrupt n vector/priority register (MPIC_IIVPR86)	32	R/W	8080_0000h	22.3.73/ 1742
5_0CD0	Internal interrupt n destination register (MPIC_IIDR86)	32	R/W	0000_0001h	22.3.74/ 1743
5_0CD8	Internal interrupt n level register (MPIC_IILR86)	32	R/W	0000_0000h	22.3.75/ 1744

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0CE0	Internal interrupt n vector/priority register (MPIC_IIVPR87)	32	R/W	8080_0000h	22.3.73/ 1742
5_0CF0	Internal interrupt n destination register (MPIC_IIDR87)	32	R/W	0000_0001h	22.3.74/ 1743
5_0CF8	Internal interrupt n level register (MPIC_IILR87)	32	R/W	0000_0000h	22.3.75/ 1744
5_0D00	Internal interrupt n vector/priority register (MPIC_IIVPR88)	32	R/W	8080_0000h	22.3.73/ 1742
5_0D10	Internal interrupt n destination register (MPIC_IIDR88)	32	R/W	0000_0001h	22.3.74/ 1743
5_0D18	Internal interrupt n level register (MPIC_IILR88)	32	R/W	0000_0000h	22.3.75/ 1744
5_0D20	Internal interrupt n vector/priority register (MPIC_IIVPR89)	32	R/W	8080_0000h	22.3.73/ 1742
5_0D30	Internal interrupt n destination register (MPIC_IIDR89)	32	R/W	0000_0001h	22.3.74/ 1743
5_0D38	Internal interrupt n level register (MPIC_IILR89)	32	R/W	0000_0000h	22.3.75/ 1744
5_0D40	Internal interrupt n vector/priority register (MPIC_IIVPR90)	32	R/W	8080_0000h	22.3.73/ 1742
5_0D50	Internal interrupt n destination register (MPIC_IIDR90)	32	R/W	0000_0001h	22.3.74/ 1743
5_0D58	Internal interrupt n level register (MPIC_IILR90)	32	R/W	0000_0000h	22.3.75/ 1744
5_0D60	Internal interrupt n vector/priority register (MPIC_IIVPR91)	32	R/W	8080_0000h	22.3.73/ 1742
5_0D70	Internal interrupt n destination register (MPIC_IIDR91)	32	R/W	0000_0001h	22.3.74/ 1743
5_0D78	Internal interrupt n level register (MPIC_IILR91)	32	R/W	0000_0000h	22.3.75/ 1744
5_0D80	Internal interrupt n vector/priority register (MPIC_IIVPR92)	32	R/W	8080_0000h	22.3.73/ 1742
5_0D90	Internal interrupt n destination register (MPIC_IIDR92)	32	R/W	0000_0001h	22.3.74/ 1743
5_0D98	Internal interrupt n level register (MPIC_IILR92)	32	R/W	0000_0000h	22.3.75/ 1744
5_0DA0	Internal interrupt n vector/priority register (MPIC_IIVPR93)	32	R/W	8080_0000h	22.3.73/ 1742
5_0DB0	Internal interrupt n destination register (MPIC_IIDR93)	32	R/W	0000_0001h	22.3.74/ 1743
5_0DB8	Internal interrupt n level register (MPIC_IILR93)	32	R/W	0000_0000h	22.3.75/ 1744
5_0DC0	Internal interrupt n vector/priority register (MPIC_IIVPR94)	32	R/W	8080_0000h	22.3.73/ 1742

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0DD0	Internal interrupt n destination register (MPIC_IIDR94)	32	R/W	0000_0001h	22.3.74/ 1743
5_0DD8	Internal interrupt n level register (MPIC_IILR94)	32	R/W	0000_0000h	22.3.75/ 1744
5_0DE0	Internal interrupt n vector/priority register (MPIC_IIVPR95)	32	R/W	8080_0000h	22.3.73/ 1742
5_0DF0	Internal interrupt n destination register (MPIC_IIDR95)	32	R/W	0000_0001h	22.3.74/ 1743
5_0DF8	Internal interrupt n level register (MPIC_IILR95)	32	R/W	0000_0000h	22.3.75/ 1744
5_0E00	Internal interrupt n vector/priority register (MPIC_IIVPR96)	32	R/W	8080_0000h	22.3.73/ 1742
5_0E10	Internal interrupt n destination register (MPIC_IIDR96)	32	R/W	0000_0001h	22.3.74/ 1743
5_0E18	Internal interrupt n level register (MPIC_IILR96)	32	R/W	0000_0000h	22.3.75/ 1744
5_0E20	Internal interrupt n vector/priority register (MPIC_IIVPR97)	32	R/W	8080_0000h	22.3.73/ 1742
5_0E30	Internal interrupt n destination register (MPIC_IIDR97)	32	R/W	0000_0001h	22.3.74/ 1743
5_0E38	Internal interrupt n level register (MPIC_IILR97)	32	R/W	0000_0000h	22.3.75/ 1744
5_0E40	Internal interrupt n vector/priority register (MPIC_IIVPR98)	32	R/W	8080_0000h	22.3.73/ 1742
5_0E50	Internal interrupt n destination register (MPIC_IIDR98)	32	R/W	0000_0001h	22.3.74/ 1743
5_0E58	Internal interrupt n level register (MPIC_IILR98)	32	R/W	0000_0000h	22.3.75/ 1744
5_0E60	Internal interrupt n vector/priority register (MPIC_IIVPR99)	32	R/W	8080_0000h	22.3.73/ 1742
5_0E70	Internal interrupt n destination register (MPIC_IIDR99)	32	R/W	0000_0001h	22.3.74/ 1743
5_0E78	Internal interrupt n level register (MPIC_IILR99)	32	R/W	0000_0000h	22.3.75/ 1744
5_0E80	Internal interrupt n vector/priority register (MPIC_IIVPR100)	32	R/W	8080_0000h	22.3.73/ 1742
5_0E90	Internal interrupt n destination register (MPIC_IIDR100)	32	R/W	0000_0001h	22.3.74/ 1743
5_0E98	Internal interrupt n level register (MPIC_IILR100)	32	R/W	0000_0000h	22.3.75/ 1744
5_0EA0	Internal interrupt n vector/priority register (MPIC_IIVPR101)	32	R/W	8080_0000h	22.3.73/ 1742
5_0EB0	Internal interrupt n destination register (MPIC_IIDR101)	32	R/W	0000_0001h	22.3.74/ 1743

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0EB8	Internal interrupt n level register (MPIC_IILR101)	32	R/W	0000_0000h	22.3.75/ 1744
5_0EC0	Internal interrupt n vector/priority register (MPIC_IIVPR102)	32	R/W	8080_0000h	22.3.73/ 1742
5_0ED0	Internal interrupt n destination register (MPIC_IIDR102)	32	R/W	0000_0001h	22.3.74/ 1743
5_0ED8	Internal interrupt n level register (MPIC_IILR102)	32	R/W	0000_0000h	22.3.75/ 1744
5_0EE0	Internal interrupt n vector/priority register (MPIC_IIVPR103)	32	R/W	8080_0000h	22.3.73/ 1742
5_0EF0	Internal interrupt n destination register (MPIC_IIDR103)	32	R/W	0000_0001h	22.3.74/ 1743
5_0EF8	Internal interrupt n level register (MPIC_IILR103)	32	R/W	0000_0000h	22.3.75/ 1744
5_0F00	Internal interrupt n vector/priority register (MPIC_IIVPR104)	32	R/W	8080_0000h	22.3.73/ 1742
5_0F10	Internal interrupt n destination register (MPIC_IIDR104)	32	R/W	0000_0001h	22.3.74/ 1743
5_0F18	Internal interrupt n level register (MPIC_IILR104)	32	R/W	0000_0000h	22.3.75/ 1744
5_0F20	Internal interrupt n vector/priority register (MPIC_IIVPR105)	32	R/W	8080_0000h	22.3.73/ 1742
5_0F30	Internal interrupt n destination register (MPIC_IIDR105)	32	R/W	0000_0001h	22.3.74/ 1743
5_0F38	Internal interrupt n level register (MPIC_IILR105)	32	R/W	0000_0000h	22.3.75/ 1744
5_0F40	Internal interrupt n vector/priority register (MPIC_IIVPR106)	32	R/W	8080_0000h	22.3.73/ 1742
5_0F50	Internal interrupt n destination register (MPIC_IIDR106)	32	R/W	0000_0001h	22.3.74/ 1743
5_0F58	Internal interrupt n level register (MPIC_IILR106)	32	R/W	0000_0000h	22.3.75/ 1744
5_0F60	Internal interrupt n vector/priority register (MPIC_IIVPR107)	32	R/W	8080_0000h	22.3.73/ 1742
5_0F70	Internal interrupt n destination register (MPIC_IIDR107)	32	R/W	0000_0001h	22.3.74/ 1743
5_0F78	Internal interrupt n level register (MPIC_IILR107)	32	R/W	0000_0000h	22.3.75/ 1744
5_0F80	Internal interrupt n vector/priority register (MPIC_IIVPR108)	32	R/W	8080_0000h	22.3.73/ 1742
5_0F90	Internal interrupt n destination register (MPIC_IIDR108)	32	R/W	0000_0001h	22.3.74/ 1743
5_0F98	Internal interrupt n level register (MPIC_IILR108)	32	R/W	0000_0000h	22.3.75/ 1744

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_0FA0	Internal interrupt n vector/priority register (MPIC_IIVPR109)	32	R/W	8080_0000h	22.3.73/ 1742
5_0FB0	Internal interrupt n destination register (MPIC_IIDR109)	32	R/W	0000_0001h	22.3.74/ 1743
5_0FB8	Internal interrupt n level register (MPIC_IILR109)	32	R/W	0000_0000h	22.3.75/ 1744
5_0FC0	Internal interrupt n vector/priority register (MPIC_IIVPR110)	32	R/W	8080_0000h	22.3.73/ 1742
5_0FD0	Internal interrupt n destination register (MPIC_IIDR110)	32	R/W	0000_0001h	22.3.74/ 1743
5_0FD8	Internal interrupt n level register (MPIC_IILR110)	32	R/W	0000_0000h	22.3.75/ 1744
5_0FE0	Internal interrupt n vector/priority register (MPIC_IIVPR111)	32	R/W	8080_0000h	22.3.73/ 1742
5_0FF0	Internal interrupt n destination register (MPIC_IIDR111)	32	R/W	0000_0001h	22.3.74/ 1743
5_0FF8	Internal interrupt n level register (MPIC_IILR111)	32	R/W	0000_0000h	22.3.75/ 1744
5_1600	Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA0)	32	R/W	8080_0000h	22.3.76/ 1745
5_1610	Messaging interrupt destination register [Group A] n (MPIC_MIDRA0)	32	R/W	0000_0001h	22.3.77/ 1746
5_1620	Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA1)	32	R/W	8080_0000h	22.3.76/ 1745
5_1630	Messaging interrupt destination register [Group A] n (MPIC_MIDRA1)	32	R/W	0000_0001h	22.3.77/ 1746
5_1640	Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA2)	32	R/W	8080_0000h	22.3.76/ 1745
5_1650	Messaging interrupt destination register [Group A] n (MPIC_MIDRA2)	32	R/W	0000_0001h	22.3.77/ 1746
5_1660	Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA3)	32	R/W	8080_0000h	22.3.76/ 1745
5_1670	Messaging interrupt destination register [Group A] n (MPIC_MIDRA3)	32	R/W	0000_0001h	22.3.77/ 1746
5_1680	Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRB0)	32	R/W	8080_0000h	22.3.78/ 1747
5_1690	Messaging interrupt destination register [Group B] n (MPIC_MIDRB0)	32	R/W	0000_0001h	22.3.79/ 1748
5_16A0	Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRB1)	32	R/W	8080_0000h	22.3.78/ 1747
5_16B0	Messaging interrupt destination register [Group B] n (MPIC_MIDRB1)	32	R/W	0000_0001h	22.3.79/ 1748
5_16C0	Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRB2)	32	R/W	8080_0000h	22.3.78/ 1747

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_16D0	Messaging interrupt destination register [Group B] n (MPIC_MIDRB2)	32	R/W	0000_0001h	22.3.79/ 1748
5_16E0	Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRB3)	32	R/W	8080_0000h	22.3.78/ 1747
5_16F0	Messaging interrupt destination register [Group B] n (MPIC_MIDRB3)	32	R/W	0000_0001h	22.3.79/ 1748
5_1C00	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA0)	32	R/W	8000_0000h	22.3.80/ 1750
5_1C10	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA0)	32	R/W	0000_0001h	22.3.81/ 1751
5_1C20	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA1)	32	R/W	8000_0000h	22.3.80/ 1750
5_1C30	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA1)	32	R/W	0000_0001h	22.3.81/ 1751
5_1C40	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA2)	32	R/W	8000_0000h	22.3.80/ 1750
5_1C50	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA2)	32	R/W	0000_0001h	22.3.81/ 1751
5_1C60	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA3)	32	R/W	8000_0000h	22.3.80/ 1750
5_1C70	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA3)	32	R/W	0000_0001h	22.3.81/ 1751
5_1C80	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA4)	32	R/W	8000_0000h	22.3.80/ 1750
5_1C90	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA4)	32	R/W	0000_0001h	22.3.81/ 1751
5_1CA0	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA5)	32	R/W	8000_0000h	22.3.80/ 1750
5_1CB0	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA5)	32	R/W	0000_0001h	22.3.81/ 1751
5_1CC0	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA6)	32	R/W	8000_0000h	22.3.80/ 1750
5_1CD0	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA6)	32	R/W	0000_0001h	22.3.81/ 1751
5_1CE0	Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRA7)	32	R/W	8000_0000h	22.3.80/ 1750
5_1CF0	Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRA7)	32	R/W	0000_0001h	22.3.81/ 1751
5_1D00	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB0)	32	R/W	8000_0000h	22.3.82/ 1752
5_1D10	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB0)	32	R/W	0000_0001h	22.3.83/ 1753
5_1D20	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB1)	32	R/W	8000_0000h	22.3.82/ 1752

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_1D30	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB1)	32	R/W	0000_0001h	22.3.83/ 1753
5_1D40	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB2)	32	R/W	8000_0000h	22.3.82/ 1752
5_1D50	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB2)	32	R/W	0000_0001h	22.3.83/ 1753
5_1D60	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB3)	32	R/W	8000_0000h	22.3.82/ 1752
5_1D70	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB3)	32	R/W	0000_0001h	22.3.83/ 1753
5_1D80	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB4)	32	R/W	8000_0000h	22.3.82/ 1752
5_1D90	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB4)	32	R/W	0000_0001h	22.3.83/ 1753
5_1DA0	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB5)	32	R/W	8000_0000h	22.3.82/ 1752
5_1DB0	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB5)	32	R/W	0000_0001h	22.3.83/ 1753
5_1DC0	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB6)	32	R/W	8000_0000h	22.3.82/ 1752
5_1DD0	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB6)	32	R/W	0000_0001h	22.3.83/ 1753
5_1DE0	Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB7)	32	R/W	8000_0000h	22.3.82/ 1752
5_1DF0	Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRB7)	32	R/W	0000_0001h	22.3.83/ 1753
5_1E00	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC0)	32	R/W	8000_0000h	22.3.84/ 1754
5_1E10	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC0)	32	R/W	0000_0001h	22.3.85/ 1755
5_1E20	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC1)	32	R/W	8000_0000h	22.3.84/ 1754
5_1E30	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC1)	32	R/W	0000_0001h	22.3.85/ 1755
5_1E40	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC2)	32	R/W	8000_0000h	22.3.84/ 1754
5_1E50	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC2)	32	R/W	0000_0001h	22.3.85/ 1755
5_1E60	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC3)	32	R/W	8000_0000h	22.3.84/ 1754
5_1E70	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC3)	32	R/W	0000_0001h	22.3.85/ 1755
5_1E80	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC4)	32	R/W	8000_0000h	22.3.84/ 1754

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5_1E90	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC4)	32	R/W	0000_0001h	22.3.85/ 1755
5_1EA0	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC5)	32	R/W	8000_0000h	22.3.84/ 1754
5_1EB0	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC5)	32	R/W	0000_0001h	22.3.85/ 1755
5_1EC0	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC6)	32	R/W	8000_0000h	22.3.84/ 1754
5_1ED0	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC6)	32	R/W	0000_0001h	22.3.85/ 1755
5_1EE0	Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRC7)	32	R/W	8000_0000h	22.3.84/ 1754
5_1EF0	Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRC7)	32	R/W	0000_0001h	22.3.85/ 1755
6_0040	Interprocessor interrupt dispatch register 0 (MPIC_CPU0_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_0050	Interprocessor interrupt dispatch register 1 (MPIC_CPU0_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_0060	Interprocessor interrupt dispatch register 2 (MPIC_CPU0_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_0070	Interprocessor interrupt dispatch register 3 (MPIC_CPU0_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_0080	Current task priority register (MPIC_CPU0_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_0090	Who am I register (MPIC_CPU0_WHOAMI)	32	R	See section	22.3.91/ 1763
6_00A0	Interrupt acknowledge register (MPIC_CPU0_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_00B0	End of interrupt register (MPIC_CPU0_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_1040	Interprocessor interrupt dispatch register 0 (MPIC_CPU1_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_1050	Interprocessor interrupt dispatch register 1 (MPIC_CPU1_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_1060	Interprocessor interrupt dispatch register 2 (MPIC_CPU1_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_1070	Interprocessor interrupt dispatch register 3 (MPIC_CPU1_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_1080	Current task priority register (MPIC_CPU1_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_1090	Who am I register (MPIC_CPU1_WHOAMI)	32	R	See section	22.3.91/ 1763
6_10A0	Interrupt acknowledge register (MPIC_CPU1_IACK)	32	R	0000_0000h	22.3.92/ 1764

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
6_10B0	End of interrupt register (MPIC_CPU1_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_2040	Interprocessor interrupt dispatch register 0 (MPIC_CPU2_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_2050	Interprocessor interrupt dispatch register 1 (MPIC_CPU2_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_2060	Interprocessor interrupt dispatch register 2 (MPIC_CPU2_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_2070	Interprocessor interrupt dispatch register 3 (MPIC_CPU2_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_2080	Current task priority register (MPIC_CPU2_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_2090	Who am I register (MPIC_CPU2_WHOAMI)	32	R	See section	22.3.91/ 1763
6_20A0	Interrupt acknowledge register (MPIC_CPU2_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_20B0	End of interrupt register (MPIC_CPU2_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_3040	Interprocessor interrupt dispatch register 0 (MPIC_CPU3_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_3050	Interprocessor interrupt dispatch register 1 (MPIC_CPU3_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_3060	Interprocessor interrupt dispatch register 2 (MPIC_CPU3_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_3070	Interprocessor interrupt dispatch register 3 (MPIC_CPU3_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_3080	Current task priority register (MPIC_CPU3_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_3090	Who am I register (MPIC_CPU3_WHOAMI)	32	R	See section	22.3.91/ 1763
6_30A0	Interrupt acknowledge register (MPIC_CPU3_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_30B0	End of interrupt register (MPIC_CPU3_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_4040	Interprocessor interrupt dispatch register 0 (MPIC_CPU4_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_4050	Interprocessor interrupt dispatch register 1 (MPIC_CPU4_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_4060	Interprocessor interrupt dispatch register 2 (MPIC_CPU4_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_4070	Interprocessor interrupt dispatch register 3 (MPIC_CPU4_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_4080	Current task priority register (MPIC_CPU4_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6_4090	Who am I register (MPIC_CPU4_WHOAMI)	32	R	See section	22.3.91/ 1763
6_40A0	Interrupt acknowledge register (MPIC_CPU4_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_40B0	End of interrupt register (MPIC_CPU4_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_5040	Interprocessor interrupt dispatch register 0 (MPIC_CPU5_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_5050	Interprocessor interrupt dispatch register 1 (MPIC_CPU5_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_5060	Interprocessor interrupt dispatch register 2 (MPIC_CPU5_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_5070	Interprocessor interrupt dispatch register 3 (MPIC_CPU5_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_5080	Current task priority register (MPIC_CPU5_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_5090	Who am I register (MPIC_CPU5_WHOAMI)	32	R	See section	22.3.91/ 1763
6_50A0	Interrupt acknowledge register (MPIC_CPU5_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_50B0	End of interrupt register (MPIC_CPU5_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_6040	Interprocessor interrupt dispatch register 0 (MPIC_CPU6_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_6050	Interprocessor interrupt dispatch register 1 (MPIC_CPU6_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_6060	Interprocessor interrupt dispatch register 2 (MPIC_CPU6_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760
6_6070	Interprocessor interrupt dispatch register 3 (MPIC_CPU6_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_6080	Current task priority register (MPIC_CPU6_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_6090	Who am I register (MPIC_CPU6_WHOAMI)	32	R	See section	22.3.91/ 1763
6_60A0	Interrupt acknowledge register (MPIC_CPU6_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_60B0	End of interrupt register (MPIC_CPU6_EOI)	32	W	0000_0000h	22.3.93/ 1765
6_7040	Interprocessor interrupt dispatch register 0 (MPIC_CPU7_IPIDR0)	32	W	0000_0000h	22.3.86/ 1757
6_7050	Interprocessor interrupt dispatch register 1 (MPIC_CPU7_IPIDR1)	32	W	0000_0000h	22.3.87/ 1758
6_7060	Interprocessor interrupt dispatch register 2 (MPIC_CPU7_IPIDR2)	32	W	0000_0000h	22.3.88/ 1760

Table continues on the next page...

MPIC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
6_7070	Interprocessor interrupt dispatch register 3 (MPIC_CPU7_IPIDR3)	32	W	0000_0000h	22.3.89/ 1761
6_7080	Current task priority register (MPIC_CPU7_CTPR)	32	R/W	0000_000Fh	22.3.90/ 1762
6_7090	Who am I register (MPIC_CPU7_WHOAMI)	32	R	See section	22.3.91/ 1763
6_70A0	Interrupt acknowledge register (MPIC_CPU7_IACK)	32	R	0000_0000h	22.3.92/ 1764
6_70B0	End of interrupt register (MPIC_CPU7_EOI)	32	W	0000_0000h	22.3.93/ 1765

22.3.1 Block revision register 1 (MPIC_BRR1)

BRR1 provides information about the MPIC IP block.

Address: 4_0000h base + 0h offset = 4_0000h

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

MPIC_BRR1 field descriptions

Field	Description
0–15 IPID	IP block ID.
16–23 IPMJ	The major revision of the IP block. NOTE: Reset value is implementation-specific.
24–31 IPMN	The minor revision of the IP block. NOTE: Reset value is implementation-specific.

22.3.2 Block revision register 2 (MPIC_BRR2)

BRR2 provides information about the IP block integration option and IP block configuration options.

MPIC Memory Map

Address: 4_0000h base + 10h offset = 4_0010h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								IPINTO								Reserved								IPCFG0							
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

MPIC_BRR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IPINTO	IP block integration options NOTE: Reset value is implementation-specific.
16–23 -	This field is reserved. Reserved
24–31 IPCFG0	IP block configuration options NOTE: Reset value is implementation-specific.

22.3.3 Interprocessor interrupt dispatch register n (MPIC_IPIDRn)

The following figure shows the four private-access IPIDRs, one for each interprocessor interrupt channel. Because external bus masters can access these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device.

Address: 4_0000h base + 40h offset + (16d × i), where i=0d to 3d

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_IPIDRn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved

Table continues on the next page...

MPIC_IPIDR*n* field descriptions (continued)

Field	Description
24 P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25 P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26 P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27 P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4
28 P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29 P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2
30 P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31 P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

22.3.4 Current task priority register (MPIC_CTPR)

There is one current task priority register per processor core on this chip. See [Global- and private-access per-CPU registers](#) for more information.

NOTE

CTPR has meaning only for interrupts routed to *int*.

Software must write the priority of the current processor core task in the CTPR for each core. The MPIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to that core if the following apply:

- The interrupt is not masked.
- The priority of the interrupt is higher than the values in the corresponding CTPR[TASKP] and ISR.

Address: 4_0000h base + 80h offset = 4_0080h

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

MPIC_CTPR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 TASKP	Task priority. Indicates the threshold that individual interrupt priorities must exceed for the interrupt request to be serviced. Task priority is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal. 0000-1111 xVPRn[PRIORITY] must exceed this value for the interrupt request to be serviced. Note the following special cases: 0000 Lowest priority. All interrupts except those whose priority are 0 can be serviced. 1111 Highest priority. No interrupts are signaled to that processor core. Hardware selects this value on a device hard reset or when the corresponding PIR[Pn] is set.

22.3.5 Who am I register (MPIC_WHOAMI)

The processor core WHOAMI *n* register can be read by a processor core to determine its physical connection to the MPIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

Address: 4_0000h base + 90h offset = 4_0090h

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

MPIC_WHOAMI field descriptions

Field	Description																
0–26 -	This field is reserved. Reserved																
27–31 ID	Returns the ID of the processor core reading this register. NOTE: ID = 11111 indicates an illegal processor ID <table> <tr><td>00000</td><td>Processor core 0</td></tr> <tr><td>00001</td><td>Processor core 1</td></tr> <tr><td>00010</td><td>Processor core 2</td></tr> <tr><td>00011</td><td>Processor core 3</td></tr> <tr><td>00100</td><td>Processor core 4</td></tr> <tr><td>00101</td><td>Processor core 5</td></tr> <tr><td>00110</td><td>Processor core 6</td></tr> <tr><td>00111</td><td>Processor core 7</td></tr> </table>	00000	Processor core 0	00001	Processor core 1	00010	Processor core 2	00011	Processor core 3	00100	Processor core 4	00101	Processor core 5	00110	Processor core 6	00111	Processor core 7
00000	Processor core 0																
00001	Processor core 1																
00010	Processor core 2																
00011	Processor core 3																
00100	Processor core 4																
00101	Processor core 5																
00110	Processor core 6																
00111	Processor core 7																

22.3.6 Interrupt acknowledge register (MPIC_IACK)**NOTE**

IACK has meaning only for interrupts routed to *int* in mixed mode (GCR[M] = 01) and should not be accessed for interrupts routed to *cint*, *mcp*, *sie* or *IRQ_OUT_B*.

NOTE

The external proxy facility mode (GCR[M] = 11), eliminates the need for the core to read the IACK register as the vector is automatically passed to the core's external proxy register (EPR) or guest external proxy register (GEPR).

In systems based on processors that implement the Power architecture, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK). Each processor core has an IACK register assigned to it. Reading IACK returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the corresponding interrupt pending register (IPR) is cleared for edge-sensitive interrupts.
- The corresponding in-service register (ISR) is updated.
- The corresponding int output signal from the MPIC is negated.

Reading IACK when no interrupt is pending returns the spurious vector value, as described in [Spurious Vector Register](#) ."

MPIC Memory Map

Address: 4_0000h base + A0h offset = 4_00A0h

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

MPIC_IACK field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only). This field is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal.

22.3.7 End of interrupt register (MPIC_EOI)**NOTE**

EOI has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint*, *mcp*, *sie* or *IRQ_OUT_B*.

Each core is assigned an EOI register. Writing to EOI signals the end of processing for the highest-priority interrupt (routed to *int*) currently in service. It also updates the corresponding ISR *n* by retiring the highest priority interrupt. Data values written to EOI are ignored, and zero is assumed.

Address: 4_0000h base + B0h offset = 4_00B0h

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

MPIC_EOI field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EOI_CODE	EOI_CODE 0000 (write only)

22.3.8 Feature reporting register (MPIC_FRR)

FRR provides information about interrupt and processor core configurations. It also informs the programming environment of the controller version.

Address: 4_0000h base + 1000h offset = 4_1000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				NIRQ								Reserved				NCPU				VID											
W																																
Reset	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	1	1	1	0	0	0	0	0	1	0		

* Notes:

- NIRQ field: Reset value is 0C3h on P4080, Rev 3; 0FBh on P4080, Rev 2.

MPIC_FRR field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–15 NIRQ	Number of interrupts. Holds the binary value of the number of the highest interrupt source supported minus one. NOTE: Reset value is implementation-specific.
16–18 -	This field is reserved. Reserved
19–23 NCPU	Number of CPUs. The number of the physical CPUs (or processor cores) supported minus one. NOTE: Reset value is implementation-specific. 00000 Single core 00001 Two cores 00111 Eight cores 01000-11111 Reserved
24–31 VID	Version ID. Reports the OpenPIC specification revision level supported by this interrupt controller implementation. The VID field's value of two (0x02) corresponds to revision 1.2 which is the revision level currently supported. NOTE: Reset value is implementation-specific.

22.3.9 Global configuration register (MPIC_GCR)

GCR controls the MPIC's operating mode, and allows software to reset the MPIC.

Address: 4_0000h base + 1020h offset = 4_1020h

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

MPIC_GCR field descriptions

Field	Description
0 RST	Reset. Setting RST forces the MPIC to be reset. Cleared automatically when the reset sequence is complete. See Resetting the MPIC , for more information.
1–2 M	Mode. MPIC operating mode. MPIC modes of operation , provides details about these modes. 00 Pass-through mode. On-chip MPIC is disabled and interrupts detected on IRQ0 are passed directly to core 0. 01 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the MPIC. 10 reserved 11 External proxy facility mode. Interrupts are delivered to the cores using the external proxy facility model.
3–31 -	This field is reserved. Reserved

22.3.10 Vendor identification register (MPIC_VIR)

VIR is defined by the OpenPIC specifications and is provided for compliance. The zero value for VIR[VENDOR_ID] indicates a generic OpenPIC-compliant device, which makes the other VIR fields meaningless.

Address: 4_0000h base + 1080h offset = 4_1080h

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

MPIC_VIR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 STEP	Stepping. Indicates the silicon revision for this device. Has no meaning if VENDOR_ID value is zero.
16–23 DEVICE_ID	Device identification. Vendor-specified identifier for this device. Has no meaning if VENDOR_ID is zero.
24–31 VENDOR_ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic OpenPIC-compliant device.

22.3.11 Processor initialization register (MPIC_PIR)

PIR provides a way for software to initiate a reset to the processor cores. Setting Pn causes the respective signal to assert. These signals are routed to the appropriate *coren_hreset* signals and cause the corresponding core to redirect program flow to the reset interrupt vector.

For proper system operation, a core should be reset in this way only if the core is already in nap or sleep state. Because a core in either state cannot perform the necessary write to cause a hard reset, a core cannot put itself into hard reset.

Each core must be reset individually. The recommended sequence for resetting multiple cores is as follows:

1. Write a 1 to the PIR[Pn] bit of the first core to be reset.
2. Read the PIR; this pushes the previous write.
3. Repeat steps 1 and 2 for the next core to be reset.
4. Repeat steps 1 and 2 for each subsequent core until all the desired cores have been reset.

Address: 4_0000h base + 1090h offset = 4_1090h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R									Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R				Reserved				P7	P6	P5	P4	P3	P2	P1	P0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_PIR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 7.
25 P6	Processor core 6 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 6.
26 P5	Processor core 5 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 5.
27 P4	Processor core 4 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 4.
28 P3	Processor core 3 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 3.
29 P2	Processor core 2 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 2.
30 P1	Processor core 1 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 1.
31 P0	Processor core 0 reset. Setting this bit causes the MPIC to assert the pic_core_hard_reset signal to core 0.

22.3.12 Processor NMI register (MPIC_PNMIR)

The PNMIR provides a way for software on another device to signal a non-maskable interrupt event to the processor cores by writing to a MPIC register. A *coren_nmi* signal to the cores is asserted when a one is written to the corresponding PNMIR field; these signals are held active until a zero is written to those same bits. This deasserts the *core_nmi* input of the corresponding core.

Address: 4_0000h base + 1098h offset = 4_1098h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_PNMIR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved

Table continues on the next page...

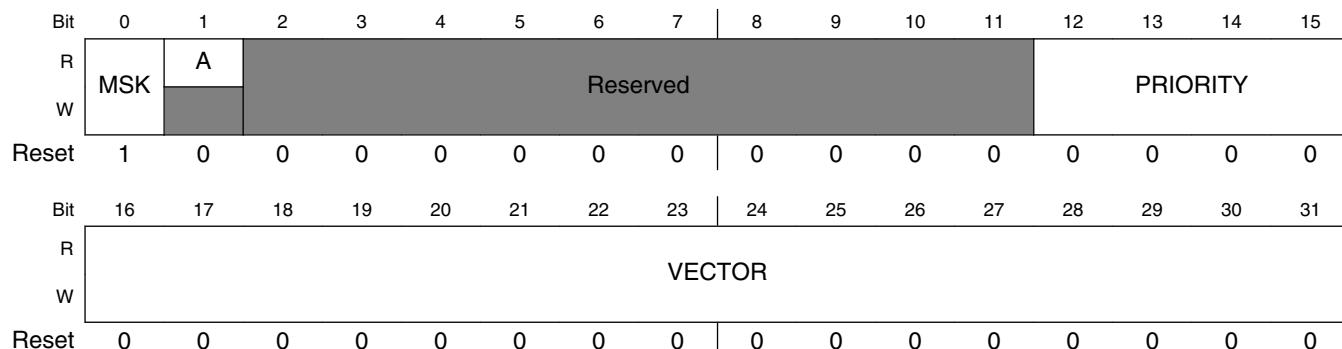
MPIC_PNMIR field descriptions (continued)

Field	Description
24 P7	Processor core 7 NMI. Setting P7 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 7.
25 P6	Processor core 6 NMI. Setting P6 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 6.
26 P5	Processor core 5 NMI. Setting P5 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 5.
27 P4	Processor core 4 NMI. Setting P4 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 4.
28 P3	Processor core 3 NMI. Setting P3 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 3.
29 P2	Processor core 2 NMI. Setting P2 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 2.
30 P1	Processor core 1 NMI. Setting P1 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 1.
31 P0	Processor core 0 NMI. Setting P0 causes the MPIC to assert the <i>pic_core_nmi</i> signal to core 0.

22.3.13 IPI vector/priority register n (MPIC_IPIVPRn)

IPIVPRs contain the interrupt vector and priority fields for the four interprocessor interrupt channels. There is one vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR *n* [A] is set.

Address: 4_0000h base + 10A0h offset + (16d × i), where i=0d to 3d

**MPIC_IPIVPRn field descriptions**

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> .

Table continues on the next page...

MPIC_IPIVPRn field descriptions (continued)

Field	Description
	0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.14 Spurious vector register (MPIC_SVR)

SVR contains the 16-bit vector returned to the processor core when the corresponding IACK register is read for a spurious interrupt.

Address: 4_0000h base + 10E0h offset = 4_10E0h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

MPIC_SVR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 VECTOR	Spurious interrupt vector. Value returned when IACK is read during a spurious vector fetch. Spurious Vector Generation , gives information about the conditions that may cause a spurious vector fetch.

22.3.15 Timer frequency reporting register [Group A] (MPIC_TFRRRA)

The TFRRs are written by software to report the clocking frequency of the MPIC timers. Note that although TFRRs are read/write, the MPIC ignores the register values.

Address: 4_0000h base + 10F0h offset = 4_10F0h

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

Reset 0

MPIC_TFRRRA field descriptions

Field	Description
0–31 FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source,-either the MPIC input clock () or the frequency of the RTC signal-to user software. See RCW Field Definitions . TFRRx is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency, as set by TCRx[CLKR]. See Timer control register [Group A] (MPIC_TCRA) ."

22.3.16 Global timer current count register [Group A] n (MPIC_GTCCRAn)

The GTCCRs contain the current count for each of the four MPIC timers in each of the two groups.

Address: 4_0000h base + 1100h offset + (64d × i), where i=0d to 3d

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

MPIC_GTCCRAn field descriptions

Field	Description
0 TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR xn [CI] goes from 1 to 0.
1–31 COUNT	Current count. Decremented while GTBCR xn [CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding GTBCR xn . Cascaded timers are reloaded with either all ones, or the GTBCR xn contents, depending on the value of TCRn[ROVR]. See Timer control register [Group A] (MPIC_TCRA) , " for more details.

22.3.17 Global timer base count register [Group A] n (MPIC_GTBCRAn)

The GTBCRs contain the base counts for each of the four MPIC timers in each of the two groups. This value is reloaded into the corresponding GTCCR xn when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR xn [CI] = 0), the timer generates an interrupt on every timer cycle.

Address: 4_0000h base + 1110h offset + (64d × i), where i=0d to 3d

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

MPIC_GTBCRAn field descriptions

Field	Description
0 CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31 BASE_CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding GTCCR xn and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the GTCCR xn at the next zero crossing of the current count.

22.3.18 Global timer vector/priority register [Group A] n (MPIC_GTVPRAn)

The GTVPRs contain the interrupt vector and the interrupt priority values for the timers. They also contain the mask and activity fields for all the timers.

Address: 4_0000h base + 1120h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSK	A							Reserved						PRIORITY	
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									VECTOR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_GTVPRAn field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.19 Global timer destination register [Group A] n (MPIC_GTDRAn)

The GTDR xn registers control the destination core to which each timer's interrupt is directed. Note that GTDR xn bits can be set independently of each other and any or all can be set for this type of interrupt.

Address: 4_0000h base + 1130h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_GTDRAn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive this interrupt 1 Directs the timer interrupt to processor core 7.
25 P6	Processor core 6. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive this interrupt 1 Directs the timer interrupt to Processor core 6.
26 P5	Processor core 5. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive this interrupt 1 Directs the timer interrupt to processor core 5.
27 P4	Processor core 4. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive this interrupt 1 Directs the timer interrupt to processor core 4.
28 P3	Processor core 3. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive this interrupt 1 Directs the timer interrupt to processors core 3.
29 P2	Processor core 2. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive this interrupt 1 Directs the timer interrupt to processor core 2.

Table continues on the next page...

MPIC_GTDRA_n field descriptions (continued)

Field	Description
30 P1	Processor core 1. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive this interrupt 1 Directs the timer interrupt to processor core 1.
31 P0	Processor core 0. Default destination after MPIC is reset. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 0 does not receive this interrupt. 1 Directs the timer interrupt to processor core 0.

22.3.20 Timer control register [Group A] (MPIC_TCRA)

The TCR registers provide various configuration options such as count frequency and roll-over behavior for the timers.

There are two choices for the clock source for the timers: a selectable frequency ratio from the MPIC input clock (), or the RTC signal. TCRs can be cascaded to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer, or one 127-bit timer (within each group).

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field, TCRx[ROVR]. Setting TCRx[ROVR] causes its GTCCR *xn* to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF_FFFF instead of its base count value. Clearing a timer's associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded, the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the TCRx[ROVR] settings.

Address: 4_0000h base + 1300h offset = 4_1300h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				ROVR		Reserved						RTM			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				CLKR		Reserved						CASC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_TCRA field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–7 ROVR	<p>Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5-7 correspond to timers 2-0. Note that global timer 3 always reloads with its GTBCR xn.</p> <ul style="list-style-type: none"> 0 The timer does not roll over. When the count reaches zero, GTCCR xn is reloaded with the GTBCR xn value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, GTCCR xn is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF). 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF). 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF). 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF). 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF). 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF). 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14 -	This field is reserved. Reserved
15 RTM	Real time mode. Specifies the clock source for the MPIC timers. <ul style="list-style-type: none"> 0 Timer clock frequency is a ratio of the frequency of the MPIC input clock as determined by the CLKR field. This is the default value. () 1 The RTC signal is used to clock the MPIC timers. If this bit is set, the CLKR field has no meaning.
16–21 -	This field is reserved. Reserved
22–23 CLKR	Clock ratio. Specifies the ratio of the timer frequency to the MPIC input clock (). The following clock ratios are supported: <ul style="list-style-type: none"> 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28 -	This field is reserved. Reserved
29–31 CASC	Cascade timers. Specifies the output of particular global timers as input to others. <ul style="list-style-type: none"> 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

22.3.21 Message register [Group A] n (MPIC_MSGRAn)

The message registers (MSGRx0-MSGRx3) can contain a 32-bit message.

Address: 4 0000h base + 1400h offset + (16d × i), where i=0d to 3d

MPIC MSGRAn field descriptions

Field	Description
0-31 MSG	Message. Contains the 32-bit message data.

22.3.22 Message enable register [Group A] (MPIC_MERA)

The MERs contain the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

Address: 4 0000h base + 1500h offset = 4 1500h

MPIC MERA field descriptions

Field	Description
0-27 -	This field is reserved. Reserved
28-31 En	Enable 3-enable 0. Used to enable interrupt generation for MSGRn (where $n = 0-3$). 0 Interrupt generation for MSGRn disabled. 1 Interrupt generation for MSGRn enabled.

22.3.23 Message status register [Group A] (MPIC_MSRA)

The MSRs contain status bits for each message register. A status bit is set when the corresponding messaging interrupt is active. Writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

Address: 4_0000h base + 1510h offset = 4_1510h

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

MPIC_MSRA field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 Sn	Status 3-status 0. Reports status of messaging interrupt n . Writing a 1 clears this field. 0 Messaging interrupt n is not active. 1 Messaging interrupt n is active.

22.3.24 Shared message signaled interrupt register [Bank A] n (MPIC_MSIRAn)

Each bank of eight MSIRs indicates which of the up to 32 interrupt sources sharing a message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Address: 4_0000h base + 1600h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SH31	SH30	SH29	SH28	SH27	SH26	SH25	SH24	SH23	SH22	SH21	SH20	SH_19	SH_18	SH_17	SH_16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH_15	SH_14	SH_13	SH_12	SH_11	SH_10	SH_9	SH_8	SH_7	SH_6	SH_5	SH_4	SH_3	SH_2	SH_1	SH_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_MSIRAn field descriptions

Field	Description
0 SH31	Message sharer 31 has a pending interrupt.
1 SH30	Message sharer 30 has a pending interrupt.
2 SH29	Message sharer 29 has a pending interrupt.
3 SH28	Message sharer 28 has a pending interrupt.
4 SH27	Message sharer 27 has a pending interrupt.
5 SH26	Message sharer 26 has a pending interrupt.
6 SH25	Message sharer 25 has a pending interrupt.
7 SH24	Message sharer 24 has a pending interrupt.
8 SH23	Message sharer 23 has a pending interrupt.
9 SH22	Message sharer 22 has a pending interrupt.

Table continues on the next page...

MPIC_MSIRAn field descriptions (continued)

Field	Description
10 SH21	Message sharer 21 has a pending interrupt.
11 SH20	Message sharer 20 has a pending interrupt.
12 SH_19	Message sharer 19 has a pending interrupt.
13 SH_18	Message sharer 18 has a pending interrupt.
14 SH_17	Message sharer 17 has a pending interrupt.
15 SH_16	Message sharer 16 has a pending interrupt.
16 SH_15	Message sharer 15 has a pending interrupt.
17 SH_14	Message sharer 14 has a pending interrupt.
18 SH_13	Message sharer 13 has a pending interrupt.
19 SH_12	Message sharer 12 has a pending interrupt.
20 SH_11	Message sharer 11 has a pending interrupt.
21 SH_10	Message sharer 10 has a pending interrupt.
22 SH_9	Message sharer 9 has a pending interrupt.
23 SH_8	Message sharer 8 has a pending interrupt.
24 SH_7	Message sharer 7 has a pending interrupt.
25 SH_6	Message sharer 6 has a pending interrupt.
26 SH_5	Message sharer 5 has a pending interrupt.
27 SH_4	Message sharer 4 has a pending interrupt.
28 SH_3	Message sharer 3 has a pending interrupt.
29 SH_2	Message sharer 2 has a pending interrupt.
30 SH_1	Message sharer 1 has a pending interrupt.
31 SH_0	Message sharer 0 has a pending interrupt.

22.3.25 Shared message signaled interrupt status register [Bank A] (MPIC_MSISRA)

MSISRa contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIRa has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIRa.

Address: 4 0000h base + 1720h offset = 4 1720h

MPIC MSISRA field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 Sn	Status n. 0 MSIRn is not active. 1 MSIRn has an active interrupt.

22.3.26 Shared message signaled interrupt index register [Bank A] (MPIC_MSIIRA)

MSIIRA provides the mechanism for setting an interrupt in the MSIRAs. When MSIIRA is written, MSIIRA[SRS] selects the register in which an interrupt bit is to be set. MSIIRA[IBS] selects the shared interrupt field in the selected MSIRA register. MSIIRA is primarily intended to support PCI Express MSIs.

This register is also aliased from MSIIRA at 04 4140h.

Address: 4_0000h base + 1740h offset = 4_1740h

MPIC_MSIIRA field descriptions

Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. 000 MSIR 0 001 MSIR 1 010 MSIR 2... 111 MSIR 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved

22.3.27 Shared message signaled interrupt register [Bank B] n (MPIC_MSIRBn)

Each bank of eight MSIRs indicates which of the up to 32 interrupt sources sharing a message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Address: 4_0000h base + 1800h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SH31	SH30	SH29	SH28	SH27	SH26	SH25	SH24	SH23	SH22	SH21	SH20	SH_19	SH_18	SH_17	SH_16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH_15	SH_14	SH_13	SH_12	SH_11	SH_10	SH_9	SH_8	SH_7	SH_6	SH_5	SH_4	SH_3	SH_2	SH_1	SH_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_MSIRBn field descriptions

Field	Description
0 SH31	Message sharer 31 has a pending interrupt.

Table continues on the next page...

MPIC_MSIRBn field descriptions (continued)

Field	Description
1 SH30	Message sharer 30 has a pending interrupt.
2 SH29	Message sharer 29 has a pending interrupt.
3 SH28	Message sharer 28 has a pending interrupt.
4 SH27	Message sharer 27 has a pending interrupt.
5 SH26	Message sharer 26 has a pending interrupt.
6 SH25	Message sharer 25 has a pending interrupt.
7 SH24	Message sharer 24 has a pending interrupt.
8 SH23	Message sharer 23 has a pending interrupt.
9 SH22	Message sharer 22 has a pending interrupt.
10 SH21	Message sharer 21 has a pending interrupt.
11 SH20	Message sharer 20 has a pending interrupt.
12 SH_19	Message sharer 19 has a pending interrupt.
13 SH_18	Message sharer 18 has a pending interrupt.
14 SH_17	Message sharer 17 has a pending interrupt.
15 SH_16	Message sharer 16 has a pending interrupt.
16 SH_15	Message sharer 15 has a pending interrupt.
17 SH_14	Message sharer 14 has a pending interrupt.
18 SH_13	Message sharer 13 has a pending interrupt.
19 SH_12	Message sharer 12 has a pending interrupt.
20 SH_11	Message sharer 11 has a pending interrupt.
21 SH_10	Message sharer 10 has a pending interrupt.
22 SH_9	Message sharer 9 has a pending interrupt.
23 SH_8	Message sharer 8 has a pending interrupt.

Table continues on the next page...

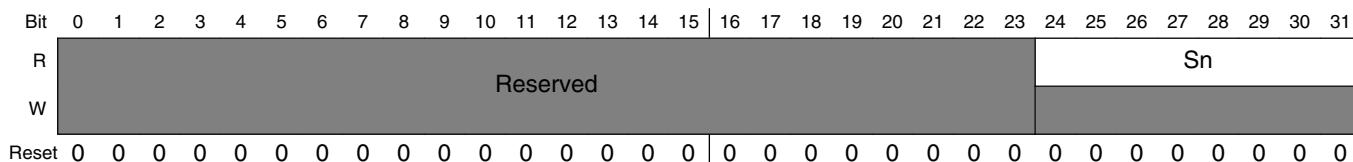
MPIC_MSIRBn field descriptions (continued)

Field	Description
24 SH_7	Message sharer 7 has a pending interrupt.
25 SH_6	Message sharer 6 has a pending interrupt.
26 SH_5	Message sharer 5 has a pending interrupt.
27 SH_4	Message sharer 4 has a pending interrupt.
28 SH_3	Message sharer 3 has a pending interrupt.
29 SH_2	Message sharer 2 has a pending interrupt.
30 SH_1	Message sharer 1 has a pending interrupt.
31 SH_0	Message sharer 0 has a pending interrupt.

**22.3.28 Shared message signaled interrupt status register [Bank B]
(MPIC_MSISRB)**

MSISRB contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIRB has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIRB.

Address: 4_0000h base + 1920h offset = 4_1920h

**MPIC_MSISRB field descriptions**

Field	Description
0–23 -	This field is reserved. Reserved
24–31 Sn	Status n. 0 MSIRn is not active. 1 MSIRn has an active interrupt.

22.3.29 Shared message signaled interrupt index register [Bank B] (MPIC_MSIIRB)

MSIIRB provides the mechanism for setting an interrupt in the MSIRBs. When MSIIRB is written, MSIIRB[SRS] selects the register in which an interrupt bit is to be set. MSIIRB[IBS] selects the shared interrupt field in the selected MSIRB register. MSIIRB is primarily intended to support PCI Express MSIs.

This register is also aliased from MSIIRB at 04_5140h.

Address: 4_0000h base + 1940h offset = 4_1940h

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

MPIC_MSIIRB field descriptions

Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. 000 MSIR 0 001 MSIR 1 010 MSIR 2... 111 MSIR 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved

22.3.30 Shared message signaled interrupt register [Bank C] n (MPIC_MSIRCn)

Each bank of eight MSIRs indicates which of the up to 32 interrupt sources sharing a message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Address: 4_0000h base + 1A00h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SH31	SH30	SH29	SH28	SH27	SH26	SH25	SH24	SH23	SH22	SH21	SH20	SH_19	SH_18	SH_17	SH_16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH_15	SH_14	SH_13	SH_12	SH_11	SH_10	SH_9	SH_8	SH_7	SH_6	SH_5	SH_4	SH_3	SH_2	SH_1	SH_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_MSIRCn field descriptions

Field	Description
0 SH31	Message sharer 31 has a pending interrupt.
1 SH30	Message sharer 30 has a pending interrupt.
2 SH29	Message sharer 29 has a pending interrupt.
3 SH28	Message sharer 28 has a pending interrupt.
4 SH27	Message sharer 27 has a pending interrupt.
5 SH26	Message sharer 26 has a pending interrupt.
6 SH25	Message sharer 25 has a pending interrupt.
7 SH24	Message sharer 24 has a pending interrupt.
8 SH23	Message sharer 23 has a pending interrupt.
9 SH22	Message sharer 22 has a pending interrupt.

Table continues on the next page...

MPIC_MSIRCn field descriptions (continued)

Field	Description
10 SH_21	Message sharer 21 has a pending interrupt.
11 SH_20	Message sharer 20 has a pending interrupt.
12 SH_19	Message sharer 19 has a pending interrupt.
13 SH_18	Message sharer 18 has a pending interrupt.
14 SH_17	Message sharer 17 has a pending interrupt.
15 SH_16	Message sharer 16 has a pending interrupt.
16 SH_15	Message sharer 15 has a pending interrupt.
17 SH_14	Message sharer 14 has a pending interrupt.
18 SH_13	Message sharer 13 has a pending interrupt.
19 SH_12	Message sharer 12 has a pending interrupt.
20 SH_11	Message sharer 11 has a pending interrupt.
21 SH_10	Message sharer 10 has a pending interrupt.
22 SH_9	Message sharer 9 has a pending interrupt.
23 SH_8	Message sharer 8 has a pending interrupt.
24 SH_7	Message sharer 7 has a pending interrupt.
25 SH_6	Message sharer 6 has a pending interrupt.
26 SH_5	Message sharer 5 has a pending interrupt.
27 SH_4	Message sharer 4 has a pending interrupt.
28 SH_3	Message sharer 3 has a pending interrupt.
29 SH_2	Message sharer 2 has a pending interrupt.
30 SH_1	Message sharer 1 has a pending interrupt.
31 SH_0	Message sharer 0 has a pending interrupt.

22.3.31 Shared message signaled interrupt status register [Bank C] (MPIC_MSISRC)

MSISRC contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIRC has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIRC.

Address: 4_0000h base + 1B20h offset = 4_1B20h

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

MPIC_MSISRC field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 Sn	Status n. 0 MSIRn is not active. 1 MSIRn has an active interrupt.

22.3.32 Shared message signaled interrupt index register [Bank C] (MPIC_MSIIRC)

MSIIRC provides the mechanism for setting an interrupt in the MSIRCs. When MSIIRC is written, MSIIRC[SRS] selects the register in which an interrupt bit is to be set. MSIIRC[IBS] selects the shared interrupt field in the selected MSIRC register. MSIIRC is primarily intended to support PCI Express MSIs.

This register is also aliased from MSIIRC at 04_6140h.

Address: 4_0000h base + 1B40h offset = 4_1B40h

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

MPIC_MSIIRC field descriptions

Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. 000 MSIR 0 001 MSIR 1 010 MSIR 2... 111 MSIR 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved

22.3.33 Timer frequency reporting register [Group B] (MPIC_TFRRB)

The TFRRs are written by software to report the clocking frequency of the MPIC timers. Note that although TFRRs are read/write, the MPIC ignores the register values.

Address: 4_0000h base + 20F0h offset = 4_20F0h

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

Reset 0

MPIC_TFRRB field descriptions

Field	Description
0–31 FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source,-either the MPIC input clock () or the frequency of the RTC signal-to user software. See RCW Field Definitions . TFRRx is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency, as set by TCRx[CLKR]. See Timer control register [Group A] (MPIC_TCRA) ."

22.3.34 Global timer current count register [Group B] n (MPIC_GTCCR n)

The GTCCRs contain the current count for each of the four MPIC timers in each of the two groups.

Address: 4_0000h base + 2100h offset + (64d × i), where i=0d to 3d

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

MPIC_GTCCR n field descriptions

Field	Description
0 TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR xn [CI] goes from 1 to 0.
1–31 COUNT	Current count. Decremented while GTBCR xn [CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding GTBCR xn . Cascaded timers are reloaded with either all ones, or the GTBCR xn contents, depending on the value of TCR n [ROVR]. See Timer control register [Group A] (MPIC_TCRA) , " for more details.

22.3.35 Global timer base count register [Group B] n (MPIC_GTBCRBn)

The GTBCRs contain the base counts for each of the four MPIC timers in each of the two groups. This value is reloaded into the corresponding GTCCR xn when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR xn [CI] = 0), the timer generates an interrupt on every timer cycle.

Address: 4_0000h base + 2110h offset + (64d × i), where i=0d to 3d

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

MPIC_GTBCRBn field descriptions

Field	Description
0 CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31 BASE_CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding GTCCR xn and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the GTCCR xn at the next zero crossing of the current count.

22.3.36 Global timer vector/priority register [Group B] n (MPIC_GTVPRBn)

The GTVPRs contain the interrupt vector and the interrupt priority values for the timers. They also contain the mask and activity fields for all the timers.

Address: 4_0000h base + 2120h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSK	A							Reserved						PRIORITY	
W																
Reset																
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									VECTOR							
W																
Reset																

MPIC_GTVPRBn field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.37 Global timer destination register [Group B] n (MPIC_GTDRBn)

The GTDR xn registers control the destination core to which each timer's interrupt is directed. Note that GTDR xn bits can be set independently of each other and any or all can be set for this type of interrupt.

Address: 4_0000h base + 2130h offset + (64d \times i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_GTDRBn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive this interrupt 1 Directs the timer interrupt to processor core 7.
25 P6	Processor core 6. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive this interrupt 1 Directs the timer interrupt to Processor core 6.
26 P5	Processor core 5. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive this interrupt 1 Directs the timer interrupt to processor core 5.
27 P4	Processor core 4. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive this interrupt 1 Directs the timer interrupt to processor core 4.
28 P3	Processor core 3. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive this interrupt 1 Directs the timer interrupt to processors core 3.
29 P2	Processor core 2. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive this interrupt 1 Directs the timer interrupt to processor core 2.

Table continues on the next page...

MPIC_GTDRBn field descriptions (continued)

Field	Description
30 P1	Processor core 1. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive this interrupt 1 Directs the timer interrupt to processor core 1.
31 P0	Processor core 0. Default destination after MPIC is reset. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 0 does not receive this interrupt. 1 Directs the timer interrupt to processor core 0.

22.3.38 Timer control register [Group B] (MPIC_TCRB)

The TCR registers provide various configuration options such as count frequency and roll-over behavior for the timers.

There are two choices for the clock source for the timers: a selectable frequency ratio from the MPIC input clock (), or the RTC signal. TCRs can be cascaded to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer, or one 127-bit timer (within each group).

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field, TCRx[ROVR]. Setting TCRx[ROVR] causes its GTCCR *xn* to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF_FFFF instead of its base count value. Clearing a timer's associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded, the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the TCRx[ROVR] settings.

Address: 4_0000h base + 2300h offset = 4_2300h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				ROVR		Reserved						RTM			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				CLKR		Reserved				CASC					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_TCRB field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–7 ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5-7 correspond to timers 2-0. Note that global timer 3 always reloads with its GTBCR xn . 0 The timer does not roll over. When the count reaches zero, GTCCR xn is reloaded with the GTBCR xn value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, GTCCR xn is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF). 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF). 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF). 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF). 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF). 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF). 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14 -	This field is reserved. Reserved
15 RTM	Real time mode. Specifies the clock source for the MPIC timers. 0 Timer clock frequency is a ratio of the frequency of the MPIC input clock () as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the MPIC timers. If this bit is set, the CLKR field has no meaning.
16–21 -	This field is reserved. Reserved
22–23 CLKR	Clock ratio. Specifies the ratio of the timer frequency to the MPIC input clock (). The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28 -	This field is reserved. Reserved
29–31 CASC	Cascade timers. Specifies the output of particular global timers as input to others. 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

22.3.39 Message register [Group B] n (MPIC_MSGRBn)

The message registers (MSGRx0-MSGRx3) can contain a 32-bit message.

Address: 4 0000h base + 2400h offset + (16d × i), where i=0d to 3d

MPIC MSGRB n field descriptions

Field	Description
0–31 MSG	Message. Contains the 32-bit message data.

22.3.40 Message enable register [Group B] (MPIC_MERB)

The MERs contain the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

Address: 4 0000h base + 2500h offset = 4 2500h

MPIC MERB field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 En	Enable 3-enable 0. Used to enable interrupt generation for MSGRn (where $n = 0\text{--}3$). 0 Interrupt generation for MSGRn disabled. 1 Interrupt generation for MSGRn enabled.

22.3.41 Message status register [Group B] (MPIC_MSRB)

The MSRs contain status bits for each message register. A status bit is set when the corresponding messaging interrupt is active. Writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

Address: 4_0000h base + 2510h offset = 4_2510h

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

MPIC_MSRB field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 Sn	Status 3-status 0. Reports status of messaging interrupt n . Writing a 1 clears this field. 0 Messaging interrupt n is not active. 1 Messaging interrupt n is active.

22.3.42 Performance monitor n mask register 0 (MPIC_PMrMR0)

Each PMnMR0 register is matched with a PM n MR1 through PM n MR5 register. Because each unreserved bit in the mask register vector (PM n MR0/1/.../5) specifies a different interrupt, only one bit in the vector can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

Address: 4_0000h base + 3000h offset + (256d \times i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved							IPI	TIMER	MSG	EXT																					
W																																
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

MPIC_PMrMR0 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved

Table continues on the next page...

MPIC_PMnMR0 field descriptions (continued)

Field	Description
8–11 IPI	Interprocessor interrupts 0-3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15 TIMER	Timer interrupts 0-3 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B). 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19 MSG	Message interrupts 0-3 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B) 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31 EXT	External interrupts IRQ[0:] 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

22.3.43 Performance monitor n mask register 1 (MPIC_PMnMR1)

Address: 4_0000h base + 3020h offset + (256d × i), where i=0d to 3d

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

MPIC_PMnMR1 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 MSIC	Shared message signaled interrupts C 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–23 MSIB	Shared message signaled interrupts B 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
24–31 MSIA	Shared message signaled interrupts A 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

22.3.44 Performance monitor n mask register 2 (MPIC_PMnMR2)

Address: 4_0000h base + 3040h offset + (256d × i), where i=0d to 3d

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

Reset 1

MPIC_PMnMR2 field descriptions

Field	Description
0–31 INT	Internal interrupts 0-31 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

22.3.45 Performance monitor n mask register 3 (MPIC_PMnMR3)

Address: 4_0000h base + 3050h offset + (256d × i), where i=0d to 3d

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

Reset 1

MPIC_PMnMR3 field descriptions

Field	Description
0–31 INT	Internal interrupts 32-63 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

22.3.46 Performance monitor n mask register 4 (MPIC_PMnMR4)

Address: 4_0000h base + 3060h offset + (256d × i), where i=0d to 3d

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

Reset 1

MPIC_PMnMR4 field descriptions

Field	Description
0–31 INT	Internal interrupts 64-95 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

22.3.47 Performance monitor n mask register 5 (MPIC_PMnMR5)

Address: 4_0000h base + 3070h offset + (256d × i), where i=0d to 3d

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

Reset 1

MPIC_PMnMR5 field descriptions

Field	Description
0–15 INT	Internal interrupts 96–111 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–31 -	This field is reserved. Reserved

22.3.48 External interrupt summary register (MPIC_ERQSR)

NOTE

The fields in ERQSR report only the current state of IRQ0-. These fields were designed to work with level-sensitive interrupts; the values returned for edge-sensitive interrupts may be unreliable.

Address: 4_0000h base + 3800h offset = 4_3800h

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

Reset 0

MPIC_ERQSR field descriptions

Field	Description
0–11 EINTn	External interrupts signal 0–11 status. Bit 0 represents external interrupt 0. 0 The corresponding external interrupt signal is not active. 1 The corresponding external interrupt signal is active.
12–31 -	This field is reserved. Reserved

22.3.49 Error interrupt summary register 0 (MPIC_EISR0)

The error interrupt is mapped to internal interrupt number 0.

Address: 4_0000h base + 3900h offset = 4_3900h

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

MPIC_EISR0 field descriptions

Field	Description
0–31 SRCn	Error interrupt source 0-31. Bit 0 represents SRC0. 0 The corresponding error interrupt source is not active. 1 The corresponding error interrupt source is active. See for the error interrupt sources reported in this register.

22.3.50 Error interrupt mask register 0 (MPIC_EIMR0)

The error interrupt is mapped to internal interrupt number 0.

Address: 4_0000h base + 3910h offset = 4_3910h

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

MPIC_EIMR0 field descriptions

Field	Description
0–31 SRCn_MSK	Error interrupt source mask 0-31. Bit 0 represents SRC0_MSK. See Table 23-2 , for the error interrupt sources associated with this register. 0 The corresponding error interrupt source is not masked and will report an error in EISR0. 1 The corresponding error interrupt source is masked and will not report an error in EISR0.

22.3.51 Watchdog status register summary register 0 (MPIC_WSRSR0)

If any of the WRSn bits in this register are set, an internal interrupt is generated. The watchdog interrupt is mapped to internal interrupt number 1.

Address: 4_0000h base + 3A00h offset = 4_3A00h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WRS7[0:1]	WRS6[0:1]	WRS5[0:1]	WRS4[0:1]	WRS3[0:1]	WRS2[0:1]	WRS1[0:1]	WRS0[0:1]								
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_WSRSR0 field descriptions

Field	Description
0–15 -	This field is reserved. Reserved, should be cleared
16–17 WRS7[0:1]	Processor 7 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
18–19 WRS6[0:1]	Processor 6 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
20–21 WRS5[0:1]	Processor 5 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
22–23 WRS4[0:1]	Processor 4 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
24–25 WRS3[0:1]	Processor 3 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
26–27 WRS2[0:1]	Processor 2 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
28–29 WRS1[0:1]	Processor 1 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
30–31 WRS0[0:1]	Processor 0 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.

22.3.52 Critical interrupt summary register 0 (MPIC_CISR0)

Address: 4_0000h base + 3B00h offset = 4_3B00h

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

MPIC_CISR0 field descriptions

Field	Description
0–11 EXTn	External interrupts 0-. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to cintn. 1 The corresponding interrupt is active and is routed to cintn (if the corresponding <i>interrupt level register field</i> is set to cint).
12–31 -	This field is reserved. Reserved

22.3.53 Critical interrupt summary register 1 (MPIC_CISR1)

Address: 4_0000h base + 3B40h offset = 4_3B40h

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

MPIC_CISR1 field descriptions

Field	Description
0–31 INTn	Internal interrupts 0-31. Bit 0 represents INT0. 0 The corresponding interrupt is not active or not routed to cintn. 1 The corresponding interrupt is active and is routed to cintn (if the corresponding <i>interrupt level register field</i> is set to cint).

22.3.54 Critical interrupt summary register 2 (MPIC_CISR2)

Address: 4_0000h base + 3B50h offset = 4_3B50h

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

MPIC_CISR2 field descriptions

Field	Description
0–31 INTn	Internal interrupts 32-63. Bit 0 represents INT32. 0 The corresponding interrupt is not active or not routed to cintn. 1 The corresponding interrupt is active and is routed to cintn (if the corresponding <i>interrupt level register field</i> is set to cint).

22.3.55 Critical interrupt summary register 3 (MPIC_CISR3)

Address: 4_0000h base + 3B60h offset = 4_3B60h

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

MPIC_CISR3 field descriptions

Field	Description
0–31 INTn	Internal interrupts 64-95. Bit 0 represents INT64. 0 The corresponding interrupt is not active or not routed to cintn. 1 The corresponding interrupt is active and is routed to cintn (if the corresponding <i>interrupt level register field</i> is set to cint).

22.3.56 Critical interrupt summary register 4 (MPIC_CISR4)

Address: 4_0000h base + 3B70h offset = 4_3B70h

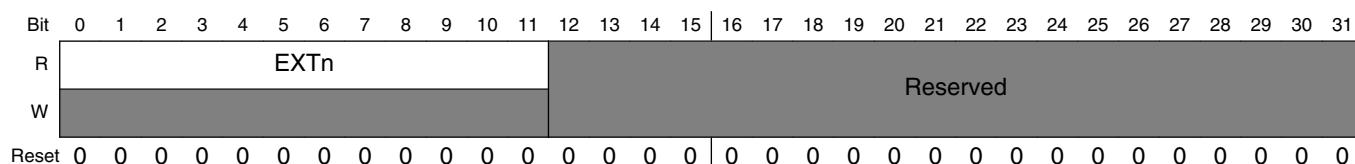
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	INTn															
W																																

MPIC_CISR4 field descriptions

Field	Description
0–15 INTn	Internal interrupts 96–111. Bit 0 represents INT96. 0 The corresponding interrupt is not active or not routed to cintn. 1 The corresponding interrupt is active and is routed to cintn (if the corresponding <i>interrupt level register field</i> is set to cint).
16–31 -	This field is reserved. Reserved

22.3.57 Machine check summary register 0 (MPIC_MCSR0)

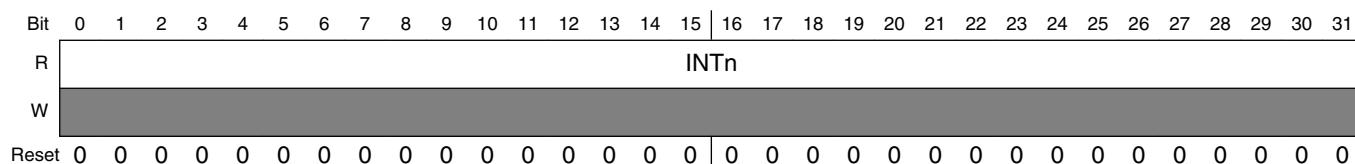
Address: 4_0000h base + 3C00h offset = 4_3C00h

**MPIC_MCSR0 field descriptions**

Field	Description
0–11 EXTn	External interrupts 0-. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to mcpn. 1 The corresponding interrupt is active and is routed to mcpn (if the corresponding <i>interrupt level register field</i> is set to mcp).
12–31 -	This field is reserved. Reserved

22.3.58 Machine check summary register 1 (MPIC_MCSR1)

Address: 4_0000h base + 3C40h offset = 4_3C40h

**MPIC_MCSR1 field descriptions**

Field	Description
0–31 INTn	Internal interrupts 0–31. Bit 0 represents INT0.

MPIC_MCSR1 field descriptions (continued)

Field	Description
	<p>0 Corresponding interrupt is not active or not routed to mcpn.</p> <p>1 The corresponding interrupt is active and is routed to the mcpn (if the corresponding <i>interrupt level register field</i> is set to mcp).</p>

22.3.59 Machine check summary register 2 (MPIC_MCSR2)

Address: 4_0000h base + 3C50h offset = 4_3C50h

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

MPIC_MCSR2 field descriptions

Field	Description
0–31 INTn	<p>Internal interrupts 32-63. Bit 0 represents INT32.</p> <p>0 The corresponding interrupt is not active or not routed to mcpn.</p> <p>1 The corresponding interrupt is active and is routed to mcpn (if the corresponding <i>interrupt level register field</i> is set to mcp).</p>

22.3.60 Machine check summary register 3 (MPIC_MCSR3)

Address: 4_0000h base + 3C60h offset = 4_3C60h

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

MPIC_MCSR3 field descriptions

Field	Description
0–31 INTn	<p>Internal interrupts 64-95. Bit 0 represents INT64.</p> <p>0 The corresponding interrupt is not active or not routed to mcpn.</p> <p>1 The corresponding interrupt is active and is routed to mcpn (if the corresponding <i>interrupt level register field</i> is set to mcp).</p>

22.3.61 Machine check summary register 4 (MPIC_MCSR4)

Address: 4_0000h base + 3C70h offset = 4_3C70h

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

MPIC_MCSR4 field descriptions

Field	Description
0–15 INTn	Internal interrupts 96–111. Bit 0 represents INT96. 0 The corresponding interrupt is not active or not routed to mcpn. 1 The corresponding interrupt is active and is routed to mcpn (if the corresponding <i>interrupt level register field</i> is set to mcp).
16–31 -	This field is reserved. Reserved

22.3.62 IRQ_OUT/Soc Interrupt Event summary register 0 (MPIC_IRQSIESR0)

Address: 4_0000h base + 3D00h offset = 4_3D00h

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

MPIC_IRQSIESR0 field descriptions

Field	Description
0–11 EXTn	External interrupts 0-. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to sien or IRQ_OUT. 1 The corresponding interrupt is active and is routed to sien or IRQ_OUT (if the corresponding <i>interrupt level register field</i> is set to sie [0:6] or IRQ_OUT).
12–31 -	This field is reserved. Reserved

22.3.63 IRQ_OUT/Soc Interrupt Event summary register 1 (MPIC_IRQSIESR1)

Address: 4_0000h base + 3D40h offset = 4_3D40h

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

MPIC_IRQSIESR1 field descriptions

Field	Description
0–31 INTn	<p>Internal interrupts 0-31 status. Bit 0 represents INT0. Bit 31 represents INT31.</p> <p>0 The corresponding interrupt is not active or not routed to sien or IRQ_OUT. 1 The corresponding interrupt is active and is routed to sien or IRQ_OUT (if the corresponding <i>interrupt level register field</i> is set to sie[0:2] or IRQ_OUT).</p>

22.3.64 IRQ_OUT/Soc Interrupt Event summary register 2 (MPIC_IRQSIESR2)

Address: 4_0000h base + 3D50h offset = 4_3D50h

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

MPIC_IRQSIESR2 field descriptions

Field	Description
0–31 INTn	<p>Internal interrupts status 32-63. Bit 0 represents INT32.</p> <p>0 The corresponding interrupt is not active or not routed to sien or IRQ_OUT 1 The corresponding interrupt is active and is routed to sien or IRQ_OUT (if the corresponding <i>interrupt level register field</i> is set to sie[0:2] or IRQ_OUT).</p>

22.3.65 IRQ_OUT/Soc Interrupt Event summary register 3 (MPIC_IRQSIESR3)

Address: 4_0000h base + 3D60h offset = 4_3D60h

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

MPIC_IRQSIESR3 field descriptions

Field	Description
0–31 INTn	Internal interrupts status 64–95. Bit 0 represents INT64. 0 The corresponding interrupt is not active or not routed to sien or IRQ_OUT 1 The corresponding interrupt is active and is routed to sien or IRQ_OUT (if the corresponding <i>interrupt level register field</i> is set to sie[0:2] or IRQ_OUT).

22.3.66 IRQ_OUT/Soc Interrupt Event summary register 4 (MPIC_IRQSIESR4)

Address: 4_0000h base + 3D70h offset = 4_3D70h

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

MPIC_IRQSIESR4 field descriptions

Field	Description
0–15 INTn	Internal interrupts status 96–111. Bit 0 represents INT96. 0 The corresponding interrupt is not active or not routed to sien or IRQ_OUT 1 The corresponding interrupt is active and is routed to sien or IRQ_OUT (if the corresponding <i>interrupt level register field</i> is set to sie[0:2] or IRQ_OUT).
16–31 -	This field is reserved. Reserved

22.3.67 Shared message signaled interrupt index register [Bank A] (alias) (MPIC_MSIIRA_alias)

MSIIRA provides the mechanism for setting an interrupt in the MSIRAs. When MSIIRA is written, MSIIRA[SRS] selects the register in which an interrupt bit is to be set MSIIRA[IBS] selects the shared interrupt field in the selected MSIRA register. MSIIRA is primarily intended to support PCI Express MSIs.

The register at this address is an alias to MSIIRA at 04_1740h.

Address: 4_0000h base + 4140h offset = 4_4140h

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

MPIC_MSIIRA_alias field descriptions

Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. Example values: 000 MSIRA 0 001 MSIRA 1 010 MSIRA 2 111 MSIRA 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. Example values: 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved

22.3.68 Shared message signaled interrupt index register [Bank B] (alias) (MPIC_MSIIRB_alias)

MSIIRB provides the mechanism for setting an interrupt in the MSIRBs. When MSIIRB is written, MSIIRB[SRS] selects the register in which an interrupt bit is to be set MSIIRB[IBS] selects the shared interrupt field in the selected MSIRB register. MSIIRB is primarily intended to support PCI Express MSIs.

The register at this address is an alias to MSIIRB at 04_1940h.

Address: 4_0000h base + 5140h offset = 4_5140h

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

MPIC_MSIIRB_alias field descriptions

Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. Example values: 000 MSIRB 0 001 MSIRB 1 010 MSIRB 2 111 MSIRB 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. Example values: 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved

22.3.69 Shared message signaled interrupt index register [Bank C] (alias) (MPIC_MSIIRC_alias)

MSIIRC provides the mechanism for setting an interrupt in the MSIRC registers. When MSIIRC is written, MSIIRC[SRS] selects the register in which an interrupt bit is to be set. MSIIRC[IBS] selects the shared interrupt field in the selected MSIRC register. MSIIRC is primarily intended to support PCI Express MSIs.

The register at this address is an alias to MSIIRC at 04_1B40h.

Address: 4_0000h base + 6140h offset = 4_6140h

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

MPIC_MSIRC_alias field descriptions

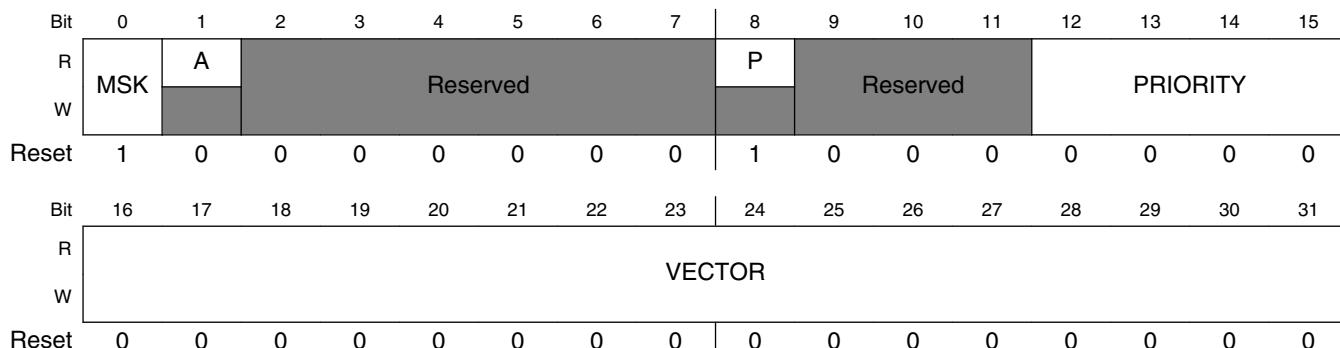
Field	Description
0–2 SRS	Shared interrupt register select. Selects the MSIR to be written. Example values: 000 MSIRC 0 001 MSIRC 1 010 MSIRC 2 111 MSIRC 7
3–7 IBS	Interrupt bit select. Selects the bit to set in the MSIR. Example values: 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29)... 11111 Set field SH31 (bit 0)
8–31 -	This field is reserved. Reserved



22.3.73 Internal interrupt n vector/priority register (MPIC_IIVPRn)

The IIVPRs have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources. These interrupts are all level-sensitive.

Address: 4_0000h base + 1_0200h offset + (32d × i), where i=0d to 111d



MPIC_IIVPRn field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.

Table continues on the next page...

MPIC_IIVPRn field descriptions (continued)

Field	Description
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7 -	This field is reserved. Reserved
8 P	Polarity. All internal interrupts are active-high. NOTE: This bit is read-only and always set specifying the polarity as active high.
9–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.74 Internal interrupt n destination register (MPIC_IIDRn)

The IIDRs have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources. If the IILR is programmed to assert int pins, only one destination bit may be set; otherwise, behavior is undefined. If the IILR is programmed to assert cint or mcp pins multiple destination pins may be set and system behavior must be managed by software.

Address: 4_0000h base + 1_0210h offset + (32d × i), where i=0d to 111d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_IIDRn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt.

Table continues on the next page...

MPIC_IIDR n field descriptions (continued)

Field	Description
	0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27 P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28 P3	Processor core 3. Indicates whether processor core 3 receives the interrupt. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29 P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30 P1	Processor core 1. Indicates whether processor core 1 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31 P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. The default destination is for processor core 0 to receive this interrupt after the MPIC is reset. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.

22.3.75 Internal interrupt n level register (MPIC_IILR n)

The IILRs specify the actual signal to assert to the destination processor.

Address: 4_0000h base + 1_0218h offset + (32d × i), where i=0d to 111d

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

MPIC_IILR_n field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 INTTGT	Interrupt Target. Specifies which signal is asserted The target processor core for <i>int</i> , <i>cint</i> , and <i>msp</i> is specified in the associated IIDRn sie [0:2] and IRQ_OUT_B override the processor specified in the associated IIDR. See SoC Interrupt Event Routing for sie [0:2] routing assignments. All other values are reserved. 0x00 int 0x01 cint 0x02 mcp 0xF0 sie 0 0xF1 sie 1 0xF2 sie 2 0xFF IRQ_OUT external output signal

22.3.76 Messaging interrupt vector/priority register [Group A] n (MPIC_MIVPRA_n)

The MIVPRs have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

Address: 4_0000h base + 1_1600h offset + (32d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSK	A							Reserved				PRIORITY			
W																
Reset																
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									VECTOR							
W																
Reset																

MPIC_MIVPRA_n field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to <i>int</i> , <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.

Table continues on the next page...

MPIC_MIVPRA_n field descriptions (continued)

Field	Description
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.77 Messaging interrupt destination register [Group A] n (MPIC_MIDRAn)

The messaging interrupt destination registers (MIDRs) control the destination for the messaging interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Address: 4_0000h base + 1_1610h offset + (32d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_MIDRAn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.

Table continues on the next page...

MPIC_MIDRAn field descriptions (continued)

Field	Description
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt . 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27 P4	Processor core 4. Indicates whether processor core 4 receives the interrupt . 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28 P3	Processor core 3. Indicates whether processor core 3 receives the interrupt . 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29 P2	Processor core 2. Indicates whether processor core 2 receives the interrupt . 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30 P1	Processor core 1. Indicates whether processor core 1 receives the interrupt . 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31 P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. The default destination is for processor core 0 to receive this interrupt after the MPIC is reset. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.

22.3.78 Messaging interrupt vector/priority register [Group B] n (MPIC_MIVPRBn)

The MIVPRs have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

Address: 4_0000h base + 1_1680h offset + (32d × i), where i=0d to 3d

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

MPIC_MIVPRB_n field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 22-656 .

22.3.79 Messaging interrupt destination register [Group B] n (MPIC_MIDRB_n)

The messaging interrupt destination registers (MIDRs) control the destination for the messaging interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Address: 4_0000h base + 1_1690h offset + (32d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_MIDRB_n field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.

Table continues on the next page...

MPIC_MIDRBn field descriptions (continued)

Field	Description
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt . 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt . 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27 P4	Processor core 4. Indicates whether processor core 4 receives the interrupt . 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28 P3	Processor core 3. Indicates whether processor core 3 receives the interrupt . 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29 P2	Processor core 2. Indicates whether processor core 2 receives the interrupt . 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30 P1	Processor core 1. Indicates whether processor core 1 receives the interrupt . 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31 P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. The default destination is for processor core 0 to receive this interrupt after the MPIC is reset. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.

22.3.80 Shared message signaled interrupt vector/priority register [Bank A] n (MPIC_MSIVPRAn)

The MSIVPRAns have the same fields and format as the GTVPRs.

Address: 4_0000h base + 1_1C00h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MSK	A							Reserved					PRIORITY		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									VECTOR							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_MSIVPRAn field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in End of interrupt register (MPIC_EOI) .

22.3.81 Shared message signaled interrupt destination register [Bank A] n (MPIC_MSIDRAn)

The MSIDRs contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Address: 4_0000h base + 1_1C10h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_MSIDRAn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27 P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28 P3	Processor core 3. Indicates whether processor core 3 receives the interrupt. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29 P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30 P1	Processor core 1. Indicates whether processor core 1 receives the interrupt.

Table continues on the next page...

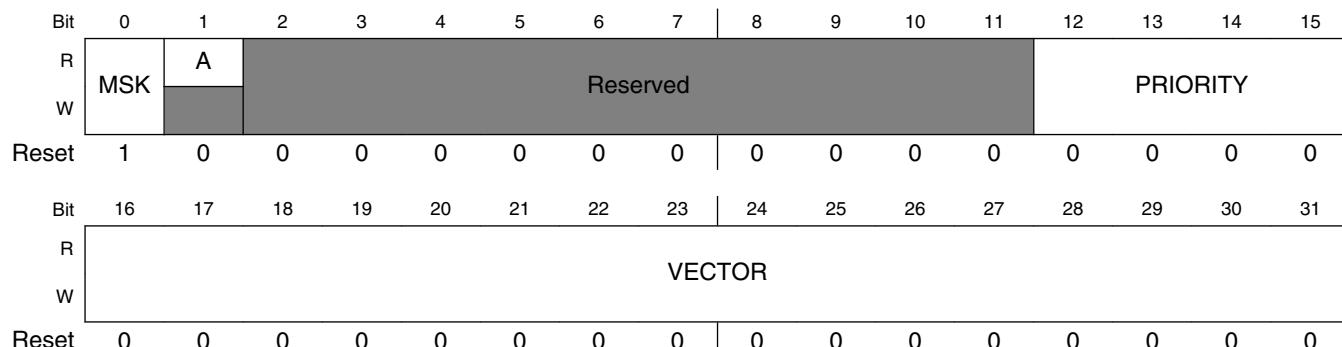
MPIC_MSIDRA_n field descriptions (continued)

Field	Description
	0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31 P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. The default destination is for core 0 to receive this shared message signaled interrupt after the MPIC is reset. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.

22.3.82 Shared message signaled interrupt vector/priority register [Bank B] n (MPIC_MSIVPRB_n)

The MSIVPRBns have the same fields and format as the GTVPRs.

Address: 4_0000h base + 1_1D00h offset + (32d × i), where i=0d to 7d

**MPIC_MSIVPRB_n field descriptions**

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, <i>cint</i> , <i>mcp</i> , <i>sie</i> [0:2], and <i>IRQ_OUT_B</i> . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.

Table continues on the next page...

MPIC_MSIVPRBn field descriptions (continued)

Field	Description
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in End of interrupt register (MPIC_EOI) .

22.3.83 Shared message signaled interrupt destination register [Bank B] n (MPIC_MSIDRBn)

The MSIDRs contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Address: 4_0000h base + 1_1D10h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_MSIDRBn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27 P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28 P3	Processor core 3. Indicates whether processor core 3 receives the interrupt.

Table continues on the next page...

MPIC_MSIDRBn field descriptions (continued)

Field	Description
	0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29 P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30 P1	Processor core 1. Indicates whether processor core 1 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31 P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. The default destination is for core 0 to receive this shared message signaled interrupt after the MPIC is reset. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.

22.3.84 Shared message signaled interrupt vector/priority register [Bank C] n (MPIC_MSIVPRCn)

The MSIVPRCns have the same fields and format as the GTVPRs.

Address: 4_0000h base + 1_1E00h offset + (32d × i), where i=0d to 7d

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

MPIC_MSIVPRCn field descriptions

Field	Description
0 MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to int, cint , mcp , sie [0:2], and IRQ_OUT_B . 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.

Table continues on the next page...

MPIC_MSIVPRCn field descriptions (continued)

Field	Description
1 A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to int. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11 -	This field is reserved. Reserved
12–15 PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to int.
16–31 VECTOR	Vector (Affects only interrupts routed to int). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in End of interrupt register (MPIC_EOI) .

22.3.85 Shared message signaled interrupt destination register [Bank C] n (MPIC_MSIDRCn)

The MSIDRs contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Address: 4_0000h base + 1_1E10h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MPIC_MSIDRCn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25 P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26 P5	Processor core 5. Indicates whether processor core 5 receives the interrupt.

Table continues on the next page...

MPIC_MSIDRC n field descriptions (continued)

Field	Description
	<p>0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.</p>
27 P4	<p>Processor core 4. Indicates whether processor core 4 receives the interrupt.</p> <p>0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.</p>
28 P3	<p>Processor core 3. Indicates whether processor core 3 receives the interrupt .</p> <p>0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.</p>
29 P2	<p>Processor core 2. Indicates whether processor core 2 receives the interrupt.</p> <p>0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.</p>
30 P1	<p>Processor core 1. Indicates whether processor core 1 receives the interrupt.</p> <p>0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.</p>
31 P0	<p>Processor core 0. Indicates whether processor core 0 receives the interrupt.</p> <p>The default destination is for core 0 to receive this shared message signaled interrupt after the MPIC is reset.</p> <p>0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0.</p>

22.3.86 Interprocessor interrupt dispatch register 0 (MPIC_CPU*n*_IPIDR0)

The following figure shows the per-CPU global access IPIDR0. Because external bus masters can access these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device.

Address: 4_0000h base + 2_0040h offset + (4096d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W									P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_CPU*n*_IPIDR0 field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25 P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26 P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27 P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4

Table continues on the next page...

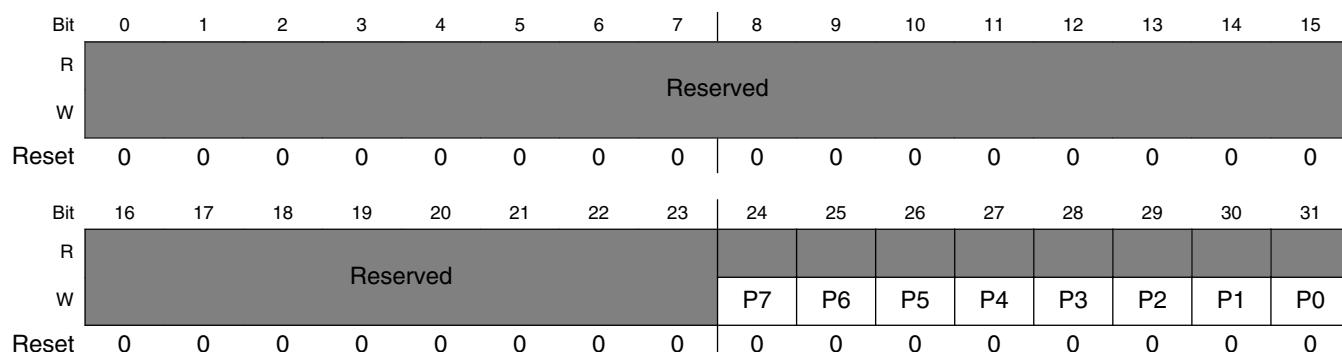
MPIC_CPU*n*_IPIDR0 field descriptions (continued)

Field	Description
28 P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29 P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2
30 P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31 P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

22.3.87 Interprocessor interrupt dispatch register 1 (MPIC_CPU*n*_IPIDR1)

The following figure shows the per-CPU global access IPIDR1. Because external bus masters can access these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device.

Address: 4_0000h base + 2_0050h offset + (4096d × i), where i=0d to 7d



MPIC_CPU n _IPIDR1 field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25 P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26 P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27 P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4
28 P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29 P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2
30 P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31 P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

22.3.88 Interprocessor interrupt dispatch register 2 (MPIC_CPU_n_IPIDR2)

The following figure shows the per-CPU global access IPIDR2. Because external bus masters can access these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device.

Address: 4_0000h base + 2_0060h offset + (4096d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W									P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MPIC_CPU_n_IPIDR2 field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25 P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26 P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27 P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4

Table continues on the next page...

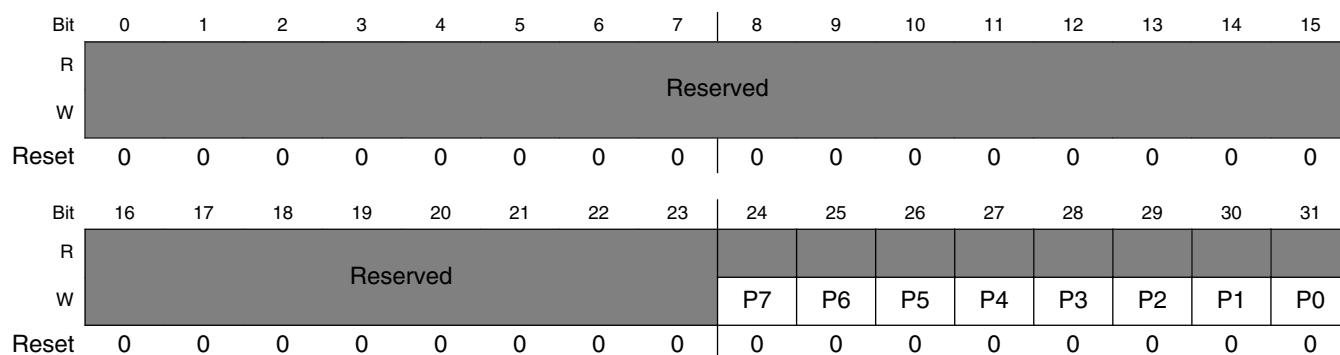
MPIC_CPU n _IPIDR2 field descriptions (continued)

Field	Description
28 P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29 P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2
30 P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31 P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

22.3.89 Interprocessor interrupt dispatch register 3 (MPIC_CPU n _IPIDR3)

The following figure shows the per-CPU global access IPIDR3. Because external bus masters can access these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device.

Address: 4_0000h base + 2_0070h offset + (4096d × i), where i=0d to 7d



MPIC_CPU n _IPIDR3 field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25 P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26 P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27 P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4
28 P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29 P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2
30 P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31 P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

22.3.90 Current task priority register (MPIC_CPU n _CTPR)

There is one CTPR per processor core on this device.

NOTE

CTPR has meaning only for interrupts routed to *int*.

Software must write the priority of the current processor core task in the CTPR for each core. The MPIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to that core if the following apply:

- The interrupt is not masked.
- The priority of the interrupt is higher than the values in the corresponding CTPR[TASKP] and ISR.

Address: 4_0000h base + 2_0080h offset + (4096d × i), where i=0d to 7d

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

MPIC_CPU*n*_CTPR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 TASKP	Task priority. Indicates the threshold that individual interrupt priorities must exceed for the interrupt request to be serviced. Task priority is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal. 0000-1111 xVPRn[PRIORITY] must exceed this value for the interrupt request to be serviced. Note the following special cases: 0000 Lowest priority. All interrupts except those whose priority are 0 can be serviced. 1111 Highest priority. No interrupts are signaled to that processor core. Hardware selects this value on a device hard reset or when the corresponding PIR[Pn] is set.

22.3.91 Who am I register (MPIC_CPU*n*_WHOAMI)

The processor core WHOAMI *n* register can be read by a processor core to determine its physical connection to the MPIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

Address: 4_0000h base + 2_0090h offset + (4096d × i), where i=0d to 7d

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

MPIC_CPU n _WHOAMI field descriptions

Field	Description																
0–26 -	This field is reserved. Reserved																
27–31 ID	Returns the ID of the processor core reading this register. NOTE: ID = 11111 indicates an illegal processor ID <table> <tr><td>00000</td><td>Processor core 0</td></tr> <tr><td>00001</td><td>Processor core 1</td></tr> <tr><td>00010</td><td>Processor core 2</td></tr> <tr><td>00011</td><td>Processor core 3</td></tr> <tr><td>00100</td><td>Processor core 4</td></tr> <tr><td>00101</td><td>Processor core 5</td></tr> <tr><td>00110</td><td>Processor core 6</td></tr> <tr><td>00111</td><td>Processor core 7</td></tr> </table>	00000	Processor core 0	00001	Processor core 1	00010	Processor core 2	00011	Processor core 3	00100	Processor core 4	00101	Processor core 5	00110	Processor core 6	00111	Processor core 7
00000	Processor core 0																
00001	Processor core 1																
00010	Processor core 2																
00011	Processor core 3																
00100	Processor core 4																
00101	Processor core 5																
00110	Processor core 6																
00111	Processor core 7																

22.3.92 Interrupt acknowledge register (MPIC_CPU n _IACK)

NOTE

IACK has meaning only for interrupts routed to *int* in mixed mode (GCR[M] = 01) and should not be accessed for interrupts routed to *cint*, *mcp*, *sie* or *IRQ_OUT_B*.

NOTE

The external proxy facility mode (GCR[M] = 11), eliminates the need for the core to read the IACK register as the vector is automatically passed to the core's external proxy register (EPR) or guest external proxy register (GEPR).

In systems based on processors that implement the Power architecture, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK). Each processor core has an IACK register assigned to it. Reading IACK returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the corresponding interrupt pending register (IPR) is cleared for edge-sensitive interrupts.
- The corresponding in-service register (ISR) is updated.
- The corresponding int output signal from the MPIC is negated.

Reading IACK when no interrupt is pending returns the spurious vector value, as described in [Spurious Vector Register](#) ."

Address: 4_0000h base + 2_00A0h offset + (4096d × i), where i=0d to 7d

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

MPIC_CPU*n*_IACK field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only). This field is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal.

22.3.93 End of interrupt register (MPIC_CPU*n*_EOI)

NOTE

EOI has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint*, *mcp*, *sie* or *IRQ_OUT_B*.

Each core is assigned an EOI register. Writing to EOI signals the end of processing for the highest-priority interrupt (routed to *int*) currently in service. It also updates the corresponding ISR *n* by retiring the highest priority interrupt. Data values written to EOI are ignored, and zero is assumed.

Address: 4_0000h base + 2_00B0h offset + (4096d × i), where i=0d to 7d

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

MPIC_CPU*n*_EOI field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 EOI_CODE	EOI_CODE 0000 (write only)

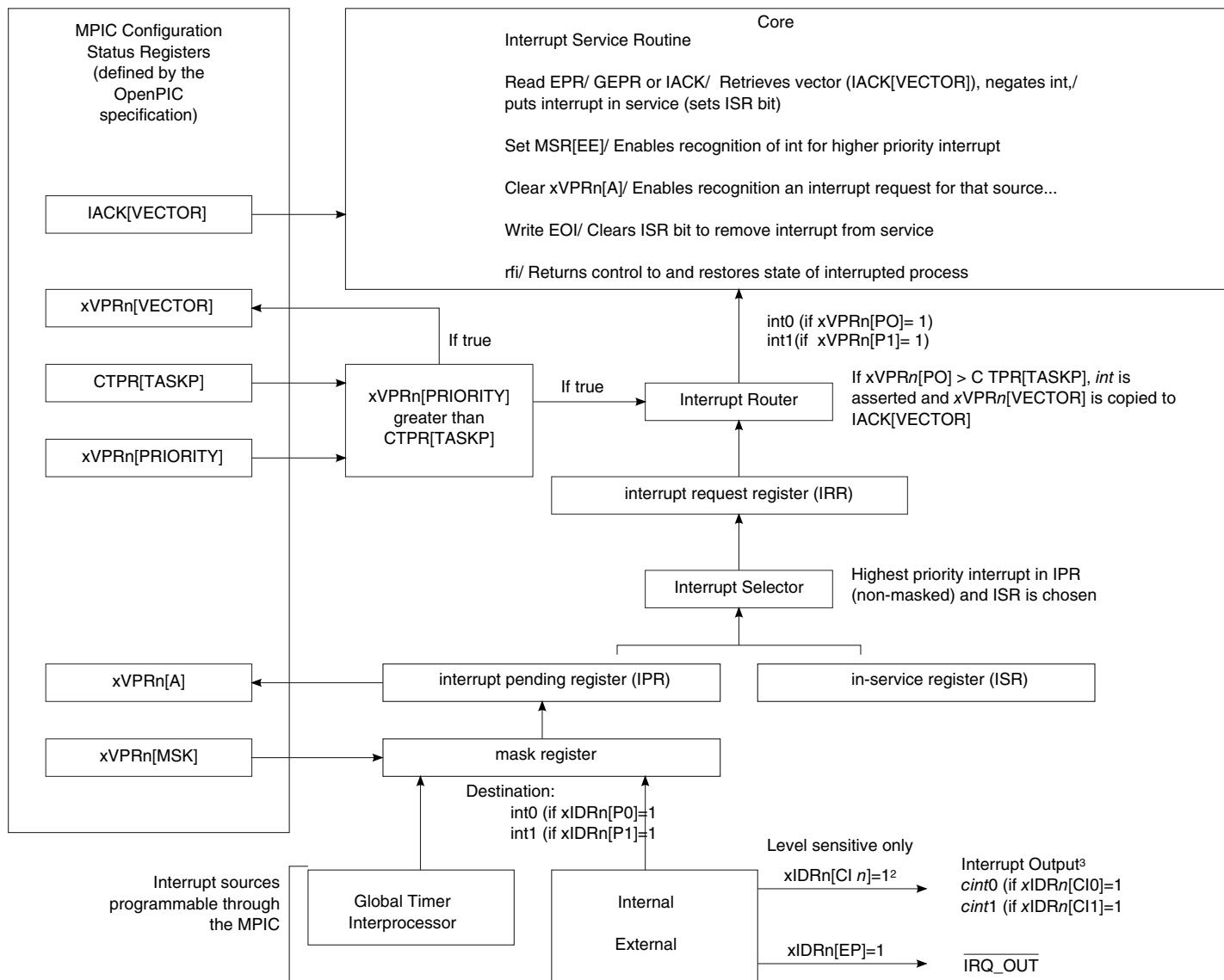
22.4 MPIC functional description

This section describes the functions of the MPIC.

22.4.1 Flow of interrupt control

This figure shows the flow of interrupts directed by the MPIC to the *int*, *cint*, *sie* or and IRQ_OUT_B outputs. Note that this diagram describes a conceptual model of an MPIC on a single processor. This logic is replicated for each implemented processor. This conceptual diagram does not fully represent all internal circuitry of the implementation

This figure focuses especially on the OpenPIC-defined logic and shows how the MPIC controls interrupt requests that target the *int* signal. The flow in this figure is from the bottom to the top, and shows at the bottom how the destination register associated with each source determines the path.



¹ If *cint*, *sie* or *IRQ_OUT* is the destination, EIVPRn[S] must be set to configure the source as level sensitive.

² If multiple destination register bits are set, MPIC behavior is undefined.

³ Although setting *CLn* directs the interrupt request to the critical interrupt output (*cintn*), integrated logic may connect this signal to a different interrupt input to the core.

Figure 22-656. MPIC Interrupt processing flow diagram for each core (n)

22.4.1.1 Interrupts Routed to *cint*, *mcp*, *sie*, or *IRQ_OUT_B*

Interrupt requests routed to *cint*, *mcp*, *sie*, or *IRQ_OUT_B* bypass the logic that is dedicated to interrupt sources that compete for *int*.

NOTE

Because interrupt sources routed to *cint*, *sie*, IRQ_OUT_B and internal interrupt sources routed to *mcp* must be level sensitive, EIVPR[S] has no effect. External interrupt sources routed to *mcp* can be level-sensitive or edge-triggered since the core always uses edge detection logic. Consequently, EIVPR[S] has no effect in this case as well. See [External interrupt n \(IRQn\) vector/priority register \(MPIC_EIVPR\)](#).

Because these interrupts bypass the OpenPIC logic, it is especially important that handlers do not read IACK. Doing so causes a spurious interrupt. Likewise, they should not write EOI.

22.4.1.2 Interrupts routed to *int*

As shown in [Figure 22-656](#), the MPIC receives interrupt requests from external and internal sources and from within the MPIC itself. As [Figure 22-656](#) shows, all of these interrupt sources can be routed to *int*; the global timer and timer processor interrupts can be directed only to *int*.

The sources' mask bits (*xVPRn[MSK]*) are tracked in the internal mask register. If a source's MSK bit is set, the mask register prevents the MPIC from asserting *int* (or *cint*, *mcp*, *sie[0:2]*, IRQ_OUT_B) on its behalf.

Unmasked interrupt requests are captured in the interrupt pending register (IPR), an internal interrupt summary register with a bit for each source. If an interrupt request is multicast, a bit is set in the IPR for each targeted processor. Although the IPR cannot be read by software, when an IPR bit is set, the corresponding source's activity bit (*xVPRn[A]*) is automatically set.

The interrupt selector monitors the IPR and the in-service register (ISR), which tracks previously taken interrupts that were superseded by a higher-priority interrupt before the interrupt handler finished. The interrupt selector recognizes the highest priority unmasked interrupt request and latches it into the interrupt request register (IRR). The source's vector (*xVPRn[VECTOR]*) is automatically passed to the core's external proxy register (EPR) or guest external proxy register (GEPR) in external proxy facility mode (GCR[M] = 11) or it is copied to IACK[VECTOR] in mixed mode (GCR[M] = 01).

If the priority ($xVPRn[\text{PRIORITY}]$) of an interrupt latched in the IRR is higher than the value in the target processor's CTPR[TASKP], the interrupt router asserts the external interrupt signal (*int*), causing that processor core to vector to its external interrupt handler.

In mixed mode (GCR[M] = 01), the interrupt handler must acknowledge the interrupt by explicitly reading the corresponding IACKregister, described in [Interrupt acknowledge register \(MPIC_IACK\)](#). The MPIC interprets this read as an interrupt acknowledge (IACK) cycle. In external proxy facility mode (GCR[M] = 11, the core automatically generates the IACK cycle (when MSR[EE] = 1) to the MPIC, although from system software perspective, the core does not acknowledge the interrupt until the external interrupt is taken.

NOTE

In external proxy facility mode, interrupt handlers should retrieve the vector from the core's EPR (or GEPR) and not read the IACK register. Reading the IACK registers in external proxy facility mode causes a spurious interrupt.

The IACK cycle not only returns the source's vector, it also negates the *int* signal to the processor (making it possible for a higher priority interrupt to assert *int*) and sets the source's bit in the ISR, indicating that this interrupt has been put in service. An interrupt remains in service from the time until the corresponding end-of-interrupt register (EOI) is written, generating what the MPIC considers an EOI signal.

This figure shows required elements in the interrupt handler.

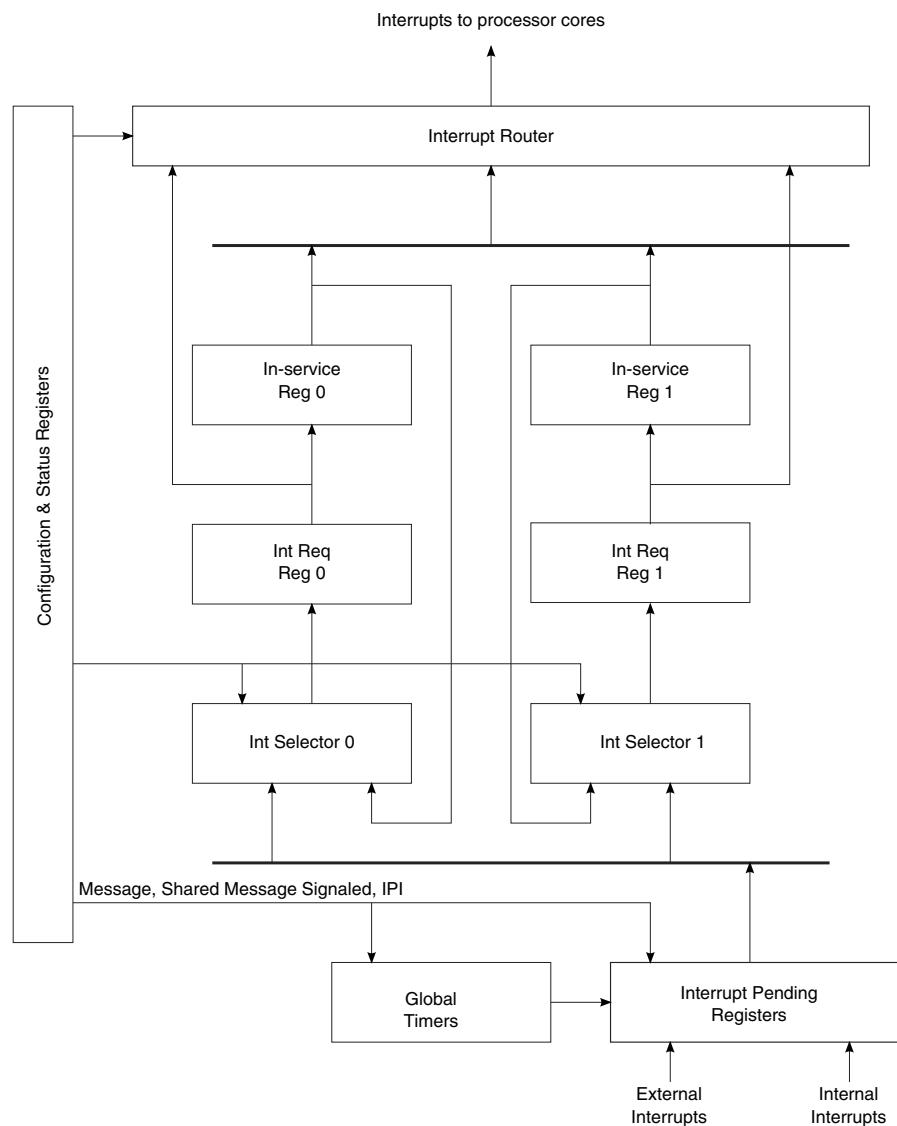


Figure 22-657. Required elements in the interrupt handler

22.4.1.2.1 Nesting of Interrupts

While an interrupt is being handled, if an interrupt request arrives with a higher $xVPRn[PRIORITY]$ value, the interrupt being serviced is superseded. As described in [Interrupts routed to *int*](#), the MPIC asserts *int*, and the newer, higher priority interrupt is handled. This happens even if software, as part of its interrupt service routine, updates the corresponding CTPR with a lower value.

Thus, although several interrupts can be in service simultaneously (and tracked by the ISR), the highest priority interrupt by that processor is always the one actively handled. When the interrupt routine completes, it performs a write EOI cycle, a side effect of which is to take the current highest priority interrupt out of service (removes it from the

ISR). At this point, the interrupt selector chooses the new highest priority interrupt request, and, assuming CTPR[TASKP] has not been updated to a value higher than the new interrupt, the MPIC asserts *int* on its behalf.

The next write EOI cycle takes the current highest priority interrupt out of service. An interrupt with lower priority than those in service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

22.4.1.2.2 Interrupt Source Priority

Each interrupt source routed to *int* is assigned a value through its *xVPRn[PRIORITY]* field. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the source is greater than the destination processor's CTPR[TASKP]. Therefore, setting *xVPRn[PRIORITY]* to zero inhibits that interrupt. Likewise, setting TASKP to 15 prevents the MPIC from delivering interrupts to that core through the *int* signal. Note that this is the reset value, preventing the MPIC from asserting *int* before the MPIC is configured.

The MPIC services simultaneous interrupts occurring with the same priority according to the following order:

1. MSGA0-MSGA3
2. MSGB0-MSGB0
3. MSIA0-MSIA7
4. MSIB0-MSIB7
5. MSIC0-MSIC7
6. IPI0-IPI3
7. Group A timer0-timer3
8. Group B timer0-timer3
9. IRQ[0:]
10. Internal0-internal 111

For example, if MSGA0, MSGA2, and IPI0 are all assigned the same priority and receive simultaneous interrupts, they are serviced in the following order:

1. MSGA0
2. MSGA2
3. IPI0

22.4.1.2.3 Interrupt Acknowledge

In mixed mode, when the MPIC causes *int* to be asserted, the external interrupt service routine acknowledges the request by reading that core's IACKregister, which at this point holds the 16-bit vector value for the interrupt source that generated the request. This is the value programmed in that source's *xVPRn[VECTOR]*. Reading IACK has the following effects:

- The *int* signal for that core is negated, making it possible for another interrupt source to signal an external interrupt to the core, and more particularly, allowing the MPIC to signal a higher-priority interrupt, as described in [Nesting of Interrupts](#).
- The source that caused that resource is represented in the internal in-service register (ISR).

The interrupt is then considered to be in service. It remains so until the processor core performs a write to the corresponding EOI. Writing to EOI is referred to as an EOI cycle.

In external proxy facility mode, the interrupt is automatically acknowledged with the same effects as reading the IACK register. Detailed description of the external proxy facility is described in the *e500mc Core Reference Manual*.

22.4.1.2.4 Spurious Vector Generation

Under certain circumstances, the MPIC has no valid vector to return to a processor core during an interrupt acknowledge cycle. In these cases, the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch:

- *int* (ipi_int_proc) is asserted in response to an externally or internally-sourced interrupt which is activated with level-sensitive logic, and the asserted level is negated before the interrupt is acknowledged.
- *int* (ipi_int_proc) is asserted for an interrupt source that is later masked (using the mask bit in the vector/priority register corresponding to that source) before the interrupt is acknowledged.
- *int* (ipi_int_proc) is asserted for an interrupt source that is later masked by an increase in the task priority level before the interrupt is acknowledged.
- An interrupt acknowledge cycle is performed by the processor core in spite of the fact that the MPIC has not asserted int (ipi_int_proc).

In all cases, a spurious vector is returned only if no pending interrupt has sufficient priority to signal an interrupt, otherwise, the vector for that interrupt source is returned.

NOTE

EOI should not be written in response to a spurious vector. Otherwise, a previously accepted interrupt might be cleared unintentionally.

22.4.2 Global- and private-access per-CPU registers

Although most MPIC registers have one address, some are replicated for each processor core in a multiprocessor device. For such registers, each core accesses its separate registers using the same address, the address decoding being sensitive to the processor core; these registers are called the per-CPU global-access registers. A copy of the per-CPU registers is available to each processor core at the same physical address; these registers are called the per-CPU private-access registers. The private-access address acts like an alias to a core's per-CPU global-access register.

For example, CTPR is the per-CPU private-access current task priority register and CPU n _CTPR are the per-CPU global-access current task priority registers.

- When Core0 accesses CTPR at 04_0080h, the MPIC aliases CPU0_CTPR at 06_0080h.
- When Core1 accesses CTPR at 04_0080h, the MPIC aliases CPU1_CTPR at 06_1080h.
- When Core2 accesses CTPR at 04_0080h, the MPIC aliases CPU2_CTPR at 06_2080h.
- When Core3 accesses CTPR at 04_0080h, the MPIC aliases CPU3_CTPR at 06_3080h.
- When Core4 accesses CTPR at 04_0080h, the MPIC aliases CPU4_CTPR at 06_4080h.
- When Core5 accesses CTPR at 04_0080h, the MPIC aliases CPU5_CTPR at 06_5080h.
- When Core6 accesses CTPR at 04_0080h, the MPIC aliases CPU6_CTPR at 06_6080h.
- When Core7 accesses CTPR at 04_0080h, the MPIC aliases CPU7_CTPR at 06_7080h.

The registers in the following table are called per-CPU registers because they are duplicated for each core in a multicore device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor core that initiates the transaction to determine the set of per-CPU registers to access.

Table 22-657. Per-CPU Register Mappings

Register name	Private-access		Global-access ¹	
	Register	Address	Register	Address
Interprocessor interrupt 0 dispatch register	IPIDR0	04_0040h	CPU n _IPI DR0	06_n040h

Table continues on the next page...

Table 22-657. Per-CPU Register Mappings (continued)

Register name	Private-access		Global-access ¹	
	Register	Address	Register	Address
Interprocessor interrupt 1 dispatch register	IPIDR1	04_0050h	CPU n _IPI DR1	06_n050h
Interprocessor interrupt 2 dispatch register	IPIDR2	04_0060h	CPU n _IPI DR2	06_n060h
Interprocessor interrupt 3 dispatch register	IPIDR3	04_0070h	CPU n _IPI DR3	06_n070h
Current task priority register	CTPR	04_0080h	CPU n _CT PR	06_n080h
Who am I register	WHOAMI	04_0090h	CPU n _WH OAMI	06_n090h
Interrupt acknowledge register	IACK	04_00A0h	CPU n _IAC K	06_n0A0h
End of interrupt register	EOI	04_00B0h	CPU n _EOI	06_n0B0h

1. n is the corresponding core number

The private-access addresses appear in the memory map at the same offset for every processor. This duplication allows user code to execute correctly in a multiprocessor environment without needing to know which core it is running on.

22.4.3 Interprocessor Interrupts

Processors can generate interprocessor interrupts that target any processor. Interrupts are initiated by writing one or several of the Pn bits in an interprocessor interrupt dispatch register (IPIDR0-IPIDR3) of one of the four IPI channels. Note that when initiating an interrupt event to more than one processor (by setting more than 1 Pn bit in the IPIDR), each recipient processor must acknowledge, take delivery and EOI that interrupt. That is, the interrupt is delivered to each processor as a unique copy. Also note that if subsequent interprocessor interrupts from a given channel to a given target processor are initiated before the first is acknowledged, only one interrupt is generated. Finally, a self interrupt occurs when a core dispatches an interprocessor interrupt event to itself.

22.4.4 Messaging Interrupts

22.4.4.1 Message Interrupts

The four MSGRs, described in [Message register \[Group A\] n \(MPIC_MSGRAn\)](#), can be used to send 32-bit messages to one or more processors. A messaging interrupt is generated by writing an MSGR if the corresponding MER bit is set and the interrupt is not masked. Reading a MSGR or writing a 1 to the status bit clears the interrupt.

22.4.4.2 Shared Message Signaled Interrupts

There are three sets of eight MSIRs (one set for each bank A, B, and C), described in [Shared message signaled interrupt register \[Bank A\] n \(MPIC_MSIRAn\)](#), that indicate which of the interrupt sources sharing the MSI register have pending interrupts. Up to 32 sources can share any individual MSI register. A shared message signaled interrupt is generated by writing to Shared Message Signaled Interrupt Index Register (MSIIR) fields SRS and IBS. This register is primarily intended to support PCI Express MSIs. The two fields, SRS and IBS, select one of the eight registers in which an interrupt bit is to be set and the bit in the selected register, respectively. The corresponding interrupt needs to be unmasked for the interrupt to happen. A read to MSI register will clear the interrupt.

22.4.5 PCI Express INTx

An INTx interrupt is signaled from the PCIe controller when it receives an in-band INTx message from a PCIe end point. These interrupts replace the out-of-band interrupts that historically were connected to the IRQ pins. Because the INTx interrupts and IRQ interrupts are generally mutually exclusive, the INTx signals are logically combined with the IRQ signals so that they share the same OpenPIC interrupt. See [PCI Express INTx/External Interrupt IRQn Sharing](#) for the association of INTx signals to IRQ signals.

22.4.6 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the internal MPIC timers. These 8 timers are organized as two independent groups (A and B) of four timers each (0-3). The timers can be individually programmed to generate a processor core interrupt when they count down to zero and can be used to generate regular periodic interrupts. The timers have the following configuration and control registers:

- Global timer current count register (GTCCRA0-GTCCRA3, GTCCRB0-GTCCRB3)
- Global timer base count register (GTBCRA0-GTBCRA3, GTBCRB0-GTBCRB3)

- Global timer vector-priority register (GTVPRRA0-GTVPRRA3, GTVPRRB0-GTVPRRB3)
- Global timer destination register (GTDRA0-GTDRA3, GTDRB0-GTDRB3)

The timer frequency should be written to the TFRR xn , described in [Timer frequency reporting register \[Group A\] \(MPIC_TFRR\)](#).

Timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in service, the subsequent interrupt is lost.

The timer control register (TCR) provides users with the ability to create timers larger than the 31-bit global timers. The timer frequency can also be changed by setting the appropriate TCR fields, as described in [Timer control register \[Group A\] \(MPIC_TCRA\)](#).

Note that when global timer interrupts are set to multicast (interrupt more than one processor by setting more than 1 Pn bit in the GTDR xn), each recipient processor must acknowledge, take delivery and EOI that interrupt. That is, the interrupt is delivered to each processor as a unique copy.

22.4.7 Resets

This section describes the behavior of the MPIC at reset and the MPIC's ability to initiate processor resets.

22.4.7.1 Resetting the MPIC

The MPIC is reset by a device power-on reset (POR) or by software that sets GCR[RST], either of which causes the following:

- All pending and in-service interrupts are cleared.
- All interrupt mask bits are set.
- Polarity, sense, external signal, critical interrupt, and activity fields are reset to default values.
- PIR, TFRR, TCR, MER x , MSR x , and MSGRx0-MSGRx3 are cleared.
- MSG and timer destination fields are set.
- The interprocessor dispatch registers are cleared.
- All timer base count values are reset to zero with count inhibited.
- CTPR[TASKP] is reset to 0x000F, disabling delivery of interrupts that target int .
- The spurious interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.

- The MPIC defaults to the pass-through mode (GCR[M] = 00).
- All other registers remain at their pre-reset programmed values.

GCR[RST] is automatically cleared when the reset sequence is complete.

22.4.7.2 Processor Core Initialization

A software reset can be routed to any of the cores by writing to the processor core initialization register (PIR). This causes the assertion of the *hresetn* output signals from the MPIC. When this occurs, the corresponding CTPR also gets written to 0x000F to prevent delivery of any interrupts to *intn*.

22.4.8 Processor Core NMI

The processor core non-maskable interrupt (PNMIR) provides a way for software on another device to signal a non-maskable interrupt event to the processor cores by writing to PNMIR register. A *coren_nmi* signal to the cores is asserted when a one is written to the corresponding PNMIR field; these signals are held active until a zero is written to those same bits. This asserts the *core_nmi* input of the corresponding core.

22.5 Initialization/Application Information

This section contains initialization and application information for the MPIC.

22.5.1 Programming Guidelines

The following subsections contain information about programming MPIC registers.

22.5.1.1 MPIC Registers

Most MPIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- Interprocessor dispatch and EOIregisters, which return zeros on reads.
- Activity bits (A) of the vector/priority registers reflect the status of the corresponding interrupt source.

- IACK, which returns the vector of the highest priority pending interrupt or the spurious vector (SVR[VECTOR]) if none is pending.
- Reserved fields always return 0.

When the MPIC is in mixed mode (GCR[M] = 01), the following guidelines are recommended:

- All MPIC registers must be located in a cache-inhibited, guarded area (configured through the core's MMU).
- The MPIC portion of the address map must be set up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.
2. Clear CTPR(CTPR = All zeros).
3. Program the MPIC to mixed mode by setting GCR[M] = 01.
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
 - Load counter with FRR[NIRQ].
 - While counter > 0, read IACK and write EOI to guarantee all the IPR and ISR bits are cleared.
6. Set the processor core CTPR values to the desired values.

Depending on the interrupt system configuration, the MPIC may generate spurious interrupts to clear interrupts latched during power-up. A spurious or non-spurious vector is returned for an interrupt acknowledge cycle in this case. See the programming note below for the non-spurious case.

NOTE

Because the default polarity/sense for external interrupts is edge-sensitive, and edge-sensitive interrupts are not cleared until they are acknowledged, it is possible for the MPIC to store spurious edges detected during power-up as pending external interrupts. If software permanently configures an external interrupt source to be edge-sensitive, it may receive the vector for the interrupt source and not a spurious interrupt vector when software clears the mask bit. This can occur once for any edge-sensitive interrupt when its mask bit is first cleared and the MPIC is in mixed mode. To avoid a false interrupt for this case, software can clear the IPR of these spurious edge detections by first configuring the polarity/sense of external interrupt sources to be level-sensitive: high-level if the input is a positive-edge source and low-level if it is a negative-edge source (while the

mask bit remains set). After this is complete, configuring the external interrupt source as edge-sensitive does not cause a false interrupt.

22.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense, or destination of an active (unmasked) interrupt source, the following steps should be taken:

1. Mask the source using the mask (MSK) bit in the vector/priority register.
2. Wait for the activity (A) bit for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.

Note that changing the destination from *int* to *cint*, *mcp*, *sie* or *IRQ_OUT_B* makes the A, and *PRIORITY* fields meaningless; however, the MSK field does affect *cint*, *mcp*, *sie[0:2]*, and *IRQ_OUT_B*.

22.5.1.3 Requirement

After issuing a reset to the core, there must be 15 EOIs before any new interrupt is sent to the core to ensure no interrupt is stuck in an in-service state. Note that each core has its own EOI register and this procedure must be performed for each core that has been reset.

Chapter 23

Interrupt Assignments

23.1 Introduction

This chapter describes the programmable interrupt controller (PIC) internal interrupt assignments for the P4080. These are the on-chip interrupt sources from peripheral logic within the integrated device.

23.2 Internal Interrupt Sources

[Table 23-1](#) shows the assignments of the internal interrupt sources and how they are mapped to the registers that control them. Only the internal interrupts used are listed; that is, the numbers are not consecutive.

Table 23-1. P4080 Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source
0	ORed Error Interrupt-See Table 23-2 for individual assignments
1	WRS
...	-
8	PAMU (access violations)
9	eLBC NOTE: The eLBC signals both event and error interrupts using this interrupt. Error interrupts for the eLBC are not sent to the ORed Error Interrupt.
...	-
12	DMA 1 channel 0
13	DMA 1 channel 1
14	DMA 1 channel 2
15	DMA 1 channel 3
16	DMA 2 channel 0
17	DMA 2 channel 1
18	DMA 2 channel 2

Table continues on the next page...

Table 23-1. P4080 Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
19	DMA 2 channel 3
20	DUART 1
21	DUART 2
22	Dual I2C 1
23	Dual I2C 2
24	PCI Express 1 INTA
25	PCI Express 2 INTA
26	PCI Express 3 INTA
27	-
28	USB 1
29	USB 2
...	-
32	eSDHC
...	-
36	Performance monitor
37	eSPI
38	-
39	GPIO
40	RapidIO outbound doorbell
41	RapidIO inbound doorbell
...	-
44	RapidIO outbound message unit 1
45	RapidIO inbound message unit 1
46	RapidIO outbound message unit 2
47	RapidIO inbound message unit 2
...	-
68	Event processing unit 1
69	Event processing unit 2
...	-
72	SEC 4.0 job queue 0
73	SEC 4.0 job queue 1
74	SEC 4.0 job queue 2
75	SEC 4.0 job queue 3
76	SEC 4.0 general error NOTE: The only SEC 4.0 general error is the RTIC.
77	Security monitor
...	-
80	Frame manager 1
81	Frame manager 2

Table continues on the next page...

Table 23-1. P4080 Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
...	-
84	Ethernet management interface 1 (MDIO 1)-FMan1:dSTEC1
85	Ethernet management interface 2 (MDIO 2)-FMan1:10GEC
...	-
88	Queue manager portal 0
89	Buffer manager portal 0
90	Queue manager portal 1
91	Buffer manager portal 1
92	Queue manager portal 2
93	Buffer manager portal 2
94	Queue manager portal 3
95	Buffer manager portal 3
96	Queue manager portal 4
97	Buffer manager portal 4
98	Queue manager portal 5
99	Buffer manager portal 5
100	Queue manager portal 6
101	Buffer manager portal 6
102	Queue manager portal 7
103	Buffer manager portal 7
104	Queue manager portal 8
105	Buffer manager portal 8
106	Queue manager portal 9
107	Buffer manager portal 9

Table 23-2 shows the ORed error interrupt sources (see [Error interrupt summary register 0 \(MPIC_EISR0\)](#)) that map to internal interrupt 0.

Table 23-2. ORed Error Interrupt Sources

ORed Interrupt	Interrupt Source	Section/Page
0	Frame manager 2 error	See the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i>
1	Frame manager 1 error	See the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i>
2	Buffer manager error	See the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i>

Table continues on the next page...

Table 23-2. ORed Error Interrupt Sources (continued)

ORed Interrupt	Interrupt Source	Section/Page
3	Queue manager error	See the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i>
4	-	-
5	Pattern match engine error	See the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i>
6-10	-	-
11	RapidIO error/eRMU error/port-write	
12	-	-
13	PCI Express 3 error	PCI Express error detect register (PEX_PEX_ERR_DR), PCI Express PME & message detect register (PEX_PEX_PME_MES_DR), and other registers mentioned in PCI Express controller internal interrupt sources
14	PCI Express 2 error	PCI Express error detect register (PEX_PEX_ERR_DR), PCI Express PME & message detect register (PEX_PEX_PME_MES_DR), and other registers mentioned in PCI Express controller internal interrupt sources
15	PCI Express 1 error	PCI Express error detect register (PEX_PEX_ERR_DR), PCI Express PME & message detect register (PEX_PEX_PME_MES_DR), and other registers mentioned in PCI Express controller internal interrupt sources
16-21	-	-
22	Memory controller 2 error	Memory error detect (DDR_ERR_DETECT)
23	Memory controller 1 error	Memory error detect (DDR_ERR_DETECT)
24-25	-	-
26	CoreNet platform cache 2 (CPC2) error	Error Detection and Reporting
27	CoreNet platform cache 1 (CPC1) error	Error Detection and Reporting
28	-	-
29	Internal RAM multi-bit ECC error	DCFG

Table continues on the next page...

Table 23-2. ORed Error Interrupt Sources (continued)

ORed Interrupt	Interrupt Source	Section/Page
30	PAMU hardware error	PAMU Operation Error Status register 1 (PAMU_POES1) and ECC Error Detect Register (PAMU_EEDET)
31	CoreNet coherency fabric (CCF) error	CCF Error Detect Register (CCF_CEDR)

23.3 PCI Express INTx/External Interrupt IRQn Sharing

The virtual INTA signals of the three PCI Express controllers are routed to dedicated internal interrupts (24, 25, and 26) in the MPIC. The other INTx signals are logically combined with the IRQn signals so that they share the external interrupt controlled by the associated EIVPRn and EIDRn registers. [Table 23-3](#) details the association of PCI Express virtual INTx signals to MPIC external interrupt IRQn signals.

Table 23-3. PCI Express INTx/External Interrupt IRQn Sharing

PCI Express Controller	INTx	IRQn
PEX1	-	IRQ0
	INTA	-
	INTB	IRQ1
	INTC	IRQ2
	INTD	IRQ3
PEX2	-	IRQ4
	INTA	-
	INTB	IRQ5
	INTC	IRQ6
	INTD	IRQ7
PEX3	-	IRQ8
	INTA	-
	INTB	IRQ9
	INTC	IRQ10
	INTD	IRQ11

23.4 SoC Interrupt Event Routing

Table 23-4 shows the routing of the MPIC SoC interrupt event outputs (*sien*), which are used to signal events to the specified blocks on the SoC.

Table 23-4. SoC Interrupt Event Routing

SoC Interrupt Event	xILRn[INTTGT]	Event Routing	Section/Page
<i>sie0</i>	0xF0	DDR controllers (all). Triggers panic button/battery backup event	Hardware Based Self-Refresh Scheme
<i>sie1</i>	0xF1	PCI Express controllers (all). Triggers generation of outbound MSI transaction.	Hardware MSI generation
<i>sie2</i>	0xF2	Reserved	-

Chapter 24

General Purpose I/O (GPIO)

24.1 GPIO overview

This chapter describes the general-purpose I/O (GPIO) module, including signal descriptions, register settings and interrupt capabilities.

This figure shows the block diagram for the GPIO module.

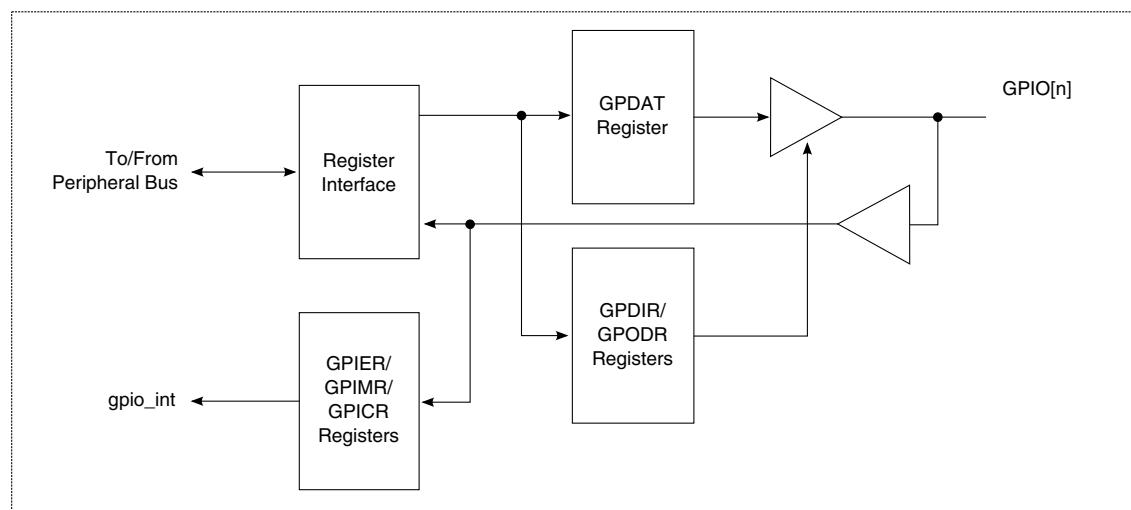


Figure 24-1. GPIO module block diagram

In general, the GPIO module supports up to 32 general-purpose I/O ports. Each port can be configured as an input or as an output. However, some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See [The GPIO module as implemented on the chip](#) for more information. If a port is configured as an input, it can optionally generate an interrupt upon detection of a change. If a port is configured as an output, it can be individually configured as an open-drain or a fully active output.

24.1.1 The GPIO module as implemented on the chip

This section provides details about how the GPIO module is integrated into this chip.

Table 24-1. GPIO module as implemented on chip

Module name	Module base address	Supported ¹ ports	Number of ports
GPIO	0x13_0000	0:31	32

1. GPIO signals are typically multiplexed with other signals. “Supported” in this context means that any signal multiplexing configuration has selected GPIO functionality. See the Signals chapter for signal multiplexing details.

24.2 GPIO features summary

The GPIO unit includes the following features:

- Supports 32 general-purpose input/output ports
- All signals are high-impedance during reset
- Open-drain capability on all ports
- All ports can optionally generate an interrupt upon changing their state
- Ports may be multiplexed with other functional signals

24.3 GPIO signal descriptions

This table provides detailed descriptions of the external GPIO signals. Note that depending on the signal multiplexing, some signals may not be available.

Table 24-2. GPIO external signals-detailed signal descriptions

Signal	I/O	Description
GPIO[0:31]	I/O	General Purpose I/O. Each signal can be individually set to act as input or output, according to application needs. Some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See The GPIO module as implemented on the chip for more information.
		State Meaning Asserted/Negated-Defined per application.
	Timing	Assertion/Negation-Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock.

24.4 GPIO Memory Map/Register Definition

The programmable register map for the GPIO module occupies 24 bytes of memory-mapped space. The full register address is comprised of the CCSR window base address, specified in CCSRBAR, plus the module base address, plus the specific register's offset within the module. The table below shows the memory map for the GPIO module.

All GPIO registers are 32 bits wide located on 32-bit address boundaries. Note that reading undefined portions of the memory map returns all zeros and writing has no effect.

GPIO memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
13_0000	GPIO direction register (GPIO_GPDIR)	32	R/W	0000_0000h	24.4.1/1789
13_0004	GPIO open drain register (GPIO_GPODR)	32	R/W	0000_0000h	24.4.2/1790
13_0008	GPIO data register (GPIO_GPDAT)	32	R/W	0000_0000h	24.4.3/1790
13_000C	GPIO interrupt event register (GPIO_GPIER)	32	w1c	See section	24.4.4/1791
13_0010	GPIO interrupt mask register (GPIO_GPIMR)	32	R/W	0000_0000h	24.4.5/1791
13_0014	GPIO interrupt control register (GPIO_GPICR)	32	R/W	0000_0000h	24.4.6/1792

24.4.1 GPIO direction register (GPIO_GPDIR)

The GPIO direction register (GPPDIR) defines the direction of the individual ports.

Address: 13_0000h base + 0h offset = 13_0000h

GPIO GPDIB field descriptions

Field	Description
0–31 DRn	<p>Direction. Indicates whether a pin is used as an input or an output.</p> <p>0 The corresponding pin is an input. 1 The corresponding pin is an output.</p>

24.4.2 GPIO open drain register (GPIO_GPODR)

The GPIO open drain register (GPODR) defines the way individual ports drive their output.

Address: 13_0000h base + 4h offset = 13_0004h

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

Reset 0

GPIO_GPODR field descriptions

Field	Description
0–31 ODn	Open drain configuration. Indicates whether a signal is actively driven as an output or is an open-drain driver. This register has no effect on signals programmed as inputs in GPDIR. 0 The corresponding signal is actively driven as an output. 1 The corresponding signal is an open-drain driver. As an output, the signal is driven active-low, otherwise it is not driven (high impedance).

24.4.3 GPIO data register (GPIO_GPDAT)

The GPIO data register (GPDAT) carries the data in/out for the individual ports.

Address: 13_0000h base + 8h offset = 13_0008h

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

Reset 0

GPIO_GPDAT field descriptions

Field	Description
0–31 Dn	Data. Writes to this register latches the data which is presented on the external pins provided the corresponding GPDIR bit is configured as an output. When GPDIR is in output mode, GPDAT read operation returns data at pin. When GPDIR is in input mode, GPDAT read operation returns state of the port.

24.4.4 GPIO interrupt event register (GPIO_GPIER)

The GPIO interrupt event register (GPIER) carries information of the events that caused an interrupt. Each bit in GPIER, corresponds to an interrupt source. GPIER bits are cleared by writing ones. However, writing zero has no effect.

NOTE

Some implementations may ignore the interrupt mask as configured in GPIMR. In these implementations, a GPIER bit can be set even though the associated interrupt is masked. See [The GPIO module as implemented on the chip](#) for more information.

Address: 13_0000h base + Ch offset = 13_000Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	EVn															
W																	w1c															
Reset	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*	n*

* Notes:

- EVn field: Undefined. User should write 1s to clear before using.

GPIO_GPIER field descriptions

Field	Description
0–31 EVn	Interrupt events. Indicates whether an interrupt event occurred on the corresponding GPIO signal. 0 No interrupt event occurred on the corresponding GPIO signal. 1 An interrupt event occurred on the corresponding GPIO signal.

24.4.5 GPIO interrupt mask register (GPIO_GPIMR)

The GPIO interrupt mask register (GPIMR) defines the interrupt masking for the individual ports. When a masked interrupt request occurs, the corresponding GPIER bit is set, regardless of the GPIMR state. When one or more non-masked interrupt events occur, the GPIO module issues an interrupt to the interrupt controller.

Address: 13_0000h base + 10h offset = 13_0010h

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

GPIO_GPIMR field descriptions

Field	Description
0–31 IMn	Interrupt mask. Indicates whether an interrupt event is masked or not masked for the corresponding GPIO signal. 0 The input interrupt signal is masked (disabled). 1 The input interrupt signal is not masked (enabled).

24.4.6 GPIO interrupt control register (GPIO_GPICR)

The GPIO interrupt control register (GPICR) determines whether the corresponding port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

Address: 13_0000h base + 14h offset = 13_0014h

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

Reset 0

GPIO_GPICR field descriptions

Field	Description
0–31 EDn	Edge detection mode. The corresponding port line asserts an interrupt request according to the following: 0 Any change on the state of the port generates an interrupt request. 1 High-to-low change on the port generates an interrupt request.

Chapter 25

Device Configuration and Pin Control

25.1 Device Configuration and Pin Control Introduction

This chapter describes the device configuration and pin control facilities of the chip. The device configuration unit provides general purpose configuration and status for the device and the pin control block implements general purpose configuration and status registers for the device, and the logic to configure the I/O pads to operate according to the requirements of the functional components. It also controls the data path between the SoC components and the I/O pads. The pin control block consists of the pins control module, the functional I/O multiplexing, and the JTAG boundary scan logic.

25.2 Device Configuration and Pin Control Features Summary

The device configuration unit features the following:

- Pin Sampling of device configuration pins at power-on reset and a corresponding POR status register for capturing the values of these configuration pins
- Reset Configuration Word Support via a set of Reset Configuration Word status registers written by the Preboot Loader during Power-on or hard reset in the PBL's RCW stage
- Boot Release Register(s) used for releasing cores for booting
- Register file for the Reset Module including:
 - Register for initiating a device RESET_REQ through software
 - Set of registers for control and status of sources on the device which can drive the device's RESET_REQ pin
 - Core and Device Disable registers used for gating off clocks for any IP blocks or cores which are not used at all by an application

- Set of Local I/O Device Number (LIODN) registers for programming the LIODN of legacy peripherals which did not originally contain this register
- Two sets of scratch registers:
 - one set of read / write scratch registers
 - one set of write-once / read scratch registers

25.3 Device Configuration and Pin Control Memory Map/ Register Definition

The following table lists the device configuration and pin control registers in offset order and provides links to the complete register descriptions.

DCFG memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0000	POR status register 1 (DCFG_PORSR1)	32	R	See section	25.3.1/1798
E_0004	POR status register 2 (DCFG_PORSR2)	32	R	See section	25.3.2/1799
E_0020	General-purpose POR configuration register (DCFG_GPPORCR1)	32	R	See section	25.3.3/1800
E_0070	Device disable register 1 (DCFG_DEVDISR1)	32	R/W	See section	25.3.4/1800
E_0074	Device disable register 2 (DCFG_DEVDISR2)	32	R/W	See section	25.3.5/1804
E_0094	Core disable register (DCFG_COREDISR)	32	R/W	0000_0000h	25.3.6/1806
E_00A0	Processor version register (DCFG_PVR)	32	R	8023_0000h	25.3.7/1808
E_00A4	System version register (DCFG_SVR)	32	R	See section	25.3.8/1809
E_00B0	Reset control register (DCFG_RSTCR)	32	R/W	0000_0000h	25.3.9/1810
E_00B4	Reset request preboot loader status register (DCFG_RSTRQPBLSR)	32	w1c	0000_0000h	25.3.10/1811
E_00C0	Reset request mask register (DCFG_RSTRQMR1)	32	R/W	0000_0000h	25.3.11/1812
E_00C8	Reset request status register (DCFG_RSTRQSR1)	32	w1c	0000_0000h	25.3.12/1813
E_00D4	Reset request WDT mask register (DCFG_RSTRQWDTMR)	32	R/W	0000_0000h	25.3.13/1815
E_00DC	Reset request WDT status register (DCFG_RSTRQWDTSR)	32	w1c	0000_0000h	25.3.14/1817
E_00E4	Boot release register (DCFG_BRR)	32	R/W	0000_0000h	25.3.15/1818
E_0100	Reset control word status registers 1-16 (DCFG_RCWSR1)	32	R	See section	25.3.16/1820
E_0104	Reset control word status registers 1-16 (DCFG_RCWSR2)	32	R	See section	25.3.16/1820

Table continues on the next page...

DCFG memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0108	Reset control word status registers 1-16 (DCFG_RCWSR3)	32	R	See section	25.3.16/ 1820
E_010C	Reset control word status registers 1-16 (DCFG_RCWSR4)	32	R	See section	25.3.16/ 1820
E_0110	Reset control word status registers 1-16 (DCFG_RCWSR5)	32	R	See section	25.3.16/ 1820
E_0114	Reset control word status registers 1-16 (DCFG_RCWSR6)	32	R	See section	25.3.16/ 1820
E_0118	Reset control word status registers 1-16 (DCFG_RCWSR7)	32	R	See section	25.3.16/ 1820
E_011C	Reset control word status registers 1-16 (DCFG_RCWSR8)	32	R	See section	25.3.16/ 1820
E_0120	Reset control word status registers 1-16 (DCFG_RCWSR9)	32	R	See section	25.3.16/ 1820
E_0124	Reset control word status registers 1-16 (DCFG_RCWSR10)	32	R	See section	25.3.16/ 1820
E_0128	Reset control word status registers 1-16 (DCFG_RCWSR11)	32	R	See section	25.3.16/ 1820
E_012C	Reset control word status registers 1-16 (DCFG_RCWSR12)	32	R	See section	25.3.16/ 1820
E_0130	Reset control word status registers 1-16 (DCFG_RCWSR13)	32	R	See section	25.3.16/ 1820
E_0134	Reset control word status registers 1-16 (DCFG_RCWSR14)	32	R	See section	25.3.16/ 1820
E_0138	Reset control word status registers 1-16 (DCFG_RCWSR15)	32	R	See section	25.3.16/ 1820
E_013C	Reset control word status registers 1-16 (DCFG_RCWSR16)	32	R	See section	25.3.16/ 1820
E_0200	Scratch read/write registers 1-4 (DCFG_SCRATCHRW1)	32	R/W	0000_0000h	25.3.17/ 1820
E_0204	Scratch read/write registers 1-4 (DCFG_SCRATCHRW2)	32	R/W	0000_0000h	25.3.17/ 1820
E_0208	Scratch read/write registers 1-4 (DCFG_SCRATCHRW3)	32	R/W	0000_0000h	25.3.17/ 1820
E_020C	Scratch read/write registers 1-4 (DCFG_SCRATCHRW4)	32	R/W	0000_0000h	25.3.17/ 1820
E_0300	Scratch write-once registers 1-4 (DCFG_SCRATCHW1R1)	32	R/W	0000_0000h	25.3.18/ 1821
E_0304	Scratch write-once registers 1-4 (DCFG_SCRATCHW1R2)	32	R/W	0000_0000h	25.3.18/ 1821
E_0308	Scratch write-once registers 1-4 (DCFG_SCRATCHW1R3)	32	R/W	0000_0000h	25.3.18/ 1821
E_030C	Scratch write-once registers 1-4 (DCFG_SCRATCHW1R4)	32	R/W	0000_0000h	25.3.18/ 1821

Table continues on the next page...

DCFG memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
E_0400	Core reset status register n (DCFG_CRSTS0)	32	w1c	0000_0000h	25.3.19/ 1821
E_0404	Core reset status register n (DCFG_CRSTS1)	32	w1c	0000_0000h	25.3.19/ 1821
E_0408	Core reset status register n (DCFG_CRSTS2)	32	w1c	0000_0000h	25.3.19/ 1821
E_040C	Core reset status register n (DCFG_CRSTS3)	32	w1c	0000_0000h	25.3.19/ 1821
E_0410	Core reset status register n (DCFG_CRSTS4)	32	w1c	0000_0000h	25.3.19/ 1821
E_0414	Core reset status register n (DCFG_CRSTS5)	32	w1c	0000_0000h	25.3.19/ 1821
E_0418	Core reset status register n (DCFG_CRSTS6)	32	w1c	0000_0000h	25.3.19/ 1821
E_041C	Core reset status register n (DCFG_CRSTS7)	32	w1c	0000_0000h	25.3.19/ 1821
E_0500	PCI Express n LIODN register (DCFG_PEX1LIODNR)	32	R/W	0000_0000h	25.3.20/ 1823
E_0504	PCI Express n LIODN register (DCFG_PEX2LIODNR)	32	R/W	0000_0000h	25.3.20/ 1823
E_0508	PCI Express n LIODN register (DCFG_PEX3LIODNR)	32	R/W	0000_0000h	25.3.20/ 1823
E_0510	RIO n LIODN register (DCFG_RIO1LIODNR)	32	R/W	0000_0000h	25.3.21/ 1824
E_0514	RIO n LIODN register (DCFG_RIO2LIODNR)	32	R/W	0000_0000h	25.3.21/ 1824
E_0520	USB n LIODN register (DCFG_USB1LIODNR)	32	R/W	0000_0000h	25.3.22/ 1824
E_0524	USB n LIODN register (DCFG_USB2LIODNR)	32	R/W	0000_0000h	25.3.22/ 1824
E_0530	eSDHC LIODN register (DCFG_eSDHCLIODNR)	32	R/W	0000_0000h	25.3.23/ 1825
E_0540	RIO message unit LIODN register (DCFG_RMULIODNR)	32	R/W	0000_0000h	25.3.24/ 1825
E_0580	DMA n LIODN register (DCFG_DMA1LIODNR)	32	R/W	0000_0000h	25.3.25/ 1826
E_0584	DMA n LIODN register (DCFG_DMA2LIODNR)	32	R/W	0000_0000h	25.3.25/ 1826
E_0604	PAMU bypass enable register (DCFG_PAMUBYPENR)	32	R	See section	25.3.26/ 1826
E_0608	DMA control register (DCFG_DMACR1)	32	R/W	0000_0000h	25.3.27/ 1827
E_0E00	Pin Multiplexing Control Register (DCFG_PMUXCR)	32	R/W	0000_0000h	25.3.28/ 1828

Table continues on the next page...

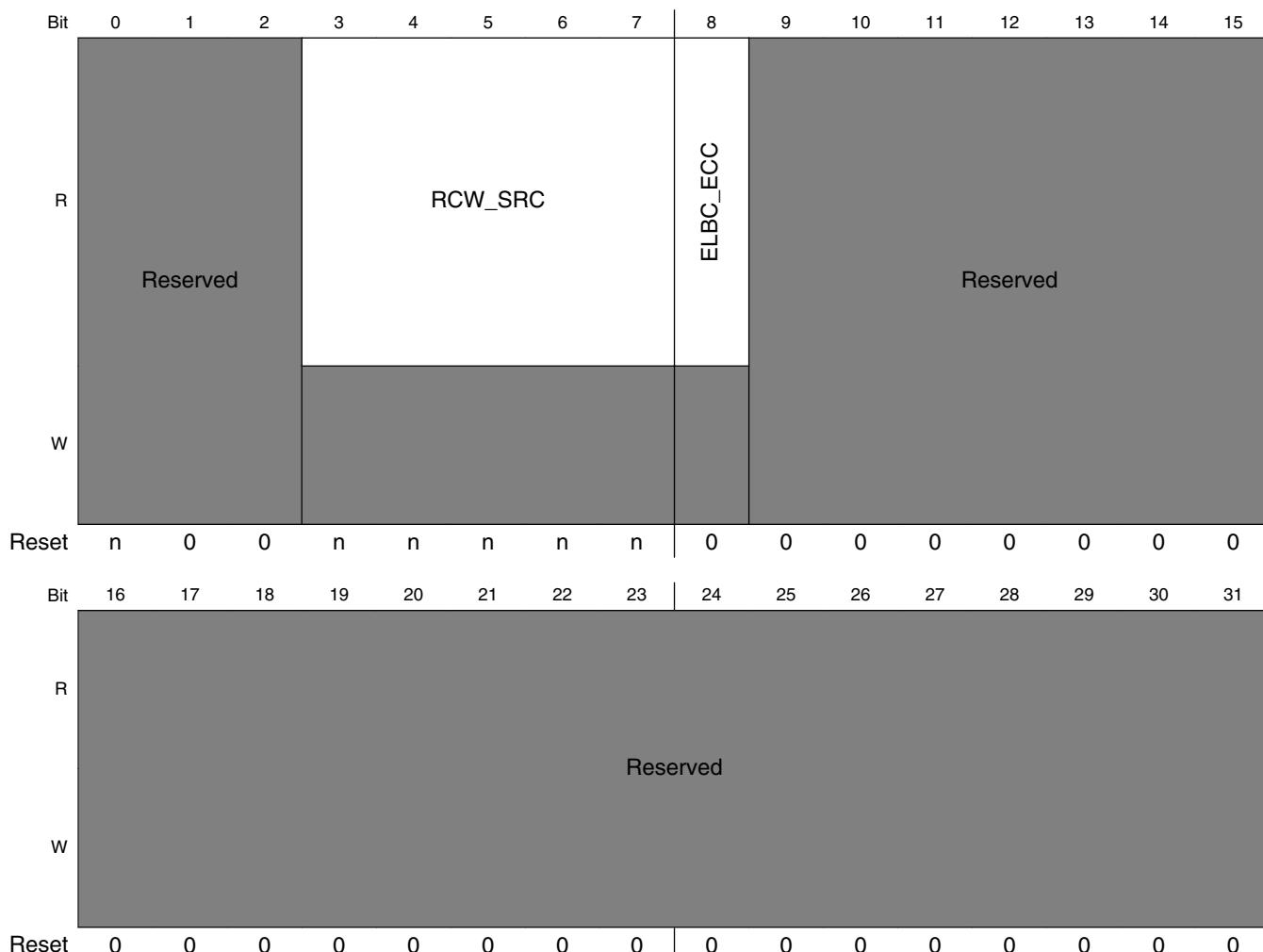
DCFG memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
E_0E20	Debug Source ID 1A (DCFG_DSRCID1A)	32	R/W	0000_0000h	25.3.29/ 1830
E_0E24	Debug Source ID 1B (DCFG_DSRCID1B)	32	R/W	0000_0000h	25.3.30/ 1831
E_0E28	Debug Source ID 1C (DCFG_DSRCID1C)	32	R/W	0000_0000h	25.3.31/ 1832
E_0E2C	Debug Source ID 1D (DCFG_DSRCID1D)	32	R/W	0000_0000h	25.3.32/ 1833
E_0E30	Debug Source ID 2A (DCFG_DSRCID2A)	32	R/W	0000_0000h	25.3.33/ 1834
E_0E34	Debug Source ID 2B (DCFG_DSRCID2B)	32	R/W	0000_0000h	25.3.34/ 1835
E_0E40	I/O Voltage Selection Status Register (DCFG_IOVSELSR)	32	R	0000_0000h	25.3.35/ 1836
E_0E60	DDR Clock Disable Register (DCFG_DDRCLKDR)	32	R/W	0000_0000h	25.3.36/ 1836
E_0E68	eLBC Clock Disable Register (DCFG_ELBCCLKDR)	32	R/W	0000_0000h	25.3.37/ 1838
E_0E80	eSDHC Polarity Configuration Register (DCFG_SDHCPCR)	32	R/W	0000_0000h	25.3.38/ 1839

25.3.1 POR status register 1 (DCFG_PORSR1)

PORSR1 captures the values of the device's POR configuration pins.

Address: E_0000h base + 0h offset = E_0000h



DCFG_PORSR1 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 RCW_SRC	Reset Configuration Word Source. Indicates which flash memory contains the RCW information. Loaded with the value of cfg_rcw_src[0:4].
8 ELBC_ECC	eLBC ECC Enable. Loaded with the value of the cfg_elbc_ecc pin at power-on reset.

Table continues on the next page...

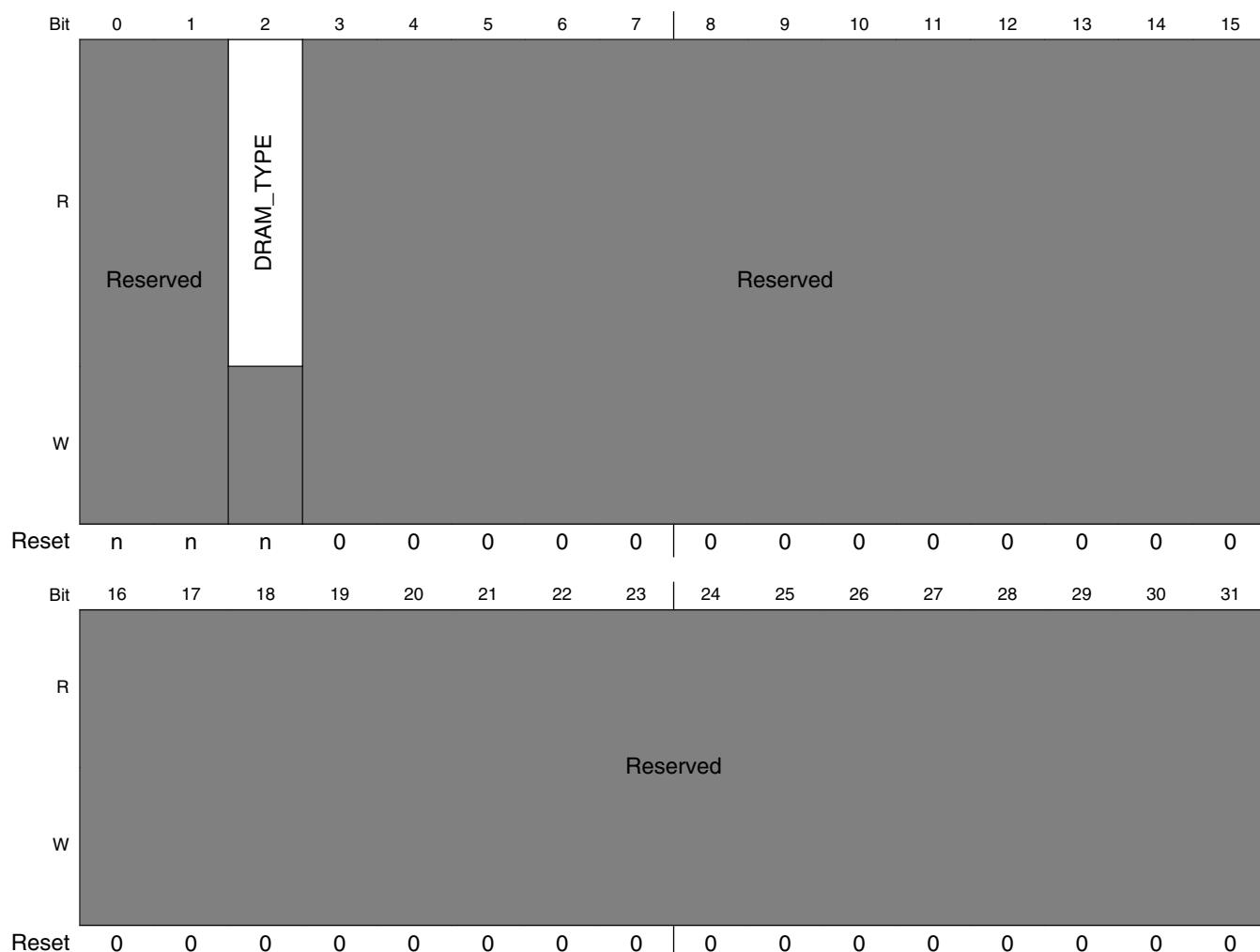
DCFG_PORSR1 field descriptions (continued)

Field	Description
	0 eLBC ECC not enabled out of reset 1 eLBC ECC enabled out of reset
9–31 -	This field is reserved. Reserved

25.3.2 POR status register 2 (DCFG_PORSR2)

PORSR2 captures the values of the device's POR configuration pins.

Address: E_0000h base + 4h offset = E_0004h



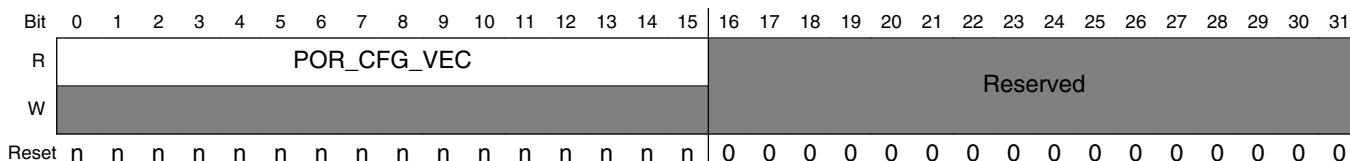
DCFG_PORSR2 field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2 DRAM_TYPE	DRAM Type Selector: 0 DDR3 (1.5 V) 1 DDR2 (1.8 V)
3–31 -	This field is reserved. Reserved

25.3.3 General-purpose POR configuration register (DCFG_GPPORCR1)

GPPORCR1 stores the value sampled from the local bus address/data signals, LAD[0:15], using sampling logic similar to that for the configuration pins . Software can use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

Address: E_0000h base + 20h offset = E_0020h

**DCFG_GPPORCR1 field descriptions**

Field	Description
0–15 POR_CFG_VEC	General-purpose POR configuration vector sampled from local bus address/data signals at the negation of PORESET_B . Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.
16–31 -	This field is reserved. Reserved

25.3.4 Device disable register 1 (DCFG_DEVDISR1)

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR1 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR1 register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

Address: E_0000h base + 70h offset = E_0070h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PEX1	PEX2	PEX3	Reserved	eRMU	SRI01	SRI02	Reserved		DMA1	DMA2	DDR1	DDR2	Reserved	DBG	
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NAL	Reserved		eLBC	USB1	USB2	Reserved	eSDHC	GPIO	eSPI	I2C1	I2C2	Reserved	DUART1	DUART2	
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	

DCFG_DEVDISR1 field descriptions

Field	Description
0 PEX1	PCI Express controller 1 disable 0 Block is enabled 1 Block is disabled
1 PEX2	PCI Express controller 2 disable 0 Block is enabled 1 Block is disabled
2 PEX3	PCI Express controller 3 disable 0 Block is enabled 1 Block is disabled
3 -	This field is reserved. Reserved
4 eRMU	RapidIO message unit disable

Table continues on the next page...

DCFG_DEVDISR1 field descriptions (continued)

Field	Description
	<p>NOTE: The RMU must be enabled to allow access to SRIO1/2 configuration registers; disabling the RMU prevents access to SRIO1/2 configuration registers.</p> <p>0 Block is enabled 1 Block is disabled</p>
5 SRIO1	<p>Serial RapidIO controller 1 disable</p> <p>NOTE: SRIO1 must be enabled to use SRIO2.</p> <p>0 Block is enabled 1 Block is disabled</p>
6 SRIO2	<p>Serial RapidIO controller 2 disable</p> <p>NOTE: SRIO1 must be enabled to use SRIO2.</p> <p>0 Block is enabled 1 Block is disabled</p>
7–8 -	This field is reserved. Reserved
9 DMA1	<p>DMA controller 1 disable</p> <p>0 Block is enabled 1 Block is disabled</p>
10 DMA2	<p>DMA controller 2 disable</p> <p>0 Block is enabled 1 Block is disabled</p>
11 DDR1	<p>DDR controller 1 disable</p> <p>0 Block is enabled 1 Block is disabled</p>
12 DDR2	<p>DDR controller 2 disable</p> <p>0 Block is enabled 1 Block is disabled</p>
13–14 -	This field is reserved. Reserved
15 DBG	<p>Debug controller disable</p> <p>0 Block is enabled 1 Block is disabled</p>
16 NAL	<p>Nexus/Aurora Link layer disable</p> <p>0 Block is enabled 1 Block is disabled</p>
17–18 -	This field is reserved. Reserved
19 eLBC	eLBC controller disable

Table continues on the next page...

DCFG_DEVDISR1 field descriptions (continued)

Field	Description
	0 Block is enabled 1 Block is disabled
20 USB1	USB controller 1 disable 0 Block is enabled 1 Block is disabled
21 USB2	USB controller 2 disable 0 Block is enabled 1 Block is disabled
22 -	This field is reserved. Reserved
23 eSDHC	eSDHC controller disable 0 Block is enabled 1 Block is disabled
24 GPIO	GPIO controller disable 0 Block is enabled 1 Block is disabled
25 eSPI	eSPI controller disable 0 Block is enabled 1 Block is disabled
26 I2C1	I2C module 1 (controllers 1 and 2) disable 0 Block is enabled 1 Block is disabled
27 I2C2	I2C module 2 (controllers 3 and 4) disable 0 Block is enabled 1 Block is disabled
28–29 -	This field is reserved. Reserved
30 DUART1	DUART controller 1 disable 0 Block is enabled 1 Block is disabled
31 DUART2	DUART controller 2 disable 0 Block is enabled 1 Block is disabled

25.3.5 Device disable register 2 (DCFG_DEVDISR2)

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR2 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR2 register must not be re-enabled via software. The only supported mechanism for re-enabling an IP block disabled in this manner is via power-on or hard reset.

Address: E_0000h base + 74h offset = E_0074h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PME	SEC	Reserved	QM_BM	Reserved	FM1	10GEC1		FM1_dTSEC1	FM1_dTSEC2	FM1_dTSEC3	FM1_dTSEC4	Reserved	FM2	10GEC2	
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FM2_dTSEC1	FM2_dTSEC2	FM2_dTSEC3	FM2_dTSEC4	Reserved											
W																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

DCFG_DEVDISR2 field descriptions

Field	Description
0 PME	Pattern match engine disable 0 Block is enabled 1 Block is disabled
1 SEC	Security disable. 0 Block is enabled 1 Block is disabled

Table continues on the next page...

DCFG_DEVDISR2 field descriptions (continued)

Field	Description
2–3 -	This field is reserved. Reserved
4 QM_BM	Queue manager/buffer manager disable 0 Block is enabled 1 Block is disabled
5 -	This field is reserved. Reserved
6 FM1	Frame manager 1 disable 0 Block is enabled 1 Block is disabled
7 10GEC1	FMan 1 10G Ethernet controller disable 0 Block is enabled 1 Block is disabled
8 FM1_dTSEC1	FMan 1 dTSEC controller 1 disable 0 Block is enabled 1 Block is disabled
9 FM1_dTSEC2	FMan 1 dTSEC controller 2 disable 0 Block is enabled 1 Block is disabled
10 FM1_dTSEC3	FMan 1 dTSEC controller 3 disable 0 Block is enabled 1 Block is disabled
11 FM1_dTSEC4	FMan 1 dTSEC controller 4 disable 0 Block is enabled 1 Block is disabled
12–13 -	This field is reserved. Reserved
14 FM2	Frame manager 2 disable 0 Block is enabled 1 Block is disabled
15 10GEC2	FMan 2 10G Ethernet controller disable 0 Block is enabled 1 Block is disabled
16 FM2_dTSEC1	FMan 2 dTSEC controller 1 disable 0 Block is enabled 1 Block is disabled
17 FM2_dTSEC2	FMan 2 dTSEC controller 2 disable

Table continues on the next page...

DCFG_DEVDIR2 field descriptions (continued)

Field	Description
	0 Block is enabled 1 Block is disabled
18 FM2_dTSEC3	FMan 2 dTSEC controller 3 disable 0 Block is enabled 1 Block is disabled
19 FM2_dTSEC4	FMan 2 dTSEC controller 4 disable 0 Block is enabled 1 Block is disabled
20–31 -	This field is reserved. Reserved

25.3.6 Core disable register (DCFG_COREDISR)

COREDISR provides a mechanism for gating clocks to any cores on the device that are not used when running an application.

COREDISR should only be configured in the following conditions:

- Before system ready, COREDISR can be programmed by an external debugger or Pre-Boot Initialization
- After system ready, a COREDISR bit can be programmed for the corresponding core by an external debugger or embedded software while the core is in boot-holdoff mode.

NOTE

Cores disabled by setting the corresponding bit in the COREDISR register must not be re-enabled via software. The only supported mechanism for re-enabling a core disabled in this manner is via power-on, hard reset, or core reset.

Address: E_0000h base + 94h offset = E_0094h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_COREDISR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 CD7	Core 7 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
25 CD6	Core 6 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
26 CD5	Core 5 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
27 CD4	Core 4 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
28 CD3	Core 3 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
29 CD2	Core 2 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
30 CD1	Core 1 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
31 CD0	Core 0 Disable. 0 Selected Core is enabled 1 Selected Core is disabled

25.3.7 Processor version register (DCFG_PVR)

The PVR contains the processor version number. It is a memory-mapped copy of the PVR in core 0 (and is therefore accessible to external devices).

Address: E_0000h base + A0h offset = E_00A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MFR_ID						Reserved		PROC_TYPE				PROC_ID			
W																
Reset	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PROCESS_REV				MFG_REV				MAJOR_REV				MINOR_REV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_PVR field descriptions

Field	Description
0–3 MFR_ID	Manufacturer ID
4–5 -	This field is reserved. Reserved
6–9 PROC_TYPE	Processor type
10–15 PROC_ID	Processor ID
16–19 PROCESS_REV	Process revision
20–23 MFG_REV	Manufacturing revision
24–27 MAJOR_REV	Processor major revision number
28–31 MINOR_REV	Processor minor revision number

25.3.8 System version register (DCFG_SVR)

The SVR contains the system version number for the device. This value can also be read through the SVR SPR of the core.

Address: E_0000h base + A4h offset = E_00A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MFG_ID				SOC_DEV_ID								SEC	SOC_VAR		
W																
Reset	1	0	0	0	0	0	1	0	0	0	0	0	n	0	0	n
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PERSONALITY								MAJOR_REV				MINOR_REV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

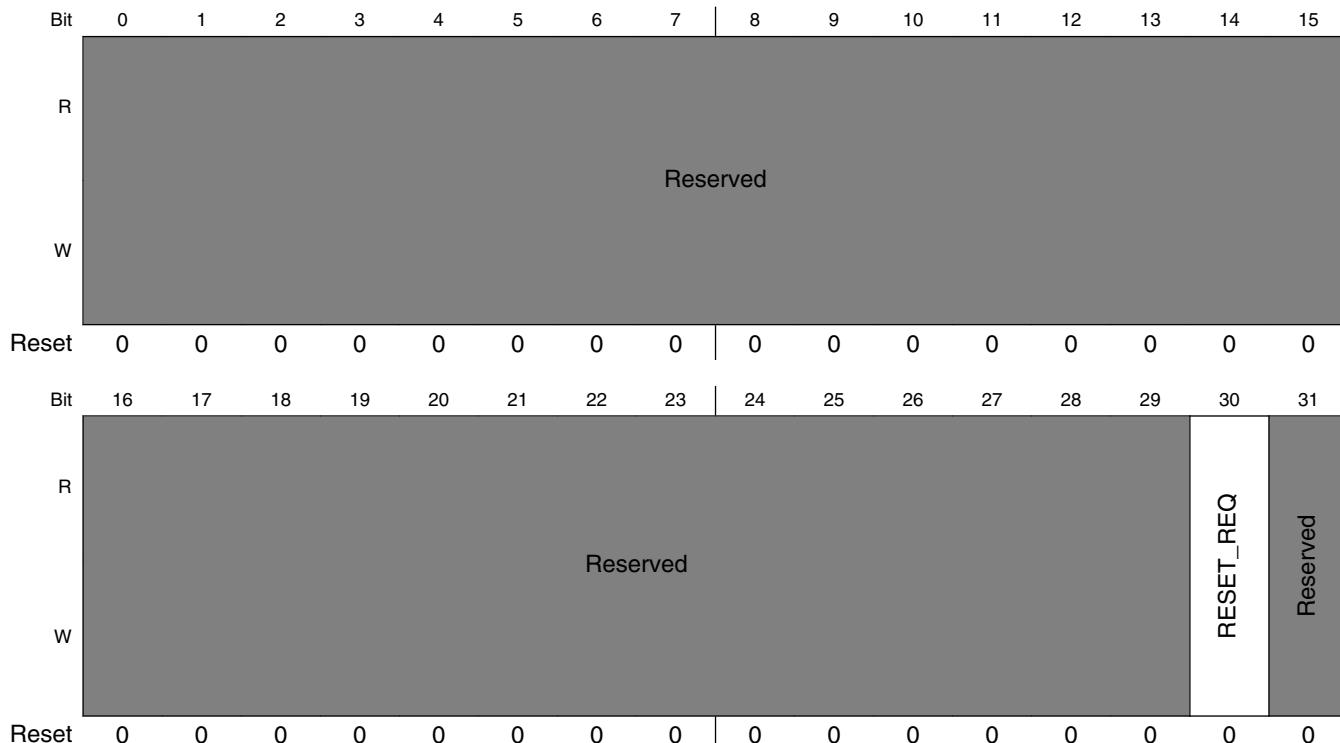
DCFG_SVR field descriptions

Field	Description
0–3 MFG_ID	Manufacturer ID 0x8:Freescale
4–11 SOC_DEV_ID	SoC device ID
12 SEC	Security 0 No security 1 Security enabled
13–15 SOC_VAR	SoC variant ID 000 P4080 001 P4040
16–23 PERSONALITY	Personality
24–27 MAJOR_REV	Device major revision number
28–31 MINOR_REV	Device minor revision number

25.3.9 Reset control register (DCFG_RSTCR)

The RSTCR contains bits that allow for software to force assertion of RESET_REQ_B . External hardware may then decide to issue the desired reset signal (PORESET_B or HRESET_B) to the device.

Address: E_0000h base + B0h offset = E_00B0h



DCFG_RSTCR field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30 RESET_REQ	Hardware reset request 0 No reset request initiated. 1 Hardware reset request initiated by software.
31 -	This field is reserved. Reserved

25.3.10 Reset request preboot loader status register (DCFG_RSTRQPBLSR)

The RSTRQPBLSR contains status bits to record the reasons for RESET_REQ_B assertion.

NOTE

This register's code is valid only when RSTRQSR[PBL_RR] is "1".

Address: E_0000h base + B4h offset = E_00B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved							ERR_CODE							Reserved																	
W								w1c																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

DCFG_RSTRQPBLSR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–14 ERR_CODE	PBL Error Code. This 7-bit bit-encoded value reflects one of 128 possible PBL errors. See PBL Error Codes , for details on the PBL error encodings.
15–31 -	This field is reserved. Reserved

25.3.11 Reset request mask register (DCFG_RSTRQMR1)

RSTRQMR1 contains mask bits for optional masking of RESET_REQ_B sources to prevent generation of such a reset request. Excludes core watchdog timer sources (see [Reset Request WDT Mask Register n](#)).

Address: E_0000h base + C0h offset = E_00C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SRI01_MSK	SRI02_MSK											CCM_MSK	PBL_MSK	SFP_MSK	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDC_MSK	MBEE_EN														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_RSTRQMR1 field descriptions

Field	Description
0 SRIO1_MSK	sRIO 1 reset request mask 0 SRIO 1 can cause a reset request 1 SRIO 1 cannot cause a reset request
1 SRIO2_MSK	sRIO 2 reset request mask (assigned in device's ArchDef spec) 0 SRIO 2 can cause a reset request 1 SRIO 2 cannot cause a reset request
2–11 -	This field is reserved. Reserved
12 CCM_MSK	CoreNet Coherency Module write miss on all local access windows during reset's PBL phase reset mask.

Table continues on the next page...

DCFG_RSTRQMR1 field descriptions (continued)

Field	Description
	0 CoreNet Coherency Module write miss can cause a reset request 1 CoreNet Coherency Module write miss cannot cause a reset request
13 PBL_MSK	PBL error reset mask 0 PBL error can cause a reset request 1 PBL error cannot cause a reset request
14 SFP_MSK	Security Fuse Processor error during POR fuse process reset mask 0 Security Fuse Processor error can cause a reset request 1 Security Fuse Processor error cannot cause a reset request
15 -	This field is reserved. Reserved
16 SDC_MSK	Security Debug Controller error mask 0 SDC can cause a reset request 1 SDC cannot cause a reset request
17 MBEE_EN	Multi-bit ECC error enable 0 Multi-bit ECC error cannot cause a reset request 1 Multi-bit ECC error can cause a reset request
18–31 -	This field is reserved. Reserved

25.3.12 Reset request status register (DCFG_RSTRQSR1)

The RSTRQSR1 contains status bits to record the reasons for RESET_REQ_B assertion. The bits here are set independent of RSTRQMR1. This means that if a reset reason occurs and is masked by RSTRQMR1, it is still recorded in RSTRQSR1. Excludes core watchdog timer sources (see [Reset Request WDT Status Register](#)).

NOTE

For the different sources captured in this register, some of these can be serviced with either PORESET_B or HRESET_B and some of these must be serviced with PORESET_B only.

Address: E_0000h base + C8h offset = E_00C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SRI01_RR	SRI02_RR	Reserved							WDT_RR	SW_RR	CCM_RR	PBL_RR	SFP_RR	SEC_M_RR	
	w1c	w1c								w1c	w1c	w1c	w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SDC_RR	MBEE_RR	Reserved													
	w1c	w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DCFG_RSTREQSR1 field descriptions

Field	Description
0 SRI01_RR	SRIO 1 reset request requires device level PORESET_B or HRESET_B 0 SRIO 1 has not caused a reset request 1 Reset request caused by SRIO 1
1 SRI02_RR	SRIO 2 reset request requires device level PORESET_B or HRESET_B 0 SRIO 2 has not caused a reset request 1 Reset request caused by SRIO 2
2–9 -	This field is reserved. Reserved
10 WDT_RR	Watchdog Timer reset requires device level PORESET_B or HRESET_B 0 All bits in RSTRQWDTSR are cleared (See Reset request WDT status register (DCFG_RSTREQWDTSR) .) 1 At least one bit in RSTRQWDTSR is set
11 SW_RR	Software settable reset requested. (See Reset Control Register .) 0 Software has not caused a reset request 1 Reset request caused by Software
12 CCM_RR	CoreNet Coherency Module write miss on all local access windows during reset's PBL phase requires device level PORESET_B or HRESET_B 0 CoreNet Coherency Module write miss has not caused a reset request 1 Reset request caused by CoreNet Coherency Module write miss

Table continues on the next page...

DCFG_RSTRQSR1 field descriptions (continued)

Field	Description
13 PBL_RR	PBL error reset request requires device level PORESET_B or HRESET_B 0 PBL has not caused a reset request 1 Reset request caused by PBL
14 SFP_RR	Security Fuse Processor error during POR fuse process caused reset request. Security Fuse Processor error requires device level PORESET_B 0 Security Fuse Processor error has not caused a reset request 1 Reset request caused by Security Fuse Processor error
15 SECM_RR	Security Monitor hard fail error. Security Monitor reached Hard Fail state and requires device level PORESET_B . 0 Security Monitor error has not caused a reset request 1 Reset request caused by Security Monitor error
16 SDC_RR	Security Debug Controller reset request after three consecutive failed challenge/response attempts Security Debug Controller requires device level PORESET_B . 0 SDC has not caused a reset request 1 Reset request caused by SDC
17 MBEE_RR	Multi-bit ECC reset request Platform internal memory multi-bit ECC error requires device level PORESET_B or HRESET_B 0 Multi-bit ECC error has not caused a reset request 1 Reset request caused by Multi-bit ECC error
18–31 -	This field is reserved. Reserved

25.3.13 Reset request WDT mask register (DCFG_RSTRQWDTMR)

RSTRQWDTMR contains mask bits for optional masking of RESET_REQ_B generation on core watchdog timer expiration (WRC/WRS=10) from the individual cores. See [Timer Control Register \(TCR\)](#) for more information.

NOTE

The lsb, bit 31, is associated with Core 0.

Address: E_0000h base + D4h offset = E_00D4h

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

Reset 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									WDT_MSK7	WDT_MSK6	WDT_MSK5	WDT_MSK4	WDT_MSK3	WDT_MSK2	WDT_MSK1	WDT_MSK0
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_RSTRQWDTMR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 WDT_MSK7	Core 7 WDT Reset Request Mask. 0 Core 7 WDT can cause a reset request 1 Core 7 WDT cannot cause a reset request
25 WDT_MSK6	Core 6 WDT Reset Request Mask. 0 Core 6 WDT can cause a reset request 1 Core 6 WDT cannot cause a reset request
26 WDT_MSK5	Core 5 WDT Reset Request Mask. 0 Core 5 WDT can cause a reset request 1 Core 5 WDT cannot cause a reset request
27 WDT_MSK4	Core 4 WDT Reset Request Mask. 0 Core 4 WDT can cause a reset request 1 Core 4 WDT cannot cause a reset request
28 WDT_MSK3	Core 3 WDT Reset Request Mask. 0 Core 3 WDT can cause a reset request 1 Core 3 WDT cannot cause a reset request
29 WDT_MSK2	Core 2 WDT Reset Request Mask. 0 Core 2 WDT can cause a reset request 1 Core 2 WDT cannot cause a reset request
30 WDT_MSK1	Core 1 WDT Reset Request Mask. 0 Core 1 WDT can cause a reset request 1 Core 1 WDT cannot cause a reset request
31 WDT_MSK0	Core 0 WDT Reset Request Mask. 0 Core 0 WDT can cause a reset request 1 Core 0 WDT cannot cause a reset request

25.3.14 Reset request WDT status register (DCFG_RSTRQWDTSR)

RSTRQWDTSR contains status bits to record the reasons for RESET_REQ_B assertion due to a core watchdog timer when configured for WRC/WRS=10. See [Timer Control Register \(TCR\)](#) for more information. Excludes device sources (see [Reset request status register \(DCFG_RSTRQSR1\)](#)).

NOTE

The lsb, bit 31, is associated with Core 0.

Address: E_0000h base + DCh offset = E_00DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									WDT_RR7	WDT_RR6	WDT_RR5	WDT_RR4	WDT_RR3	WDT_RR2	WDT_RR1	WDT_RR0
	Reserved								w1c							
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_RSTRQWDTSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 WDT_RR7	Core 7 WDT Reset Request Status. 0 Core 7 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 7 WDT configured for WRC/WRS=10 has caused a reset request
25 WDT_RR6	Core 6 WDT Reset Request Status. 0 Core 6 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 6 WDT configured for WRC/WRS=10 has caused a reset request

Table continues on the next page...

DCFG_RSTRQWDTSR field descriptions (continued)

Field	Description
26 WDT_RR5	Core 5 WDT Reset Request Status. 0 Core 5 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 5 WDT configured for WRC/WRS=10 has caused a reset request
27 WDT_RR4	Core4 WDT Reset Request Status. 0 Core 4 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 4 WDT configured for WRC/WRS=10 has caused a reset request
28 WDT_RR3	Core 3 WDT Reset Request Status. 0 Core 3 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 3 WDT configured for WRC/WRS=10 has caused a reset request
29 WDT_RR2	Core 2 WDT Reset Request Status. 0 Core 2 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 2 WDT configured for WRC/WRS=10 has caused a reset request
30 WDT_RR1	Core 1 WDT Reset Request Status. 0 Core 1 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 1 WDT configured for WRC/WRS=10 has caused a reset request
31 WDT_RR0	Core 0 WDT Reset Request Status. 0 Core 0 WDT configured for WRC/WRS=10 has not caused a reset request 1 Core 0 WDT configured for WRC/WRS=10 has caused a reset request

25.3.15 Boot release register (DCFG_BRR)

BRR contains control bits for enabling boot for each core. On exiting HRESET or PORESET, the RCW[BOOT_HO] field optionally allows for logical core 0 to be released for booting or to remain in boot holdoff. See [RCW Field Definitions](#), for more information. All other cores remain in boot holdoff until their corresponding bit is set.

Address: E_0000h base + E4h offset = E_00E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_BRR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 CR7	Core 7 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
25 CR6	Core 6 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
26 CR5	Core 5 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
27 CR4	Core 4 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
28 CR3	Core 3 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
29 CR2	Core 2 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
30 CR1	Core 1 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
31 CR0	Core 0 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting

25.3.16 Reset control word status registers 1-16 (DCFG_RCWSR n)

RCWSR n contains the reset configuration word information written with the RCW values read from non-volatile memory at device reset. See [Reset Configuration Word \(RCW\)](#) for more information.

Address: E_0000h base + 100h offset + (4d × i), where i=0d to 15d

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

DCFG_RCWSR n field descriptions

Field	Description
0–31 RCW	These registers capture the 512-byte RCW. Register n contains the read only-value of RCW bits (n-1)*32 through (n*32)-1 loaded in power-on reset's reset configuration stage.

25.3.17 Scratch read/write registers 1-4 (DCFG_SCRATCHRW n)

The SCRATCHRW n provides read / write scratch register locations available to the user.

NOTE

When performing secure boot, SCRATCHRW1 is used to store a 32-bit address used during the secure boot process to access the first location of external secure boot code. The PBL pre-boot initialization process must initialize these registers for proper secure boot operation.

Address: E_0000h base + 200h offset + (4d × i), where i=0d to 3d

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

DCFG_SCRATCHRW n field descriptions

Field	Description
0–31 VAL	32-bit scratch contents

25.3.18 Scratch write-once registers 1-4 (DCFG_SCRATCHW1Rn)

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a Power-on or Hard reset.

Address: E_0000h base + 300h offset + (4d × i), where i=0d to 3d

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

DCFG_SCRATCHW1Rn field descriptions

Field	Description
0–31 VAL	32-bit scratch contents

25.3.19 Core reset status register n (DCFG_CRSTSRn)

CRSTSR n contains the reset status bits for each core on the device.

A power-on reset of the device causes the following to occur:

1. RST_PORST is set.
2. All other bits are cleared.

A hard reset of the device causes the following to occur:

1. RST_PORST remains unchanged (PORST resources are not affected by hard reset).
2. RST_HRST is set.
3. All other bits are cleared.

If an application prefers that RST_PORST is not set after hard reset, then a write-1-clear must be performed on RST_PORST upon exiting power-on reset.

- Bits 0-27 reflect conditions for this specific core, i.e., not device conditions
- Bits 28-31 reflect device conditions which in turn affect this specific core

Device Configuration and Pin Control Memory Map/Register Definition

Address: E_0000h base + 400h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R					RST_WRT				RST_MPIC				RST_CORE			
	Reserved					Reserved				Reserved				Reserved		
W					w1c				w1c				w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	READY													RST_HRST		RST_PORST
									Reserved							
W	w1c													w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CRSTS_{Rn} field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 RST_WRT	Core was reset in response to watchdog timer expiration that was made visible to the platform 0 no reset 1 reset occurred and WRC/WRS=11
5–7 -	This field is reserved. Reserved
8 RST_MPIC	Core was reset in response to MPIC reset request 0 no reset 1 reset occurred
9–11 -	This field is reserved. Reserved
12 RST_CORE	Core was reset in response to internal core request to reset itself by setting bit DBCR0[RST] register 0 no reset 1 reset occurred
13–15 -	This field is reserved. Reserved

Table continues on the next page...

DCFG_CRSTS n field descriptions (continued)

Field	Description
16 READY	<p>Core ready pin.</p> <p>NOTE: This bit is cleared on a device power-on or hard reset. Upon completion of power-on or hard reset processing, this bit may be automatically set for a core if none of the conditions specified are true.</p> <p>Core is in the ready state after device has successfully passed through the System Ready point and the core is currently not in any of the following states:</p> <ul style="list-style-type: none"> • Core Dozing • Core Napping • Core Debug Halted • Core Debug Stopped • Core held in reset via a warm reset from the MPIC <p>0 Core n not ready 1 Core n ready</p>
17–29 -	This field is reserved. Reserved
30 RST_HRST	Core was reset due to a hard reset initiated by external assertion of HRESET
31 RST_PORST	Core was reset due to a PORSET

25.3.20 PCI Express n LIODN register (DCFG_PEX n LIODNR)

The PEX n LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: E_0000h base + 500h offset + (4d × i), where i=0d to 2d

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

Reset 0

DCFG_PEX n LIODNR field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–31 LDEVNUM	Logical Device Number

25.3.21 RIO n LIODN register (DCFG_RIOOnLIODNR)

The RIO n LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: E_0000h base + 510h offset + (4d × i), where i=0d to 1d

DCFG_RIOnLIODNR field descriptions

Field	Description
0–19 - Reserved	This field is reserved. Reserved
20–31 LDEVENUM	Logical Device Number

25.3.22 USB n LIODN register (DCFG_USBnLIODNR)

The USB n LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: E 0000h base + 520h offset + (4d × i), where i=0d to 1d

DCFG USBnLIODNR field descriptions

Field	Description
0–19 - 	This field is reserved. Reserved
20–31 LDEVENUM	Logical Device Number

25.3.23 eSDHC LIODN register (DCFG_eSDHCLIODNR)

The eSDHCLIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: E 0000h base + 530h offset = E 0530h

DCFG eSDHCLIODNR field descriptions

Field	Description
0–19 - Reserved	This field is reserved.
20–31 LDEVNUM	Logical Device Number

25.3.24 RIO message unit LIODN register (DCFG_RMULIODNR)

The RMULIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: E_{0000h} base + 540h offset = E_{0540h}

DCFG RMULIODNR field descriptions

Field	Description
0–19 - Reserved	This field is reserved.
20–31 LDEVNUM	Logical Device Number

25.3.25 DMA n LIODN register (DCFG_DMAnLIODNR)

The DMA n LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this. See [DMA in multi-partition systems](#) for information regarding special considerations for DMA related to the PAMU.

Address: E_0000h base + 580h offset + (4d × i), where i=0d to 1d

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

DCFG_DMAnLIODNR field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–31 LDEVNUM	Logical Device Number

25.3.26 PAMU bypass enable register (DCFG_PAMUBYPENR)

PAMUBYPENR contains control bits for enabling bypass mode on each PAMU.

Address: E_0000h base + 604h offset = E_0604h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																	PBYP[1-16]								Reserved							
W	n*	n*	n*	n*	n*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- PBYP[1-16] field: If secure boot is enabled (either by RCW[SB_EN] or the ITS fuse in the SFP block), these bits reset to 0; if secure boot is not enabled, these bits reset to 1.

DCFG_PAMUBYPENR field descriptions

Field	Description
0–15 PBYP[1-16]	PAMU Bypass n 0 PAMU n is not bypassed 1 PAMU n is bypassed

Table continues on the next page...

DCFG_PAMUBYPENR field descriptions (continued)

Field	Description
16–31 -	This field is reserved. Reserved

25.3.27 DMA control register (DCFG_DMCR1)

The DMCR1 contains bits for allowing DMA transactions from internal sources on the device.

Address: E_0000h base + 608h offset = E_0608h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA1_0	DMA1_1	DMA1_2	DMA1_3	DMA2_0	DMA2_1	DMA2_2	DMA2_3								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W									Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_DMCR1 field descriptions

Field	Description
0–1 DMA1_0	00 DMA1's Ch0 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved
2–3 DMA1_1	00 DMA1's Ch1 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved
4–5 DMA1_2	00 DMA1's Ch2 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved
6–7 DMA1_3	00 DMA1's Ch3 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved
8–9 DMA2_0	00 DMA2's Ch0 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved

Table continues on the next page...

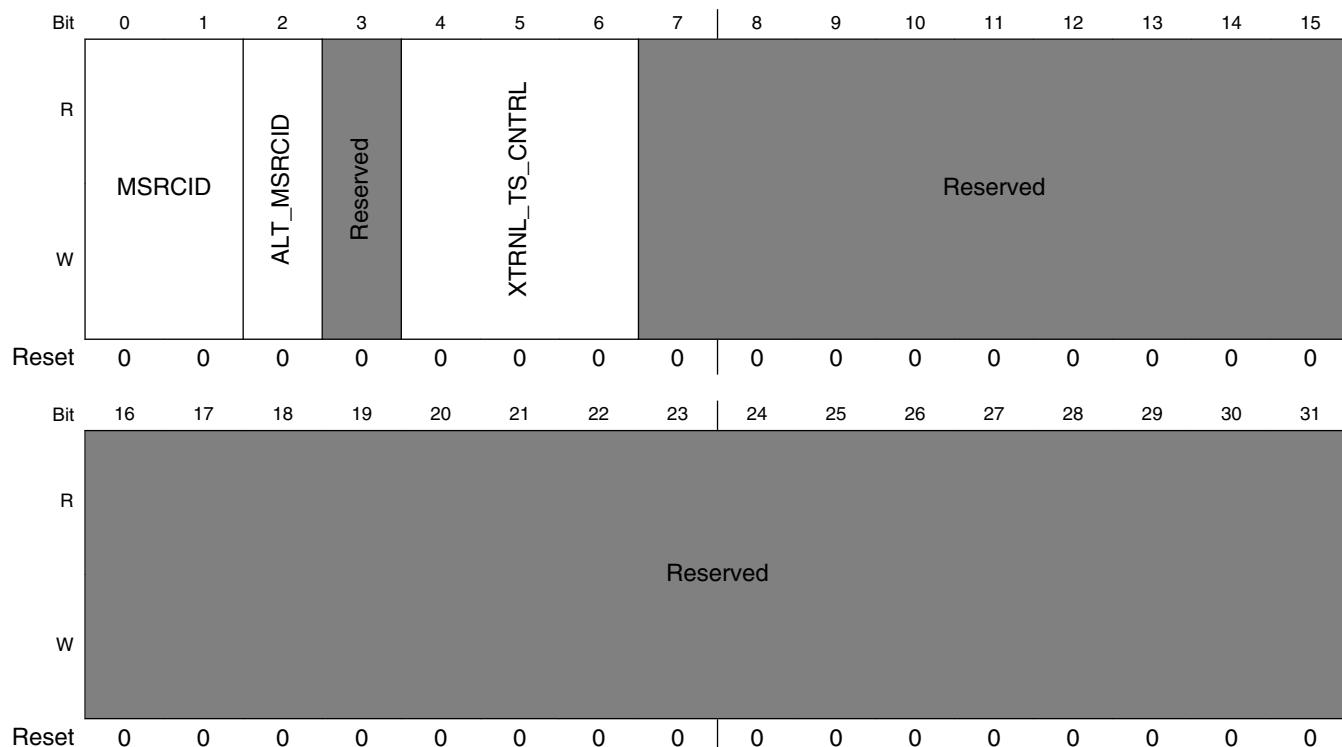
DCFG_DMCR1 field descriptions (continued)

Field	Description	
10–11 DMA2_1	00 DMA2's Ch1 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved	
12–13 DMA2_2	00 DMA2's Ch2 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved	
14–15 DMA2_3	00 DMA2's Ch3 I/F <- EPU 01 Reserved 10 Reserved 11 Reserved	
16–31 -	This field is reserved. Reserved	

25.3.28 Pin Multiplexing Control Register (DCFG_PMUXCR)

The PMUXCR allows for pin muxing control, beyond that conveyed by RCW and/or POR configuration pins, for pins shared by multiple IP blocks.

Address: E_0000h base + E00h offset = E_0E00h



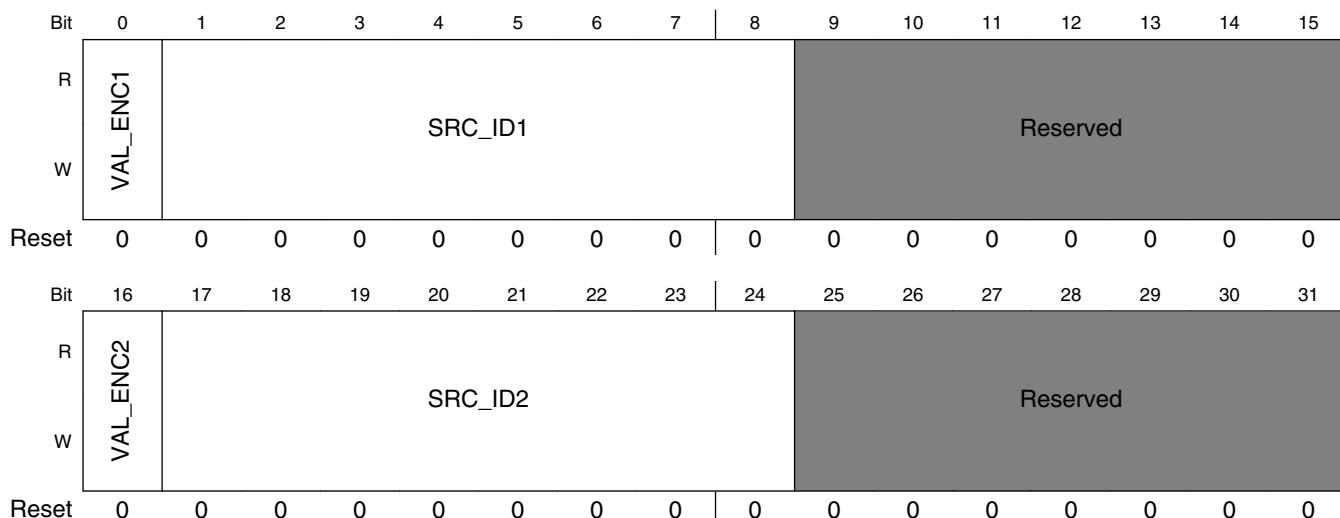
DCFG_PMUXCR field descriptions

Field	Description
0–1 MSRCID	Indicates which memory interface's source ID information is broadcast on the dedicated MSRCID pins. 00 Reserved 01 Reserved 10 eLBC
2 ALT_MSRCID	Indicates which memory interface's source ID information is broadcast on the DMA2 pins, when those pins are selected as the alternate output function 0 Reserved (default) 1 eLBC
3 -	This field is reserved. Reserved
4–6 XTRNL_TS_ CNTRL	External timestamp control. XTRNL_TS_CNTRL configures the routing of the timestamp device level pins (EC_XTRNL_RX_STMP n and EC_XTRNL_TX_STMP n) to the possible receivers on Frame Managers 1 and 2. The user must additionally select, via individual 1G MAC configuration registers, whether that given 1G MAC will be enabled to use the external timestamp signaling. See the <i>QorIQ Data Path Acceleration Architecture (DPA) Reference Manual</i> (DPAARM) for more information. NOTE: Note that external SFD can be enabled for only one dTSEC in a given frame manager and only one 1G MAC can have external timestamping enabled for a given set of EC STMP n signals. NOTE: The 10G MAC's do not support external timestamping. 0xx (default) EC_XTRNL_RX_STMP1 and EC_XTRNL_TX_STMP1 timestamp signals routed to all four 1G MAC's of FMan1. EC_XTRNL_RX_STMP2 and EC_XTRNL_TX_STMP2 timestamp signals routed to all four 1G MAC's of FMan2. 100 EC_XTRNL_RX_STMP1 and EC_XTRNL_TX_STMP1 timestamp signals routed to 1G MAC1 of FMan1, and EC_XTRNL_RX_STMP2 and EC_XTRNL_TX_STMP2 timestamp signals routed to 1G MAC's 2, 3, 4 of FMan1. This allows for external timestamping support for a RGMII and SGMII on FMan1. 101 EC_XTRNL_RX_STMP1 and EC_XTRNL_TX_STMP1 timestamp signals routed to 1G MAC's 1 and 2 of FMan1, and EC_XTRNL_RX_STMP2 and EC_XTRNL_TX_STMP2 timestamp signals routed to 1G MAC's 3 and 4 of FMan1. This allows for external timestamping support for two SGMIIIs, while RGMII is used on FMan1. 110 EC_XTRNL_RX_STMP1 and EC_XTRNL_TX_STMP1 timestamp signals routed to 1G MAC1 of FMan2, and EC_XTRNL_RX_STMP2 and EC_XTRNL_TX_STMP2 timestamp signals routed to 1G MAC's 2, 3, 4 of FMan2. This allows for external timestamping support for a RGMII and SGMII on FMan2. 111 EC_XTRNL_RX_STMP1 and EC_XTRNL_TX_STMP1 timestamp signals routed to 1G MAC's 1 and 2 of FMan2, and EC_XTRNL_RX_STMP2 and EC_XTRNL_TX_STMP2 timestamp signals routed to 1G MAC's 3 and 4 of FMan2. This allows for ext timestamping support for two SGMIIIs, while RGMII is used on FMan2.
7–31 -	This field is reserved. Reserved

25.3.29 Debug Source ID 1A (DCFG_DSRCID1A)

The DSRCID1A allows for mapping specific 8-bit source ID values to 3-bit encoded values 0b001, and 0b010, to be sent out on debug pins MSRCID[0:2]. Note that the encoded value of 0b000 on MSRCID[0:2] pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID1A-DSRCID1D.

Address: E_0000h base + E20h offset = E_0E20h



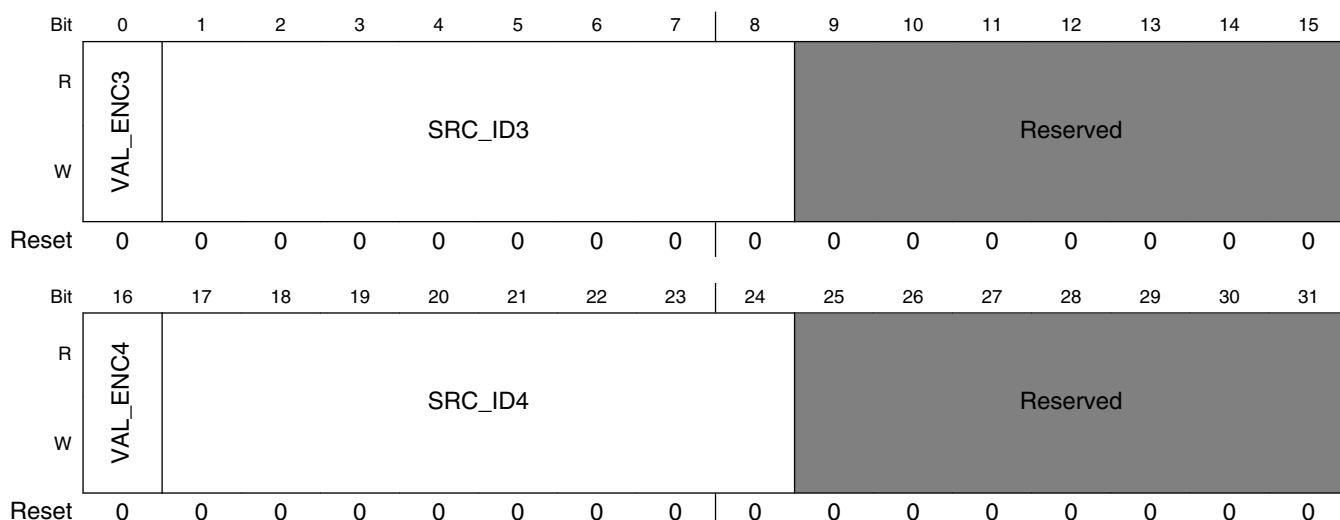
DCFG_DSRCID1A field descriptions

Field	Description
0 VAL_ENC1	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID1 field.
1–8 SRC_ID1	Source ID value to be mapped to encoded value 0b001 on pins MSRCID[0:2].
9–15 -	This field is reserved. Reserved
16 VAL_ENC2	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID2 field.
17–24 SRC_ID2	Source ID value to be mapped to encoded value 0b010 on pins MSRCID[0:2].
25–31 -	This field is reserved. Reserved

25.3.30 Debug Source ID 1B (DCFG_DSRCID1B)

The DSRCID1B allows for mapping specific 8-bit source ID values to 3-bit encoded values 0b011, and 0b100, to be sent out on debug pins MSRCID[0:2]. Note that the encoded value of 0b000 on MSRCID[0:2] pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID1A-DSRCID1D.

Address: E_0000h base + E24h offset = E_0E24h



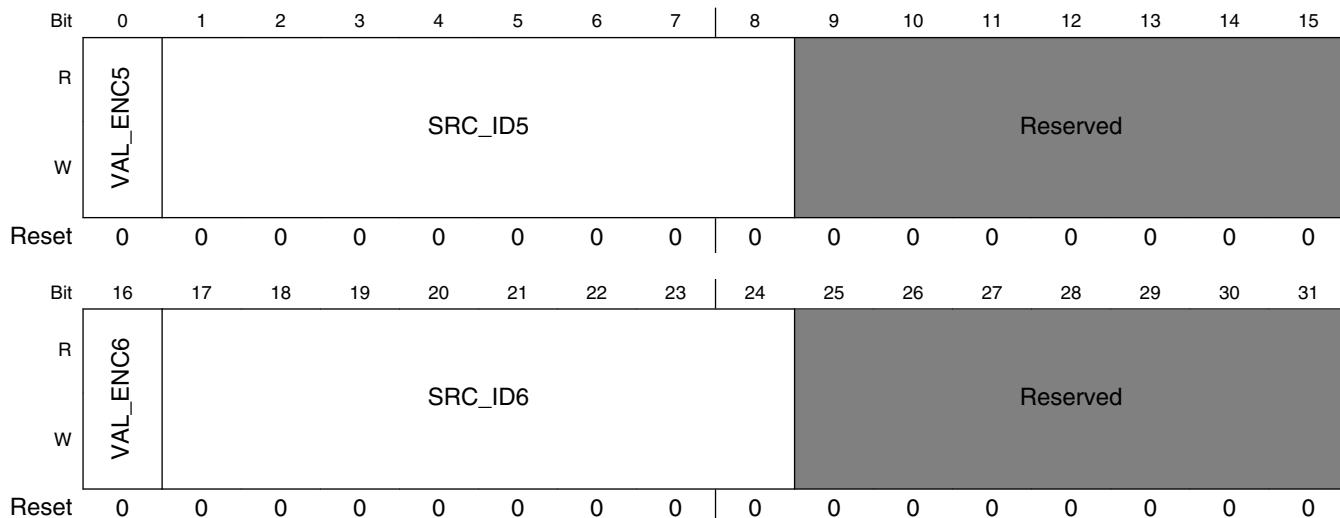
DCFG_DSRCID1B field descriptions

Field	Description
0 VAL_ENC3	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID3 field.
1–8 SRC_ID3	Source ID value to be mapped to encoded value 0b011 on pins MSRCID[0:2].
9–15 -	This field is reserved. Reserved
16 VAL_ENC4	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID4 field.
17–24 SRC_ID4	Source ID value to be mapped to encoded value 0b100 on pins MSRCID[0:2].
25–31 -	This field is reserved. Reserved

25.3.31 Debug Source ID 1C (DCFG_DSRCID1C)

The DSRCID1C allows for mapping specific 8-bit source ID values to 3-bit encoded values 0b101, and 0b110, to be sent out on debug pins MSRCID[0:2]. Note that the encoded value of 0b000 on MSRCID[0:2] pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID1A-DSRCID1D.

Address: E_0000h base + E28h offset = E_0E28h



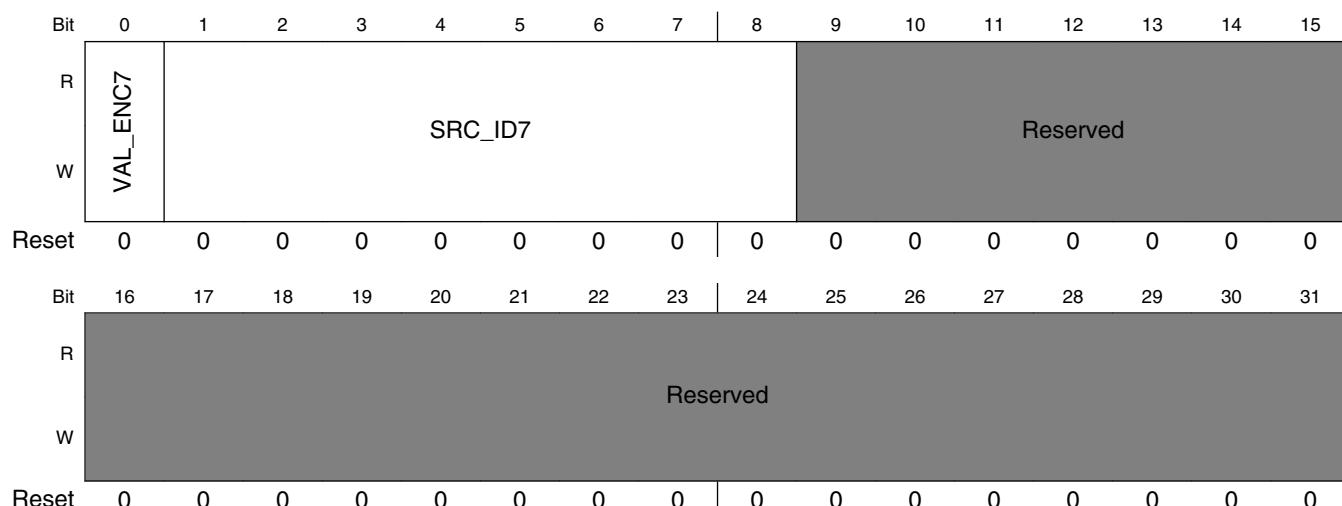
DCFG_DSRCID1C field descriptions

Field	Description
0 VAL_ENC5	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID5 field.
1–8 SRC_ID5	Source ID value to be mapped to encoded value 0b101 on pins MSRCID[0:2].
9–15 -	This field is reserved. Reserved
16 VAL_ENC6	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID6 field.
17–24 SRC_ID6	Source ID value to be mapped to encoded value 0b110 on pins MSRCID[0:2].
25–31 -	This field is reserved. Reserved

25.3.32 Debug Source ID 1D (DCFG_DSRCID1D)

The DSRCID1D allows for mapping specific 8-bit source ID values to 3-bit encoded value 0b111, to be sent out on debug pins MSRCID[0:2]. Note that the encoded value of 0b000 on MSRCID[0:2] pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID1A-DSRCID1D.

Address: E_0000h base + E2Ch offset = E_0E2Ch



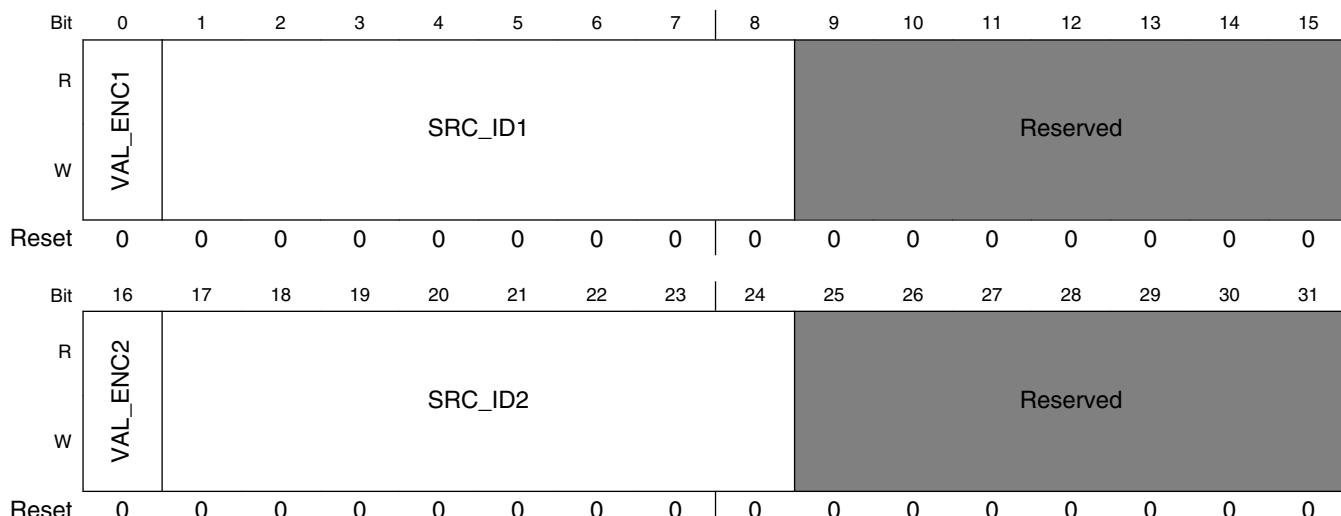
DCFG_DSRCID1D field descriptions

Field	Description
0 VAL_ENC7	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID7 field.
1–8 SRC_ID7	Source ID value to be mapped to encoded value 0b111 on pins MSRCID[0:2].
9–31 -	This field is reserved. Reserved

25.3.33 Debug Source ID 2A (DCFG_DSRCID2A)

The DSRCID2A allows for mapping specific 8-bit source ID values to 2-bit encoded values 0b01, and 0b10, to be sent out on DMA2 pins DMA2_DACK0_B and DMA2_DDONE0_B, when ALT_MSRCID is selected as the alternate output function per the RCW[DMA2] configuration field. Note that the encoded value of 0b00 on DMA2 pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID2A-DSRCID2B.

Address: E_0000h base + E30h offset = E_0E30h



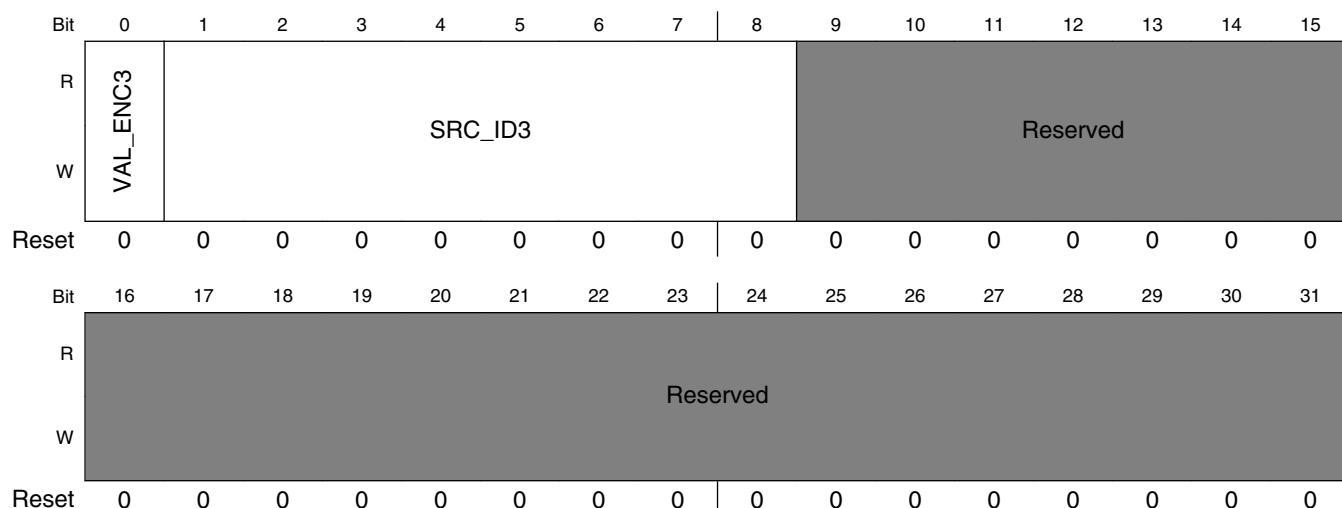
DCFG_DSRCID2A field descriptions

Field	Description
0 VAL_ENC1	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID1 field.
1–8 SRC_ID1	Source ID value to be mapped to encoded value 0b01 on DMA2 pins DMA2_DACK0_B and DMA2_DDONE0_B.
9–15 -	This field is reserved. Reserved
16 VAL_ENC2	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID2 field.
17–24 SRC_ID2	Source ID value to be mapped to encoded value 0b10 on DMA2 pins DMA2_DACK0_B and DMA2_DDONE0_B.
25–31 -	This field is reserved. Reserved

25.3.34 Debug Source ID 2B (DCFG_DSRCID2B)

The DSRCID2B allows for mapping specific 8-bit source ID values to 2-bit encoded value 0b11, to be sent out on DMA2 pins DMA2_DACK0_B and DMA2_DDONE0_B, when ALT_MSRCID is selected as the alternate output function per the RCW[DMA2] configuration field. Note that the encoded value of 0b00 on DMA2 pins identifies that the architected source ID associated with the transaction on the memory interface being monitored is not a match on any of the validated architected source IDs as defined by the VAL_ENC_x and SRC_ID_x fields described for registers DSRCID2A-DSRCID2B.

Address: E_0000h base + E34h offset = E_0E34h



DCFG_DSRCID2B field descriptions

Field	Description
0 VAL_ENC3	Indicates that a valid architected source ID to debug source ID mapping exists, as determined by the contents of SRC_ID3 field.
1–8 SRC_ID3	Source ID value to be mapped to encoded value 0b11 on DMA2 pins DMA2_DACK0_B and DMA2_DDONE0_B.
9–31 -	This field is reserved. Reserved

25.3.35 I/O Voltage Selection Status Register (DCFG_IOVSELSR)

The IOVSELSR allows to read the state of the IO_VSEL[0:4] pins which select the I/O voltage for pins on the BVDD, CVDD, and LVDD I/O voltage planes. See [Table 3-2](#) for more information.

Address: E_0000h base + E40h offset = E_0E40h

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

DCFG_IOVSELSR field descriptions

Field	Description
0–4 IO_VSEL	State of the IO_VSEL[0:4] signals sampled at PORRESET, which configure the I/O voltage for the BVDD, CVDD, and LVDD power planes according to Table 3-2 .
5–31 -	This field is reserved. Reserved

25.3.36 DDR Clock Disable Register (DCFG_DDRCLKDR)

DDRCLKDR allows for specific, unused clocks of both of the DDR Controllers' interfaces to be released to high impedance, thereby reducing power consumption.

Address: E_0000h base + E60h offset = E_0E60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved								D1_MCK0_DIS	D1_MCK1_DIS	D1_MCK2_DIS	D1_MCK3_DIS	D1_MCK4_DIS	D1_MCK5_DIS		
W									0	0	0	0	0	0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								D2_MCK0_DIS	D2_MCK1_DIS	D2_MCK2_DIS	D2_MCK3_DIS	D2_MCK4_DIS	D2_MCK5_DIS		
W									0	0	0	0	0	0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

DCFG_DDRCLKDR field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10 D1_MCK0_DIS	DDR Controller 1 clock 0 disable. The output is tri-stated, when disabled. 0 D1_MCK[0] is enabled 1 D1_MCK[0] is disabled
11 D1_MCK1_DIS	DDR Controller 1 clock 1 disable. The output is tri-stated, when disabled. 0 D1_MCK[1] is enabled 1 D1_MCK[1] is disabled
12 D1_MCK2_DIS	DDR Controller 1 clock 2 disable. The output is tri-stated, when disabled. 0 D1_MCK[2] is enabled 1 D1_MCK[2] is disabled
13 D1_MCK3_DIS	DDR Controller 1 clock 3 disable. The output is tri-stated, when disabled. 0 D1_MCK[3] is enabled 1 D1_MCK[3] is disabled
14 D1_MCK4_DIS	DDR Controller 1 clock 4 disable. The output is tri-stated, when disabled. 0 D1_MCK[4] is enabled 1 D1_MCK[4] is disabled
15 D1_MCK5_DIS	DDR Controller 1 clock 5 disable. The output is tri-stated, when disabled. 0 D1_MCK[5] is enabled 1 D1_MCK[5] is disabled
16–25 -	This field is reserved. Reserved
26 D2_MCK0_DIS	DDR Controller 2 clock 0 disable. The output is tri-stated, when disabled. 0 D2_MCK[0] is enabled 1 D2_MCK[0] is disabled
27 D2_MCK1_DIS	DDR Controller 2 clock 1 disable. The output is tri-stated, when disabled. 0 D2_MCK[1] is enabled 1 D2_MCK[1] is disabled
28 D2_MCK2_DIS	DDR Controller 2 clock 2 disable. The output is tri-stated, when disabled. 0 D2_MCK[2] is enabled 1 D2_MCK[2] is disabled
29 D2_MCK3_DIS	DDR Controller 2 clock 3 disable. The output is tri-stated, when disabled. 0 D2_MCK[3] is enabled 1 D2_MCK[3] is disabled
30 D2_MCK4_DIS	DDR Controller 2 clock 4 disable. The output is tri-stated, when disabled. 0 D2_MCK[4] is enabled 1 D2_MCK[4] is disabled

Table continues on the next page...

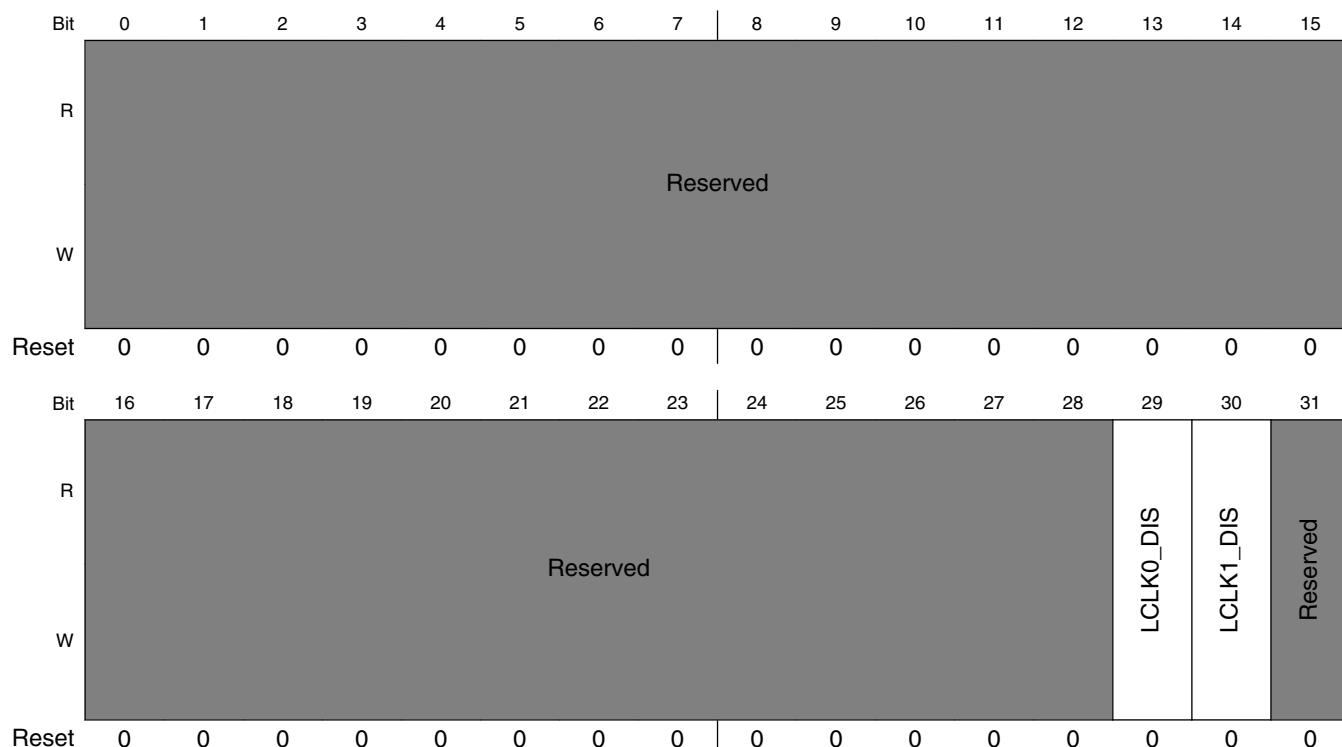
DCFG_DDRCLKDR field descriptions (continued)

Field	Description
31 D2_MCK5_DIS	DDR Controller 2 clock 5 disable. The output is tri-stated, when disabled. 0 D2_MCK[5] is enabled 1 D2_MCK[5] is disabled

25.3.37 eLBC Clock Disable Register (DCFG_ELBCCCLKDR)

The ELBCCCLKDR allows for specific, unused clocks of the eLBC interface to be not driven/high-impedance, thereby reducing power consumption.

Address: E_0000h base + E68h offset = E_0E68h



DCFG_ELBCCCLKDR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29 LCLK0_DIS	eLBC clock 0 disable. The output is not driven and in a high impedance state, when disabled. 0 LCLK[0] is enabled 1 LCLK[0] is disabled

Table continues on the next page...

DCFG_ELBCLKDR field descriptions (continued)

Field	Description
30 LCLK1_DIS	eLBC clock 1 disable. The output is not driven and in a high impedance state, when disabled. 0 LCLK[1] is enabled 1 LCLK[1] is disabled
31 -	This field is reserved. Reserved

25.3.38 eSDHC Polarity Configuration Register (DCFG_SDHCPCR)

The SDHCPCR allows for specific polarity control of the eSDHC input signals.

Address: E_0000h base + E80h offset = E_0E80h

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

DCFG_SDHCPCR field descriptions

Field	Description
0 CD_INV	eSDHC Card Detect Invert 0 SDHC_CD signal to the eSDHC module has the same polarity as the I/O Pin. 1 SDHC_CD signal to the eSDHC module has inverted polarity in relation to the I/O Pin.
1 WP_INV	eSDHC Write Protect Invert 0 SDHC_WP signal to the eSDHC module has the same polarity as the I/O Pin. 1 SDHC_WP signal to the eSDHC module has inverted polarity in relation to the I/O Pin.
2–31 -	This field is reserved. Reserved

Chapter 26

Run Control/Power Management (RCPM)

26.1 RCPM overview

This chapter provides the specification and programming model for the power management functions of the run control/power management unit (RCPM).

The run control/power management unit (RCPM) performs all device-level tasks associated with device run control and power management.

The RCPM communicates extensively with the embedded cores and coherency fabric to achieve both device run-control and power-management functions.

This document focuses solely on the mechanisms of the RCPM unit that control the power management aspects of the device and each of the processor cores within the device.

Dynamic power management minimizes power consumption when a block is idle. Software can access RCPM memory-mapped registers (CDOZCR, CNAPCR, and POWMGTCSR[SLP]) to put specific cores in doze/nap state or put the device in sleep mode. The RCPM supports several wake up sources through internal timers and internal and external interrupts.

26.1.1 Power Management Features

Power management functions include the following:

- Software-controlled core doze/nap and device sleep mode
- Supports OpenPIC interrupt based wake-up and the following wake-up sources:
 - Wake on internal timer event
 - Wake on internal and external interrupt event
- Enter device sleep through core or external master (for example, PCI Express).
- Core power management
 - Independent power management control of each core

- Core doze state where:
 - Instruction fetching is suspended
 - All previously fetched instructions are completed
 - Core is halted but clocks remain active
- Core nap state where:
 - Entered from the core doze state
 - Core caches (for the napping cores) must be explicitly flushed before entering core nap to maintain cache coherence.
 - Clocks are gated off (however, the timebase continues to run)
- Entry into core power management states via:
 - RCPM CDOZCR for request to enter the core's doze state
 - RCPM CNAPCR for request to enter the core's nap state
 - Event processing unit (EPU) triggering to enter the core's doze mode
- Independent core wake-up from:
 - Unmasked interrupt request
 - Unmasked critical interrupt
 - Debug interrupt
- Device power management:
 - Device sleep state where:
 - All cores are in core nap
 - IP blocks are halted and clocks gated off
 - Entry into device sleep state by setting POWMGTCR[SLP]
 - Selective power down of IP blocks using IPPDEXPR
 - Independent device wake-up from:
 - Unmasked interrupt request
 - Unmasked critical interrupt
 - Unmasked machine check
 - Unmasked NMI
- Core control
 - Per core enabling/disabling of core timebase
 - Per core selection of timebase clock

26.1.2 Modes of Operation

The RCPM unit supports the following power management modes of operation:

- Reset modes
- Power management modes
 - Core(s) full on mode
 - Core(s) in DOZE state
 - Core(s) in NAP state

- Device full on mode
- Device in SLEEP state

26.1.2.1 Mode Summary-Core and Device Levels

Table 26-1 summarizes the core/device power management modes.

Table 26-1. Core Power Management Mode Summary

Device Mode Name	Re su ma ble	Servic es Snoo ps	Device Clocks	Core Clocks	Book E Core State Entry	Description
Power Management Modes						
Core Dozing	Y	Y	Y	Y	Seq: • Full On -> Core Halted	The core referred to is currently in the <i>dozing</i> state.
Core Napping	Y	N	Y	N	Seq: • Full On -> Core Halted • Core Halted -> Core Stopped	The core referred to is currently in the <i>napping</i> state.
NOTE: Coherency maintained only if all core caches (L1, L2) are flushed before entering core nap mode.						

Table 26-2. Device Power Management Mode Summary

Device Mode Name	Re su ma ble	Servic es Snoo ps	Device Clocks	Core Clocks	Book E Core State Entry	Description
Power Management Modes						
Device Sleeping	Y	N	N	N	Seq: • Full On -> Core Halted • Core Halted -> Core Stopped	Platform clock to all IP are gated off. Core timebase is turned-off.

26.1.2.2 Modes Entry and Exit

Table 26-3 summarizes the core power management modes entry and exit.

Table 26-3. Core Power Management Mode: Entry and Exit

Device Mode Name	Entry via ...	Exit via ...	Book E Core State Entry
Power Management Modes			
Core Dozing	<ul style="list-style-type: none"> Hypervisor set corresponding bit in RCPM CDOZCR[C_DOZ_RQ] to 1 	<ul style="list-style-type: none"> Clear CDOZCR[C_DOZ_RQ] Core interrupt without RCPM CPMIMR masking. Core critical interrupt without RCPM CPMCIMR masking. Core external machine check without RCPM CPMMCIMR masking. Core NMI without RCPM CPMNMIMR masking. The following core wake up events will unconditionally wake up core from power management state: <ul style="list-style-type: none"> Decrementer exception (qualified by MSR[EE] MSR[GS]) FIT exception (qualified by MSR[EE] MSR[GS]) Watchdog exception (qualified by MSR[CE] MSR[GS]) Performance interrupt (qualified by MSR[EE] MSR[GS]) Machine check asynchronous exception (exclude external initiate machine check and NMI) (qualified by MSR[ME] MSR[GS]). This includes core unrecoverable failure during snoop processing. Doorbell interrupt (qualified by MSR[EE] MSR[GS]). Doorbell critical interrupt (qualified by MSR[CE] MSR[GS]). Guest doorbell (qualified by MSR[EE] MSR[GS]). Guest doorbell critical (qualified by MSR[CE] & MSR[GS]). Guest doorbell machine check event (qualified by MSR[ME] & MSR[GS]). Core Hard reset event Core debug halt request Wake on LAN - Magic packet, if programmed Wake on USB - plug/unplug, if programmed Wake on eSDHC - Card detect, if programmed Wake on GPIO if programmed 	Seq: <ul style="list-style-type: none"> Full On -> Core Halted
Core Napping	<ul style="list-style-type: none"> Hypervisor set corresponding bit in RCPM CNAPCR[C_NAP_RQ] to 1. MPIC core warm reset request assertion¹ Core disable request by setting COREDISR[CORE] = 1 	<ul style="list-style-type: none"> Clear RCPM CNAPCR[C_NAP_RQ] Core interrupt without RCPM CPMIMR masking. Core critical interrupt without RCPM CPMCIMR masking. Core external machine check without RCPM CPMMCIMR masking. Core NMI without RCPM CPMNMIMR masking. 	Seq: <ul style="list-style-type: none"> Full On -> Core Halted Core Halted -> Core Stopped

Table 26-3. Core Power Management Mode: Entry and Exit

Device Mode Name	Entry via ...	Exit via ...	Book E Core State Entry
		<ul style="list-style-type: none"> The following core wake up events will unconditionally wake up core from power management state: <ul style="list-style-type: none"> Decrementer exception (qualified by MSR[EE] MSR[GS]) FIT exception (qualified by MSR[EE] MSR[GS]) Watchdog exception (qualified by MSR[CE] MSR[GS]) MPIC core warm reset request deassertion Core debug halt request Wake on LAN - Magic packet, if programmed Wake on USB - plug/unplug, if programmed Wake on eSDHC - Card detect, if programmed Wake on GPIO if programmed 	

1. A warm reset request is serviced by placing the core in the nap state prior to reset assertion in order for the CCM to be able to properly handle the core being reset.

Table 26-4 summarizes the device sleeping power management mode entry and exit.

Table 26-4. Device Power Management Mode: Entry and Exit

Device Mode Name	Entry via ...	Exit via ...	Book E Core State Entry
Power Management Modes			
Device Sleeping	<ul style="list-style-type: none"> Set RCPM POWMGTCSR[SLP] POWMGTCSR[SLP] to 1 	<ul style="list-style-type: none"> Clear RCPM POWMGTCSR[SLP] Core interrupt without RCPM/CPMIMR masking. Core critical interrupt without RCPM/CPMCIMR masking. Core external machine check without RCPM/CPMMCMR masking. Core NMI without RCPM/CPMNIMR masking. The following core wake up events will unconditionally wake up core from power management state: <ul style="list-style-type: none"> Decrementer exception (qualified by MSR[EE] MSR[GS]) FIT exception (qualified by MSR[EE] MSR[GS]) Watchdog exception (qualified by MSR[CE] MSR[GS]) Core debug halt request Wake on LAN - Magic packet, if programmed Wake on USB - plug/unplug, if programmed Wake on eSDHC - Card detect, if programmed Wake on GPIO if programmed 	Seq: <ul style="list-style-type: none"> Full On -> Core Halted Core Halted -> Core Stopped

26.1.2.3 Reset Modes

26.1.2.3.1 Power-On Reset State

During power-on reset, the following hardware structures are reset:

- All CCSR registers
- SoC power management state machine
- Core power management state machine
- Interrupt sampling module

26.1.2.3.2 Hard Reset State

During hard reset, the following are affected:

- All CCSR registers
- SoC power management state machine
- Core power management state machine

26.1.2.4 Power Management Modes

Hypervisor software can place the device in core doze or nap mode by writing to RCPM CDOZCR or CNAPCR registers. In addition, Hypervisor or external masters can write to memory-mapped POWMGTCR in RCPM to cause the device to enter sleep mode.

26.1.2.4.1 Core Dozing Mode

In core dozing mode, the core suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1/L2 cache are still supported and thus the data in the data cache is kept coherent. Interrupts directed to the core are monitored by the device and cause RCPM to use the defined handshake mechanism to exit the core from doze mode to allow the core to recognize and process the interrupt.

The core's timer facilities are still enabled during core doze mode, and core timebase interrupts can be generated. All device logic external to the core remains fully operational in core doze mode.

26.1.2.4.2 Core Napping Mode

In core napping mode, all clocks internal to the core are turned off except for its timer facilities clock (the core timebase). The L1/L2 cache do not respond to snoops in core nap mode, so if coherency is required, the L1/L2 cache must be flushed before entering core nap mode.

All device logic external to core remains fully operational in core nap mode.

It is not possible for the device to place the core into nap mode. This is because the device cannot initiate a cache flush of the core.

26.1.2.4.3 Device Sleeping Mode

In device sleeping mode, all clocks internal to the core are turned off as well as the clock in device logic so that only the modules which are required to wake up the device still has a running clock. Core timebase is turned-off.

The modules which can be used as a wake-up source are internal timers, internal and external interrupts.

After the core and I/O interfaces have shut down, ASLEEP pin is asserted.

26.2 RCPM Signal Descriptions

[Table 26-5](#) describes the RCPM signals.

Table 26-5. RCPM Detailed Signal Descriptions

Signal	I/ O	Description
ASLEEP	O	Asleep. After negation of PORESET_B, ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.
		State Meaning Asserted- Indicates that the device is either still in its power-on reset sequence or it has reached a sleep state after a power-down command is issued by software. Negated- The device is not in sleep mode. (It has either exited from a power-down state, or has completed the POR sequence.)
		Timing Assertion- May occur at any time; may be asserted asynchronously to the input clocks. Negation- Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous.
	I	Real Time Clock. Refer to the chip hardware specifications for timing specification. The signal can be optionally use to clock the global timers in the programmable interrupt controller.
CKSTP_OUT_B	O	Timing Assertion/Negation - See the hardware specification of device for specific timing information for this signal.
		Checkstop Out

Table continues on the next page...

Table 26-5. RCPM Detailed Signal Descriptions (continued)

Signal	I/O	Description
	State Meaning	Asserted- Indicates that the device is in a checkstop state. The rest of the device logic remains functional. Negated- Indicates normal operation. After CKSTP_OUT_B has been asserted, it is negated after the next negation (low-to-high transition) of PORESET_B.
	Timing	Assertion- May occur at any time; may be asserted asynchronously to the input clocks. Negation- Must remain asserted until the device has been reset with a PORESET_B.

26.3 RCPM Memory Map/Register Definition

This section identifies power management resources that are not included as part of a processor core or platform IP.

RCPM memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_2004	Core doze status register (RCPM_CDOZSR)	32	R	0000_0000h	26.3.1/1849
E_200C	Core doze control register (RCPM_CDOZCR)	32	R/W	0000_0000h	26.3.2/1850
E_2014	Core nap status register (RCPM_CNAPSR)	32	R	0000_0000h	26.3.3/1850
E_201C	Core nap control register (RCPM_CNAPCR)	32	R/W	0000_0000h	26.3.4/1851
E_2024	Core doze previous status register (RCPM_CDOZPSR)	32	w1c	0000_0000h	26.3.5/1852
E_202C	Core nap previous status register (RCPM_CNAPPSR)	32	w1c	0000_0000h	26.3.6/1852
E_2034	Core wait status register (RCPM_CWAITSR)	32	R	0000_0000h	26.3.7/1853
E_203C	Core watchdog detect status register (RCPM_CWDTDTSR)	32	R	0000_0000h	26.3.8/1854
E_2040	Power management control and status register (RCPM_POWMGTCR)	32	R/W	0000_0000h	26.3.9/1855
E_2050	IP powerdown exception control register (RCPM_IPPDEXPCR)	32	R/W	0000_0000h	26.3.10/1856
E_2064	Core Power Management Interrupt Mask Register (RCPM_CPMIMR)	32	R/W	0000_0000h	26.3.11/1859
E_206C	Core Power Management Critical Interrupt Mask Register (RCPM_CPMCIMR)	32	R/W	0000_0000h	26.3.12/1859
E_2074	Core Power Management Machine Check Mask Register (RCPM_CPMMMCIMR)	32	R/W	0000_0000h	26.3.13/1860
E_207C	Core Power Management NMI Mask Register (RCPM_CPMNMIMR)	32	R/W	0000_0000h	26.3.14/1861
E_2084	Core timebase enable register (RCPM_CTBENR)	32	R/W	0000_0000h	26.3.15/1861

Table continues on the next page...

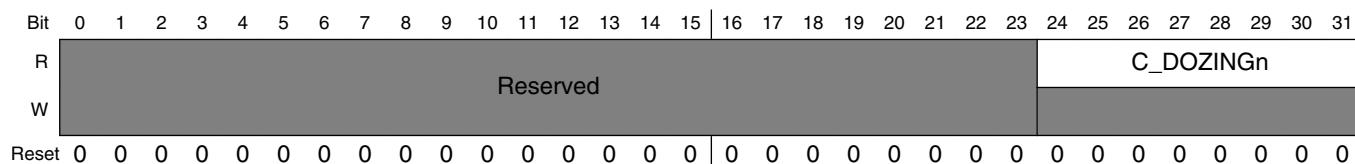
RCPM memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_208C	Core timebase clock select register (RCPM_CTBCKSELR)	32	R/W	0000_0000h	26.3.16/ 1862
E_2094	Core timebase halt control register (RCPM_CTBHLTCR)	32	R/W	0000_0000h	26.3.17/ 1862
E_20A4	Core machine check mask control register (RCPM_CMCPMASKCR)	32	R/W	0000_0000h	26.3.18/ 1863

26.3.1 Core doze status register (RCPM_CDOZSR)

The core doze status register (CDOZSR) is used for reporting doze status per core.

Address: E_2000h base + 4h offset = E_2004h

**RCPM_CDOZSR field descriptions**

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_DOZINGn	Core doze status. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Core is not in the core dozing mode 1 Core is in the core dozing mode

26.3.2 Core doze control register (RCPM_CDOZCR)

The core doze control register (CDOZCR) is used for requesting that a core be placed in the doze state.

Address: E_2000h base + Ch offset = E_200Ch

RCPM CDOZCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_DOZ_RQn	<p>Core doze request</p> <p>These bits are cleared by the interrupt event.</p> <p>NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.</p> <p>0 No request to put this core into its doze mode</p> <p>1 Request to put this core into its doze mode</p>

26.3.3 Core nap status register (RCPM_CNAPSR)

The core nap status register (CNAPSR) is used for reporting nap status per core.

Address: E 2000h base + 14h offset = E 2014h

RCPM CNAPSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_NAPPINGn	Core nap status NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.

Table continues on the next page...

RCPM_CNAPSR field descriptions (continued)

Field	Description
0	Core is not in its napping mode
1	Core is in its napping mode

26.3.4 Core nap control register (RCPM_CNAPCR)

The core nap control register (CNAPCR) is used for requesting that a core be placed in the nap state. Each core must be put into nap mode individually. The recommended sequence for putting multiple cores into nap mode is as follows:

1. Write a 1 to the CNAPCR[C_NAP_RQ n] bit of the first core to be put into nap mode.
2. Read the CNAPCR; this pushes the previous write.
3. Repeat steps 1 and 2 for the next core to be put into nap mode.
4. Repeat steps 1 and 2 for each subsequent core until all the desired cores have been put into nap mode.

Similarly, each core must be brought out of nap mode individually by clearing the corresponding bit in the CNAPCR and then reading back the CNAPCR to push each individual write.

Address: E_2000h base + 1Ch offset = E_201Ch

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

RCPM_CNAPCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_NAP_RQn	Core nap request These bits are cleared by the interrupt event. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 No request to put this core into its nap mode 1 Request to put this core into its nap mode

26.3.5 Core doze previous status register (RCPM_CDOZPSR)

The core doze previous status register (CDOZPSR) is used for reporting previous doze status per core. It is used by software to determine which power saving state a core was in before it was awakened by an interrupt.

Address: E_2000h base + 24h offset = E_2024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RCPM_CDOZPSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_PDOZINGn	Core doze previous status These bits are set when the corresponding CDOZSR bit has a 1-to-0 transition that it is caused by an interrupt. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Core was not in the core doze mode before being awakened by an interrupt 1 Core was in the core doze mode before being awakened by an interrupt

26.3.6 Core nap previous status register (RCPM_CNAPPSR)

The core nap previous status register (CNAPPSR) is used for reporting the previous nap status per core. It is used by software to determine which power saving state a core was in before it was awakened by an interrupt.

Address: E_2000h base + 2Ch offset = E_202Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RCPM_CNAPPSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_PNAPINGn	Core nap previous status. These bits are set when the corresponding CNAPSR bit has a 1-to-0 transition that is caused by an interrupt. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Core was not in the core nap mode before being awakened by an interrupt. 1 Core was in the core nap mode before being awakened by an interrupt.

26.3.7 Core wait status register (RCPM_CWAITSR)

The core wait status register (CWAITSR) is used for reporting wait status per core.

Address: E_2000h base + 34h offset = E_2034h

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

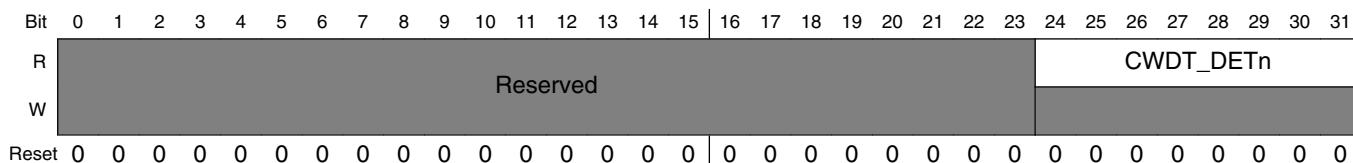
RCPM_CWAITSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 C_WAITINGn	Core wait status NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Core is not in its wait mode 1 Core is in its wait mode

26.3.8 Core watchdog detect status register (RCPM_CWDTDSR)

The core WDT detect status register (CWDTDSR) is used for reporting detection of WDT timeout per core .

Address: E_2000h base + 3Ch offset = E_203Ch



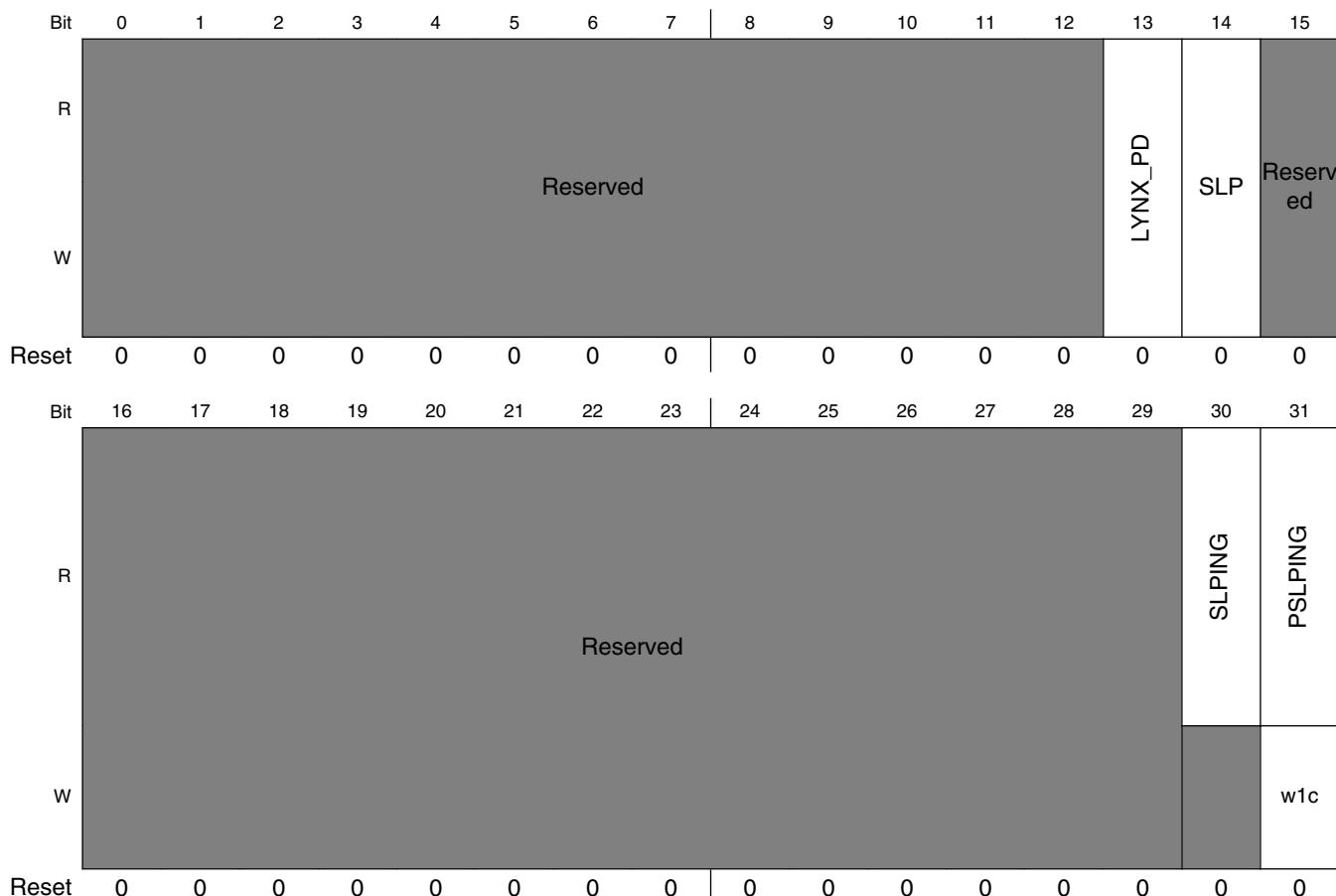
RCPM_CWDTDSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CWDT_DETn	Core WDT Detect Status. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Core has <i>not</i> signalled a WRS condition to be serviced by RCPM 1 Core <i>has</i> signalled a WRS condition to be serviced by RCPM

26.3.9 Power management control and status register (RCPM_POWMGTCsr)

The power management control and status register (POWMGTCsr) performs power management at the device level. This register is used for entering the device sleep state. In addition, it provides status indicating successful entry into the device sleep state.

Address: E_2000h base + 40h offset = E_2040h



RCPM_POWMGTCsr field descriptions

Field	Description
0–12 -	This field is reserved. Reserved
13 LYNX_PD	SerDes and protocol converter power down control

Table continues on the next page...

RCPM_POWMGCSR field descriptions (continued)

Field	Description
	<p>0 SerDes and protocol converter are not powered down in device sleep state 1 SerDes and protocol converter are powered down in device sleep state (for further power saving). If this bit is set, after the system wakes up from device sleep, the SerDes has to go through the training sequence to return to functioning.</p>
14 SLP	<p>Device sleep mode request. This bit is cleared by the interrupt event.</p> <p>0 No request to put device in sleep mode 1 Request to place device in sleep mode. Core instruction fetching is halted, snooping of L1 and L2 cache s is disabled, and most functional blocks are shut down in the cores and the system logic.</p>
15–29 -	This field is reserved. Reserved
30 SLPING	<p>Device sleep status</p> <p>NOTE: This bit cannot reflect if the device has successfully reached the sleep state because once the device reaches that state, there is no means on the device that is active for reading the register.</p> <p>0 Device is not attempting to reach sleep mode 1 The device is attempting to sleep because POWMGCSR[SLP] is set. Most functional blocks in the core and device are shut down or are attempting to shut down.</p>
31 PSLPING	<p>Previous device sleep status</p> <p>PSLPING is used by software to know which state the device was prior to being awakened. Before setting POWMGCSR[SLP], software should clear PSLPING by writing a 1 to this bit (w1c).</p> <p>This bit is set when SLPING has a 1-to-0 transition and it is caused by the interrupt.</p> <p>0 Device was not in sleep mode before being awakened by the interrupt 1 Device was in sleep mode before being awakened by the interrupt</p>

26.3.10 IP powerdown exception control register (RCPM_IPPDEXPCR)

The IP powerdown exception control register (IPPDEXPCR) provides a mechanism for excluding certain IP blocks from device sleep mode to make these IP blocks available as sources for wake-up events. For example,

- Wake on LAN (magic packet)-requires excluding the Ethernet controller from device sleep
- Wake on USB (unplug/plug event)-requires excluding the USB controller from device sleep
- Wake on eSDHC (card detect event)-requires excluding the eSDHC controller from device sleep
- Wake on GPIO-requires excluding the GPIO controller from device sleep

Address: E_2000h base + 50h offset = E_2050h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	eSDHC	FM1_dTSEC1	FM_dTSEC2	FM_dTSEC3	FM1_dTSEC4	10GEC1	FM2_dTSEC1	FM2_dTSEC2	FM2_dTSEC3	FM2_dTSEC4	10GEC2	Reserved	USB1	USB2	GPIO	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RCPM_IPPDEXPCR field descriptions

Field	Description
0 eSDHC	eSDHC powerdown exception 0 eSDHC is powered down during device sleep. 1 eSDHC is not powered down during device sleep.
1 FM1_dTSEC1	FM_dTSEC1 powerdown exception 0 FM_dTSEC1 is powered down during device sleep. 1 FM_dTSEC1 is not powered down during device sleep.
2 FM_dTSEC2	FM_dTSEC2 powerdown exception 0 FM_dTSEC2 is powered down during device sleep. 1 FM_dTSEC2 is not powered down during device sleep.
3 FM_dTSEC3	FM_dTSEC3 powerdown exception 0 FM_dTSEC3 is powered down during device sleep. 1 FM_dTSEC3 is not powered down during device sleep.
4 FM1_dTSEC4	FM_dTSEC4 powerdown exception 0 FM_dTSEC4 is powered down during device sleep. 1 FM_dTSEC4 is not powered down during device sleep.
5 10GEC1	10GEC 1 powerdown exception 0 10GEC 1 is powered down during device sleep. 1 10GEC 1 is not powered down during device sleep.

Table continues on the next page...

RCPM_IPPDEXPCR field descriptions (continued)

Field	Description
6 FM2_dTSEC1	FM2_dTSEC1 powerdown exception 0 FM2_dTSEC1 is powered down during device sleep. 1 FM2_dTSEC1 is not powered down during device sleep.
7 FM2_dTSEC2	FM2_dTSEC2 powerdown exception 0 FM2_dTSEC2 is powered down during device sleep. 1 FM2_dTSEC2 is not powered down during device sleep.
8 FM2_dTSEC3	FM2_dTSEC3 powerdown exception 0 FM2_dTSEC3 is powered down during device sleep. 1 FM2_dTSEC3 is not powered down during device sleep.
9 FM2_dTSEC4	FM2_dTSEC4 powerdown exception 0 FM2_dTSEC4 is powered down during device sleep. 1 FM2_dTSEC4 is not powered down during device sleep.
10 10GEC2	10GEC2 powerdown exception 0 10GEC2 is powered down during device sleep. 1 10GEC2 is not powered down during device sleep.
11 -	This field is reserved. Reserved
12 USB1	USB1 powerdown exception 0 USB1 is powered down during device sleep. 1 USB1 is not powered down during device sleep.
13 USB2	USB2 powerdown exception 0 USB2 is powered down during device sleep. 1 USB2 is not powered down during device sleep.
14 GPIO	GPIO powerdown exception 0 GPIO is powered down during device sleep. 1 GPIO is not powered down during device sleep.
15–31 -	This field is reserved. Reserved

26.3.11 Core Power Management Interrupt Mask Register (RCPM_CPMIMR)

The core power management interrupt mask register (CPMIMR) is used for masking interrupts (*intn*) as a means for waking up from a lower power state.

Address: E_2000h base + 64h offset = E_2064h

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

RCPM_CPMIMR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CIMn	Core Interrupt Power Management Wake Up Mask. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Pending core interrupts cause wake up from a low power mode 1 Pending core interrupts are masked as a wake up condition

26.3.12 Core Power Management Critical Interrupt Mask Register (RCPM_CPMCIMR)

The core power management critical interrupt mask register (CPMCIMR) is used for masking critical interrupts (*cintn*) as a means for waking up from a lower power state.

Address: E_2000h base + 6Ch offset = E_206Ch

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

RCPM_CPMCIMR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved

Table continues on the next page...

RCPM_CPMCIMR field descriptions (continued)

Field	Description
24–31 CCIMn	<p>Core Critical Interrupt Power Management Wake Up Mask.</p> <p>NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.</p> <p>0 Pending core critical interrupts cause wake up from a low power mode 1 Pending core critical interrupts are masked as a wake up condition</p>

26.3.13 Core Power Management Machine Check Mask Register (RCPM_CPMMMC MR)

The core power management machine check mask register (CPMMCMR) is used for masking machine checks (*mcpn*) as a means for waking up from a lower power state.

Address: E_2000h base + 74h offset = E_2074h

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

Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0

RCPM_CPMMMC MR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CMC M _n	<p>Core Machine Check Power Management Wake Up Mask.</p> <p>NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.</p> <p>0 Pending core machine checks cause wake up from a low power mode 1 Pending core machine checks are masked as a wake up condition</p>

26.3.14 Core Power Management NMI Mask Register (RCPM_CPMNMIMR)

The core power management NMI mask register (CPMNMIMR) is used for masking NMIs (*coren_nmi*) as a means for waking up from a lower power state.

Address: E_2000h base + 7Ch offset = E_207Ch

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

RCPM_CPMNMIMR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CNMIMn	Core NMI Power Management Wake Up Mask. NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. 0 Pending core NMIs cause wake up from a low power mode 1 Pending core NMIs are masked as a wake up condition

26.3.15 Core timebase enable register (RCPM_CTBENR)

The core timebase enable register (CTBENR) provides a mechanism for enabling clocks to any core timebases on the device.

Address: E_2000h base + 84h offset = E_2084h

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

RCPM_CTBENR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CTBENn	Core timebase enable NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.

Table continues on the next page...

RCPM_CTBENR field descriptions (continued)

Field	Description
0	Selected core timebase is disabled
1	Selected core timebase is enabled

26.3.16 Core timebase clock select register (RCPM_CTBCKSELR)

The core timebase clock select register (CTBCKSELR) selects the clock source for each core's timebase.

Address: E_2000h base + 8Ch offset = E_208Ch

RCPM_CTBCKSELR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CTBCKSELn	Core n timebase input clock select NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7. <ul style="list-style-type: none"> <li data-bbox="362 1012 926 1020">0 Core 0 timebase is clocked with platform clock/16 <li data-bbox="362 1020 926 1028">1 Core 0 timebase is clocked with the RTC signal

26.3.17 Core timebase halt control register (RCPM_ CTBHLTCR)

The core timebase halt control register (CTBHLTCR) defines the timebase behavior when the core is in halted state. This feature provides the user with more flexible timebase control during multi-core programming debug.

Address: E_2000h base + 94h offset = E_2094h

RCPM CTBHLTCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CTBHLTn	<p>Core timebase behavior during core halted</p> <p>NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.</p> <ul style="list-style-type: none"> 0 Core timebase is stopped when core is halted 1 Core timebase is not stopped when core is halted

26.3.18 Core machine check mask control register (RCPM_CMCPMASKCR)

The core machine check mask control register (CMCPMASKCR) controls whether machine check out for each core is sent to the concentrator for all cores.

Address: E 2000h base + A4h offset = E 20A4h

RCPM CMCPMASKCR field descriptions

Field	Description				
0–23 -	This field is reserved. Reserved				
24–31 CMCPMASKn	<p>Mask core n machine check output</p> <p>NOTE: bit 31 corresponds to Core 0, bit 24 corresponds to Core 7.</p> <table> <tr> <td data-bbox="375 1238 429 1246">0</td> <td data-bbox="429 1238 980 1246">Core n machine check output sent to core concentrator</td> </tr> <tr> <td data-bbox="375 1246 429 1254">1</td> <td data-bbox="429 1246 980 1254">Core n machine check output is not sent to core concentrator</td> </tr> </table>	0	Core n machine check output sent to core concentrator	1	Core n machine check output is not sent to core concentrator
0	Core n machine check output sent to core concentrator				
1	Core n machine check output is not sent to core concentrator				

26.4 RCPM Functional Description

26.4.1 Core Power Management

The RCPM unit supports minimizing power consumption through software-controlled core power management states—core doze and core nap. The RCPM can handshake with the core to halt the instruction sequence and handshake with the CCF to gracefully stop

the internal CoreNet network. The RCPM allows several wake-up event sources to cause an exit from core doze and core nap modes (internal timer, internal and external interrupts). The wake-up events are mapped to OpenPIC interrupts to generate a wake-up interrupt to the RCPM.

26.4.1.1 Core Doze Operation

26.4.1.1.1 Core Doze Operation Entry

Core doze operation can be triggered by the following:

- EPU event trigger via CGACRE
- Device event trigger via CGACRD
- Setting corresponding bit(s) in RCPM CDOZCR via software

26.4.1.1.2 Core Doze Operation Exit

Exit from core doze mode (wake up) can be triggered by the following:

- Corresponding bit(s) in RCPM CDOZCR are clear
- Core IRQ interrupt without RCPM CPMIMR masking
- Core critical interrupt without RCPM CPMCIMR masking
- Core debug interrupt
- Core time based interrupt
- Wake on LAN-magic packet
- Wake on USB-unplug/plug
- Wake on eSDHC-card detect
- Wake on GPIO
- Core unrecoverable failure
- Core hard reset event

26.4.1.1.3 Core Doze Operation Sequence

The sequence of core doze operation is described as follows:

1. The RCPM core power management control logic takes the core doze request, and requests the core to halt. The power management halt request is recorded in core SBSR0[PM_HALT].
2. When the core receives a power management halt request, it stops decoding instructions, unless there is a **lw** or **stmw** in progress, in which case it stops after all of the micro-operations have been decoded. Next, the core waits for the completion

buffer to drain. To make sure that all load/store operations can be seen globally, the sequencer logic jams an **msync** instruction and waits for it to deallocate.

3. After that, the core declares its halted state and drives an internal *core_halted* signal to the RCPM. At this point, the core is dozing.

A core in the halted state (dozing) can still handle snoop requests.

26.4.1.2 Core Nap Operation

26.4.1.2.1 Core Nap Operation Entry

Core nap operation can be triggered by the following mechanism:

- Setting corresponding bit(s) in RCPM CNAPCR via software

In addition, the following actions cause the hardware to trigger a core nap-like operation to bring the core(s) into stopped state gracefully. In these two cases, core nap mode is used to gracefully stop the core to prevent a CoreNet hang.

- Setting the corresponding bit(s) in the device configuration COREDISR to disable the core(s)
- Setting the corresponding bit(s) in the MPIC PIR to initiate a core warm reset

26.4.1.2.2 Core Nap Operation Exit

Exit from core nap mode (wake up) can be triggered by the following:

- Corresponding bit(s) in RCPM CNAPCR are clear
- Core IRQ interrupt without RCPM CPMIMR masking
- Core critical interrupt without RCPM CPMCIMR masking
- Core time based interrupt
- Wake on LAN-magic packet
- Wake on USB-unplug/plug
- Wake on eSDHC-card detect
- Wake on GPIO
- Core unrecoverable failure
- Core hard reset event

26.4.1.2.3 Core Nap Operation Sequence

The sequence of core nap operation is described as follows:

1. The RCPM core power management control logic takes a core nap request and starts the core nap sequence. The first part of the core nap sequence is to bring the core to a halted state, similar to the core doze sequence.
2. After the core is halted, the RCPM unit sends a core-specific stop request to the CCF. This is done so the CCF can drain any snoop transactions on the fly to reach a transaction boundary where it can stop.
3. The CCF puts the processor into a no-snoop mode and stops sending any new snoops to the processor. Note that there might be snoop transactions already enroute to the core which are not stopped.
4. The CCF ensures that all activity from the core related to snoops delivered to the core has been received into the CCF. The CCF then signals to the RCPM that core activity has stopped. This ensures that the CCF has extracted all response transactions from the various internal buffers that may have been written to by the core. At this point, core is napping.

There is no need to handshake between the core power management control logic and the SoC power management control logic in core nap operation because it is assumed that software has flushed the core L1/L2 cache data prior to triggering the core nap operation. For debugging events, the debugger needs to perform the necessary work to make sure the coherence state is preserved. The power management stop request is recorded in SBSR0[PM_STOP] in the core.

26.4.2 Device Power Management

The RCPM unit supports minimizing power consumption through a software-controlled device power management state-device sleep. The RCPM can handshake with various blocks at the SoC level to gracefully stop traffic and direct the memory controller to put DDR into self-refresh mode (if enabled). The RCPM allows several wake-up event sources to cause an exit from device sleep mode (internal timer, internal and external interrupts). The wake-up events are mapped to OpenPIC interrupts to generate a wake-up interrupt to the RCPM.

26.4.2.1 Device Sleep Operation

Prior to entering device sleep mode, software is expected to quiesce all external devices/internal blocks through configuration. The RCPM hardware is designed to implement sanity check for the idle status of internal blocks. Core timebase is turned off in device sleep mode.

26.4.2.1.1 Device Sleep Operation Entry

Device sleep operation can be triggered by the following:

- Setting RCPM POWMGTCsr[SLP] to 1

26.4.2.1.2 Device Sleep Operation Exit

Exit from device sleep mode (wake up) can trigger by the following:

- Core IRQ interrupt without RCPM CPMIMR masking.
- Core critical interrupt without RCPM CPMCIMR masking.
- Wake on LAN-magic packet
- Wake on USB-unplug/plug (via MPIC)
- Wake on eSDHC-card detect (via MPIC)
- Wake on GPIO
- Core unrecoverable failure
- Core hard reset event

26.4.2.1.3 Device Sleep Operation Sequence

The sequence of device sleep operation is described as follows:

1. The RCPM device power management control logic receives a device sleep request and begins the device sleep sequence.
2. The RCPM checks the internal blocks idle status and confirms that the blocks report idle.
3. The RCPM drives a stop request to all master IP blocks in the device, and waits for the stop acknowledgement from all master IP blocks. After a master IP block receives a stop request, it stops mastering any new transactions. After all transaction tenures have completed (request, response, and data tenures), it signals a stop acknowledgement back to the RCPM.
4. The RCPM drives a stop request to the IP blocks which can serve as both master and target device, and waits for a stop acknowledgement from all master/target IPs. During this process, a deadlock case could happen. If the deadlock case has been detected by the RCPM, it deasserts the stop request to these IP blocks and lets the transaction go through to resolve the deadlock.
5. The RCPM drives a stop request to all target IP blocks in the device, and waits for a stop acknowledgement from all target IP blocks.
6. At this point, the device is in the sleeping state. The RCPM drives the power down bus to the clock control unit to gate off the clocks to all IP blocks in the device except for those IP blocks necessary for wake up (for example, the MPIC or the USB).

26.4.3 Core Timebase and Decrementer Operation

26.4.3.1 Timebase Operation in Normal and Debug Device Modes

Core timebase is controlled by CTBENU and CTBSEL register in RCPM as well as the core's TCR. Besides SoC level control, core stops timer in the cases shown in [Table 26-24](#).

Assuming core is not in either HRESET, LSRL, or debug session, and also timer is enabled by RCPM:

Table 26-24. Timebase Operation

Mode	CTBE N[TBE N]	Debug Mode	IDM / EDM	DBCR 0 [FT]	EDBCR 0 [EFT]	TB Timer State	Description
Any Mode	0	No	-	-	-	Stopped	Timebase is disabled through CTBEN. Disable performed before entering core.
Normal Op	1					Active	Timebase is active
Core Dozing						Active	Optionally disabled in software if needed.
Core Napping						Active	Optionally disabled in software if needed.
Device Sleeping						Stopped	Timebase is always stopped on entering device sleeping.
Debug Interrupt in IDM mode		Yes	IDM	0	-	Active	Debug Interrupt is being processed.
				1		Stopped	

NOTE

There is a change between the control of timebase between the e500-v2 and e500mc. For the former, this control was selected in the core's HID0 register, whereas for the e500mc, this is controlled via the CTBENU, L and CTBSELU,L registers.

26.5 Initialization Information

This section describes RCPM initialization. RCPM uses both POR reset and hard reset for hardware initialization. Interrupt sampling, RUNN counter, and core group logic are initialized only through POR reset. The SoC power management module and core power

management module are initialized through hard reset. RCPM CCSR/DCSR are reset either by POR reset or hard reset depending on their functionality. For more information, see the Programmable Registers section.

26.6 Application Information

26.6.1 Magic Packet Wakeup

To enable magic packet wakeup feature in device sleep mode, the frame manager module associated with the dTSEC receiving magic packet need to be enable.

26.6.2 Proper Use of Core Timebase

See [Core Timebase and Decrementer Operation](#).

Appendix A

Terminology, Conventions, and Resources

A.1 Acronyms and abbreviations

This table describes commonly-used acronyms and abbreviations used in this document.

Table A-1. Acronyms and abbreviations

Acronym/ Abbreviation	Meaning
8b/10b	8-bit/10-bit encoding
AIC	Antenna interface controller
AMC	Advanced mezzanine card
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCF	coreNet coherence fabric
CCSR	Configuration control and status register
CLASS	Chip-level arbitration and switching system
CRC	Cyclic redundancy check
DCSR	Debug configuration and status register
DDR	Double data-rate
DIP	Dual inline package
DPLL	Digital phase-locked loop
DTLB	Data translation lookaside buffer
dTSEC	Data path three-speed Ethernet controller
DUART	Dual universal asynchronous receiver/transmitter
ECC	Error checking and correction
ECM	e500 coherency module

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
EEST	Enhanced Ethernet serial transceiver
EHCI	Enhanced host controller interface
eLBC	Enhanced local bus controller
EPROM	Erasable programmable read-only memory
EEPROM	Electrically-erasable programmable read-only memory
eTSEC	Enhanced three-speed Ethernet controller
FCS	Frame-check sequence
FIFO	First in, first out
GCI	General circuit interface
GMII	Gigabit media-independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
I2C	Inter-integrated circuit
IFC	Intergrated Flash controller
IPG	Interpacket gap
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
HSSI	High-speed serial interface
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least significant byte
lsb	Least significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MII	Media independent interface
MMU	Memory management unit
MSB	Most significant byte
msb	Most significant bit
NMI	Non-maskable interrupt
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
OC	
OSI	Open systems interconnection
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
PLL	Phase-locked loop
POR	Power-on reset
PRI	Primary rate interface
PWM	Pulse-width modulation
RAID	Redundant array of independent drives
RGMII	Reduced gigabit media-independent interface
RIO	RapidIO
RTOS	Real-time operating system
RWITM	Read with intent to modify
RMW	Read-modify-write
Rx	Receiver
RxBD	Receive buffer descriptor
SATA	Serial advanced technology attachment
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SD/MMC	Secure digital/multimedia card
SDOS	SmartDSP operating system
SEC	Security Engine
SerDes	Serializer/Deserializer
SFD	Start frame delimiter
SGMII	Serial gigabit media-independent interface
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
Tx	Transmitter
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
uTCA	Micro telecommunications computing platform
UTP	Unshielded twisted pair
VA	Virtual address
ZBT	Zero bus turnaround

A.2 Notational conventions

This table shows notational conventions used in this content.

Table A-2. Notational conventions

Convention	Definition
General	
Cleared	When a bit takes the value zero, it is said to be cleared.
Set	When a bit takes the value one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
<i>italics</i>	Italics can indicate the following: <ul style="list-style-type: none"> Variable command parameters, for example, bcctrx Titles of publications Internal signals, for example, <i>core int</i>
0x	Prefix to denote hexadecimal number
h	Suffix to denote hexadecimal number
0b	Prefix to denote binary number
b	Suffix to denote binary number
rA, rB	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REGISTER[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized x indicates an alphanumeric variable
n	An italicized n indicates either: <ul style="list-style-type: none"> An integer variable A general-purpose bitfield unknown
¬	NOT logical operator

Table continues on the next page...

Table A-2. Notational conventions (continued)

Convention	Definition
&	AND logical operator
	OR logical operator
	Concatenation, for example, TCR[WPEXT] TCR[WP]
Signals	
OVERBAR	An overbar indicates that a signal is active-low.
<i>lowercase_italics</i>	Lowercase italics is used to indicate internal signals
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration.
Register access	
Reserved	Ignored for the purposes of determining access type
R/W	Indicates that all non-reserved fields in a register are read/write
R	Indicates that all non-reserved fields in a register are read only
W	Indicates that all non-reserved fields in a register are write only
w1c	Indicates that all non-reserved fields in a register are cleared by writing ones to them

A.3 Related resources

This table shows related resources that may be helpful. Additional literature is published as new processors become available. For current literature, visit www.freescale.com or contact your local FAE.

Table A-3. Related resources

Resource	Purpose
Reference manual	Provides details about individual implementations
Hardware specifications	Provides specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations
Chip errata	Provides additional or corrective information for a particular device mask set. Individual errata items are published cumulatively.
Application note	Addresses specific design issues useful to programmers and engineers working with Freescale processors

Appendix B

P4081

B.1 Introduction

The QorIQ P4081 advanced, multicore processor combines eight e500mc Power Architecture® processor cores with high-performance datapath acceleration logic and network and peripheral bus interfaces.

The P4081 is a reduced feature version of the P4080. The P4081 combines eight e500mc Power Architecture cores and two memory complexes (CoreNet platform cache and DDR3 memory controller) with most of the same high-performance datapath acceleration, networking, and peripheral bus interfaces.

This figure shows the major functional units within the chip.

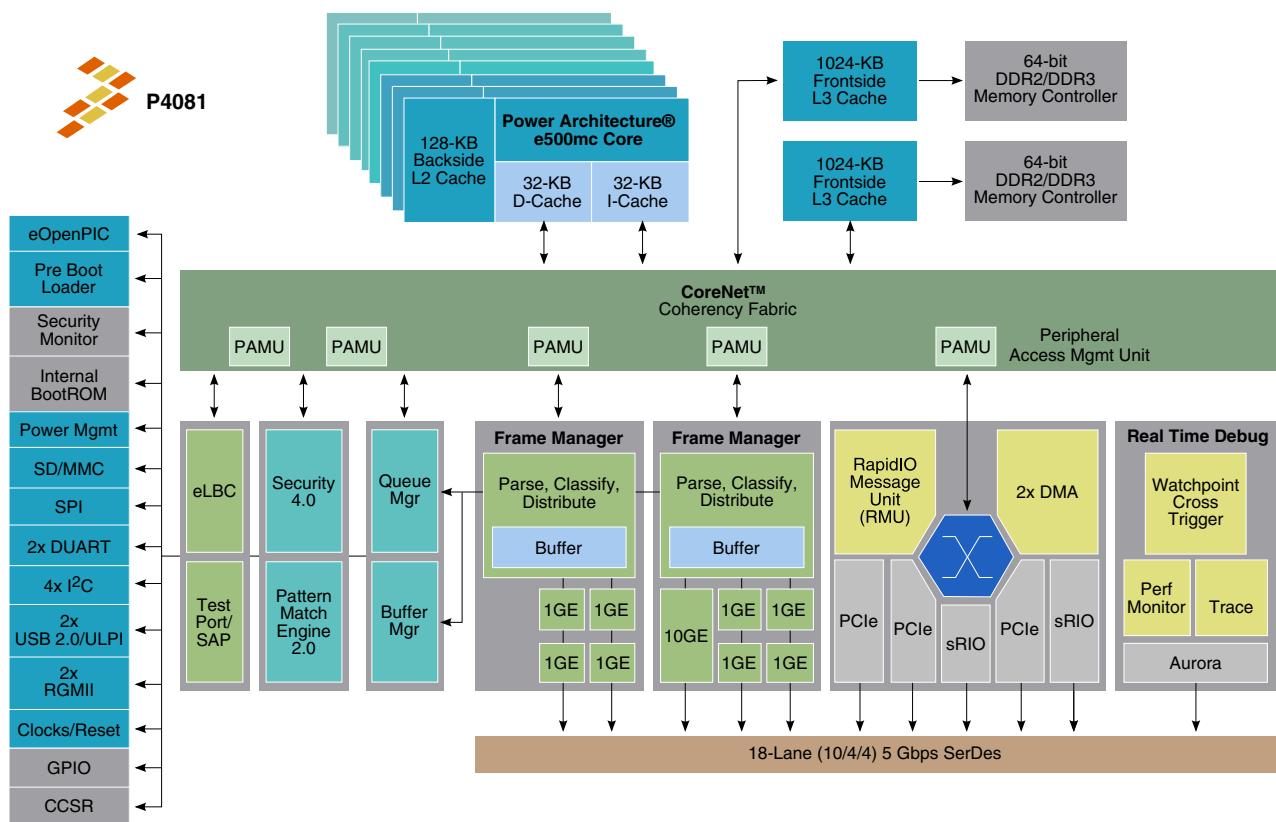


Figure B-1. P4081 block diagram

B.2 Features of P4081

Clocking

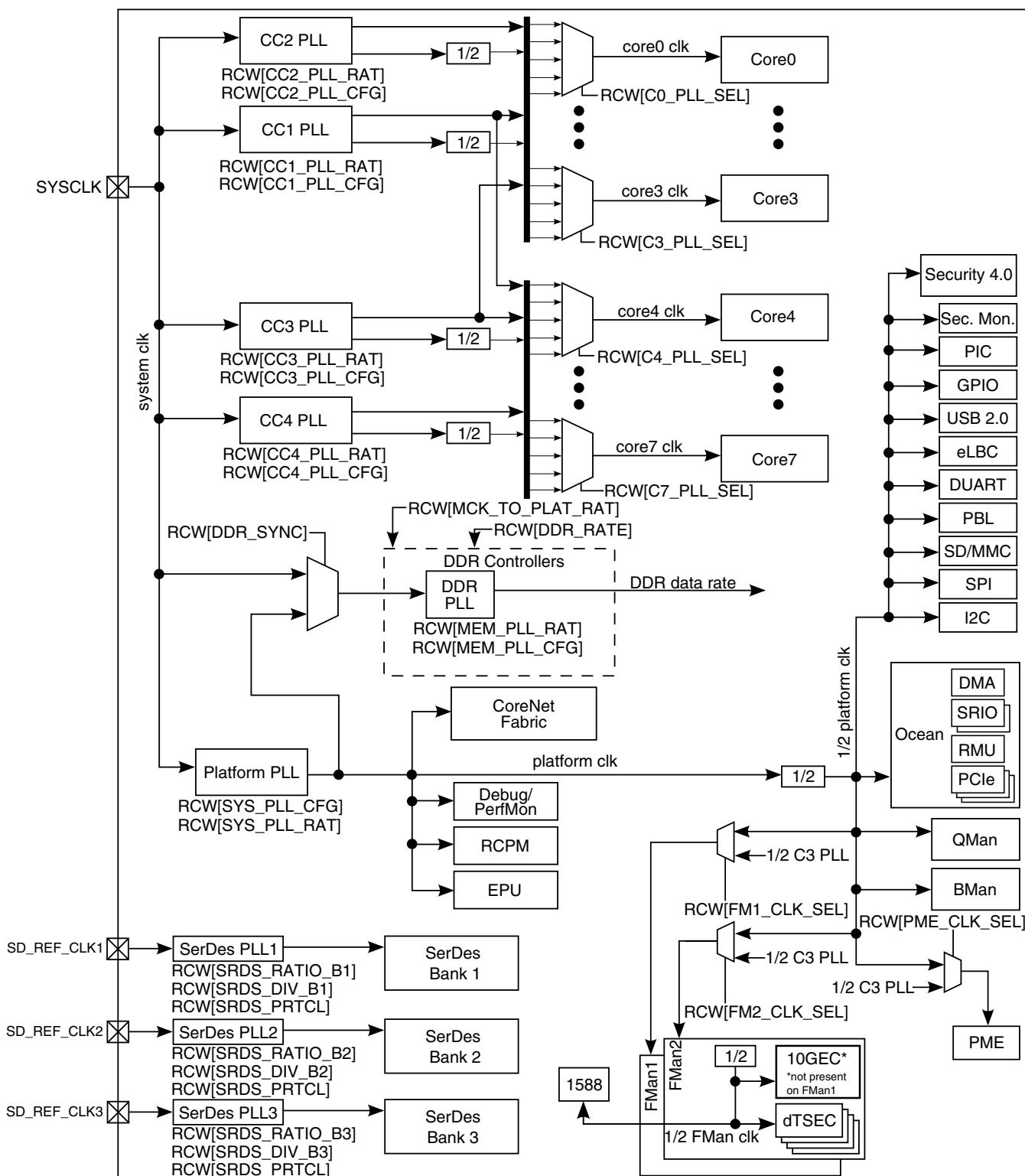


Figure B-2. P4081 logic clock subsystem

B.3 Overview of Differences

Table B-1. Comparison between P4080 and P4081

Feature	P4080	P4081
Peripherals		
Number of Frame Managers	2	2
Total number of 1GE MACs	5 per FM	5 per FM
Total number of 10GE MACs	1 per FM	1 on FMan2

B.4 SerDes Lane Assignments and Multiplexing

The SerDes lanes correspond to the SD_TXn and SD_RXn signals as shown in [Table B-2](#). [Table B-2](#) also shows the lane assignments for the controllers that can use the SerDes.

Table B-2. SerDes Bank/Lane/Signal Assignments

Bank	Lane	Signal	Controller Lane Assignments							
			PCI Express			SRI0		Debug	Frame Manager	
			PCIe1	PCIe2	PCIe3	sRIO1	sRIO2		FM1	FM2
1	A	SD_TX00/SD_RX00_B SD_RX00/SD_RX00_B	0				3			
	B	SD_TX01/SD_RX01_B SD_RX01/SD_RX01_B	1				2			
	C	SD_TX02/SD_RX02_B SD_RX02/SD_RX02_B	2		0		1			
	D	SD_TX03/SD_RX03_B SD_RX03/SD_RX03_B	3		1		0			
	E	SD_TX04/SD_RX04_B SD_RX04/SD_RX04_B	4	0		3				dTSEC1
	F	SD_TX05/SD_RX05_B SD_RX05/SD_RX05_B	5	1		2	0			dTSEC2
	G	SD_TX06/SD_RX06_B SD_RX06/SD_RX06_B	6	2		1				dTSEC3
	H	SD_TX07/SD_RX07_B SD_RX07/SD_RX07_B	7	3		0				dTSEC4
	I	SD_TX08/SD_RX08_B						1		

Table continues on the next page...

Table B-2. SerDes Bank/Lane/Signal Assignments (continued)

Bank	Lane	Signal	Controller Lane Assignments							
			PCI Express			SRIO		Debug	Frame Manager	
			PCIe1	PCIe2	PCIe3	sRIO1	sRIO2		FM1	FM2
		SD_RX08/SD_RX08_B								
	J	SD_TX09/SD_TX09_B SD_RX09/SD_RX09_B						0		
2	A	SD_TX10/SD_RX10_B SD_RX10/SD_RX10_B			0					dTSEC1 / 10GEC0
	B	SD_TX11/SD_RX11_B SD_RX11/SD_RX11_B			1					dTSEC2 / 10GEC1
	C	SD_TX12/SD_RX12_B SD_RX12/SD_RX12_B			2					dTSEC3 / 10GEC2
	D	SD_TX13/SD_RX13_B SD_RX13/SD_RX13_B			3					dTSEC4 / 10GEC3
3	A	SD_TX14/SD_RX14_B SD_RX14/SD_RX14_B							dTSEC1	
	B	SD_TX15/SD_RX15_B SD_RX15/SD_RX15_B							dTSEC2	
	C	SD_TX16/SD_RX16_B SD_RX16/SD_RX16_B							dTSEC3	
	D	SD_TX17/SD_RX17_B SD_RX17/SD_RX17_B							dTSEC4	

Table B-3 describes the protocol multiplexing options for the SerDes lanes. As shown in Table B-3, each option is selected by a specific encoding of RCW[SRDS_PRTCL]. See [RCW Field Definitions](#), for more information.

Table B-3. SerDes Lane Multiplexing/Configuration

SRDS PRTC L	Bank 1										Bank 2				Bank 3				
	A	B	C	D	E	F	G	H	I	J	A	B	C	D	A	B	C	D	
0x02	PCIe1 (2.5G)										Debug (5/2.5G)	XAUI FM2 10GEC				Reserved ¹			
0x05	PCIe1 (5/2.5G)				PCIe2 (5/2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹				
0x08	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	PCIe2 (5/2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹						

Table continues on the next page...

Table B-3. SerDes Lane Multiplexing/Configuration (continued)

SRDS PRTC L	Bank 1										Bank 2				Bank 3				
	A	B	C	D	E	F	G	H	I	J	A	B	C	D	A	B	C	D	
0x0D	PCIe1 (5/2.5G)			PCIe2 (5/2.5G)		2x SGMII FM2 dTSEC[3:4] ²		Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹					
0x0E	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	PCIe2 (5/2.5G)		2x SGMII FM2 dTSEC[3:4] ²		Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹						
0x0F	PCIe1 (5/2.5G)			4x SGMII FM2 dTSEC[1:4] ²				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹					
0x10	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	4x SGMII FM2 dTSEC[1:4] ²				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹						
0x13	sRIO2 (2.5G)			sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹					
0x16	sRIO2 (3.125G)			sRIO1 (3.125G)				Debug (3.125G)		4x SGMII FM2 dTSEC[1:4]				4x SGMII FM1 dTSEC[1:4]					
0x19	sRIO2 (3.125G)			sRIO1 (3.125G)				Debug (3.125G)		PCIe3 (5/2.5G)				4x SGMII FM1 dTSEC[1:4]					
0x1D	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	-	sRIO2 (2.5G)	-	sRIO1 (2.5G)	Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹						
0x22	PCIe1 (5/2.5G)			sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹					
0x25	PCIe1 (5/2.5G)	PCIe3 (5/2.5G)	sRIO1 (2.5G)				Debug (5/2.5G)		XAUI FM2 10GEC				Reserved ¹						

1. Bank 3 cannot be used in this configuration.
2. When SerDes bank 1 is configured for 5G operation (by RCW[SRDS_RATIO_B1] and the reference clock), lanes configured as SGMII operate at 1.25Gbps regardless of the corresponding SRDS_DIV_B1 settings.

Appendix C

P4080 Reference Manual Revision History

C.1 Substantive changes from revision 1 to revision 2

Substantive changes from revision 1 to revision 2 are as follows:

C.1.1 Overview Revision History

Reference	Description
Network Interfaces	Revised note regarding 1588 time-stamping in SGMII 10/100 mode to specify that P4080 Rev 3 supports time-stamping in SGMII 10/100 mode and P4080 Rev 2 does not support time-stamping in SGMII 10/100 mode.
Debug support	Revised section and added note regarding contacting third-party tool developers for information on debug capabilities.

C.1.2 Memory Map

Reference	Description
Table 2-104	Edited comments to CCSR block base Address map table to better communicate the Local Configuration Control, and LAWs start offsets .
Memory Map Overview	Updated the last sentence of the third paragraph to read as follows: There is also a fixed (8 Mbytes of local memory space) default boot window from 0_FF80_0000h to 0xFFFF_FFFFh.

Substantive changes from revision 1 to revision 2

C.1.3 Signal Descriptions Revision History

Reference	Description
SerDes Control Memory Map	In the register definitions, replaced all instances of "XGMII," with "XAUI."
Configuration Signals Sampled at Reset	Changed HRESET_B to PORESET_B
SerDes PLL reset and reconfiguration	Added section
Bank 1 Receive Equalization Control Register 0 Lane n (Signals_B1RECRn0), Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRn0), and Bank 2 Receive Equalization Control Register 0 Lane n (Signals_B2RECRn0)	Added new registers for Rx equalization control
Bank 1 TTL Control Register 0 - Lane x (Signals_B1TTLCRn0), Bank 2 TTL Control Register 0 - Lane x (Signals_B2TTLCRn0), and Bank 3 TTL Control Register 0 - Lane x (Signals_B3TTLCRn0)	<ul style="list-style-type: none"> Revised reset value Added note to FLT_SEL field description Revised recommended settings per protocol for FLT_SEL field Added new field, PM_DIS, at bit 17 Added recommended settings per protocol to Reserved bit 0

C.1.4 Reset, Clocking, and Initialization

Reference	Description
RCW Field Definitions	Added note to RCW[BOOT_LOC] regarding booting from PCIe in RC mode.
Core n clock control/status register (Clocking_CLKCnCSR)	Changed "Core 0 clock control/status register" to "Core n clock control/status register"
DRAM type select	Revised cfg_dram_type=1 to select DDR2 technology (1.8 V).
Boot space translation attribute register (LCC_BSTAR)	Changed the equation for value of the the BSTAR[SIZE] field to $2^{(SIZE+1)}$.
System Control Signals	Added ASLEEP and TMP_DETECT_B signals to the system controls table.
External Signal Descriptions	Removed the duplicate row for CLK_OUT signal.
PLLCGSR and PLLDGSR	Added bit 0 KILL and updated the bit width of CFG bits from 25-30 to 23-30
Table 4-1, Table 4-4	Updated signal names for SerDes reference clocks to SD_REF_CLKn_P/ SD_REF_CLKn_N for positive and negative terminals.
Boot Space Translation	Updated the core's first instruction fetch effective address from 0x0_FFFF_FFE0 to 0x0_FFFF_FFC0.

C.1.5 Pre-Boot Loader Revision History

Reference	Description
Software Restrictions	Added bullet stating that the Flush command is restricted to CCSR space.
Registers Written by PBL During RCW and PBI Phases	Added the following statement:

Reference	Description
	If the interface is the PBI/RCW source, the registers list of that interface resets back to the POR value, when PBL is completed. Alternatively, if the registers are modified by some external source, then its value does not reset back to POR value when PBL is completed.

C.1.6 Secure Boot and Trust Architecture Revision History

Reference	Description
SM_HP Security Interrupt Control Register (SECMON_HPSICR) and SM_HP Security Violation Control Register (SECMON_HPSVCR)	Changed the number of SVI_ENn bits in HPSICR and SV_CFGn bits in HPSVCR (they were showing 6 but the chip only has 4)
Security monitor	Removed "A new feature in the P4080 is the pre-boot loader, derived from the micro sequencer previously dedicated to the PowerQUICC I2C interface."
Security fuse processor	Added reference to "whitepaper Manufacturing for Trust for more information"
Code signing	Added new figure and removed Although Figure 6-3 shows the hash being calculated over a plaintext image, it is possible (even advantageous) for portions of the code to be encrypted (indirectly) with the OTPMK . This prevents attackers from stealing the code from flash
Pre-boot phase	Removed "In Figure the PBI Image is shown as also containing commands which cause the PBL to copy the ESBC to main memory."
ISBC phase	Removed "by the PAMU's"
ESBC U-Boot and boot script	Replaced Mini uboot by ESBC u-boot and changed the title to ESBC u-boot and boot script
#secure_boot_and_trust_architecture	Removed sections ESBC client, Next executable, Chain of trust summary, Operational Modes
Fuse programming	Added "The fuse programming sequence and example values are included in the Freescale whitepaper, Manufacturing for Trust"
Fuse read errors	Added " If fuse errors are suspected in other registers, software is responsible for checking the actual values against expected values. It is also possible to calculate a checksum over the original values and storing this checksum in one of the OEM Section Scratchpad Registers (SFP_OSSR)."
OEM Security Policy Register (SFP_OSPR) , Debug Permissions Register (SFP_DPR) , Debug Challenge Value Register n (SFP_DCVRn) , Debug Response Value Register n (SFP_DRVRn) , One Time Programmable Master Key n (SFP_OTPMKRn)	Added note "This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection."
Freescale Section Write Protect Register (SFP_FSWPR) , Freescale Unique ID Register (SFP_FUIDR) , Freescale Scratch Pad Fuse Register n (SFP_FSPFRn)	Added note "This register is protected by the Freescale write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection."
Security fuse processor (SFP) memory map	Removed SFP Configuration Register (SFP_SFPCR)

Table continues on the next page...

Substantive changes from revision 1 to revision 2

Reference	Description
OEM Security Policy Register (SFP_OSPR)	Removed note "This register is writable only until programmed into the fuse array." Added updated register description. Added bit 28 SEC_DIS.
OEM Security Policy Register (SFP_OSPR)	Removed note "It is recommended that the OVPR[ITF] always be set equal to..."
Debug Permissions Register (SFP_DPR), Freescale Unique ID Register (SFP_FUIDR), Debug Challenge Value Register n (SFP_DCVRn), Debug Response Value Register n (SFP_DRVRn), Super Root Key Hash n (SFP_SRKHRn), OEM Unique ID Register (SFP_OUIDR)	Removed the note "This register is writable only until programmed into the fuse array."
Debug Challenge Value Register n (SFP_DCVRn)	Added Example to the register description
Debug Response Value Register n (SFP_DRVRn)	Added new code, an example and the sections after the example to the register description
One Time Programmable Master Key n (SFP_OTPMKRn)	Removed the note "This register is writable only until programmed into the fuse array." Added sentence "Reads to these registers return all 0s whether an OTPMK has been programmed or not."
Security fuse processor (SFP) memory map	Removed "OEM Validation Policy Register" and "OEM Section Checksum Register"
OEM Scratch Pad Fuse Register (SFP_OSPFR)	Changed OEM Scratch Pad Fuse Register into OEM Scratch Pad Fuse Register 0 and OEM Scratch Pad Fuse Register 1
Security Monitor memory map/register definition	Removed "There are fields in the SM_HP command register that can be also written in nonfunctional SSM states, as follows:...."
SM_HP Command Register (SECMON_HPCOMR)	Added <ul style="list-style-type: none"> • Check <ul style="list-style-type: none"> • SSM_ST, SW_SV, SW_FSV – these bits are writable by the ISBC, which is the only software running during Check state. • Soft Fail <ul style="list-style-type: none"> • SSM_ST, SW_SV, SW_FSV, HAC_STOP, HAC_LOAD, and HAC_CLEAR bits
SM_HP Security Interrupt Control Register (SECMON_HPSICR)	Changed bit 0 to reserved
SM_HP Security Violation Control Register (SECMON_HPSVCR)	Changed bit 0 to reserved
SM_HP Security Violation Status Register (SECMON_HPSVSR)	Added " This register is writable in the Soft Fail state. "
SM_HP Security Interrupt Control Register (SECMON_HPSICR)	Added bits 31-28, SVI_EN0_RTIC, SVI_EN1_SFP, SVI_EN2_SDC, SVI_EN3_TMP and updated the bitfield description table.
SM_HP Security Violation Control Register (SECMON_HPSVCR)	Added bits 28-31, SV_CFG3_TMP, SV_CFG2_SDC, SV_CFG1_SFP, SV_CFG0_RTIC and updated the bitfield description table.
SM_HP Security Violation Status Register (SECMON_HPSVSR)	Added bits 28-31, SEC_VIO0 RTIC, SEC_VIO1 SFP, SEC_VIO2 SDC, SEC_VIO3 TMP and updated the bitfield description table. Added "This register is writable in the Soft Fail state."
SM_HP High Assurance Counter Initial Value Register (SECMON_PHACIVR)	Changed IV in the register name to Initial Value. Added " The high assurance counter decrements at the operating frequency of the Security Monitor, generally 400MHz (half of platform frequency). The maximum delay is ~10 seconds" to the register description.
SM_HP High Assurance Counter Register (SECMON_PHACR)	Added "The SM_HP High Assurance Counter Register contains the value of the High Assurance Counter. The High Assurance Counter is a delay

Table continues on the next page...

Reference	Description
	introduced before the System Security Monitor transitions from Soft Fail to Hard Fail State if this transition is enabled" to the register description.
Figure 6-41Security monitor	Removed the LP section from the figure. Removed "Although the P4080 is the first QorIQ device to incorporate the Security Monitor, the logical module has been used in other Freescale devices..."
Security monitor features summary	Changed the Security Violation inputs from six to four and modified the list.
Operational states	Added " Access to persistent and ephemeral secrets locked out, some zeroization of secrets within the SEC" and "Hard fail-system hard reset is requested. All behaviors of Soft fail, plus zeroization of some SoC caches and main memory, SoC reset initiated"
Security Monitor functional description	Added "Security Monitor incorporates a security state machine (SSM) that detects security violation inputs and manages the system security state in accordance with configured security policy. The Security Monitor is also responsible for checking the OTPMK's hamming code and passing..."
Security Monitor functional description	Added "In the Trust Architecture 1.0, the Security Monitor module only operates while the SoC is powered on. In other versions of the trust architecture..."
SM_HP description	Added "The SM_HP contains all security monitor status and configuration registers. The configuration registers control security violation detection and response"
Initialization guidelines	Added figure "Relationship Between Registers"
Trust Architecture overview	Removed sections "Power glitch detector (PGD)", "LP security violation/tamper policy", "Security monitor interrupts, alarms, and security violations".
State definitions	Removed the word "CoreNet", the statement "Except for a few bits in the HP command register, the Security Monitor registers..." . Modified the wording of "Upon transitioning to Soft Fail, the Security Monitor signals the SEC to stop using the OTPMK and JDKEKs, and zeroize any ephemeral secrets within the SEC. The security monitor also generates an interrupt to inform software of the soft fail, and if configured to do so, starts the HAC_Counter count down toward Hard Fail. "
Secure boot sequence	Added new figure
Code signing	Added new figure
SEC 4.0	Removed "also incorporates a 16-Kbyte on-chip RAM with an ACU that can be used for secure storage of cryptographic credentials."
Security fuse processor	Removed "If the OEM doesn't like the values in the SFP shadow registers prior to burning, these values can be cleared via HRESET (not PORESET)..."
Secure boot sequence	Added "Although the figure shows the hash being calculated over a single plaintext image, it is possible (even advantageous) for secure boot to..."
ESBC U-Boot and boot script	Modified the text in the section. Replaced mini u-boot by esbc u-boot
Security monitor	Modified the entire section.
Security fuse processor	Removed Operational Modes section
OEM Security Policy Register (SFP_OSPR)	Removed "The OEM security policy register and the Freescale section write protect register are mirror registers that can be used prevent writes to other mirror registers, thereby preventing programming of the fuse array. Setting bits in the WP bit prevents writes to the corresponding (Freescale or OEM) mirror registers until the next PORESET, at which time the WP bits are cleared. The write protect bits must be programmed into the fuse array to provide permanent protection against fuse array updates." and

Table continues on the next page...

Substantive changes from revision 1 to revision 2

Reference	Description
	"There are other ways to trigger the device to perform secure boot, however no other method is guaranteed to work in the presence of an attacker with physical access to the system."
Freescale Section Write Protect Register (SFP_FSWPR)	Removed "To prevent updates to the Freescale unique ID, this register always set in production devices purchased from Freescale and its authorized distributors."
One Time Programmable Master Key n (SFP OTPMKRn), Super Root Key Hash n (SFP_SRKHRn), Debug Response Value Register n (SFP_DVRn)	Added the bit order positions of keys in the registers and modified the bitfield description table
Security Monitor functional description	Removed section Security Monitor Clock Sources
Operational states	Modified the last sentence to say "Hard fail-system hard reset is requested. All behaviors of Soft fail, plus zeroization of some SoC caches and main memory, SoC reset initiated."
Security monitor	Modified the text in the section
SM_HP description	Removed "The SM_HP contains all the interfaces to the platform core and to the SM_LP section. Access to SM_LP registers can only be done through the SM_HP, and only when it is powered up"
External tamper-detection	Corrected erroneous text regarding the description of external tamper detection that appears in some manuals.
Debug Error Syndrome Status Register (SFP_DESSR)	Changed the size of SFP_DESSR[ECC_Syndrome] to 6-bits (bits 25-30).
Signals	Added new section .
Debug Response Value Register n (SFP_DVRn)	Removed "24" in the C code "24 uint64 mask = ((uint64) 1...
Security monitor	Reorganized Security Monitor section and most significantly moved the Security Monitor Memory Map under the Security Monitor section.
Security Monitor functional description	Revised the section to eliminate the bulleted list and clarified that Trust Architecture 1.0 devices only support the HP section.
Debug Response Value Register n (SFP_DVRn)	Added statement, "Reads to these registers return all 0s regardless of whether the a DRV has been programmed or not." to register description. Deleted repeated paragraph that starts, "The process for programming the debug response value..." Deleted repeated paragraph that starts, "Note that it is the OEM's responsibility to control access to..."
One Time Programmable Master Key n (SFP OTPMKRn)	Removed extra/contradictory register/key bits mapping in the OTPMKn bitfield description.
Transitioning among system security monitor states (Formerly, "Security state machine" and "State definitions")	Combined the two sections into a single section. Revised the security states diagram
Security state machine	Replaced security monitor state diagram with a simplified version.
State definitions	Changed "Start" to "Init" In the table, removed transition from Trusted to Secure.
Instruction Register (SFP_INGR)	Added note stating INGR is not self clearing.
Debug Challenge Value Register n (SFP_DCVRn)	Changed to Write to DCVR 1: c0c0c0c0 Write to DCVR 0: d0d0d0d0 from

Table continues on the next page...

Reference	Description
	Write to DCVR 0: c0c0c0c0 Write to DCVR 1: d0d0d0d0 changed the bitfield description of DCVRn bits to DCVR1 is the upper part (bits 0-31) and DCVR0 is the lower part (bits 32-63) of the 64-bit debug challenge value.
Signals	Updated the description of TMP_DETECT_B.
SM_HP Version ID Register 1 (SECMON_HPV IDR1) and SM_HP Version ID Register 2 (SECMON_HPV IDR2)	Changed register reset values to n. Added IP_ERA bitfield to HPVIDR2.
External tamper-detection	Added "as per the TMP_DETECT_B voltage in the device hardware specification"
External tamper-detection and Signals	Updated TMP_DETECT as TMP_DETECT_B and LP_TMP_DETECT as LP_TMP_DETECT_B throughout.
Debug Response Value Register n (SFP_DRVRn)	Corrected field description in table to state: DRVR1 is the upper part (bits 0-31) and DRVR0 is the lower part (bits 32-63) of the 64-bit debug response value. Both of these registers reset to 0x0000_0000 unless programmed into the fuse array.

C.1.7 e500mc Core Integration Revision History

Reference	Description
Performance Monitor Details	Modified notes for performance monitor events 16, 73, 126, and 86-89 to state that these events are supported on P4080 Rev. 3 but not on P4080, Rev.2.

C.1.8 CoreNet Platform Cache (CPC) Revision History

Reference	Description
Supported SRAM Mode Configurations	Removed configuration with CPC SRAM Mode Enabled=Yes, CPC Interleaving=No, and DDR1 and DDR2 used with memory controller cache line or page interleaving enabled.
Enabling the CPC after Power-On Reset	Revised the steps for enabling CPC after Power-On Reset.
CPC partition allocation register 0 (CPC_CPCPAR0), CPC partition allocation register n (CPC_CPCPARn)	Removed LOADEC from DRDALLOC, DRDPEALLOC, and DRDFEALLOC allocation policies.
CPC partition allocation register 0 (CPC_CPCPAR0), CPC partition allocation register n (CPC_CPCPARn)	Changed bits 22, 23 and 25 to reserved. Changed "Data RWITM, WBI,CTM and CTT transactions that miss in the CPC" to "Transactions resulting from a Store operation that missed in a core cache" in the description of DSTALLOC bits.

Table continues on the next page...

Reference	Description
	<p>Replaced WCO with "Writes carrying cast-out data from core caches" in the description of WCOALLOC bits.</p> <p>Replaced ReadU transactions by "Decorated Load operations" in the description of READUALLOC bits.</p> <p>Replaced WriteU/notify transaction by "Decorated Store operations" in the description of WRITEUALLOC bit description.</p> <p>Added link to "Decorated Storage".</p> <p>Replaced CoreNet by "Explicitly" in the description of CNSTASHEN bit description."</p>
Error CaptureCoreNet Platform Cache (CPC) Memory Map	Removed CPC error attribute register (CPCx_CPCERRATTR).
CPC partition ID register 0 (CPC_CPCPIR0), CPC partition ID register n (CPC_CPCPIRn)	Added link to LAW registers.

C.1.9 CoreNet Coherency Fabric (CCF) Revision History

Reference	Description
CoreNet Coherency Fabric (CCF) Memory Map	Added SIDMR0-SIDMR31 and CIDMR0-CIDMR31 registers
CCF Error Capture Attribute Register (CCF_CECAR)	Changed the access type for all non-reserved bits in CCF_CECAR to write-1-to-clear (w1c)
CCF Introduction	Added note about WIMG bits and adhering to all power architecture rules.
CSDID n Port Mapping Register (CCF_CIDMRn)	Changed the offset values for CIDMR1-CIDMR31 (previous values were incorrect).

C.1.10 Peripheral Access Management Unit (PAMU)

No substantive changes

C.1.11 DDR Memory Controller Revision History

Reference	Description
DDR SDRAM Interface Operation	Added 8 Gbit DRAM support
Supported DDR SDRAM Organizations	Added 8 Gbit DRAM support rows to the table
DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)	Added setting xx00100 to DDR_SDRAM_CFG[BA_INTLV_CTL]

Table continues on the next page...

Reference	Description
DDR Functional Description	Changed "Bank sizes up to 4 Gbytes are supported, providing up to a maximum of 16 Gbytes of DDR main memory." to "Bank sizes up to 8 Gbytes are supported, providing up to a maximum of 32 Gbytes of DDR main memory."
Error Management	Changed the error reporting via "critical" and "machine check" interrupts to "error" interrupt. (It is up to the MPIC programming how DDR error interrupts are reported to the core.)
Supported DDR Interleaving Configurations	In the Supported DDR Interleaving Configurations table, removed four configurations where Memory controller interleaving is enabled (INTLV_EN=1) and External memory banks 0/1 and 2/3 are interleaved (BA_INTLV_CTL=0x60). These configurations are not supported.
Supported DDR Interleaving Configurations	In the Supported DDR Interleaving Configurations table, added footnotes describing inaccessible chip selects when controller-interleaving is enabled.
DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL)	Added statement to register description: Before issuing a command via the DDR_SDRAM_MD_CNTL register, the DDR interface should be idle. This can be done by setting DDR_SDRAM_CFG[MEM_HALT] and disabling refreshes by clearing DDR_INTERVAL[REFINT]. If there are memory contents that need to be preserved during this time, then software should also force any required refresh commands while DDR_INTERVAL[REFINT] is cleared.
DDR training initialization extended address (DDR_DDR_INIT_EXT_ADDRESS)	Changed description of setting 0 of bitfield UIA to read "This will be the first valid address in each enabled chip select."
DDR SDRAM Refresh Timing	Revised the DDR SDRAM Bank Staggered Auto Refresh Timing figure to show the staggered refresh occurs on MCS[0,3]_B and then MCS[1,2]_B. Also added note below the figure.
Figure 11-230	Updated text in figure title from "2566432-Mbyte" to "256-Mbyte"
Self-Refresh in Sleep Mode	Added note: Deep sleep mode is not supported for DDR3 registered DIMMs.
DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_Cfg_2)	Added statement to description of DLL_RST_DIS: If self-refresh is to be used with DDR3, this bit should be set.
DDR SDRAM timing configuration 3 (DDR_TIMING_Cfg_3), DDR SDRAM timing configuration 1 (DDR_TIMING_Cfg_1)	Changed description of TIMING_Cfg_3[EXT_REFREC] from "This field is concatenated with TIMING_Cfg_1[REFREC] to obtain an 8-bit value" to "This field is concatenated with TIMING_Cfg_1[REFREC] to obtain an 9-bit value...." Changed description of TIMING_Cfg_1[REFREC] from "This field is concatenated with TIMING_Cfg_3[EXTREFREC] to obtain a 7-bit value" to "This field is concatenated with TIMING_Cfg_3[EXTREFREC] to obtain a 9-bit value".
Memory data path read capture ECC (DDR_CAPTURE_ECC)	Updated description of field ECE.
DDR Memory Test Pattern n Register (DDR_DDR_MTPn)	Added the following note at the end of register description: Memory test read compares data that is written in this register.
Memory Interface Signals	Updated MAPAR_ERR_B timing for registered DIMMs: "Driven by the registered DDR3 DIMMs 3 DRAM cycles after the parity bit has been driven by the memory controller (for DDR2 DIMMs, driven 1 cycle after the parity bit has been driven by the memory controller)."
DDR Memory Controller Memory Map	For DDRx_DDRCDR_1 bit field definitions, removed content that referred to "calibrate the DDR drivers to 18ohms. However, this should ..."
DDR Memory Controller	Removed section 11.5.4.1, DDR SDRAM Interface Timing/ Clock Distribution

C.1.12 I2C Revision History

Reference	Description
I2C digital filter sampling rate register (I2C_I2CDFSRR)	Updated reset value for I2CDFSRR register to 0x10.

C.1.13 Enhanced Local Bus Controller (eLBC) Revision History

Reference	Description
Options register 0 layout for FCM Mode (eLBC_ORf0) and Options register 0 layout for UPM Mode (eLBC_ORu0)	Updated the reset values of Options register 0 layout for FCM Mode (ORf0) and Options register 0 layout for UPM Mode (ORu0).
External address latch enable signal (LALE)	Removed the paragraph "If the RCW is loaded by the eLBC, LALE may remain at an unknown value for up to 8 cycles after PORESET negation. Thus, LALE should....."
Configuration register (eLBC_LBCR)	Added 32 as the value of 'n' in the bitfield description of LRDY in the local bus configuration register.
Table 13-199	For OR0 register field SCY, changed the settings from "From por_cfg_scy[1:3]" to "010."
eLBC external signal descriptions	In the Name column, added LGTA_B in the "LGPL4/LFRB_B/LUPWAIT/LPBSE" group.
Table 13-208	Added commands FMR and LSOR to the table.
External access termination (LGTA_B)	Updated Figure 13-82 : - Removed the first (the upper one) LCLK clock signal - Updated "Sample point" as "Sample point for LGTA signal"
Figure 13-70	Removed t _{LSKEW} from figure and updated LCLK to start from rising edge.
eLBC external signal descriptions	Removed all instances of MDVAL and MSRCID debug signals.
Configuration register (eLBC_LBCR)	Updated the ABSWP and AS16 bit description for 16-bit port size.
General-purpose chip-select machine (GPCM)	Revised the addressing in the "Enhanced local bus to GPCM device interface" figure.

C.1.14 Enhanced Serial Peripheral Interface Revision History

Reference	Description
16-bit address example, 24-bit address example	Swapped the 5th and 6th espi intialisation sequence.

Table continues on the next page...

Reference	Description
eSPI CS0 mode register (ESPI_SPMODE0), eSPI CS1 mode register (ESPI_SPMODE1), eSPI CS2 mode register (ESPI_SPMODE2), and eSPI CS3 mode register (ESPI_SPMODE3)	In SPMODEn[LENn] bitfield description, added text "Supports a range from 1-bit to 16-bit data characters."
eSPI mode register (ESPI_SPMODE) and eSPI CS0 mode register (ESPI_SPMODE0)	Updated reset value of SPMODE and SPMODE0 to 0x0000_0000

C.1.15 Enhanced Secure Digital Host Controller Revision History

Reference	Description
Enhanced Secure Digital Host Controller (eSDHC) Memory Map	Reinstated the note regarding register access restriction prior to the memory map table.
Table 15-1	For SDHC_DAT3, updated the "Function" column from "4/8-bit mode: DAT3 line or configured as card detection pin" to "4/8-bit mode: DAT3 line". Updated the "Pull up/Pull down required" column from "10 k–100 kW to OVDD is required during the normal operating conditions. Board should have a 100 KW pull-down resistor if DAT3 is used for an SD card detection. In this case, the pull-up resistor has to be disconnected during the identification phase. The pull-up resistor has to be connected and the pull-down resistor should be switched out after the identification phase" to "Pull up. Do not use DAT3 pin as a CD pin."
Interrupt status (eSDHC_IRQSTAT)	Added a note to IRQSTAT[CCE] and IRQSTAT[CIE].
Host controller version (eSDHC_HOSTVER)	For HOSTVER(VVN), added bit value definition for 01.
System control (eSDHC_SYSCTL)	Changed the maximum SD clock frequency to 52 MHz for SYSCTL(16-23). Also, updated the second example given to yield a clock value of 400 kHz.
Host controller capabilities (eSDHC_HOSTCAPBLT)	Changed reset value for HOSTCAPBLT bits 5, 6, 8, and 11. Added the qualifier "on P4080 Rev2," to the notes on HOSTCAPBLT bits 5, 6, 8, and 11 that apply only to P4080 Rev 2.
Clock generator	Removed the word "base" from the following sentence: The frequency of clock output from this stage, DIV, can be base, base/2, base/4, ..., or base/256.
Command transfer type (eSDHC_XFERTYP)	In the Relation between parameters and name of response type table, updated the fourth row to add R7 as the response type.
System control (eSDHC_SYSCTL)	Updated the RSTA, RSTC, and RSTD bits access to write-only reads zero.
System control (eSDHC_SYSCTL) and Host controller version (eSDHC_HOSTVER)	Updated Reset value of HOSTVER, and added external spec reference to description field of SYSCTL register..

C.1.16 Universal Serial Bus Interface Revision History

Reference	Description
Master Interface Data Burst Size (USB_BURTSIZE), Transmit FIFO Tuning Controls (USB_TXFILLTUNING), and Endpoint Flush (USB_ENDPTFLUSH)	Updated the following register names: <ul style="list-style-type: none"> Name of BURTSIZE from Programmable burst size to Master interface data burst size Name of TXFILLTUNING from Host TT transmit pre-buffer packet tuning to Transmit FIFO tuning controls Name of ENDPTFLUSH from Endpoint de-initialize to Endpoint flush
Capability Register Length (USB_CAPLENGTH)	Updated the description of CAPLENGTH bit.
Host Controller Structural Parameters (USB_HCSPARAMS)	Added following in description of USB_HCSPARAMS[PPC]: ULPI Mode: 0 – USBDR will write 0 for DrvVbus bit of OTGControl register in PHY. 1 – USBDR will write 1 for DrvVbus bit of OTGControl register in PHY. The OTGControl register is defined in ULPI specification.
Identification register (USB_USB_ID)	Updated bit fields for ID register.
USB memory map/register definition	Updated descriptions for USBSTS[PCI], PORTSC[CCS], PORTSC[PP], PORTSC[PIC], and USBCMD[RST] bit fields.
Table 16-75	Updated the Queue Head Layout table.
Device controller initialization	Added paragraph starting with "To configure the external ULPI PHY....." below the second paragraph.
Host controller initialization	Added paragraph starting with "To configure the external ULPI PHY....." below the first paragraph.
Control (USB_CONTROL)	Removed "UTMI mode: This bit is used to enable the... 0 Disable".
Figure 16-71	Updated number of elements from 6 to 32.
PHY clocks	Removed sentence "The clock is...".
PHY interface	Added first sentence The ULPI interface connects to an external PHY.
Control (USB_CONTROL) and Host controller initialization	Removed instances of 'PHY_CLK_VALID'.
USB_BURTSIZE	Removed register.
Transmit FIFO Tuning Controls (USB_TXFILLTUNING) and USB Device Mode (USB_USBMODE)	Updated reset value and added note in Transmit FIFO Tuning Controls (USB_TXFILLTUNING) .
Age Count Threshold (USB_AGE_CNT_THRESH)	Updated register description, removed internal content.
Port Status/Control (USB_PORTSC)	Updated reset value and description of [PP] bit.
Control (USB_CONTROL)	Updated description of the USB_EN bit.
Config1, Config2, and Status1	Added registers.
Transmit FIFO Tuning Controls (USB_TXFILLTUNING)	Updated note at the bottom of register description.
Host controller initialization	The sentence, "Optionally modify the ..." is removed.
Host Controller Structural Parameters (USB_HCSPARAMS)	"UTMI mode:" is removed from the description of PPC bit field.
Endpoint NAK Indicator Register (USB_ENDPTNAK), Endpoint NAK Indication	Added registers.

Table continues on the next page...

Reference	Description
Enable Register (USB_ENPTNAKEN), General purpose timer load n (USB_GPTIMERnLD), and Master Interface Data Burst Size (USB_BURSTSIZE)	
Overview	Updated diagram to remove on chip Phy
Transmit FIFO Tuning Controls (USB_TXFILLTUNING)	TXFILLTUNING register bitfield diagram now matches intent of read only.
Device controller initialization and Host controller initialization	Removed instances of USB_CONTROL[PHY_CLK_SEL].
System Interface Control (USB_SI_CTRL)	Stated limitation of 32 byte fetch mode in SI_CTRL, and updated reg description of TSFILLTUNING.

C.1.17 DUART Revision History

Reference	Description
Baud-Rate Generator Logic	Removed reference to ipg_clk. Clarified that this clock is the DUART module input clock.
DUART Features Summary	Moved the UART block diagram to Overview section.

C.1.18 PCI Express Interface Controller Revision History

Reference	Description
PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)	Added bit 13 ADVNFE Advisory Non-Fatal Error Status to the Correctable_Error_Status_Register and bit 13 ADVNFEM Advisory Non-Fatal Error Mask to the Correctable_Error_Mask_Register.
PCI Express Link Status Register (Link_Status_Register)	Changed the access of the Link Status Register[LABS, LBMS] to w1c. Added the following note to the bitfield descriptions: 'This bifield is write-1-to-clear in RC mode; it is read-only in EP mode'.
Type 0 configuration header registers and Type 1 configuration header registers	In the Type 0 header figure, moved the Expansion ROM Base Address to 30h. In the Type 1 header figure, added Expansion ROM Base Address at 38h.
PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)	Updated the reset value to 0000_00B0.
PCI Express Command Register (Command_Register)	Revised notes for RC mode in Bus_master and Memory_space bit fields to describe unique treatment of transactions targeting PEXCSRBAR space.
PCI Express Command Register (Command_Register)	Added Note "The above description does not apply for the inbound memory transaction to PEXCSRBAR, which is processed normally even when this bit is cleared, as long as the Memory Space bit is set." to bit 2 "Bus_master"

Table continues on the next page...

Substantive changes from revision 1 to revision 2

Reference	Description
PCI Express Link Capabilities Register (Link_Capabilities_Register)	For bit 9-4, description is updated as: Reset value depends upon the maximum link width of the PCI Express controller is capable of supporting.
PCI Express configuration address register (PEX_PEX_CONFIG_ADDR)	Revised the last sentence of register description to state, "the register address is: Extended register number II Register number II 00b."
Introduction	Added details about the implemented PCI Express controllers.
PCI Express error detect register (PEX_PEX_ERR_DR)	Revised bitfield description for PEX_ERR_DR[PAT].
PCI Express Interrupt Pin Register (Interrupt_Pin_Register)	Added a statement to register description that the register only applies to EP mode. Added footnote to register reset valuedistinguishing RC mode and EP mode reset values.
Figure 18-296	Changed Next Pointer at 0x4D to say "(EP: 0x88/RC: NULL)"
PCI Express Interrupt Pin Register (Interrupt_Pin_Register)	Added a statement to register description that the register only applies to EP mode.
PCI Express Link Capabilities Register (Link_Capabilities_Register)	Updated the description of the following bits: Port Number, L1_EX_LAT, L0s_EX_LAT and ASPM.
Outbound ATMU message generation and Inbound messages	Moved discussion of Outbound ATMU messages and Inbound messages into the chapter's functional description.
PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)	In the bitfield description for Advanced_Error_Capabilities_and_Control_Register[ECRCCE], added qualifier "On P4080, Rev 2," to note stating, "End-to-end CRC (ECRC) errors on packets may not be detected even if ECRC checking is enabled...." and added the statement, "ECRC checking is fully functional on P4080, Rev 3.
N_FTS Control Register (PEX_N_FTS_CTL) and ACK Replay Time-Out Register (PEX_ACK_REPLY_TIMEOUT)	Added new registers, N_FTS_Control_Register at offset 41Ch and ACK Replay Time-Out Register at 438h
Configuring ACK replay time-out when ASPM is enabled	Added new section.
PCI Express Interrupt Line Register (Interrupt_Line_Register)	In Interrupt_Line_Register[Interrupt_Line] bit description, added text "N/A for RC mode."
Type 0 configuration header registers Type 1 configuration header registers	Added the note "The BIST register (0x0F) is optional and reserved on the PCI Express controller."
PCI Express Power Management Timer Register (Power_Management_Timer_Register)	Corrected the reset value - the values for L1_WAIT_TIMER and L0s_TIME_IN were transposed.
PCI Express error capture status register (PEX_PEX_ERR_CAP_STAT)	
PCI Express MSI inbound window attributes register (PEX_PEXMSIWAR)	Updated TRGT bit reset to nnnn.
PCI Express MSI inbound window attributes register (PEX_PEXMSIWAR)	Updated the IWS bit setting.
PCI Express configuration address register (PEX_PEX_CONFIG_ADDR)	Revised the last sentence of register description to state, "the register address is: Extended register number II Register number II 00b."

C.1.19 Serial RapidIO Interface Revision History

Reference	Description
Accessing configuration registers using rapidIO packets	Removed the subheading, "Inbound Maintenance Accesses." Also, reorganized and modified the entire section.
SRIO	Corrected register naming typo's within P1PCSECCSR0, P2PCSECCSR0, P1ADIDCSR, P2ADIDCSR reg descriptions.

C.1.20 DMA Controller Revision History

Reference	Description
DMA channel operation	Updated the second sentence of the first paragraph as follows: In both modes, a channel can be activated by clearing and setting MRn[CS], or through the single-write start mode using MRn[CDSM/SWSM] and MRn[SRW], or through an external control mode using MRn[ECS_EN] and the external DMA_DREQ signal.
Table 20-3	Modified the introductory text to the following: This table describes the external DMA control signals. These signals are only utilized when operating in external master mode. See External control mode transfer .
DMA mode register (DMA_MRn)	In the bitfield description for DMAx_MRn[XFE], changed "...the new chaining features," to, "striding feature in direct mode or disable list chaining feature in chaining mode"
Basic chaining, single-write start mode	In step 1 of the sequence, changed "Set the mode register current descriptor start mode bit, MRn[CDSM_SWSM], and the extended features enable bit MRn[XFE]." to "Set the mode register current descriptor start mode bit, MRn[CDSM_SWSM], and clear the extended features enable bit MRn[XFE].".
DMA mode register (DMA_MRn)	Changed the bitfield description of DMAx_MRn[XFE] to: 0 Disable striding feature in direct mode or disable list chaining feature in chaining mode. 1 Enable striding feature in direct mode or enable list chaining feature in chaining mode.
DMA signal descriptionsSignal overview	Added note "External Control is supported on channel 0 only".
DMA features summary	Removed bullet regarding support for three priority levels.
DMA errors	Removed bullet, "Transfer started with a priority of three".
DMA mode register (DMA_MRn)	Updated description of bit setting '1111' of DMA_MRn[BWC].

Substantive changes from revision 1 to revision 2

C.1.21 Data Path Acceleration Architecture (DPAA) Overview and P4080 DPAA Implementation

Reference	Description
P4080 Datapath Three Speed Ethernet Controller (dTSEC) Implementation	Revised bullet to say that P4080 does not support RX preamble extraction.
Single-Buffer Frames	Updated description.
Figure 21-2	Updated Frame Descriptor to include LIODN_offset.

C.1.22 Multicore Programmable Interrupt Controller (MPIC) Revision History

Reference	Description
Internal interrupt n level register (MPIC_IILRn)	Added MPIC_IIVPR111, MPIC_IIDR111, MPIC_IILR111
Internal interrupt n vector/priority register (MPIC_IIVPRn)	
Internal interrupt n destination register (MPIC_IIDRn)	
Global- and private-access per-CPU registers	Added new section describing the global- and private-access per-CPU registers.
Feature reporting register (MPIC_FRR)	Updated FRR[NIRQ] value for P4080, Rev 3
Who am I register (MPIC_WHOAMI), Who am I register (MPIC_CPU _n _WHOAMI)	Updated reset value of ID bit to nnnnn.

C.1.23 Interrupt Assignments Revision History

Reference	Description
Internal Interrupt Sources	In Table 23-2, added PEX_PME_MES_DR to entries for PCI Express 1, PCI Express 2, and PCI Express 3.
Internal Interrupt Sources	In Table 23-2, added xref to PCI express controller internal interrupt sources to entries for PCI Express 1, PCI Express 2, and PCI Express 3.

C.1.24 General Purpose I/O (GPIO) Revision History

Reference	Description
GPIO interrupt event register (GPIO_GPIER)	Added note saying "Some implementations may....."
throughout	Updated whole chapter as generic and added chip specific topic "The GPIO module as implemented on the chip"

Table continues on the next page...

Reference	Description
GPIO data register (GPIO_GPDAT)	Updated the GPDAT[Dn] bit description.
GPIO interrupt event register (GPIO_GPIER)	Updated the GPIER reset value to "Undefined. User should write 1s to clear before using."

C.1.25 Device Configuration and Pin Control Revision History

Reference	Description
Device Configuration and Pin Control Memory Map/Register Definition	In the memory map table, changed the register Access value for RSTRQPBLSR (at offset 0B4h) to w1c.
Pin Multiplexing Control Register (DCFG_PMUXCR)	Replaced the cross-reference to the Timer Control Register (TMR_CTRL) with a reference to the <i>QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual</i> (DPAARM).
Device Configuration and Pin Control Memory Map/Register Definition	In PMUXCR revised MSRCID and ALT_MSRCID bitfields to only support eLBC. Also revised the register descriptions for Debug Source ID 2A and 2B registers.
Reset request WDT status register (DCFG_RSTREQWDTSR)	Added xref to TCR register.

C.1.26 Run Control/Power Management (RCPM) Revision History

Reference	Description
Device Sleeping Mode and RCPM Signal Descriptions	Changed the pin ASLEEP from active-low to active-high.
RCPM Memory Map/Register Definition	In the memory map table, changed the register Access value for CDOZPSR (at offset 024h) to w1c.
Run Control/Power Management (RCPM)	Removed references to Domain Deep Sleep Mode. This chip does not support Deep Sleep mode.
Core Timebase and Decrementer Operation	Removed introductory text.

C.1.27 Debug and Performance Monitoring Overview Revision History

Reference	Description
Debug and Performance Monitoring Overview	Removed chapter from manual. Please contact your preferred third-party tools vendor for information and guidance for leveraging debug capabilities of QorIQ products.

C.2 Substantive changes from revision 0 to revision 1

Substantive changes from revision 0 to revision 1 are as follows:

C.2.1 Overview Revision History

Reference	Description
e500 Core and Cache Memory Complex	In the bulleted statement that begins "Two cache line reads..." changed "(1024 bytes)" to "(1024 bits)."

C.2.2 Memory Map Revision History

Reference	Description
throughout	Modified text to eliminate spurious or redundant details that are specified elsewhere in the document.

C.2.3 Signal Descriptions Revision History

Reference	Description
SerDes Bank n Reset Control Register (Signals_SRDSBnRSTCTL)	In the description for SRDSBnRSTCTL[PSCL], revised sentence that said, "This value should be changed only during reset = 1." to say "This value should be changed only when RSTREQ = 1."
MPIC and GPIO Signal Multiplexing	Added note regarding setting MPIC_EIVPRn polarity to active high when IRQ field selects GPIO functionality
SerDes Control Memory Map	Updated the offsets for B1GCRB1-B1GCRJ1.

C.2.4 Reset, Clocking, and Initialization Revision History

Reference	Description
Power-On Reset Sequence	Revised step 1 of the sequence to indicate that PORESET is asserted as the power is applied to the chip. This brings the sequence in agreement with the requirements in the hardware specifications.
throughout	Changed active-low signals from overbar to _B.

Table continues on the next page...

Reference	Description
Updating CCSRBARs	Revised the text in step 1 of the procedure for better clarity.
RCW Field Definitions	Added note to IRQ field (bits 373-381) regarding setting MPIC_EIVPRn polarity to active high when IRQ field selects GPIO functionality.
RCW Field Definitions	In RCW[SYS_PLL_RAT], added settings for 11:1 and 12:1; also added a note stating, "All ratios may not be supported due to frequency restrictions. Refer to the hardware specifications for the supported frequencies."
RCW Field Definitions	In EC1 and EC2 bitfields, changed "MAC" to "dTSEC" to be consistent with all other references in the manual.
Hard Coded RCW Options	Changed "FM1 MAC1" to "FM1-dTSEC" to be consistent with all other references in the manual.
RCW Field Definitions	In EC1 and EC2 bitfields, revised the note clarifying incompatibilities between RGMII mode and the corresponding dTSEC's SerDes lanes being powered up.
Clocking Memory Map/Register Definition	Changed access type for PLLnCGSR from R (read-only) to R/W (read/write) because the KILL bitfield is writable.
Table 4-38	Updated clocking configuration status register fields in the table
RCW Field Definitions	In CC3_PLL_CFG and CC3_PLL_RAT bitfields, revised the notes regarding P4040 recommended settings to state in P4040 mode, set CC3_PLL_RAT = 0_1000 and set CC3_PLL_CFG to the correct value depending on the frequency of SYSCLK.

C.2.5 Pre-Boot Loader Revision History

Reference	Description
PBL Error Codes	Added note to error encoding 0x21 stating that some eLBC errors flagged in LTESR will be reported as a data transfer error from memory devices (encoding 0x78).
Software Recommendations	Removed bullet "PBI commands should not directly initialize the direction (input/output) of the GPIOs by writing to GPIO CCSR space."

C.2.6 Secure Boot and Trust Architecture Revision History

Reference	Description
Security monitor features summary	Added bulleted list under Security State Machine "Receives configuration input....."
Security monitor clock sources	Added "This clock can be gated outside of the....." Added "This clock can be gated outside of the....."

C.2.7 e500mc Core Integration Revision History

There were no changes.

C.2.8 CoreNet Platform Cache (CPC) Revision History

Reference	Description
CPC configuration and status register 0 (CPC_CPCCSR0)	Added CPCPE field (bit 1) to CPCCSR0.
Features	Revised section heading to "Features" and deleted several bullets describing details that might be interpreted incorrectly.
Features	Revised bullet describing replacement algorithm to include other configurable options (Streaming PLRU and FIFO)
SRAM Mode Registers	Revised paragraph describing speculative request behavior when CPC is configured as SRAM.
Decorated Store Operations	Added footnote to Address Alignment column head explaining that Addr[60:63] refers to the core's effective address not the physical address.
CPC configuration and status register 0 (CPC_CPCCSR0)	In CPCCSR0 register description, moved sequence for modifying CPCCSR0 to new topic within Programming Examples section. In CPCHDBCR0[SPEC_DIS] bit field description, added note saying: This bit should be set to a 1 if CPC SRAM is enabled and the SRAM address space does not overlap an enabled address region in the DRAM controller. In CPCERRATTR[Snarf], updated bit field description to: Transaction was initiated by an intervention clean ("snarf") copyback to memory (for example, a Read transaction that receives intervention data from a core cache). In CPCPAR0[DSTALLOC] bit field settings, updated "WBI" to "writeback WBI".
CPC error injection control register (CPC_CPCERRINJCTL)	In CPCERRINJCTL[TERRIEN] bit field description, changed CPCERRINJLO[0:31] to CPCERRINJLO[11:31].
Modifying CPC Control and Status Registers	Added new topic "Modifying CPC Control and Status Registers" under "Programming Examples".
SRAM Mode Registers	Added sentence: "Additionally, the interleave granularity must be less than or equal to the size of an individual CPC's SRAM."
Decorated Store Operations	In "Decorated Storage Operations" table, changed inverted "increment" and "decrement" that appear in "Function" column as needed to match the description in "Update Operation" column.

C.2.9 CoreNet Coherency Fabric (CCF) Revision History

Reference	Description
CCF Error Detect Register (CCF_CEDR) and CCF Error Capture Attribute Register (CCF_CECAR)	Changed access type for all bits in CCF_CEDR and CCF_CECAR to write-1-to-clear (w1c)

C.2.10 Peripheral Access Management Unit (PAMU) Revision History

Reference	Description
PAMU Memory Map	Removed PAMU_EEDIS register. Error checking must be enabled for normal operation which was the default setting for PAMU_EEDIS..
System Set-Up for PAMU Operation	Added note regarding PAMU bypass mode.

C.2.11 DDR Memory Controller Revision History

Reference	Description
DDR Memory Controller Memory Map	Removed DDR_EOR.
Memory data path read capture ECC (DDR_CAPTURE_ECC)	Updated the bitfield description of ECE in CAPTURE_ECC to say: 0:7 -8-bit ECC for the 16 bits in beats 0 & 4 in 16-bit bus mode; should be ignored for 32-bit and 64-bit mode 8:15 -8-bit ECC for the 16 bits in beats 1 & 5 in 16-bit bus mode; should be ignored for 32-bit and 64-bit bus mode 16:23 -8-bit ECC for the 16 bits in beats 2 & 6 in 16-bit bus mode; for the 32 bits in beats 0 & 2 & 4 & 6 in 32-bit bus mode; should be ignored for 64-bit mode 24:31 -8-bit ECC for the 16 bits in beats 3 & 7 in 16-bit bus mode; for the 32 bits in beats 1 & 3 & 5 & 7 in 32-bit bus mode; should be used for every beat in 64-bit mode
DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)	Changed the bit 14 to Reserved.
DDR write leveling control (DDR_DDR_WRLVL_CNTL)	Changed the binaries for 6 clocks and 14 clocks in the WRLVL_SMPL bitfield description. Changed 1010 to 0110 for 6 clocks and 1110 for the 14 clocks in DDR_WRLVL_CNTL.
DDR Memory Test Control Register (DDR_DDR_MTCR)	Added the following note: "The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN] to DDR_MTCR."
DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)	Modified the bitfield description for chips that support DDR3/DDR2 to: ACT_PD_EXIT Active: powerdown exit timing (DDR2: tXARD and tXARDS; DDR3: tXP). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. ODT_PD_EXIT ODT: powerdown exit timing (DDR2: tAXPD, DDR3: set to 1). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. For devices that support DDR3/DDR3L to ACT_PD_EXIT Active: powerdown exit timing (tXP). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. ODT_PD_EXIT ODT: powerdown exit timing (set to 1). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted in TIMING_CFG_0 register."
DDR Self Refresh Counter (DDR_DDR_SR_CNTR)	DDR_SR_CNTR[SR_IT] 0001 setting changed to reserved for chips that support DDR3/3L and for devices with DDR2/3, changed to say "(DDR2 only)". Replaced the unit of "clock" with "data rate".
DDR Signals Overview	Removed footnotes to the table.
DDR Signals Overview	Changed reset for D1_MDIC[0:1] and D2_MDIC[0:1] to "See hardware specifications."

Table continues on the next page...

Substantive changes from revision 0 to revision 1

Reference	Description
Memory Interface Signals	In the description column for Dn_MDIC[0:1], removed the specific value for pull-up/pull-down resistors and added statement to "See the chip's hardware specifications for specific values."

C.2.12 I2C Revision History

There were no changes.

C.2.13 Enhanced Local Bus Controller (eLBC) Revision History

Reference	Description
eLBC external signal descriptions	Rewrote (simplified) description of LA in detailed signal table.
External access termination (LGTA_B)	In the first paragraph, changed "LGTA should be asserted for at least one bus cycle to be effective" to "LGTA must be asserted for two bus cycles to be effective."
Figure 13-70	Updated figure
Table 13-208	Added commands FMR and LSOR to the table

C.2.14 Enhanced Serial Peripheral Interface Revision History

Reference	Description
eSPI mode register (ESPI_SPMODE)	Added HO_ADJ field (bits 13-15) to SPMODE register.
eSPI detailed signal descriptions	Changed MOSI to I/O (was just O) because it can serve as a second input for Winbond dual output read.

C.2.15 Enhanced Secure Digital Host Controller Revision History

Reference	Description
Present state (eSDHC_PRSSTAT)	Changed clock name reference in PRSSTAT[SDOFF] from SDHC to SD
Enhanced Secure Digital Host Controller (eSDHC) Memory Map	In the bitfield description for ESDHC_PROCTL[D3CD], removed sentence regarding eSDHC DAT3 signal is used as chip select for SPI mode. This statement does not apply for this chip.
Commands for MMC/SD	In the Commands for MMC/SD/SDIO/CE-ATA table, added CMD8 row for SD Cards. In addition, added the following note: "CMD8 differs for MMC and SD

Table continues on the next page...

Reference	Description
	cards. For SD cards, CMD8 is referred to as SEND_IF_COND. For MMC cards, CMD8 is referred to as SEND_EXT_CSD"
Reset	Removed reference to SDIO in the figure, figure caption and the pseudocode

C.2.16 Universal Serial Bus Interfaces Revision History

Reference	Description
Ping Control	Added following: Note For high-speed bulk and control...
Queue Head Transfer Overlay	Added following text in description of "NakCnt": Note that in host mode, the NAK counter is decremented after receiving a NYET to an OUT Token. Thus, the NAK counter may be lower than expected.
ULPI_VIEWPORT	In USBI_ULPI_VIEWPORT, added the note that begins with "Read or write to the ULPI PHY extended register set..." before the register figure.
PORTSC	In the Port Status and Control Register, added the following in the bit description of bits 19-16 [PTC]: "Note that for K_STATE test mode (Test_K), a controller reset (Host mode) or power cycle (Device mode) is required after the test completes."
	In the Programmable burst size register, added following: "Note that BURSTSIZE[TXPBURST] and BURSTSIZE[RXPBURST] are effectively write-only fields. The actual value written in the register is used correctly for the burst length, but the fields always return 10h."
CONTROL	Changed USBI_CONTROL[PHY_CLK_VALID] to w1c.
PORTR	In the bitfield description for USBn_PORTR[PTS], removed the 10 - Reserved entry.
Host Controller Initialization and Device Controller Initialization	Removed first step in the procedure regarding configuring the external ULPI PHY. This removed step concerned UTMI PHY which this chip does not support.
Device Controller Initialization	Changed register name in procedures from "USBGP" to "USB_CONTROL."
CONTROL	Added USBn_CONTROL[PHY_CLK_SEL and PHY_CLK_VALID] fields as they are necessary to initialize the USB controller.

C.2.17 DUART Revision History

There were no changes.

C.2.18 PCI Express Interface Controller Revision History

Reference	Description
Error capture registers (inbound error)	Removed description of PEX_ERR_CAP_Rn from register section and added it to functional description (new sections error_capture_registers_inbound and error_capture_registers_outbound)
Error capture registers (outbound error)	

Table continues on the next page...

Substantive changes from revision 0 to revision 1

Reference	Description
PEX	
Initial credit advertisement	In Table 18-363, change values from: PH=4, PD=(256/16)x4=64, NPH = 4 Change To: PH=6, PD=(256/16)x6=96, NPH = 8
PCI Express error disable register (PEX_PEX_ERR_DISR)	Revised sense/polarity in descriptions for PEX_ERR_DISR[IMBAIE, IIOBAIE, and LDDED].
PCI Express error capture status register (PEX_PEX_ERR_CAP_STAT)	Revised the encoded values in the bitfield description for GSID.

C.2.19 Serial RapidIO Interface Revision History

There were no changes.

C.2.20 DMA Revision History

Reference	Description
DMA current list descriptor address register (DMA_CLSDARn)	The name of DMA n current list descriptor address register (DMA $_n$ _CLSDAR $_n$) was mislabeled in the last revision and has been updated.

C.2.21 Data Path Acceleration Architecture (DPAA) Overview and P4080 DPAA Implementation Revision History

Reference	Description
P4080 Frame Manager (FMan) Implementation	Under 10GEC PortID (RX/TX), removed FM1 and FM2 qualifiers and removed "0x11/0x31." There is one set of PortIDs 0x10/0x30; the Port IDs are the same for FM1 and FM2.

C.2.22 Multicore Programmable Interrupt Controller (MPIC) Revision History

Reference	Description
MPIC modes of operation	Added External Proxy Facility Mode(GCR[M] = 11), Mixed Mode(GCR[M] = 01), and Pass-Through Mode(GCR[M] = 00) sub-sections under Modes of Operation section

Table continues on the next page...

Reference	Description
Internal interrupt n vector/priority register (MPIC_IIVPRn) and Internal interrupt n destination register (MPIC_IIDRn)	Removed reference to non-existent "Table 1-8" in descriptions for MPIC_IIDRn and MPIC_IIVPRn.

C.2.23 Interrupt Assignments Revision History

Reference	Description
Internal Interrupt Sources	<ul style="list-style-type: none"> In Table 23-1, for internal interrupt 9, added a note stating interrupt 9 is used for both eLBC event and error interrupts. In Table 23-2, deleted ORed interrupt 19.
Internal Interrupt Sources	In Table 23-1, for internal interrupts 12-19, renumbered the DMA channels 0-3 to match their channel numbers in the rest of this document.

C.2.24 General Purpose I/O (GPIO) Revision History

Reference	Description
GPIO interrupt mask register (GPIO_GPIMR)	Updated bitfield description for GPIMR[IMn].

C.2.25 Device Configuration and Pin Control Revision History

Reference	Description
	Changed registers at 0E_0544h and 0E_0548h to Reserved. These registers are not supported on P4080.
System version register (DCFG_SVR)	Modified the bitfield description for the MFG_ID and SOC_DEV_ID bits.
PAMU bypass enable register (DCFG_PAMUBYPENR)	Changed reset value for PAMUBYPEN and added footnote to say that the reset value depends on whether secure boot is enabled.

C.2.26 Run Control/Power Management (RCPM) Revision History

Reference	Description
Core doze status register (RCPM_CDOZSR)	Removed paragraph under Core doze status register (RCPM_CDOZSR) at offset 0E_2004h describing microarchitectural details not relevant to the programming model.
RCPM Memory Map/Register Definition	Added note to bit field descriptions of all registers describing the correspondence of core numbers to bits.

C.2.27 Debug and Performance Monitoring Overview Revision History

There were no changes.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, Energy Efficient Solutions logo, Kinetis, mobileGT, PowerQUICC, Processor Expert, QorIQ, Qorivva, StarCore, Symphony, and VortiQa are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Airfast, BeeKit, BeeStack, CoreNet, Flexis, Layerscape, MagniV, MXC, Platform in a Package, QorIQ Qonverge, QUICC Engine, Ready Play, SafeAssure, SafeAssure logo, SMARTMOS, Tower, TurboLink, Vybrid, and Xtrinsic are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010–2014 Freescale Semiconductor, Inc.

Document Number P4080RM
Revision 2, 05/2014

