MPC8572E PowerQUICC™ III Integrated Host Processor Family Reference Manual

Supports
MPC8572E
MPC8572

MPC8572ERM
Rev. 2
05/2008

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: MPC8572ERM
Rev. 2, 05/2008

Part I—Overview I
Overview 1
Memory Map 2
Signal Descriptions 3
Reset, Clocking, and Initialization 4
Part II—e500 Core Complex and L2 Cache II
Core Complex Overview 5
Core Register Summary 6
L2 Look-Aside Cache/SRAM 7
Part III—Memory, Security, and I/O Interfaces III
e500 Coherency Module 8
DDR Memory Controllers 9
Programmable Interrupt Controller (PIC) 10
Security Engine (SEC) 3.0 11
I2C Interfaces 12
DUART 13
Enhanced Local Bus Controller 14
Enhanced Three-Speed Ethernet Controllers 15
10/100 Fast Ethernet Controller 16
Pattern Matcher (PM) 1.1 17
Table Lookup Unit 18
DMA Controllers 19
Serial RapidIO Interface 20
PCI Express Interface Controller 21
General Purpose I/O (GPIO) 22
Part IV—Global Functions and Debug IV
Global Utilities 22
Device Performance Monitor 23
Debug Features and Watchpoint Facility 24

Revision History A

Glossary GLO
Index IND

I **Part I—Overview**
1 Overview
2 Memory Map
3 Signal Descriptions
4 Reset, Clocking, and Initialization
II **Part II—e500 Core Complex and L2 Cache**
5 Core Complex Overview
6 Core Register Summary
7 L2 Look-Aside Cache/SRAM
III **Part III—Memory, Security, and I/O Interfaces**
8 e500 Coherency Module
9 DDR Memory Controllers
10 Programmable Interrupt Controller (PIC)
11 Security Engine (SEC) 3.0
12 I2C Interfaces
13 DUART
14 Enhanced Local Bus Controller
15 Enhanced Three-Speed Ethernet Controllers
16 10/100 Fast Ethernet Controller
17 Pattern Matcher (PM) 1.1
18 Table Lookup Unit
19 DMA Controllers
20 Serial RapidIO Interface
21 PCI Express Interface Controller
22 General Purpose I/O (GPIO)
IV **Part IV—Global Functions and Debug**
23 Global Utilities
24 Device Performance Monitor
25 Debug Features and Watchpoint Facility

A Revision History

GLO Glossary
IND Index

Contents

Paragraph Number	Title	Page Number

About This Book

Audience cxxvii
Organization cxxvii
Suggested Reading cxxx
General Information cxxx
Related Documentation cxxx
Conventions cxxxi
Signal Conventions cxxxii
Acronyms and Abbreviations cxxxii

Part I
Overview

Chapter 1
Overview

1.1 Introduction 1-1
1.2 MPC8572E Overview 1-2
1.2.1 Key Features 1-2
1.3 MPC8572E Architecture Overview 1-3
1.3.1 e500 Core Overview 1-3
1.3.2 On-Chip Memory Unit 1-4
1.3.3 e500 Coherency Module (ECM) 1-4
1.3.4 DDR SDRAM Controllers 1-4
1.3.5 Programmable Interrupt Controller (PIC) 1-5
1.3.6 Integrated Security Engine (SEC) 1-5
1.3.7 I^2C Controllers 1-6
1.3.8 Boot Sequencer 1-6
1.3.9 Dual Universal Asynchronous Receiver/Transmitter (DUART) 1-6
1.3.10 Enhanced Local Bus Controller 1-7
1.3.11 Enhanced Three-Speed Ethernet Controllers (eTSECs) 1-7
1.3.12 10/100 Fast Ethernet Maintenance Interface (FEC) 1-9
1.3.13 Pattern Matching Engine (PME) with DEFLATE Engine 1-9
1.3.14 Table Lookup Unit (TLU) 1-9
1.3.15 OceaN Switch Fabric 1-10
1.3.16 Integrated DMA 1-10
1.3.17 High Speed I/O Interfaces 1-11

Contents

Paragraph Number	Title	Page Number
1.3.18	Serial RapidIO Interface	1-11
1.3.18.1	RapidIO Message Unit	1-11
1.3.19	PCI Express Interfaces	1-12
1.3.20	Power Management	1-12
1.3.21	Clocking	1-12
1.3.22	Address Map	1-13
1.3.23	Processing Across the On-Chip Fabric	1-13
1.3.24	Data Processing with the e500 Coherency Module	1-13
1.4	MPC8572E Application Examples	1-14
1.4.1	Dual-Core Device Application	1-14
1.4.2	Virtual Private Network (VPN) Access Router	1-16
1.4.3	High-Performance Communication System Using Distributed Processing	1-16
1.4.4	High-Performance Communication System	1-17
1.4.5	RAID Controller Application	1-17
1.4.6	SerDes Application	1-18
1.4.7	DSP Farm Application	1-18

Chapter 2
Memory Map

2.1	Local Memory Map Overview and Example	2-1
2.2	Address Translation and Mapping	2-3
2.2.1	SRAM Windows	2-4
2.2.2	Window into Configuration Space	2-4
2.2.3	Local Access Windows	2-4
2.2.3.1	Local Access Register Memory Map	2-5
2.2.3.2	Local Access IP Block Revision Register 1 (LAIPBRR1)	2-6
2.2.3.3	Local Access IP Block Revision Register 2 (LAIPBRR2)	2-6
2.2.3.4	Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR11)	2-7
2.2.3.5	Local Access Window *n* Attributes Registers (LAWAR0–LAWAR11)	2-8
2.2.3.6	Precedence of Local Access Windows	2-9
2.2.3.7	Configuring Local Access Windows	2-9
2.2.3.8	Distinguishing Local Access Windows from Other Mapping Functions	2-9
2.2.3.9	Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects	2-9
2.2.4	Outbound Address Translation and Mapping Windows	2-10
2.2.5	Inbound Address Translation and Mapping Windows	2-10
2.2.5.1	Serial RapidIO Inbound ATMU	2-10
2.2.5.2	PCI Express Inbound ATMU	2-10
2.2.5.3	Illegal Interaction Between Inbound ATMUs and Local Access Windows	2-11

Contents

Paragraph Number	Title	Page Number
2.3	Configuration, Control, and Status Register Map	2-11
2.3.1	Accessing CCSR Memory from the Local Processor	2-12
2.3.2	Accessing CCSR Memory from External Masters	2-13
2.3.3	Organization of CCSR Memory	2-13
2.3.4	General Utilities Registers	2-14
2.3.5	Interrupt Controller and CCSR	2-15
2.3.6	Serial RapidIO and CCSR	2-16
2.3.7	Device-Specific Utilities	2-16
2.3.8	Accessing Reserved Registers and Bits	2-17
2.4	Complete CCSR Map	2-17

Chapter 3
Signal Descriptions

3.1	Signals Overview	3-1
3.2	Configuration Signals Sampled at Reset	3-23
3.3	Output Signal States During Reset	3-25

Chapter 4
Reset, Clocking, and Initialization

4.1	Overview	4-1
4.2	External Signal Descriptions	4-1
4.2.1	System Control Signals	4-2
4.2.2	Clock Signals	4-3
4.3	Memory Map/Register Definition	4-4
4.3.1	Local Configuration Control	4-4
4.3.1.1	Accessing Configuration, Control, and Status Registers	4-5
4.3.1.1.1	Updating CCSRBAR	4-5
4.3.1.1.2	Configuration, Control, and Status Base Address Register (CCSRBAR)	4-6
4.3.1.2	Accessing Alternate Configuration Space	4-6
4.3.1.2.1	Alternate Configuration Base Address Register (ALTCBAR)	4-7
4.3.1.2.2	Alternate Configuration Attribute Register (ALTCAR)	4-7
4.3.1.3	Boot Page Translation	4-8
4.3.1.3.1	Boot Page Translation Register (BPTR)	4-9
4.3.2	Boot Sequencer	4-9
4.4	Functional Description	4-9
4.4.1	Reset Operations	4-9
4.4.1.1	Soft Reset	4-9
4.4.1.2	Hard Reset	4-10
4.4.2	Power-On Reset Sequence	4-10

Contents

Paragraph Number	Title	Page Number
4.4.3	Power-On Reset Configuration	4-12
4.4.3.1	System PLL Ratio	4-13
4.4.3.2	DDR PLL Ratio	4-13
4.4.3.3	e500 Core PLL Ratios	4-14
4.4.3.4	Boot ROM Location	4-15
4.4.3.5	Host/Agent Configuration	4-16
4.4.3.6	I/O Port Selection	4-17
4.4.3.7	CPU Boot Configuration	4-19
4.4.3.8	Boot Sequencer Configuration	4-20
4.4.3.9	DDR SDRAM Type	4-21
4.4.3.10	FEC Configuration	4-21
4.4.3.11	eTSEC SGMII Mode	4-21
4.4.3.12	SGMII SerDes Reference Clock Configuration	4-22
4.4.3.13	eTSEC1 and eTSEC2 Width	4-23
4.4.3.14	eTSEC3 and eTSEC4 Width	4-24
4.4.3.15	eTSEC1 Protocol	4-24
4.4.3.16	eTSEC2 Protocol	4-25
4.4.3.17	eTSEC3 Protocol	4-25
4.4.3.18	eTSEC4 Protocol	4-26
4.4.3.19	RapidIO Device ID	4-27
4.4.3.20	RapidIO System Size	4-27
4.4.3.21	Memory Debug Configuration	4-28
4.4.3.22	DDR Debug Configuration	4-28
4.4.3.23	General-Purpose POR Configuration	4-28
4.4.3.24	eLBC FCM ECC Configuration	4-29
4.4.3.25	SerDes1 Enable	4-29
4.4.3.26	SGMII SerDes Enable	4-29
4.4.3.27	Engineering Use POR Configuration	4-30
4.4.4	Clocking	4-30
4.4.4.1	System Clock and DDR Controller Complex Clock	4-30
4.4.4.2	RapidIO and PCI Express Clocks	4-31
4.4.4.3	SGMII Clocks	4-31
4.4.4.3.1	Minimum Frequency Requirements	4-32
4.4.4.4	Ethernet Clocks	4-32
4.4.4.5	Real Time Clock	4-32

Part II
e500 Core Complex and L2 Cache

Chapter 5
Core Complex Overview

Contents

Paragraph Number	Title	Page Number
5.1	Overview	5-1
5.1.1	Upward Compatibility	5-3
5.1.2	Core Complex Summary	5-3
5.2	e500 Processor and System Version Numbers	5-4
5.3	Features	5-5
5.3.1	e500v2 Differences	5-10
5.4	Instruction Set	5-11
5.5	Instruction Flow	5-13
5.5.1	Initial Instruction Fetch	5-13
5.5.2	Branch Detection and Prediction	5-13
5.5.3	e500 Execution Pipeline	5-14
5.6	Programming Model	5-16
5.7	On-Chip Cache Implementation	5-18
5.8	Interrupts and Exception Handling	5-18
5.8.1	Exception Handling	5-18
5.8.2	Interrupt Classes	5-19
5.8.3	Interrupt Types	5-19
5.8.4	Upper Bound on Interrupt Latencies	5-20
5.8.5	Interrupt Registers	5-20
5.9	Memory Management	5-22
5.9.1	Address Translation	5-24
5.9.2	MMU Assist Registers (MAS0–MAS4 and MAS6–MAS7)	5-25
5.9.3	Process ID Registers (PID0–PID2)	5-26
5.9.4	TLB Coherency	5-26
5.10	Memory Coherency	5-26
5.10.1	Atomic Update Memory References	5-27
5.10.2	Memory Access Ordering	5-27
5.10.3	Cache Control Instructions	5-27
5.10.4	Programmable Page Characteristics	5-27
5.11	Core Complex Bus (CCB)	5-27
5.12	Performance Monitoring	5-28
5.12.1	Global Control Register	5-28
5.12.2	Performance Monitor Counter Registers	5-28
5.12.3	Local Control Registers	5-29
5.13	Legacy Support of Power Architecture Technology	5-29
5.13.1	Instruction Set Compatibility	5-29
5.13.1.1	User Instruction Set	5-29
5.13.1.2	Supervisor Instruction Set	5-30
5.13.2	Memory Subsystem	5-30
5.13.3	Exception Handling	5-30
5.13.4	Memory Management	5-30

Contents

Paragraph Number	Title	Page Number
5.13.5	Reset	5-30
5.13.6	Little-Endian Mode	5-31
5.14	PowerQUICC III Implementation Details	5-31

Chapter 6
Core Register Summary

6.1	Overview	6-1
6.1.1	Register Set	6-1
6.2	Register Model for 32-Bit Implementations	6-3
6.2.1	Special-Purpose Registers (SPRs)	6-4
6.3	Registers for Computational Operations	6-8
6.3.1	General-Purpose Registers (GPRs)	6-8
6.3.2	Integer Exception Register (XER)	6-8
6.4	Registers for Branch Operations	6-9
6.4.1	Condition Register (CR)	6-9
6.4.2	Link Register (LR)	6-11
6.4.3	Count Register (CTR)	6-11
6.5	Processor Control Registers	6-11
6.5.1	Machine State Register (MSR)	6-11
6.5.2	Processor ID Register (PIR)	6-13
6.5.3	Processor Version Register (PVR)	6-13
6.5.4	System Version Register (SVR)	6-14
6.6	Timer Registers	6-14
6.6.1	Timer Control Register (TCR)	6-14
6.6.2	Timer Status Register (TSR)	6-15
6.6.3	Time Base Registers	6-16
6.6.4	Decrementer Register	6-16
6.6.5	Decrementer Auto-Reload Register (DECAR)	6-17
6.7	Interrupt Registers	6-17
6.7.1	Interrupt Registers Defined by the Embedded and Base Categories	6-17
6.7.1.1	Save/Restore Register 0 (SRR0)	6-17
6.7.1.2	Save/Restore Register 1 (SRR1)	6-17
6.7.1.3	Critical Save/Restore Register 0 (CSRR0)	6-17
6.7.1.4	Critical Save/Restore Register 1 (CSRR1)	6-18
6.7.1.5	Data Exception Address Register (DEAR)	6-18
6.7.1.6	Interrupt Vector Prefix Register (IVPR)	6-18
6.7.1.7	Interrupt Vector Offset Registers (IVOR*n*)	6-18
6.7.1.8	Exception Syndrome Register (ESR)	6-19
6.7.2	Additional Interrupt Registers	6-20
6.7.2.1	Machine Check Save/Restore Register 0 (MCSRR0)	6-20

Contents

Paragraph Number	Title	Page Number
6.7.2.2	Machine Check Save/Restore Register 1 (MCSRR1)	6-20
6.7.2.3	Machine Check Address Register (MCAR/MCARU)	6-21
6.7.2.4	Machine Check Syndrome Register (MCSR)	6-21
6.8	Software-Use SPRs (SPRG0–SPRG7 and USPRG0)	6-22
6.9	Branch Target Buffer (BTB) Registers	6-23
6.9.1	Branch Buffer Entry Address Register (BBEAR)	6-23
6.9.2	Branch Buffer Target Address Register (BBTAR)	6-23
6.9.3	Branch Unit Control and Status Register (BUCSR)	6-24
6.10	Hardware Implementation-Dependent Registers	6-25
6.10.1	Hardware Implementation-Dependent Register 0 (HID0)	6-25
6.10.2	Hardware Implementation-Dependent Register 1 (HID1)	6-26
6.11	L1 Cache Configuration Registers	6-28
6.11.1	L1 Cache Control and Status Register 0 (L1CSR0)	6-28
6.11.2	L1 Cache Control and Status Register 1 (L1CSR1)	6-29
6.11.3	L1 Cache Configuration Register 0 (L1CFG0)	6-30
6.11.4	L1 Cache Configuration Register 1 (L1CFG1)	6-31
6.12	MMU Registers	6-32
6.12.1	Process ID Registers (PID0–PID2)	6-32
6.12.2	MMU Control and Status Register 0 (MMUCSR0)	6-32
6.12.3	MMU Configuration Register (MMUCFG)	6-32
6.12.4	TLB Configuration Registers (TLB*n*CFG)	6-33
6.12.4.1	TLB0 Configuration Register 0 (TLB0CFG)	6-33
6.12.4.2	TLB1 Configuration Register 1 (TLB1CFG)	6-34
6.12.5	MMU Assist Registers	6-34
6.12.5.1	MAS Register 0 (MAS0)	6-34
6.12.5.2	MAS Register 1 (MAS1)	6-35
6.12.5.3	MAS Register 2 (MAS2)	6-36
6.12.5.4	MAS Register 3 (MAS3)	6-37
6.12.5.5	MAS Register 4 (MAS4)	6-38
6.12.5.6	MAS Register 6 (MAS6)	6-38
6.12.5.7	MAS Register 7 (MAS7)	6-39
6.13	Debug Registers	6-39
6.13.1	Debug Control Registers (DBCR0–DBCR2)	6-39
6.13.1.1	Debug Control Register 0 (DBCR0)	6-39
6.13.1.2	Debug Control Register 1 (DBCR1)	6-41
6.13.1.3	Debug Control Register 2 (DBCR2)	6-42
6.13.2	Debug Status Register (DBSR)	6-43
6.13.3	Instruction Address Compare Registers (IAC1–IAC2)	6-45
6.13.4	Data Address Compare Registers (DAC1–DAC2)	6-45
6.14	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)	6-45

Contents

Paragraph Number	Title	Page Number
6.14.1	Accumulator (ACC)	6-47
6.15	Performance Monitor Registers (PMRs)	6-48
6.15.1	Global Control Register 0 (PMGC0, UPMGC0)	6-49
6.15.2	Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3)	6-50
6.15.3	Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3)	6-51
6.15.4	Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3)	6-52

Chapter 7
L2 Look-Aside Cache/SRAM

7.1	L2 Cache Overview	7-1
7.1.1	L2 Cache and SRAM Features	7-1
7.2	L2 Cache and SRAM Organization	7-4
7.2.1	Accessing the On-Chip Array as an L2 Cache	7-5
7.2.2	Accessing the On-Chip Array as an SRAM	7-5
7.2.3	Connection of the On-Chip Memory to the System	7-7
7.3	Memory Map/Register Definition	7-8
7.3.1	L2/SRAM Register Descriptions	7-9
7.3.1.1	L2 Control Register (L2CTL)	7-9
7.3.1.2	L2 Cache Way Allocation for Processors Register (L2CWAP)	7-13
7.3.1.3	L2 Cache External Write Registers	7-15
7.3.1.3.1	L2 Cache External Write Address Registers 0–3 (L2CEWAR*n*)	7-15
7.3.1.3.2	L2 Cache External Write Address Registers Extended Address (L2CEWAREA*n*)	7-16
7.3.1.3.3	L2 Cache External Write Control Registers 0–3 (L2CEWCR*n*)	7-16
7.3.1.4	L2 Memory-Mapped SRAM Registers	7-17
7.3.1.4.1	L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR*n*)	7-18
7.3.1.4.2	L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA*n*)	7-19
7.3.1.5	L2 Error Registers	7-19
7.3.1.5.1	Error Injection Registers	7-20
7.3.1.5.2	Error Control and Capture Registers	7-22
7.4	External Writes to the L2 Cache (Cache Stashing)	7-27
7.4.1	Stash-Only Cache Regions	7-28
7.5	L2 Cache Timing	7-28
7.6	L2 Cache and SRAM Coherency	7-29
7.6.1	L2 Cache Coherency Rules	7-29
7.6.2	Memory-Mapped SRAM Coherency Rules	7-31
7.7	L2 Cache Locking	7-31
7.7.1	Locking the Entire L2 Cache	7-31
7.7.2	Locking Programmed Memory Ranges	7-32

Contents

Paragraph Number	Title	Page Number
7.7.3	Locking Selected Lines	7-32
7.7.4	Clearing Locks on Selected Lines	7-32
7.7.5	Flash Clearing of Instruction and Data Locks	7-32
7.7.6	Locks with Stale Data	7-33
7.8	PLRU L2 Replacement Policy	7-33
7.8.1	PLRU Bit Update Considerations	7-34
7.8.2	Allocation of Lines	7-34
7.9	L2 Cache Operation	7-35
7.9.1	Initialization	7-35
7.9.1.1	L2 Cache Initialization	7-35
7.9.1.2	Memory-Mapped SRAM Initialization	7-35
7.9.2	Flash Invalidation of the L2 Cache	7-36
7.9.3	Managing Errors	7-36
7.9.3.1	ECC Errors	7-36
7.9.3.2	Tag Parity Errors	7-36
7.9.4	L2 Cache States	7-36
7.9.5	L2 State Transitions	7-37
7.9.6	Error Checking and Correcting (ECC)	7-41

Part III
Memory, Security, and I/O Interfaces

Chapter 8
e500 Coherency Module

8.1	Introduction	8-1
8.1.1	Overview	8-2
8.1.2	Features	8-2
8.2	Memory Map/Register Definition	8-3
8.2.1	Register Descriptions	8-3
8.2.1.1	ECM CCB Address Configuration Register (EEBACR)	8-3
8.2.1.2	ECM CCB Port Configuration Register (EEBPCR)	8-4
8.2.1.3	ECM IP Block Revision Register 1 (EIPBRR1)	8-5
8.2.1.4	ECM IP Block Revision Register 2 (EIPBRR2)	8-6
8.2.1.5	ECM Error Detect Register (EEDR)	8-6
8.2.1.6	ECM Error Enable Register (EEER)	8-7
8.2.1.7	ECM Error Attributes Capture Register (EEATR)	8-8
8.2.1.8	ECM Error Low Address Capture Register (EELADR)	8-9
8.2.1.9	ECM Error High Address Capture Register (EEHADR)	8-9
8.3	Functional Description	8-10
8.3.1	I/O Arbiter	8-10

Contents

Paragraph Number	Title	Page Number
8.3.2	CCB Arbiter	8-10
8.3.3	Transaction Queue	8-10
8.3.4	Global Data Multiplexor	8-11
8.3.5	CCB Interface	8-11
8.4	Initialization/Application Information	8-11

Chapter 9
DDR Memory Controllers

9.1	Introduction	9-1
9.2	Features	9-2
9.2.1	Modes of Operation	9-3
9.3	External Signal Descriptions	9-3
9.3.1	Signals Overview	9-3
9.3.2	Detailed Signal Descriptions	9-6
9.3.2.1	Memory Interface Signals	9-7
9.3.2.2	Clock Interface Signals	9-11
9.3.2.3	Debug Signals	9-11
9.4	Memory Map/Register Definition	9-11
9.4.1	Register Descriptions	9-13
9.4.1.1	Chip Select Memory Bounds (CS*n*_BNDS)	9-13
9.4.1.2	Chip Select Configuration (CS*n*_CONFIG)	9-14
9.4.1.3	Chip Select Configuration 2 (CSn_CONFIG_2)	9-17
9.4.1.4	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)	9-17
9.4.1.5	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)	9-19
9.4.1.6	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)	9-21
9.4.1.7	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)	9-23
9.4.1.8	DDR SDRAM Control Configuration (DDR_SDRAM_CFG)	9-25
9.4.1.9	DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)	9-28
9.4.1.10	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)	9-30
9.4.1.11	DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)	9-31
9.4.1.12	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)	9-32
9.4.1.13	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)	9-34
9.4.1.14	DDR SDRAM Data Initialization (DDR_DATA_INIT)	9-35
9.4.1.15	DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)	9-35
9.4.1.16	DDR Initialization Address (DDR_INIT_ADDR)	9-36
9.4.1.17	DDR Initialization Enable Extended Address (DDR_INIT_EXT_ADDR)	9-36
9.4.1.18	DDR SDRAM Timing Configuration 4 (TIMING_CFG_4)	9-37
9.4.1.19	DDR SDRAM Timing Configuration 5 (TIMING_CFG_5)	9-39
9.4.1.20	DDR ZQ Calibration Control (DDR_ZQ_CNTL)	9-40
9.4.1.21	DDR Write Leveling Control (DDR_WRLVL_CNTL)	9-42

Contents

Paragraph Number	Title	Page Number
9.4.1.22	DDR Debug Status Register 1 (DDRDSR_1)	9-45
9.4.1.23	DDR Debug Status Register 2 (DDRDSR_2)	9-46
9.4.1.24	DDR Control Driver Register 1 (DDRCDR_1)	9-46
9.4.1.25	DDR Control Driver Register 2 (DDRCDR_2)	9-49
9.4.1.26	DDR IP Block Revision 1 (DDR_IP_REV1)	9-50
9.4.1.27	DDR IP Block Revision 2 (DDR_IP_REV2)	9-50
9.4.1.28	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	9-51
9.4.1.29	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	9-51
9.4.1.30	Memory Data Path Error Injection Mask ECC (ERR_INJECT)	9-52
9.4.1.31	Memory Data Path Read Capture High (CAPTURE_DATA_HI)	9-52
9.4.1.32	Memory Data Path Read Capture Low (CAPTURE_DATA_LO)	9-53
9.4.1.33	Memory Data Path Read Capture ECC (CAPTURE_ECC)	9-53
9.4.1.34	Memory Error Detect (ERR_DETECT)	9-54
9.4.1.35	Memory Error Disable (ERR_DISABLE)	9-55
9.4.1.36	Memory Error Interrupt Enable (ERR_INT_EN)	9-56
9.4.1.37	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)	9-57
9.4.1.38	Memory Error Address Capture (CAPTURE_ADDRESS)	9-58
9.4.1.39	Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS)	9-59
9.4.1.40	Single-Bit ECC Memory Error Management (ERR_SBE)	9-59
9.5	Functional Description	9-60
9.5.1	DDR SDRAM Interface Operation	9-64
9.5.1.1	Supported DDR SDRAM Organizations	9-64
9.5.2	DDR SDRAM Address Multiplexing	9-66
9.5.3	JEDEC Standard DDR SDRAM Interface Commands	9-70
9.5.4	DDR SDRAM Interface Timing	9-72
9.5.4.1	Clock Distribution	9-75
9.5.5	DDR SDRAM Mode-Set Command Timing	9-76
9.5.6	DDR SDRAM Registered DIMM Mode	9-77
9.5.7	DDR SDRAM Write Timing Adjustments	9-77
9.5.8	DDR SDRAM Refresh	9-78
9.5.8.1	DDR SDRAM Refresh Timing	9-79
9.5.8.2	DDR SDRAM Refresh and Power-Saving Modes	9-79
9.5.8.2.1	Self-Refresh in Sleep Mode	9-81
9.5.9	DDR Data Beat Ordering	9-82
9.5.10	Page Mode and Logical Bank Retention	9-82
9.5.11	Error Checking and Correcting (ECC)	9-83
9.5.12	Error Management	9-85
9.6	Initialization/Application Information	9-86
9.6.1	Programming Differences Between Memory Types	9-88
9.6.2	DDR SDRAM Initialization Sequence	9-93

Contents

Paragraph Number	Title	Page Number
9.6.3	Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System	9-93
9.6.3.1	Hardware Based Self-Refresh	9-93
9.6.3.2	Software Based Self-Refresh	9-93
9.6.3.3	Bypassing Re-initialization During Battery-Backed Operation	9-94

Chapter 10
Programmable Interrupt Controller (PIC)

10.1	Introduction	10-1
10.1.1	Overview	10-1
10.1.2	The PIC in Multiple-Processor Implementations	10-4
10.1.3	Interrupts to the Processor Core	10-4
10.1.4	Modes of Operation	10-5
10.1.4.1	Mixed Mode (GCR[M] = 1)	10-5
10.1.4.2	Pass-Through Mode (GCR[M] = 0)	10-5
10.1.5	Interrupt Sources	10-6
10.1.5.1	Interrupt Routing—Mixed Mode	10-6
10.1.5.2	Interrupt Destinations	10-7
10.1.5.3	Internal Interrupt Sources	10-7
10.2	External Signal Descriptions	10-9
10.2.1	Signal Overview	10-9
10.2.2	Detailed Signal Descriptions	10-9
10.3	Memory Map/Register Definition	10-10
10.3.1	Global Registers	10-20
10.3.1.1	Block Revision Register 1 (BRR1)	10-20
10.3.1.2	Block Revision Register 2 (BRR2)	10-21
10.3.1.3	Feature Reporting Register (FRR)	10-21
10.3.1.4	Global Configuration Register (GCR)	10-22
10.3.1.5	Vendor Identification Register (VIR)	10-23
10.3.1.6	Processor Core Initialization Register (PIR)	10-23
10.3.1.7	Interprocessor Interrupt Vector/Priority Registers (IPIVPR0–IPIVPR3)	10-24
10.3.1.8	Spurious Vector Register (SVR)	10-25
10.3.2	Global Timer Registers	10-25
10.3.2.1	Timer Frequency Reporting Register (TFRRA–TFRRB)	10-25
10.3.2.2	Global Timer Current Count Registers (GTCCRA0–GTCCRA3, GTCCRB0–GTCCRB3)	10-26
10.3.2.3	Global Timer Base Count Registers (GTBCRA0–GTBCRA3, GTBCRB0–GTBCRB3)	10-26
10.3.2.4	Global Timer Vector/Priority Registers (GTVPRA0–GTVPRA3, GTVPRB0–GTVPRB3)	10-27

Contents

Paragraph Number	Title	Page Number
10.3.2.5	Global Timer Destination Registers (GTDRA0–GTDRA3, GTDRB0–GTDRB3)	10-28
10.3.2.6	Timer Control Registers (TCRA–TCRB)	10-28
10.3.3	IRQ_OUT and Critical Interrupt Summary Registers	10-30
10.3.3.1	External Interrupt Summary Register (ERQSR)	10-31
10.3.3.2	IRQ_OUT Summary Register 0 (IRQSR0)	10-31
10.3.3.3	IRQ_OUT Summary Register 1 (IRQSR1)	10-32
10.3.3.4	IRQ_OUT Summary Register 2 (IRQSR2)	10-33
10.3.3.5	Critical Interrupt Summary Register 0 (CISR0)	10-33
10.3.3.6	Critical Interrupt Summary Register 1 (CISR1)	10-34
10.3.3.7	Critical Interrupt Summary Register 2 (CISR2)	10-34
10.3.4	Performance Monitor Mask Registers (PMMRs)	10-34
10.3.4.1	Performance Monitor Mask Registers 0 (PM0MR0–PM3MR0)	10-35
10.3.4.2	Performance Monitor Mask Registers 1 (PM0MR1–PM3MR1)	10-36
10.3.4.3	Performance Monitor Mask Registers 2 (PM0MR2–PM3MR2)	10-36
10.3.5	Message Registers	10-36
10.3.5.1	Message Registers (MSGR0–MSGR7)	10-37
10.3.5.2	Message Enable Register (MER)	10-37
10.3.5.3	Message Status Register (MSR)	10-38
10.3.6	Shared Message Signaled Registers	10-38
10.3.6.1	Shared Message Signaled Interrupt Registers (MSIR0–MSIR7)	10-39
10.3.6.2	Shared Message Signaled Interrupt Status Register (MSISR)	10-39
10.3.6.3	Shared Message Signaled Interrupt Index Register (MSIIR)	10-40
10.3.6.4	Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)	10-40
10.3.6.5	Shared Message Signaled Interrupt Destination Registers 0–7 (MSIDR*n*)	10-41
10.3.7	Interrupt Source Configuration Registers	10-42
10.3.7.1	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)	10-43
10.3.7.2	External Interrupt Destination Registers (EIDR0–EIDR11)	10-44
10.3.7.3	Internal Interrupt Vector/Priority Registers (IIVPRn)	10-45
10.3.7.4	Internal Interrupt Destination Registers (IIDRn)	10-46
10.3.7.5	Messaging Interrupt Vector/Priority Registers (MIVPRn)	10-47
10.3.7.6	Messaging Interrupt Destination Registers (MIDR0–MIDR7)	10-48
10.3.8	Per-CPU (Private Access) Registers	10-49
10.3.8.1	Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)	10-50
10.3.8.2	Processor Core Current Task Priority Registers 0–1 (CTPR0–CTPR1)	10-51
10.3.8.3	Who Am I Registers 0–1 (WHOAMI0–WHOAMI1)	10-52
10.3.8.4	Processor Core Interrupt Acknowledge Registers 0–1 (IACK0–IACK1)	10-52
10.3.8.5	Processor Core End of Interrupt Registers (EOI0–EOI1)	10-53
10.4	Functional Description	10-54
10.4.1	Flow of Interrupt Control	10-54
10.4.1.1	Interrupts Routed to cint or IRQ_OUT	10-54

Contents

Paragraph Number	Title	Page Number
10.4.1.2	Interrupts Routed to *int*	10-54
10.4.1.2.1	Nesting of Interrupts	10-56
10.4.1.2.2	Interrupt Source Priority	10-56
10.4.1.2.3	Interrupt Acknowledge	10-57
10.4.1.2.4	Spurious Vector Generation	10-57
10.4.2	Interprocessor Interrupts	10-57
10.4.3	Message Interrupts	10-58
10.4.4	Shared Message Signaled Interrupts	10-58
10.4.5	PCI Express INTx	10-58
10.4.6	Global Timers	10-59
10.4.7	Resets	10-60
10.4.8	Resetting the PIC	10-60
10.4.8.1	Processor Core Initialization	10-60
10.5	Initialization/Application Information	10-60
10.5.1	Programming Guidelines	10-60
10.5.1.1	PIC Registers	10-60
10.5.1.2	Changing Interrupt Source Configuration	10-62

Chapter 11
Security Engine (SEC) 3.0

11.1	Architecture Overview	11-3
11.1.1	Descriptors	11-5
11.1.2	Polychannel	11-6
11.1.3	Controller	11-7
11.1.3.1	Channel-Controlled Access to EUs	11-8
11.1.3.2	Host-Controlled Access to EUs	11-8
11.1.4	Execution Units (EUs)	11-9
11.1.4.1	Common EU Interface	11-9
11.1.4.2	Public Key Execution Unit (PKEU)	11-9
11.1.4.2.1	Elliptic Curve Operations	11-9
11.1.4.2.2	Modular Exponentiation Operations	11-10
11.1.4.3	Data Encryption Standard Execution Unit (DEU)	11-10
11.1.4.4	Advanced Encryption Standard Execution Unit (AESU)	11-11
11.1.4.5	Arc Four Execution Unit (AFEU)	11-12
11.1.4.6	Message Digest Execution Unit (MDEU)	11-12
11.1.4.7	Kasumi Execution Unit (KEU)	11-12
11.1.4.8	Cyclical Redundancy Check Unit (CRCU)	11-13
11.1.4.9	Random Number Generator (RNGU)	11-13
11.2	Configuration of Internal Memory Space	11-13
11.3	Descriptor Overview	11-21

Contents

Paragraph Number	Title	Page Number
11.3.1	Descriptor Structure	11-22
11.3.2	Descriptor Format: Header Dword	11-23
11.3.2.1	Selecting Execution Units—EU_SEL0 and EU_SEL1	11-25
11.3.2.2	Selecting Descriptor Type—DESC_TYPE	11-26
11.3.3	Descriptor Format: Pointer Dwords	11-27
11.3.4	Link Table Format	11-29
11.4	Descriptor Types	11-32
11.5	Polychannel	11-35
11.5.1	Arbitration Among Channels	11-36
11.5.1.1	Arbitration for Use of the Controller	11-36
11.5.1.2	Arbitration for Use of Execution Units	11-36
11.5.1.3	Arbitration Algorithms	11-37
11.5.1.3.1	Round-Robin Arbitration	11-37
11.5.1.3.2	Priority Arbitration	11-37
11.5.2	Polychannel Registers	11-38
11.5.2.1	Traffic Counters	11-38
11.5.2.1.1	Fetch FIFO Enqueue Counter	11-38
11.5.2.1.2	Descriptor Finished Counter	11-38
11.5.2.1.3	Data Bytes In Counter	11-38
11.5.2.1.4	Data Bytes Out Counter	11-39
11.5.3	Channel Interrupts	11-39
11.5.3.1	Channel Done Interrupt	11-40
11.5.3.2	Channel Error Interrupt	11-40
11.5.4	Channel Registers	11-40
11.5.4.1	Channel Configuration Register (CCR)	11-41
11.5.4.2	Channel Status Register (CSR)	11-44
11.5.4.3	Current Descriptor Pointer Register (CDPR)	11-47
11.5.4.4	Fetch FIFO (FF)	11-47
11.5.4.5	Descriptor Buffer (DB)	11-48
11.5.4.6	Scatter and Gather Link Tables (SLT, GLT)	11-48
11.6	Controller	11-49
11.6.1	Bus Transfers	11-49
11.6.1.1	System Bus Master Read	11-50
11.6.1.2	System Bus Master Write	11-51
11.6.2	Controller Interrupts	11-51
11.6.2.1	Controller Primary Interrupt	11-51
11.6.2.2	Controller Secondary Interrupt	11-52
11.6.3	Controller Registers	11-52
11.6.3.1	EU Assignment Status Register (EUASR)	11-52
11.6.3.2	Interrupt Enable Register (IER)	11-53
11.6.3.3	Interrupt Status Register (ISR)	11-56

Contents

Paragraph Number	Title	Page Number
11.6.3.4	Interrupt Clear Register (ICR)	11-57
11.6.3.5	ID Register	11-59
11.6.3.6	IP Block Revision Register	11-59
11.6.3.7	Master Control Register (MCR)	11-60
11.6.4	Snooping by Caches	11-61
11.7	Execution Units	11-62
11.7.1	Advanced Encryption Standard Execution Unit (AESU)	11-62
11.7.1.1	AESU Mode Register	11-63
11.7.1.2	AESU Key Size Register	11-67
11.7.1.3	AESU Data Size Register	11-68
11.7.1.4	AESU Reset Control Register	11-68
11.7.1.5	AESU Status Register	11-70
11.7.1.6	AESU Interrupt Status Register	11-71
11.7.1.7	AESU Interrupt Mask Register	11-73
11.7.1.8	ICV Size Register	11-74
11.7.1.9	AESU ICV Size Register	11-75
11.7.1.10	AESU End_of_Message Register	11-75
11.7.1.11	AESU Context Registers	11-75
11.7.1.11.1	Context for CBC, CBC-RBP, OFB, and CFB128 Cipher Modes	11-76
11.7.1.11.2	Context for Counter Cipher Mode	11-76
11.7.1.11.3	Context for SRT Cipher Mode	11-77
11.7.1.11.4	Context and Operation for XCBC-MAC Cipher Mode	11-78
11.7.1.11.5	Context and Operation for CMAC (OMAC1) Cipher Mode	11-79
11.7.1.11.6	Context for CCM Cipher Mode	11-80
11.7.1.11.7	Context and Operation for GCM Cipher Mode	11-83
11.7.1.11.8	AESU Key Registers	11-92
11.7.1.11.9	AESU FIFOs	11-92
11.7.2	ARC Four Execution Unit (AFEU)	11-92
11.7.2.1	AFEU Mode Register	11-93
11.7.2.2	AFEU Key Size Register	11-94
11.7.2.3	AFEU Context/Data Size Register	11-94
11.7.2.4	AFEU Reset Control Register	11-95
11.7.2.5	AFEU Status Register	11-96
11.7.2.6	AFEU Interrupt Status Register	11-97
11.7.2.7	AFEU Interrupt Mask Register	11-98
11.7.2.8	AFEU End of Message Register	11-100
11.7.2.9	AFEU Context	11-100
11.7.2.9.1	Writing AFEU Context	11-100
11.7.2.9.2	Reading AFEU Context	11-101
11.7.2.10	AFEU Key Registers	11-101
11.7.2.10.1	AFEU FIFOs	11-101

Contents

Paragraph Number	Title	Page Number
11.7.3	Cyclical Redundancy Check Unit (CRCU)	11-102
11.7.3.1	ICV Checking	11-102
11.7.3.2	CRCU Mode Register	11-103
11.7.3.3	CRCU Key Size Register	11-104
11.7.3.4	CRCU Data Size Register	11-104
11.7.3.5	CRCU Reset Control Register	11-104
11.7.3.6	CRCU Control Register	11-105
11.7.3.7	CRCU Status Register	11-106
11.7.3.8	CRCU Interrupt Status Register	11-107
11.7.3.9	CRCU Interrupt Mask Register	11-108
11.7.3.10	CRCU ICV Size Register	11-109
11.7.3.11	CRCU End of Message Register	11-110
11.7.3.12	CRCU Received ICV Register	11-110
11.7.3.13	CRCU Context Register	11-110
11.7.3.14	CRCU Key Register	11-111
11.7.3.15	CRCU FIFO	11-112
11.7.4	Data Encryption Standard Execution Unit (DEU)	11-112
11.7.4.1	DEU Mode Register	11-112
11.7.4.2	DEU Key Size Register	11-113
11.7.4.3	DEU Data Size Register	11-114
11.7.4.4	DEU Reset Control Register	11-115
11.7.4.5	DEU Status Register	11-116
11.7.4.6	DEU Interrupt Status Register	11-117
11.7.4.7	DEU Interrupt Mask Register	11-118
11.7.4.8	DEU End_of_Message Register	11-120
11.7.4.9	DEU IV Register	11-121
11.7.4.10	DEU Key Registers	11-121
11.7.4.11	DEU FIFOs	11-121
11.7.5	Kasumi Execution Unit (KEU)	11-121
11.7.5.1	KEU Mode Register (KEUMR)	11-122
11.7.5.2	KEU Key Size Register (KEUKSR)	11-124
11.7.5.3	KEU Data Size Register (KEUDSR)	11-124
11.7.5.4	KEU Reset Control Register (KEURCR)	11-125
11.7.5.5	KEU Status Register (KEUSR)	11-126
11.7.5.6	KEU Interrupt Status Register (KEUISR)	11-127
11.7.5.7	KEU Interrupt Mask Register (KEUIMR)	11-129
11.7.5.8	KEU Data Out Register (F9 MAC) (KEUDOR)	11-130
11.7.5.9	KEU End of Message Register (KEUEMR)	11-131
11.7.5.10	KEU IV_1 Register (KEUIV1)	11-131
11.7.5.11	KEU ICV_In Register (KEUICV)	11-132
11.7.5.12	KEU IV_2 Register (Fresh) (KEUIV2)	11-133

Contents

Paragraph Number	Title	Page Number
11.7.5.13	KEU Context Data Registers (KEUC*n*)	11-133
11.7.5.14	KEU Key Data Registers_1 and _2 (Confidentiality Key) (KEUKD*n*)	11-133
11.7.5.15	KEU Key Data Registers _3 and _4 (Integrity Key) (KEUKD*n*)	11-134
11.7.5.16	KEU FIFOs	11-135
11.7.6	Message Digest Execution Unit (MDEU)	11-135
11.7.6.1	ICV Checking	11-135
11.7.6.2	MDEU Mode Register	11-136
11.7.6.3	Recommended Settings for MDEU Mode Register	11-139
11.7.6.4	MDEU Key Size Register	11-140
11.7.6.5	MDEU Data Size Register	11-140
11.7.6.6	MDEU Reset Control Register	11-141
11.7.6.7	MDEU Status Register	11-142
11.7.6.8	MDEU Interrupt Status Register	11-143
11.7.6.9	MDEU Interrupt Mask Register	11-145
11.7.6.10	MDEU ICV Size Register	11-146
11.7.6.11	MDEU End_of_Message Register	11-146
11.7.6.12	MDEU Context Registers	11-147
11.7.6.13	MDEU Key Registers	11-150
11.7.6.14	MDEU FIFOs	11-150
11.7.7	Public Key Execution Units (PKEU)	11-150
11.7.7.1	PKEU Mode Register	11-151
11.7.7.2	PKEU Key Size Register	11-151
11.7.7.3	PKEU AB Size Register	11-153
11.7.7.4	PKEU Data Size Register	11-153
11.7.7.5	PKEU Reset Control Register	11-154
11.7.7.6	PKEU Status Register	11-155
11.7.7.7	PKEU Interrupt Status Register	11-156
11.7.7.8	PKEU Interrupt Mask Register	11-157
11.7.7.9	PKEU End_of_Message Register	11-158
11.7.7.10	PKEU Parameter Memories	11-159
11.7.7.10.1	PKEU Parameter Memory A	11-159
11.7.7.10.2	PKEU Parameter Memory B	11-159
11.7.7.10.3	PKEU Parameter Memory E	11-159
11.7.7.10.4	PKEU Parameter Memory N	11-159
11.7.8	Random Number Generator (RNGU)	11-159
11.7.8.1	RNGU Mode Register	11-160
11.7.8.2	RNGU Data Size Register	11-161
11.7.8.3	RNGU Reset Control Register	11-161
11.7.8.4	RNGU Status Register	11-162
11.7.8.5	RNGU Interrupt Status Register	11-163
11.7.8.6	RNGU Interrupt Mask Register	11-164

Contents

Paragraph Number	Title	Page Number
11.7.8.7	RNGU End_of_Message Register	11-165
11.7.8.8	RNGU ENTROPY	11-165
11.7.8.9	RNGU FIFO	11-166
11.8	Power Saving Mode	11-166

Chapter 12
I^2C Interfaces

12.1	Introduction	12-1
12.1.1	Overview	12-2
12.1.2	Features	12-2
12.1.3	Modes of Operation	12-2
12.2	External Signal Descriptions	12-3
12.2.1	Signal Overview	12-3
12.2.2	Detailed Signal Descriptions	12-3
12.3	Memory Map/Register Definition	12-4
12.3.1	Register Descriptions	12-5
12.3.1.1	I^2C Address Register (I2CADR)	12-6
12.3.1.2	I^2C Frequency Divider Register (I2CFDR)	12-6
12.3.1.3	I^2C Control Register (I2CCR)	12-7
12.3.1.4	I^2C Status Register (I2CSR)	12-9
12.3.1.5	I^2C Data Register (I2CDR)	12-10
12.3.1.6	Digital Filter Sampling Rate Register (I2CDFSRR)	12-11
12.4	Functional Description	12-11
12.4.1	Transaction Protocol	12-11
12.4.1.1	START Condition	12-12
12.4.1.2	Slave Address Transmission	12-12
12.4.1.3	Repeated START Condition	12-13
12.4.1.4	STOP Condition	12-13
12.4.1.5	Protocol Implementation Details	12-13
12.4.1.5.1	Transaction Monitoring—Implementation Details	12-14
12.4.1.5.2	Control Transfer—Implementation Details	12-14
12.4.1.6	Address Compare—Implementation Details	12-15
12.4.2	Arbitration Procedure	12-15
12.4.2.1	Arbitration Control	12-15
12.4.3	Handshaking	12-16
12.4.4	Clock Control	12-16
12.4.4.1	Clock Synchronization	12-16
12.4.4.2	Input Synchronization and Digital Filter	12-16
12.4.4.2.1	Input Signal Synchronization	12-17
12.4.4.2.2	Filtering of SCL and SDA Lines	12-17

Contents

Paragraph Number	Title	Page Number
12.4.4.3	Clock Stretching	12-17
12.4.5	Boot Sequencer Mode	12-17
12.4.5.1	EEPROM Calling Address	12-19
12.4.5.2	EEPROM Data Format	12-19
12.5	Initialization/Application Information	12-21
12.5.1	Initialization Sequence	12-21
12.5.2	Generation of START	12-22
12.5.3	Post-Transfer Software Response	12-22
12.5.4	Generation of STOP	12-23
12.5.5	Generation of Repeated START	12-23
12.5.6	Generation of SCL When SDA Low	12-23
12.5.7	Slave Mode Interrupt Service Routine	12-23
12.5.7.1	Slave Transmitter and Received Acknowledge	12-24
12.5.7.2	Loss of Arbitration and Forcing of Slave Mode	12-24
12.5.8	Interrupt Service Routine Flowchart	12-24

Chapter 13
DUART

13.1	Overview	13-1
13.1.1	Features	13-1
13.1.2	Modes of Operation	13-2
13.2	External Signal Descriptions	13-3
13.2.1	Signal Overview	13-3
13.2.2	Detailed Signal Descriptions	13-3
13.3	Memory Map/Register Definition	13-3
13.3.1	Register Descriptions	13-5
13.3.1.1	Receiver Buffer Registers (URBR*n*) (ULCR[DLAB] = 0)	13-5
13.3.1.2	Transmitter Holding Registers (UTHR*n*) (ULCR[DLAB] = 0)	13-5
13.3.1.3	Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (ULCR[DLAB] = 1)	13-6
13.3.1.4	Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)	13-8
13.3.1.5	Interrupt ID Registers (UIIR*n*) (ULCR[DLAB] = 0)	13-8
13.3.1.6	FIFO Control Registers (UFCR*n*) (ULCR[DLAB] = 0)	13-10
13.3.1.7	Line Control Registers (ULCR*n*)	13-11
13.3.1.8	Modem Control Registers (UMCR*n*)	13-13
13.3.1.9	Line Status Registers (ULSR*n*)	13-14
13.3.1.10	Modem Status Registers (UMSR*n*)	13-15
13.3.1.11	Scratch Registers (USCR*n*)	13-16
13.3.1.12	Alternate Function Registers (UAFR*n*) (ULCR[DLAB] = 1)	13-16
13.3.1.13	DMA Status Registers (UDSR*n*)	13-17

Contents

Paragraph Number	Title	Page Number
13.4	Functional Description	13-18
13.4.1	Serial Interface	13-19
13.4.1.1	START Bit	13-19
13.4.1.2	Data Transfer	13-20
13.4.1.3	Parity Bit	13-20
13.4.1.4	STOP Bit	13-20
13.4.2	Baud-Rate Generator Logic	13-20
13.4.3	Local Loopback Mode	13-21
13.4.4	Errors	13-21
13.4.4.1	Framing Error	13-21
13.4.4.2	Parity Error	13-21
13.4.4.3	Overrun Error	13-21
13.4.5	FIFO Mode	13-21
13.4.5.1	FIFO Interrupts	13-22
13.4.5.2	DMA Mode Select	13-22
13.4.5.3	Interrupt Control Logic	13-22
13.5	DUART Initialization/Application Information	13-23

Chapter 14
Enhanced Local Bus Controller

14.1	Introduction	14-1
14.1.1	Overview	14-2
14.1.2	Features	14-2
14.1.3	Modes of Operation	14-3
14.1.3.1	eLBC Bus Clock and Clock Ratios	14-3
14.1.3.2	Source ID Debug Mode	14-4
14.2	External Signal Descriptions	14-4
14.3	Memory Map/Register Definition	14-8
14.3.1	Register Descriptions	14-10
14.3.1.1	Base Registers (BR0–BR7)	14-10
14.3.1.2	Option Registers (OR0–OR7)	14-12
14.3.1.2.1	Address Mask	14-13
14.3.1.2.2	Option Registers (OR*n*)—GPCM Mode	14-14
14.3.1.2.3	Option Registers (OR*n*)—FCM Mode	14-16
14.3.1.2.4	Option Registers (OR*n*)—UPM Mode	14-19
14.3.1.3	UPM Memory Address Register (MAR)	14-20
14.3.1.4	UPM Mode Registers (M*x*MR)	14-21
14.3.1.5	Memory Refresh Timer Prescaler Register (MRTPR)	14-23
14.3.1.6	UPM/FCM Data Register (MDR)	14-23
14.3.1.7	Special Operation Initiation Register (LSOR)	14-24

Contents

Paragraph Number	Title	Page Number
14.3.1.8	UPM Refresh Timer (LURT)	14-25
14.3.1.9	Transfer Error Status Register (LTESR)	14-26
14.3.1.10	Transfer Error Check Disable Register (LTEDR)	14-28
14.3.1.11	Transfer Error Interrupt Enable Register (LTEIR)	14-29
14.3.1.12	Transfer Error Attributes Register (LTEATR)	14-30
14.3.1.13	Transfer Error Address Register (LTEAR)	14-31
14.3.1.14	Local Bus Configuration Register (LBCR)	14-32
14.3.1.15	Clock Ratio Register (LCRR)	14-33
14.3.1.16	Flash Mode Register (FMR)	14-34
14.3.1.17	Flash Instruction Register (FIR)	14-36
14.3.1.18	Flash Command Register (FCR)	14-37
14.3.1.19	Flash Block Address Register (FBAR)	14-38
14.3.1.20	Flash Page Address Register (FPAR)	14-38
14.3.1.21	Flash Byte Count Register (FBCR)	14-40
14.4	Functional Description	14-40
14.4.1	Basic Architecture	14-42
14.4.1.1	Address and Address Space Checking	14-42
14.4.1.2	External Address Latch Enable Signal (LALE)	14-42
14.4.1.3	Data Transfer Acknowledge (TA)	14-43
14.4.1.4	Data Buffer Control (LBCTL)	14-44
14.4.1.5	Atomic Operation	14-44
14.4.1.6	Parity Generation and Checking (LDP)	14-45
14.4.1.7	Bus Monitor	14-45
14.4.1.8	PLL Bypass Mode	14-45
14.4.2	General-Purpose Chip-Select Machine (GPCM)	14-46
14.4.2.1	GPCM Read Signal Timing	14-47
14.4.2.2	GPCM Write Signal Timing	14-49
14.4.2.3	Chip-Select Assertion Timing	14-50
14.4.2.3.1	Programmable Wait State Configuration	14-51
14.4.2.3.2	Chip-Select and Write Enable Negation Timing	14-51
14.4.2.3.3	Relaxed Timing	14-52
14.4.2.3.4	Output Enable (LOE) Timing	14-54
14.4.2.3.5	Extended Hold Time on Read Accesses	14-54
14.4.2.4	External Access Termination (LGTA)	14-55
14.4.2.5	GPCM Boot Chip-Select Operation	14-56
14.4.3	Flash Control Machine (FCM)	14-57
14.4.3.1	FCM Buffer RAM	14-58
14.4.3.1.1	Buffer Layout and Page Mapping for Small-Page NAND Flash Devices	14-59
14.4.3.1.2	Buffer Layout and Page Mapping for Large-Page NAND Flash Devices	14-60
14.4.3.1.3	Error Correcting Codes and the Spare Region	14-61
14.4.3.2	Programming FCM	14-63

Contents

Paragraph Number	Title	Page Number
14.4.3.2.1	FCM Command Instructions	14-64
14.4.3.2.2	FCM No-Operation Instruction	14-64
14.4.3.2.3	FCM Address Instructions	14-64
14.4.3.2.4	FCM Data Read Instructions	14-65
14.4.3.2.5	FCM Data Write Instructions	14-65
14.4.3.3	FCM Signal Timing	14-66
14.4.3.3.1	FCM Chip-Select Timing	14-66
14.4.3.3.2	FCM Command, Address, and Write Data Timing	14-66
14.4.3.3.3	FCM Ready/Busy Timing	14-68
14.4.3.3.4	FCM Read Data Timing	14-68
14.4.3.3.5	FCM Extended Read Hold Timing	14-69
14.4.3.4	FCM Boot Chip-Select Operation	14-70
14.4.3.4.1	FCM Bank 0 Reset Initialization	14-70
14.4.3.4.2	Boot Block Loading into the FCM Buffer RAM	14-71
14.4.4	User-Programmable Machines (UPMs)	14-72
14.4.4.1	UPM Requests	14-73
14.4.4.1.1	Memory Access Requests	14-74
14.4.4.1.2	UPM Refresh Timer Requests	14-75
14.4.4.1.3	Software Requests—RUN Command	14-75
14.4.4.1.4	Exception Requests	14-75
14.4.4.2	Programming the UPMs	14-76
14.4.4.2.1	UPM Programming Example (Two Sequential Writes to the RAM Array)	14-76
14.4.4.2.2	UPM Programming Example (Two Sequential Reads from the RAM Array)	14-77
14.4.4.3	UPM Signal Timing	14-77
14.4.4.4	RAM Array	14-78
14.4.4.4.1	RAM Words	14-78
14.4.4.4.2	Chip-Select Signal Timing (CST*n*)	14-82
14.4.4.4.3	Byte Select Signal Timing (BST*n*)	14-82
14.4.4.4.4	General-Purpose Signals (G*n*T*n*, GO*n*)	14-83
14.4.4.4.5	Loop Control (LOOP)	14-83
14.4.4.4.6	Repeat Execution of Current RAM Word (REDO)	14-83
14.4.4.4.7	Address Multiplexing (AMX)	14-84
14.4.4.4.8	Data Valid and Data Sample Control (UTA)	14-85
14.4.4.4.9	LGPL[0:5] Signal Negation (LAST)	14-86
14.4.4.4.10	Wait Mechanism (WAEN)	14-86
14.4.4.5	Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge	14-87
14.4.4.6	Extended Hold Time on Read Accesses	14-87
14.5	Initialization/Application Information	14-88
14.5.1	Interfacing to Peripherals in Different Address Modes	14-88
14.5.1.1	Multiplexed Address/Data Bus for 32-Bit Addressing	14-88
14.5.1.2	Peripheral Hierarchy on the Local Bus for High Bus Speeds	14-89

Contents

Paragraph Number	Title	Page Number
14.5.1.3	GPCM Timings	14-89
14.5.2	Bus Turnaround	14-90
14.5.2.1	Address Phase after Previous Read	14-90
14.5.2.2	Read Data Phase after Address Phase	14-90
14.5.2.3	Read-Modify-Write Cycle for Parity Protected Memory Banks	14-91
14.5.2.4	UPM Cycles with Additional Address Phases	14-91
14.5.3	Interface to Different Port-Size Devices	14-91
14.5.4	Command Sequence Examples for NAND Flash EEPROM	14-93
14.5.4.1	NAND Flash Soft Reset Command Sequence Example	14-93
14.5.4.2	NAND Flash Read Status Command Sequence Example	14-94
14.5.4.3	NAND Flash Read Identification Command Sequence Example	14-94
14.5.4.4	NAND Flash Page Read Command Sequence Example	14-95
14.5.4.5	NAND Flash Block Erase Command Sequence Example	14-95
14.5.4.6	NAND Flash Program Command Sequence Example	14-96
14.5.5	Interfacing to Fast-Page Mode DRAM Using UPM	14-97
14.5.6	Interfacing to ZBT SRAM Using UPM	14-102

Chapter 15
Enhanced Three-Speed Ethernet Controllers

15.1	Overview	15-1
15.2	Features	15-2
15.3	Modes of Operation	15-5
15.4	External Signals Description	15-6
15.4.1	Detailed Signal Descriptions	15-9
15.5	Memory Map/Register Definition	15-15
15.5.1	Top-Level Module Memory Map	15-15
15.5.2	Detailed Memory Map	15-16
15.5.3	Memory-Mapped Register Descriptions	15-26
15.5.3.1	eTSEC General Control and Status Registers	15-26
15.5.3.1.1	Controller ID Register (TSEC_ID)	15-26
15.5.3.1.2	Controller ID Register (TSEC_ID2)	15-27
15.5.3.1.3	Interrupt Event Register (IEVENT)	15-28
15.5.3.1.4	Interrupt Mask Register (IMASK)	15-32
15.5.3.1.5	Error Disabled Register (EDIS)	15-34
15.5.3.1.6	Ethernet Control Register (ECNTRL)	15-35
15.5.3.1.7	Pause Time Value Register (PTV)	15-38
15.5.3.1.8	DMA Control Register (DMACTRL)	15-39
15.5.3.1.9	TBI Physical Address Register (TBIPA)	15-40
15.5.3.2	eTSEC Transmit Control and Status Registers	15-41
15.5.3.2.1	Transmit Control Register (TCTRL)	15-41

Contents

Paragraph Number	Title	Page Number
15.5.3.2.2	Transmit Status Register (TSTAT)	15-42
15.5.3.2.3	Default VLAN Control Word Register (DFVLAN)	15-46
15.5.3.2.4	Transmit Interrupt Coalescing Register (TXIC)	15-47
15.5.3.2.5	Transmit Queue Control Register (TQUEUE)	15-48
15.5.3.2.6	TxBD Ring 0–3 Weighting Register (TR03WT)	15-49
15.5.3.2.7	TxBD Ring 4–7 Weighting Register (TR47WT)	15-49
15.5.3.2.8	Transmit Data Buffer Pointer High Register (TBDBPH)	15-50
15.5.3.2.9	Transmit Buffer Descriptor Pointers 0–7 (TBPTR0–TBPTR7)	15-50
15.5.3.2.10	Transmit Descriptor Base Address High Register (TBASEH)	15-51
15.5.3.2.11	Transmit Descriptor Base Address Registers (TBASE0–TBASE7)	15-52
15.5.3.2.12	Transmit Time Stamp Identification Register (TMR_TXTS1–2_ID)	15-52
15.5.3.2.13	Transmit Time Stamp Register (TMR_TXTS1–2_H/L)	15-53
15.5.3.3	eTSEC Receive Control and Status Registers	15-53
15.5.3.3.1	Receive Control Register (RCTRL)	15-54
15.5.3.3.2	Receive Status Register (RSTAT)	15-56
15.5.3.3.3	Receive Interrupt Coalescing Register (RXIC)	15-59
15.5.3.3.4	Receive Queue Control Register (RQUEUE)	15-60
15.5.3.3.5	Receive Bit Field Extract Control Register (RBIFX)	15-61
15.5.3.3.6	Receive Queue Filer Table Address Register (RQFAR)	15-63
15.5.3.3.7	Receive Queue Filer Table Control Register (RQFCR)	15-64
15.5.3.3.8	Receive Queue Filer Table Property Register (RQFPR)	15-65
15.5.3.3.9	Maximum Receive Buffer Length Register (MRBLR)	15-68
15.5.3.3.10	Receive Data Buffer Pointer High Register (RBDBPH)	15-69
15.5.3.3.11	Receive Buffer Descriptor Pointers 0–7 (RBPTR0–RBPTR7)	15-69
15.5.3.3.12	Receive Descriptor Base Address High Register (RBASEH)	15-70
15.5.3.3.13	Receive Descriptor Base Address Registers (RBASE0–RBASE7)	15-71
15.5.3.3.14	Receive Stamp Register (TMR_RXTS_H/L)	15-71
15.5.3.4	MAC Functionality	15-72
15.5.3.4.1	Configuring the MAC	15-72
15.5.3.4.2	Controlling CSMA/CD	15-72
15.5.3.4.3	Handling Packet Collisions	15-72
15.5.3.4.4	Controlling Packet Flow	15-73
15.5.3.4.5	Controlling PHY Links	15-74
15.5.3.5	MAC Registers	15-75
15.5.3.5.1	MAC Configuration 1 Register (MACCFG1)	15-75
15.5.3.5.2	MAC Configuration 2 Register (MACCFG2)	15-76
15.5.3.5.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)	15-78
15.5.3.5.4	Half-Duplex Register (HAFDUP)	15-79
15.5.3.5.5	Maximum Frame Length Register (MAXFRM)	15-80
15.5.3.5.6	MII Management Configuration Register (MIIMCFG)	15-80
15.5.3.5.7	MII Management Command Register (MIIMCOM)	15-82

Contents

Paragraph Number	Title	Page Number
15.5.3.5.8	MII Management Address Register (MIIMADD)	15-82
15.5.3.5.9	MII Management Control Register (MIIMCON)	15-83
15.5.3.5.10	MII Management Status Register (MIIMSTAT)	15-83
15.5.3.5.11	MII Management Indicator Register (MIIMIND)	15-84
15.5.3.5.12	Interface Status Register (IFSTAT)	15-84
15.5.3.5.13	MAC Station Address Part 1 Register (MACSTNADDR1)	15-85
15.5.3.5.14	MAC Station Address Part 2 Register (MACSTNADDR2)	15-86
15.5.3.5.15	MAC Exact Match Address 1–15 Part 1 Registers (MAC01ADDR1–MAC15ADDR1)	15-86
15.5.3.5.16	MAC Exact Match Address 1–15 Part 2 Registers (MAC01ADDR2–MAC15ADDR2)	15-87
15.5.3.6	MIB Registers	15-87
15.5.3.6.1	Transmit and Receive 64-Byte Frame Counter (TR64)	15-88
15.5.3.6.2	Transmit and Receive 65- to 127-Byte Frame Counter (TR127)	15-89
15.5.3.6.3	Transmit and Receive 128- to 255-Byte Frame Counter (TR255)	15-89
15.5.3.6.4	Transmit and Receive 256- to 511-Byte Frame Counter (TR511)	15-90
15.5.3.6.5	Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)	15-90
15.5.3.6.6	Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)	15-91
15.5.3.6.7	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)	15-91
15.5.3.6.8	Receive Byte Counter (RBYT)	15-92
15.5.3.6.9	Receive Packet Counter (RPKT)	15-92
15.5.3.6.10	Receive FCS Error Counter (RFCS)	15-93
15.5.3.6.11	Receive Multicast Packet Counter (RMCA)	15-93
15.5.3.6.12	Receive Broadcast Packet Counter (RBCA)	15-94
15.5.3.6.13	Receive Control Frame Packet Counter (RXCF)	15-94
15.5.3.6.14	Receive Pause Frame Packet Counter (RXPF)	15-95
15.5.3.6.15	Receive Unknown Opcode Packet Counter (RXUO)	15-95
15.5.3.6.16	Receive Alignment Error Counter (RALN)	15-96
15.5.3.6.17	Receive Frame Length Error Counter (RFLR)	15-96
15.5.3.6.18	Receive Code Error Counter (RCDE)	15-97
15.5.3.6.19	Receive Carrier Sense Error Counter (RCSE)	15-97
15.5.3.6.20	Receive Undersize Packet Counter (RUND)	15-98
15.5.3.6.21	Receive Oversize Packet Counter (ROVR)	15-98
15.5.3.6.22	Receive Fragments Counter (RFRG)	15-99
15.5.3.6.23	Receive Jabber Counter (RJBR)	15-99
15.5.3.6.24	Receive Dropped Packet Counter (RDRP)	15-100
15.5.3.6.25	Transmit Byte Counter (TBYT)	15-100
15.5.3.6.26	Transmit Packet Counter (TPKT)	15-101
15.5.3.6.27	Transmit Multicast Packet Counter (TMCA)	15-101
15.5.3.6.28	Transmit Broadcast Packet Counter (TBCA)	15-102

Contents

Paragraph Number	Title	Page Number
15.5.3.6.29	Transmit Pause Control Frame Counter (TXPF)	15-102
15.5.3.6.30	Transmit Deferral Packet Counter (TDFR)	15-103
15.5.3.6.31	Transmit Excessive Deferral Packet Counter (TEDF)	15-103
15.5.3.6.32	Transmit Single Collision Packet Counter (TSCL)	15-104
15.5.3.6.33	Transmit Multiple Collision Packet Counter (TMCL)	15-104
15.5.3.6.34	Transmit Late Collision Packet Counter (TLCL)	15-105
15.5.3.6.35	Transmit Excessive Collision Packet Counter (TXCL)	15-105
15.5.3.6.36	Transmit Total Collision Counter (TNCL)	15-106
15.5.3.6.37	Transmit Drop Frame Counter (TDRP)	15-106
15.5.3.6.38	Transmit Jabber Frame Counter (TJBR)	15-107
15.5.3.6.39	Transmit FCS Error Counter (TFCS)	15-107
15.5.3.6.40	Transmit Control Frame Counter (TXCF)	15-108
15.5.3.6.41	Transmit Oversize Frame Counter (TOVR)	15-108
15.5.3.6.42	Transmit Undersize Frame Counter (TUND)	15-109
15.5.3.6.43	Transmit Fragment Counter (TFRG)	15-109
15.5.3.6.44	Carry Register 1 (CAR1)	15-110
15.5.3.6.45	Carry Register 2 (CAR2)	15-111
15.5.3.6.46	Carry Mask Register 1 (CAM1)	15-112
15.5.3.6.47	Carry Mask Register 2 (CAM2)	15-114
15.5.3.6.48	Receive Filer Rejected Packet Counter (RREJ)	15-115
15.5.3.7	Hash Function Registers	15-115
15.5.3.7.1	Individual/Group Address Registers 0–7 (IGADDR*n*)	15-116
15.5.3.7.2	Group Address Registers 0–7 (GADDR*n*)	15-116
15.5.3.8	FIFO Registers	15-117
15.5.3.8.1	FIFO Configuration Register (FIFOCFG)	15-117
15.5.3.9	DMA Attribute Registers	15-119
15.5.3.9.1	Attribute Register (ATTR)	15-119
15.5.3.9.2	Attribute Extract Length and Extract Index Register (ATTRELI)	15-120
15.5.3.10	Lossless Flow Control Configuration Registers	15-120
15.5.3.10.1	Receive Queue Parameters 0–7 (RQPRM0–PQPRM7)	15-121
15.5.3.10.2	Receive Free Buffer Descriptor Pointer Registers 0–7 (RFBPTR0–RFBPTR7)	15-121
15.5.3.11	Hardware Assist for IEEE 1588 Compatible Timestamping	15-122
15.5.3.11.1	Timer Control Register (TMR_CTRL)	15-122
15.5.3.11.2	Timer Event Register (TMR_TEVENT)	15-124
15.5.3.11.3	Timer Event Mask Register (TMR_TEMASK)	15-125
15.5.3.11.4	Timer PTP Packet Event Register (TMR_PEVENT)	15-126
15.5.3.11.5	Timer Event Mask Register (TMR_PEMASK)	15-127
15.5.3.11.6	Timer Status Register (TMR_STAT)	15-127
15.5.3.11.7	Timer Counter Register (TMR_CNT_H/L)	15-128
15.5.3.11.8	Timer Drift Compensation Addend Register (TMR_ADD)	15-129

Contents

Paragraph Number	Title	Page Number
15.5.3.11.9	Timer Accumulator Register (TMR_ACC)	15-130
15.5.3.11.10	Timer Prescale Register (TMR_PRSC)	15-130
15.5.3.11.11	Timer Offset Register (TMROFF_H/L)	15-131
15.5.3.11.12	Alarm Time Comparator Register (TMR_ALARM1–2_H/L)	15-131
15.5.3.11.13	Timer Fixed Interval Period Register (TMR_FIPER1–3)	15-132
15.5.3.11.14	External Trigger Stamp Register (TMR_ETTS1–2_H/L)	15-133
15.5.4	Ten-Bit Interface (TBI)	15-134
15.5.4.1	TBI Transmit Process	15-134
15.5.4.1.1	Packet Encapsulation	15-134
15.5.4.1.2	8B10B Encoding	15-134
15.5.4.1.3	Preamble Shortening	15-134
15.5.4.2	TBI Receive Process	15-135
15.5.4.2.1	Synchronization	15-135
15.5.4.2.2	Auto-Negotiation for 1000BASE-X	15-135
15.5.4.3	TBI MII Set Register Descriptions	15-135
15.5.4.3.1	Control Register (CR)	15-136
15.5.4.3.2	Status Register (SR)	15-137
15.5.4.3.3	AN Advertisement Register (ANA)	15-138
15.5.4.3.4	AN Link Partner Base Page Ability Register (ANLPBPA)	15-140
15.5.4.3.5	AN Expansion Register (ANEX)	15-141
15.5.4.3.6	AN Next Page Transmit Register (ANNPT)	15-142
15.5.4.3.7	AN Link Partner Ability Next Page Register (ANLPANP)	15-143
15.5.4.3.8	Extended Status Register (EXST)	15-144
15.5.4.3.9	Jitter Diagnostics Register (JD)	15-144
15.5.4.3.10	TBI Control Register (TBICON)	15-145
15.6	Functional Description	15-146
15.6.1	Connecting to Physical Interfaces on Ethernet	15-146
15.6.1.1	Media-Independent Interface (MII)	15-147
15.6.1.2	Reduced Media-Independent Interface (RMII)	15-148
15.6.1.3	Gigabit Media-Independent Interface (GMII)	15-149
15.6.1.4	Reduced Gigabit Media-Independent Interface (RGMII)	15-149
15.6.1.5	Ten-Bit Interface (TBI)	15-150
15.6.1.6	Reduced Ten-Bit Interface (RTBI)	15-151
15.6.1.7	Ethernet Physical Interfaces Signal Summary	15-153
15.6.1.8	SGMII Interface	15-157
15.6.2	Connecting to FIFO Interfaces	15-157
15.6.2.1	Flow Control	15-159
15.6.2.2	CRC Appending and Checking	15-159
15.6.2.3	8-Bit GMII-Style Packet FIFO Mode	15-160
15.6.2.4	8-Bit Encoded Packet FIFO Mode	15-161
15.6.2.5	16-Bit GMII-Style Packet FIFO Mode	15-161

Contents

Paragraph Number	Title	Page Number
15.6.2.6	16-Bit Encoded Packet FIFO Mode	15-163
15.6.2.7	FIFO Interface Signal Summary	15-164
15.6.3	Gigabit Ethernet Controller Channel Operation	15-165
15.6.3.1	Initialization Sequence	15-165
15.6.3.1.1	Hardware Controlled Initialization	15-165
15.6.3.1.2	User Initialization	15-165
15.6.3.2	Soft Reset and Reconfiguring Procedure	15-166
15.6.3.3	Gigabit Ethernet Frame Transmission	15-167
15.6.3.4	Gigabit Ethernet Frame Reception	15-168
15.6.3.5	Ethernet Preamble Customization	15-169
15.6.3.5.1	User-Defined Preamble Transmission	15-170
15.6.3.5.2	User-Visible Preamble Reception	15-170
15.6.3.6	RMON Support	15-171
15.6.3.7	Frame Recognition	15-171
15.6.3.7.1	Destination Address Recognition and Frame Filtering	15-172
15.6.3.7.2	Hash Table Algorithm	15-173
15.6.3.8	Magic Packet Mode	15-175
15.6.3.9	Flow Control	15-175
15.6.3.10	Interrupt Handling	15-176
15.6.3.10.1	Interrupt Coalescing	15-177
15.6.3.10.2	Interrupt Coalescing By Frame Count Threshold	15-177
15.6.3.10.3	Interrupt Coalescing By Timer Threshold	15-178
15.6.3.11	Inter-Frame Gap Time	15-179
15.6.3.12	Internal and External Loop Back	15-179
15.6.3.13	Error-Handling Procedure	15-179
15.6.4	TCP/IP Off-Load	15-181
15.6.4.1	Frame Control Blocks	15-182
15.6.4.2	Transmit Path Off-Load and Tx PTP Packet Parsing	15-182
15.6.4.3	Receive Path Off-Load	15-184
15.6.5	Quality of Service (QoS) Provision	15-186
15.6.5.1	Receive Queue Filer	15-186
15.6.5.1.1	Filing Rules	15-187
15.6.5.1.2	Comparing Properties with Bit Masks	15-188
15.6.5.1.3	Special-Case Rules	15-189
15.6.5.1.4	Filer Interrupt Events	15-189
15.6.5.1.5	Setting Up the Receive Queue Filer Table	15-190
15.6.5.1.6	Filer Example—802.1p Priority Filing	15-190
15.6.5.1.7	Filer Example—IP Diff-Serv Code Points Filing	15-191
15.6.5.1.8	Filer Example—TCP and UDP Port Filing	15-191
15.6.5.2	Transmission Scheduling	15-192
15.6.5.2.1	Priority-Based Queuing (PBQ)	15-193

Contents

Paragraph Number	Title	Page Number
15.6.5.2.2	Modified Weighted Round-Robin Queuing (MWRR)	15-193
15.6.6	Lossless Flow Control	15-194
15.6.6.1	Back Pressure Determination via Free Buffers	15-194
15.6.6.2	Software Use of Hardware-Initiated Back Pressure	15-196
15.6.6.2.1	Initialization	15-196
15.6.6.2.2	Operation	15-197
15.6.7	Hardware Assist for IEEE 1588 Compatible Timestamping	15-197
15.6.7.1	Features	15-197
15.6.7.2	Timer Logic Overview	15-198
15.6.7.3	Time-Stamp Insertion on the Received Packets	15-198
15.6.7.3.1	Timestamp Point	15-198
15.6.7.4	PTP Packet Parsing	15-199
15.6.7.4.1	General Purpose Filer Rule	15-200
15.6.8	Buffer Descriptors	15-200
15.6.8.1	Data Buffer Descriptors	15-201
15.6.8.2	Transmit Data Buffer Descriptors (TxBD)	15-202
15.6.8.3	Receive Buffer Descriptors (RxBD)	15-205
15.7	Initialization/Application Information	15-207
15.7.1	Interface Mode Configuration	15-207
15.7.1.1	MII Interface Mode	15-208
15.7.1.2	GMII Interface Mode	15-212
15.7.1.3	TBI Interface Mode	15-216
15.7.1.4	RGMII Interface Mode	15-220
15.7.1.5	RMII Interface Mode	15-224
15.7.1.6	RTBI Interface Mode	15-228
15.7.1.7	8-Bit FIFO Mode	15-232
15.7.1.8	16-Bit FIFO Mode	15-235
15.7.1.9	SGMII Interface Support	15-238

Chapter 16
10/100 Fast Ethernet Controller

16.1	Introduction	16-1
16.1.1	Fast Ethernet Controller Overview	16-5
16.2	Features	16-5
16.3	Modes of Operation	16-6
16.4	External Signals Description	16-6
16.4.1	Detailed Signal Descriptions	16-7
16.5	Memory Map/Register Definition	16-8
16.5.1	Top-Level Module Memory Map	16-8
16.5.2	Detailed Memory Map—Control/Status Registers	16-9

Contents

Paragraph Number	Title	Page Number
16.5.3	Memory-Mapped Register Descriptions	16-12
16.5.3.1	FEC General Control and Status Registers	16-12
16.5.3.1.1	Interrupt Event Register (IEVENT)	16-12
16.5.3.1.2	Interrupt Mask Register (IMASK)	16-15
16.5.3.1.3	Error Disabled Register (EDIS)	16-17
16.5.3.1.4	Minimum Frame Length Register (MINFLR)	16-17
16.5.3.1.5	Pause Time Value Register (PTV)	16-18
16.5.3.1.6	DMA Control Register (DMACTRL)	16-19
16.5.3.2	FEC FIFO Control and Status Registers	16-20
16.5.3.2.1	FIFO Pause Control Register (FIFO_PAUSE_CTRL)	16-21
16.5.3.2.2	FIFO Transmit Threshold Register (FIFO_TX_THR)	16-21
16.5.3.2.3	FIFO Transmit Starve Register (FIFO_TX_STARVE)	16-22
16.5.3.2.4	FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)	16-22
16.5.3.3	FEC Transmit Control and Status Registers	16-23
16.5.3.3.1	Transmit Control Register (TCTRL)	16-23
16.5.3.3.2	Transmit Status Register (TSTAT)	16-24
16.5.3.3.3	TxBD Data Length Register (TBDLEN)	16-25
16.5.3.3.4	Current Transmit Buffer Descriptor Pointer Register (CTBPTR)	16-26
16.5.3.3.5	Transmit Buffer Descriptor Pointer Register (TBPTR)	16-26
16.5.3.3.6	Transmit Descriptor Base Address Register (TBASE)	16-27
16.5.3.3.7	Out-of-Sequence TxBD Register (OSTBD)	16-27
16.5.3.3.8	Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)	16-29
16.5.3.4	FEC Receive Control and Status Registers	16-30
16.5.3.4.1	Receive Control Register (RCTRL)	16-30
16.5.3.4.2	Receive Status Register (RSTAT)	16-30
16.5.3.4.3	RxBD Data Length Register (RBDLEN)	16-31
16.5.3.4.4	Current Receive Buffer Descriptor Pointer Register (CRBPTR)	16-32
16.5.3.4.5	Maximum Receive Buffer Length Register (MRBLR)	16-32
16.5.3.4.6	Receive Buffer Descriptor Pointer Register (RBPTR)	16-33
16.5.3.4.7	Receive Descriptor Base Address Register (RBASE)	16-33
16.5.3.5	MAC Functionality	16-33
16.5.3.5.1	Configuring the MAC	16-34
16.5.3.5.2	Controlling CSMA/CD	16-34
16.5.3.5.3	Handling Packet Collisions	16-34
16.5.3.5.4	Controlling Packet Flow	16-35
16.5.3.5.5	Controlling PHY Links	16-35
16.5.3.6	MAC Registers	16-36
16.5.3.6.1	MAC Configuration Register 1 (MACCFG1)	16-36
16.5.3.6.2	MAC Configuration Register 2 (MACCFG2)	16-38
16.5.3.6.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)	16-39
16.5.3.6.4	Half-Duplex Register (HAFDUP)	16-40

Contents

Paragraph Number	Title	Page Number
16.5.3.6.5	Maximum Frame Length Register (MAXFRM)	16-41
16.5.3.6.6	MII Management Configuration Register (MIIMCFG)	16-41
16.5.3.6.7	MII Management Command Register (MIIMCOM)	16-42
16.5.3.6.8	MII Management Address Register (MIIMADD)	16-43
16.5.3.6.9	MII Management Control Register (MIIMCON)	16-44
16.5.3.6.10	MII Management Status Register (MIIMSTAT)	16-44
16.5.3.6.11	MII Management Indicator Register (MIIMIND)	16-45
16.5.3.6.12	Interface Control (Interface Control)—User	16-45
16.5.3.6.13	Interface Status Register (IFSTAT)	16-47
16.5.3.6.14	Station Address Register Part 1 (MACSTNADDR1)	16-47
16.5.3.6.15	Station Address Register Part 2 (MACSTNADDR2)	16-48
16.5.3.7	Hash Function Registers	16-48
16.5.3.7.1	Individual Address Registers 0–7 (IADDR*n*)	16-49
16.5.3.7.2	Group Address Registers 0–7 (GADDR*n*)	16-49
16.5.3.8	Attribute Registers	16-50
16.5.3.8.1	Attribute Register (ATTR)	16-50
16.5.3.8.2	Attribute Extract Length and Extract Index Register (ATTRELI)	16-51
16.6	Functional Description	16-52
16.6.1	Connecting to Physical Interfaces	16-52
16.6.1.1	Media-Independent Interface (MII)	16-52
16.6.2	FEC Channel Operation	16-54
16.6.2.1	Initialization Sequence	16-54
16.6.2.1.1	Hardware Controlled Initialization	16-54
16.6.2.1.2	User Initialization	16-54
16.6.2.2	Soft Reset and Reconfiguring Procedure	16-55
16.6.2.3	FEC Frame Transmission	16-56
16.6.2.4	FEC Frame Reception	16-57
16.6.2.5	Frame Recognition	16-59
16.6.2.5.1	Destination Address Recognition	16-59
16.6.2.5.2	Hash Table Algorithm	16-61
16.6.2.5.3	CRC Computation Examples	16-61
16.6.2.6	Flow Control	16-62
16.6.2.7	Interrupt Handling	16-63
16.6.2.8	Inter-Packet Gap Time	16-64
16.6.2.9	Internal and External Loop Back	16-64
16.6.2.10	Error-Handling Procedure	16-65
16.6.3	Buffer Descriptors	16-66
16.6.3.1	Transmit Data Buffer Descriptor (TxBD)	16-67
16.6.3.2	Receive Buffer Descriptor (RxBD)	16-69
16.6.4	Data Extraction to the L2 Cache	16-71
16.7	Initialization/Application Information	16-72

Contents

Paragraph Number	Title	Page Number
16.7.1	Interface Mode Configuration	16-72
16.7.1.1	MII Interface Mode	16-72

Chapter 17
Pattern Matcher (PM) 1.1

17.1	Preface	17-1
17.1.1	Conventions	17-1
17.1.1.1	Glossary	17-2
17.2	Introduction	17-3
17.2.1	Overview	17-3
17.2.2	Features	17-4
17.3	Memory Map and Register Definition	17-6
17.3.1	DMA Engine Common Control Registers	17-14
17.3.1.1	Interrupt Status Register Common Control (ISRCC)	17-14
17.3.1.2	Interrupt Enable Register Common Control (IERCC)	17-16
17.3.1.3	Interrupt Status Disable Register Common Control (ISDRCC)	17-17
17.3.1.4	Interrupt Inhibit Register Common Control (IIRCC)	17-18
17.3.1.5	Statistics Counters Overflow Status (SCOS)	17-19
17.3.1.6	Performance Monitor Threshold (PERFT)	17-20
17.3.1.7	Channel Scheduling Control Register (CSCR)	17-21
17.3.2	Key Element Scanner Control Registers	17-22
17.3.2.1	Statistic—Number of Input Bytes Scanned for Pattern Matching (STNIB)	17-22
17.3.2.2	Statistic—Number of Input SUIs (STNIS)	17-23
17.3.2.3	Statistic—Number of Trigger Hits Against 1-Byte Triggers (STNTH1)	17-23
17.3.2.4	Statistic—Number of Trigger Hits Against 2-Byte Triggers (STNTH2)	17-24
17.3.2.5	Statistic—Number of Trigger Hits Against Variable Length Triggers (STNTHL)	17-24
17.3.2.6	Statistic—Number of Trigger Hits against special triggers (STNTHS)	17-25
17.3.2.7	Statistic—Number of Confidence Stage Hits (STNCH)	17-25
17.3.2.8	Software Database Version Register / Scratch Pad (SWDB)	17-26
17.3.2.9	KES Variable Length Trigger Size (KVLTS)	17-26
17.3.2.10	KES Error Configuration (KEC)	17-27
17.3.2.11	PME Error Handling Disables (PEHD)	17-27
17.3.3	Data Examination Engine Control Registers	17-29
17.3.3.1	Statistic—Number of Pattern Matches (STNPM)	17-29
17.3.3.2	Statistic—Number of SUIs with at Least 1 Pattern Match (STNS1M)	17-29
17.3.3.3	DXE Pattern Range Counter Configuration (DRCC)	17-30
17.3.3.4	Statistic—Number of Pattern Match Attempts Within a Range of Indexes (STNPMR)	17-31

Contents

Paragraph Number	Title	Page Number
17.3.3.5	Pattern Description and Stateful Rule Base Address High and Low Registers (PDSRBAH and PDSRBAL)	17-31
17.3.3.6	DXE Memory Control Register (DMCR)	17-32
17.3.3.7	DXE Error Configuration (DEC)	17-33
17.3.4	Stateful Rule Engine Control Registers	17-34
17.3.4.1	Statistic—Number of DXE Generated Stateful Rule Executions (STNDSR)	17-34
17.3.4.2	Statistic—Number of End of SUI Generated Stateful Rule Executions (STNESR)	17-34
17.3.4.3	Statistic—Number of SUIs with at Least 1 Report (STNS1R)	17-35
17.3.4.4	Statistic—Number of Output Bytes Produced in Reports (STNOB)	17-35
17.3.4.5	SRE Context Base Address High and Low Registers (SCBARH and SCBARL)	17-36
17.3.4.6	SRE Memory Control Register (SMCR)	17-37
17.3.4.7	SRE Configuration Register (SREC)	17-38
17.3.4.8	SRE Rule Reset Vector 0–7 (SRRV0–7)	17-39
17.3.4.9	SRE Rule Reset First Index (SRRFI)	17-39
17.3.4.10	SRE Rule Reset 32-Byte Increment (SRRI)	17-40
17.3.4.11	SRE Rule Reset Repetitions (SRRR)	17-40
17.3.4.12	SRE Rule Reset Work Configuration (SRRWC)	17-41
17.3.4.13	SRE Free Running Counter Configuration (SFRCC)	17-42
17.3.4.14	SRE Error Configuration 1 (SEC1)	17-43
17.3.4.15	SRE Error Configuration 2 (SEC2)	17-43
17.3.4.16	SRE Error Configuration 3 (SEC3)	17-44
17.3.5	Deflate Engine Registers	17-44
17.3.5.1	Statistic—Deflate Bytes Consumed (STDBC)	17-44
17.3.5.2	Statistic—Deflate Bytes Produced (STDBP)	17-45
17.3.5.3	Statistic—Deflate Work Units Consumed (STDWC)	17-45
17.3.5.4	Deflate Bytes Produced Per Work Unit Limit (DBPPWL)	17-46
17.3.6	Free Buffer Manager Common Control Registers	17-46
17.3.6.1	Free Buffer List Buffer Size Registers (FBL0SIZE to FBL7SIZE)	17-46
17.3.6.2	Free Buffer List Assignment A Register (FBLAAR)	17-47
17.3.6.3	Free Buffer List Assignment B Register (FBLABR)	17-48
17.3.6.4	Free Buffer Manager Control Register (FBM_CR)	17-50
17.3.7	Memory Interface Arbiter Registers	17-51
17.3.7.1	MIA Byte Count Register (MIA_BYC)	17-51
17.3.7.2	MIA Block Count Register (MIA_BLC)	17-51
17.3.7.3	MIA Count Enable Register (MIA_CE)	17-52
17.3.7.4	MIA Control Register (MIA_CR)	17-55
17.3.7.5	MIA Arbitration Control Register (MIA_ACR)	17-56
17.3.7.6	MIA Local Snoop Hit Count Registers (MIA_LSHC0/1/2)	17-58
17.3.7.7	MIA Cancelled Write Count Registers (MIA_CWC0/1/2)	17-58

Contents

Paragraph Number	Title	Page Number
17.3.7.8	PM IP Block Revision 1 Register (PM_IP_REV_1)	17-59
17.3.7.9	PM IP Block Revision 2 Register (PM_IP_REV_2)	17-59
17.3.8	DMA Engine Channel 0 Registers	17-60
17.3.8.1	Interrupt Status Register 0 (ISR0)	17-60
17.3.8.2	Interrupt Enable Register 0 (IER0)	17-62
17.3.8.3	Interrupt Status Disable Register 0 (ISDR0)	17-63
17.3.8.4	Interrupt Inhibit Register 0 (IIR0)	17-64
17.3.8.5	Interrupt Interval Control Register 0 (IICR0)	17-65
17.3.8.6	Notifications Produced Index Register Channel 0 (NPIR0)	17-65
17.3.8.7	Commands Consumed Index Register Channel 0 (CCIR0)	17-66
17.3.8.8	Free Buffer Deallocate FIFO Consumer Index Register Channel 0 (FBCIR0)	17-67
17.3.8.9	Channel Error Status Channel 0 (CHERR0)	17-68
17.3.8.10	Notifications Consumed Index Register Channel 0 (NCIR0)	17-68
17.3.8.11	Commands Produced Index Register Channel 0 (CPIR0)	17-69
17.3.8.12	Commands Expired Index Register Channel 0 (CEIR0)	17-70
17.3.8.13	Free Buffer Deallocate FIFO Producer Index Register Channel 0 (FBPIR0)	17-70
17.3.8.14	Output Truncated Incidence Counter Channel 0 (TRUNCIC0)	17-71
17.3.8.15	Read Byte Counter Channel 0 (RBC0)	17-72
17.3.8.16	Statistics Counters Overflow Status Register Channel 0 (SCOS0)	17-72
17.3.8.17	Channel Reset and Control Register Channel 0 (CRCR0)	17-73
17.3.8.18	Command FIFO Base Address High and Low Registers Channel 0 (CFIFO_BH0 and CFIFO_BL0)	17-74
17.3.8.19	Command FIFO Depth Register Channel 0 (CFIFO_DEPTH0)	17-75
17.3.8.20	Command FIFO Threshold Register Channel 0 (CFIFO_THRES0)	17-76
17.3.8.21	Notification FIFO Base Address High and Low Registers Channel 0 (NFIFO_BH0 and NFIFO_BL0)	17-76
17.3.8.22	Notification FIFO Depth Register Channel 0 (NFIFO_DEPTH0)	17-78
17.3.8.23	Notification FIFO Threshold Register Channel 0 (NFIFO_THRES0)	17-78
17.3.8.24	Notification Deflate Length Limit Register Channel 0 (NDLL0)	17-79
17.3.8.25	Notification Result Length Limit Register Channel 0 (NRLL0)	17-79
17.3.8.26	Free Buffer Deallocate FIFO Base Address High and Low Registers Channel 0 (FBFIFO_BH0 and FBFIFO_BL0)	17-80
17.3.8.27	Free Buffer Deallocate FIFO Depth Register Channel 0 (FBFIFO_DEPTH0)	17-81
17.3.8.28	Free Buffer Deallocate FIFO Threshold Register Channel 0 (FBFIFO_THRES0)	17-82
17.3.8.29	Stream Context Configuration Register Channel 0 (SC_CONFIG0)	17-82
17.3.8.30	Stream Context Base Address High and Low Registers Channel 0 (SC_BH0 and SC_BL0)	17-83
17.3.8.31	Residue Configuration Register Channel 0 (RES_CONFIG0)	17-84
17.3.8.32	Residue Base Address High and Low Registers Channel 0 (RES_BH0 and RES_BL0)	17-85

Contents

Paragraph Number	Title	Page Number
17.3.8.33	Cache Awareness Control Register Channel 0 (CACR0)	17-86
17.3.8.34	Memory Transaction Priority Control Register Channel 0 (MTPCR0)	17-88
17.3.9	Free Buffer Manager Channel 0 Registers	17-89
17.3.9.1	Free Buffer List Threshold A Register Channel 0 (FBL_THRES_A0)	17-89
17.3.9.2	Free Buffer List High and Low Watermark A Register Channel 0 (FBL_HLWM_A0)	17-90
17.3.9.3	Free Buffer List Allocation and Deallocation Counter Register A Channel 0 (FBL_AD_CA0)	17-91
17.3.9.4	Free Buffer List On Deck Disable A Register Channel 0 (FBL_ODD_A0)	17-91
17.3.9.5	Free Buffer List Physical Head Address High and Low A Registers Channel 0 (FBL_PH_AH0 and FBL_PH_AL0)	17-92
17.3.9.6	Free Buffer List Virtual Head Address High and Low A Registers Channel 0 (FBL_VH_AH0 and FBL_VH_AL0)	17-93
17.3.9.7	Free Buffer List Physical Tail Address High and Low A Registers Channel 0 (FBL_PT_AH0 and FBL_PT_AL0)	17-94
17.3.9.8	Free Buffer List Number of Buffers A Register Channel 0 (FBL_NB_A0)	17-95
17.3.9.9	Free Buffer List Buffer Size A Register Channel 0 (FBL_BS_A0)	17-96
17.3.9.10	Free Buffer List Virtual On Deck Pointer A High and Low Registers Channel 0 (FBL_PD_AH0 and FBL_PD_AL0)	17-96
17.3.9.11	Free Buffer List Threshold B Register Channel 0 (FBL_THRES_B0)	17-97
17.3.9.12	Free Buffer List High and Low Watermark B Register Channel 0 (FBL_HLWM_B0)	17-97
17.3.9.13	Free Buffer List Allocation and Deallocation Counter Register B Channel 0 (FBL_AD_CB0)	17-98
17.3.9.14	Free Buffer List On Deck Disable B Register Channel 0 (FBL_ODD_B0)	17-98
17.3.9.15	Free Buffer List Physical Head Address High and Low B Registers Channel 0 (FBL_PH_BH0 and FBL_PH_BL0)	17-98
17.3.9.16	Free Buffer List Virtual Head Address High and Low B Registers Channel 0 (FBL_VH_BH0 and FBL_VH_BL0)	17-98
17.3.9.17	Free Buffer List Physical Tail Address High and Low B Registers Channel 0 (FBL_PT_BH0 and FBL_PT_BL0)	17-98
17.3.9.18	Free Buffer List Number of Buffers B Register Channel 0 (FBL_NB_B0)	17-98
17.3.9.19	Free Buffer List Buffer Size B Register Channel 0 (FBL_BS_B0)	17-99
17.3.9.20	Free Buffer List Virtual On Deck Pointer B High and Low Registers Channel 0 (FBL_PD_BH0 and FBL_PD_BL0)	17-99
17.4	Functional Description	17-99
17.4.1	Key Element Scanner	17-101
17.4.1.1	Trigger Stage	17-104
17.4.1.1.1	Variable Length Trigger	17-105
17.4.1.1.2	2-Byte Trigger	17-108
17.4.1.1.3	1-Byte Trigger	17-109

Contents

Paragraph Number	Title	Page Number
17.4.1.1.4	Special Triggers	17-110
17.4.1.2	Confidence Stage	17-111
17.4.1.2.1	Variable Length Trigger to Confidence Mapping	17-114
17.4.1.2.2	2-Byte Trigger to Confidence Mapping	17-115
17.4.1.2.3	1-Byte Trigger to Confidence Mapping	17-116
17.4.1.2.4	Special Trigger to Confidence Mapping	17-116
17.4.1.2.5	Confidence Collision Resolution	17-116
17.4.1.2.6	Confidence Mask Codes	17-118
17.4.2	Data Examination Engine	17-119
17.4.2.1	Pattern Description	17-120
17.4.2.1.1	Anchor Instruction	17-123
17.4.2.1.2	Element Compare Instruction	17-123
17.4.2.1.3	Repetition	17-124
17.4.2.1.4	Capture	17-125
17.4.2.1.5	Inconclusive Match	17-125
17.4.2.1.6	Test Line Block Format	17-127
17.4.2.1.7	Pattern Description Memory Space	17-135
17.4.2.1.8	Test Line Examples	17-136
17.4.3	Stateful Rule Engine	17-137
17.4.3.1	Stateful Rule Physical Structure	17-138
17.4.3.2	SRE Instruction Set	17-140
17.4.3.2.1	Instruction Set Operands	17-140
17.4.3.2.2	Instruction Set Description	17-142
17.4.3.3	SRE Context Table	17-150
17.4.3.4	SRE Report Format	17-152
17.4.3.4.1	Simple Match Report	17-153
17.4.3.4.2	Simple End of SUI Report	17-153
17.4.3.4.3	Reaction Reports	17-153
17.4.3.4.4	Verbose Reports	17-153
17.4.4	Deflate Engine	17-156
17.4.4.1	Huffman Decoding Engine	17-156
17.4.4.2	Data Expansion Engine	17-157
17.4.5	DMA Engine	17-157
17.4.5.1	Command and Notification FIFO Operation	17-159
17.4.5.2	Command FIFO Entries	17-161
17.4.5.2.1	Command FIFO Command Codes	17-161
17.4.5.2.2	Command FIFO Stream ID	17-163
17.4.5.2.3	Command FIFO Input Data Descriptor	17-164
17.4.5.2.4	Command FIFO Result and Deflate Data Descriptors	17-168
17.4.5.2.5	Command FIFO Modifier Bits	17-168
17.4.5.2.6	Command FIFO Pass Through Context	17-168

Contents

Paragraph Number	Title	Page Number
17.4.5.3	Notification FIFO Entries	17-168
17.4.5.3.1	Notification FIFO Stream ID	17-169
17.4.5.3.2	Notification FIFO Output Data Descriptor	17-169
17.4.5.3.3	Notification FIFO Exception Code and Truncation Indication	17-170
17.4.5.3.4	Notification FIFO Pass Through Context	17-170
17.4.5.4	Interrupts	17-170
17.4.5.5	Free Buffer Deallocate FIFO	17-173
17.4.5.6	Error Handling	17-174
17.4.5.6.1	No-Halt Errors	17-174
17.4.5.6.2	Channel Errors	17-174
17.4.5.6.3	Common-Control Errors	17-175
17.4.5.7	Channel Scheduling	17-176
17.4.5.8	Cache Awareness	17-177
17.4.5.9	System Memory Transaction Priority	17-178
17.4.5.10	DMA Engine Data Structures	17-178
17.4.6	Free Buffer Manager	17-203
17.4.6.1	Free Buffer Linked List Structural Specification	17-204
17.4.6.2	Free Buffer List Depletion	17-205
17.4.6.3	FBM Cache Awareness and Memory Transaction Priority	17-206
17.4.7	Memory Interface Arbiter	17-206
17.4.7.1	MIA Read and Write Ports	17-206
17.4.7.2	MIA Read/Write Port Arbitration	17-207
17.4.7.2.1	Serialized Mode	17-207
17.4.7.2.2	Read Priority Weighted Mode	17-208
17.4.7.3	MIA Local Snoop and Write Cancellation	17-208
17.4.8	Internal Register Interface (IRI)	17-209
17.4.9	Performance Monitor Event Interface	17-209
17.4.9.1	System Memory Access Performance Monitoring	17-210
17.4.9.2	Internal Sequencer Performance Monitoring	17-211
17.5	Initialization Information	17-212
17.5.1	Initialization Sequence	17-212
17.5.1.1	Common-Control Register Space	17-212
17.5.1.2	Channel Register Space	17-212
17.5.2	Interrupt Initialization	17-213
17.5.2.1	Interrupt Registers	17-213
17.5.2.2	Channel FIFO Interrupts	17-214
17.5.2.2.1	Command FIFO	17-214
17.5.2.2.2	Notification FIFO	17-215
17.5.2.2.3	Free Buffer Deallocate FIFO	17-215
17.5.2.3	Free Buffer List Depletion	17-215
17.5.2.4	Channel Error Interrupts	17-215

Contents

Paragraph Number	Title	Page Number
17.5.2.5	Common-Control Error Interrupts	17-216
17.5.3	Bring Down Procedure	17-216
17.5.3.1	Syncing	17-216
17.5.3.1.1	Teardown	17-217
17.5.3.2	Reset	17-217
17.6	Application Information	17-217

Chapter 18
Table Lookup Unit

18.1	Introduction	18-1
18.2	Features	18-2
18.3	Modes of Operation	18-3
18.4	Memory Map/Register Definition	18-4
18.4.1	Top-Level Module Memory Map	18-4
18.4.2	Detailed Memory Map—Control/Status Registers	18-5
18.4.3	TLU Register Descriptions	18-8
18.4.3.1	TLU General Control/Status Registers	18-8
18.4.3.1.1	TLU Identifier1 Register (TLU_ID1)	18-8
18.4.3.1.2	TLU Identifier2 Register (TLU_ID2)	18-9
18.4.3.1.3	Interrupt Event Register (IEVENT)	18-9
18.4.3.1.4	Interrupt Mask Register (IMASK)	18-10
18.4.3.1.5	Interrupt Error Attributes Register (IEATR)	18-11
18.4.3.1.6	Interrupt Error Address Register (IEADD)	18-12
18.4.3.1.7	Interrupt Error Disable Register (IEDIS)	18-12
18.4.3.1.8	Memory Bank 0–3 Base Registers (MBANK0–MBANK3)	18-13
18.4.3.2	TLU Physical Table Configuration Registers	18-14
18.4.3.2.1	Physical Table 0–31 Configuration Registers (PTBL0–PTBL31)	18-14
18.4.3.3	TLU Statistics Counters	18-16
18.4.3.3.1	Memory Reads Count Register (CRAMR)	18-16
18.4.3.3.2	Memory Writes Count Register (CRAMW)	18-16
18.4.3.3.3	Total Find Count Register (CFIND)	18-17
18.4.3.3.4	Hash-Trie-Key Table Find Count Register (CTHTK)	18-17
18.4.3.3.5	CRT Table Find Count Register (CTCRT)	18-18
18.4.3.3.6	Chained Hash Table Find Count Register (CTCHS)	18-18
18.4.3.3.7	Flat Data Table Lookup Count Register (CTDAT)	18-19
18.4.3.3.8	Successful Find Count register (CHITS)	18-19
18.4.3.3.9	Failed Find Count Register (CMISS)	18-20
18.4.3.3.10	Hash Collision Count Register (CHCOL)	18-20
18.4.3.3.11	CRT Level Count Register (CCRTL)	18-21
18.4.3.3.12	Counter Carries-Out Register (CARO)	18-21

Contents

Paragraph Number	Title	Page Number
18.4.3.3.13	Counter Carry Mask Register (CARM)	18-22
18.4.3.4	TLU Command/Response Registers	18-23
18.4.3.4.1	TLU Command Operation Register (CMDOP)	18-23
18.4.3.4.2	TLU Command Index Register (CMDIX)	18-25
18.4.3.4.3	TLU Command Status and Response Register (CSTAT)	18-25
18.4.3.5	TLU Key/Data Registers	18-26
18.4.3.5.1	Key/Data Words 0–7 Buffer Registers (KD0B–KD7B)	18-26
18.5	Functional Description	18-27
18.5.1	Background	18-27
18.5.2	TLU Command Set	18-28
18.5.2.1	Write—Write Data to Table Index Command	18-29
18.5.2.1.1	Write Command Format	18-30
18.5.2.1.2	Write Command Response	18-31
18.5.2.2	Add—Add Data to Table Index Command	18-31
18.5.2.2.1	Add Command Format	18-31
18.5.2.2.2	Add Command Response	18-32
18.5.2.3	Read—Read Data from Table Index Command	18-32
18.5.2.3.1	Read Command Format	18-32
18.5.2.3.2	Read Command Response	18-33
18.5.2.4	Acchash—Accumulate to Hash Command	18-33
18.5.2.4.1	Acchash Command Format	18-34
18.5.2.4.2	Acchash Command Response	18-34
18.5.2.5	Find—Find Key Command	18-35
18.5.2.5.1	Find Command Format	18-35
18.5.2.5.2	Find Command Response	18-35
18.5.2.6	Findr—Find Key and Read Table Command	18-36
18.5.2.6.1	Findr Command Format	18-36
18.5.2.6.2	Findr Command Response	18-37
18.5.2.7	Findw—Find Key and Write Table Command	18-37
18.5.2.7.1	Findw Command Format	18-38
18.5.2.7.2	Findw Command Response	18-38
18.5.2.8	Error Handling	18-39
18.5.3	TLU Tables and Operation	18-39
18.5.3.1	Data Simple Table	18-41
18.5.3.2	Hash Simple Table	18-41
18.5.3.3	Trie Simple Table	18-42
18.5.3.4	Key Simple Table	18-44
18.5.3.5	Index-VPT-Data Simple Table	18-45
18.5.3.5.1	FAIL ETYPE	18-47
18.5.3.5.2	DATA ETYPE	18-47
18.5.3.5.3	SIMPLE ETYPE	18-47

Contents

Paragraph Number	Title	Page Number
18.5.3.5.4	HASH ETYPE	18-48
18.5.3.5.5	RUNDELTA ETYPE	18-48
18.5.4	Exact Match Algorithms	18-51
18.5.4.1	Flat Data	18-51
18.5.4.2	Hash-Trie-Key	18-52
18.5.4.3	Chained-Hash	18-54
18.5.5	Longest-Prefix Match Algorithms	18-58
18.5.5.1	Compressed Radix Trie (CRT)	18-58
18.5.6	TLU Hash Function	18-61
18.5.7	Initializing and Maintaining Tables	18-62
18.5.7.1	Configuration of Local Bus	18-62

Chapter 19
DMA Controllers

19.1	Introduction	19-1
19.1.1	Block Diagram	19-1
19.1.2	Overview	19-2
19.1.3	Features	19-2
19.1.4	Modes of Operation	19-2
19.2	External Signal Description	19-5
19.2.1	Signal Overview	19-5
19.2.2	Detailed Signal Descriptions	19-5
19.3	Memory Map/Register Definition	19-6
19.3.1	DMA Register Descriptions	19-9
19.3.1.1	Mode Registers (MR*n*)	19-9
19.3.1.2	Status Registers (SR*n*)	19-12
19.3.1.3	Current Link Descriptor Address Registers (CLNDAR*n* and ECLNDAR*n*)	19-13
19.3.1.4	Source Attributes Registers (SATR*n*)	19-15
19.3.1.5	Source Address Registers (SAR*n*)	19-16
19.3.1.5.1	Source Address Registers for RapidIO Maintenance Reads (SAR*n*)	19-17
19.3.1.6	Destination Attributes Registers (DATR*n*)	19-17
19.3.1.7	Destination Address Registers (DAR*n*)	19-19
19.3.1.7.1	Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)	19-19
19.3.1.8	Byte Count Registers (BCR*n*)	19-20
19.3.1.9	Next Link Descriptor Address Registers (NLNDAR*n* and ENLNDAR*n*)	19-21
19.3.1.10	Current List Descriptor Address Registers (CLSDAR*n* and ECLSDAR*n*)	19-22
19.3.1.11	Next List Descriptor Address Registers (NLSDAR*n* and ENLSDAR*n*)	19-23

Contents

Paragraph Number	Title	Page Number
19.3.1.12	Source Stride Registers (SSR*n*)	19-24
19.3.1.13	Destination Stride Registers (DSR*n*)	19-24
19.3.1.14	DMA General Status Register (DGSR)	19-25
19.4	Functional Description	19-27
19.4.1	DMA Channel Operation	19-27
19.4.1.1	Basic DMA Mode Transfer	19-27
19.4.1.1.1	Basic Direct Mode	19-28
19.4.1.1.2	Basic Direct Single-Write Start Mode	19-28
19.4.1.1.3	Basic Chaining Mode	19-29
19.4.1.1.4	Basic Chaining Single-Write Start Mode	19-29
19.4.1.2	Extended DMA Mode Transfer	19-30
19.4.1.2.1	Extended Direct Mode	19-30
19.4.1.2.2	Extended Direct Single-Write Start Mode	19-30
19.4.1.2.3	Extended Chaining Mode	19-30
19.4.1.2.4	Extended Chaining Single-Write Start Mode	19-31
19.4.1.3	External Control Mode Transfer	19-31
19.4.1.4	Channel Continue Mode for Cascading Transfer Chains	19-32
19.4.1.4.1	Basic Mode	19-33
19.4.1.4.2	Extended Mode	19-33
19.4.1.5	Channel Abort	19-33
19.4.1.6	Bandwidth Control	19-33
19.4.1.7	Channel State	19-34
19.4.1.8	Illustration of Stride Size and Stride Distance	19-34
19.4.2	DMA Transfer Interfaces	19-35
19.4.3	DMA Errors	19-35
19.4.4	DMA Descriptors	19-35
19.4.5	Limitations and Restrictions	19-38
19.5	DMA System Considerations	19-39
19.5.1	Unusual DMA Scenarios	19-41
19.5.1.1	DMA to e500 Core	19-41
19.5.1.2	DMA to Ethernet	19-41
19.5.1.3	DMA to Configuration, Control, and Status Registers	19-41
19.5.1.4	DMA to I^2C	19-42
19.5.1.5	DMA to DUART	19-42

Chapter 20
Serial RapidIO Interface

20.1	Overview	20-1
20.2	Features	20-1
20.3	Modes of Operation	20-3

Contents

Paragraph Number	Title	Page Number
20.3.1	RapidIO Port	20-3
20.3.2	Message Unit	20-3
20.4	1x/4x LP-Serial Signal Descriptions	20-3
20.4.1	Serial Rapid I/O Interface Overview	20-4
20.4.2	Serial Rapid I/O Interface Detailed Signal Descriptions	20-4
20.4.2.1	SD1_TX[4:7]/SD1_TX[4:7]—Outputs	20-4
20.4.2.2	SD1_RX[4:7]/SD1_RX[4:7]—Inputs	20-4
20.5	Memory Map/Register Definition	20-4
20.6	RapidIO Endpoint Configuration Register Definitions	20-10
20.6.1	RapidIO Architectural Registers	20-10
20.6.1.1	Device Identity Capability Register (DIDCAR)	20-10
20.6.1.2	Device Information Capability Register (DICAR)	20-11
20.6.1.3	Assembly Identity Capability Register (AIDCAR)	20-12
20.6.1.4	Assembly Information Capability Register (AICAR)	20-12
20.6.1.5	Processing Element Features Capability Register (PEFCAR)	20-13
20.6.1.6	Source Operations Capability Register (SOCAR)	20-14
20.6.1.7	Destination Operations Capability Register (DOCAR)	20-15
20.6.1.8	Mailbox Command and Status Register (MCSR)	20-16
20.6.1.9	Port-Write and Doorbell Command and Status Register (PWDCSR)	20-18
20.6.1.10	Processing Element Logical Layer Control Command and Status Register (PELLCCSR)	20-19
20.6.1.11	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	20-19
20.6.1.12	Base Device ID Command and Status Register (BDIDCSR)	20-20
20.6.1.13	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	20-21
20.6.1.14	Component Tag Command and Status Register (CTCSR)	20-21
20.6.2	RapidIO Extended Features Space, 1x/4x LP-Serial Registers	20-22
20.6.2.1	Port Maintenance Block Header 0 Register (PMBH0)	20-22
20.6.2.2	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	20-22
20.6.2.3	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	20-23
20.6.2.4	General Control Command and Status Register (GCCSR)	20-23
20.6.2.5	Link Maintenance Request Command and Status Register (LMREQCSR)	20-24
20.6.2.6	Link Maintenance Response Command and Status Register (LMRESPCSR)	20-25
20.6.2.7	Local ackID Status Command and Status Register (LASCSR)	20-25
20.6.2.8	Error and Status Command and Status Register (ESCSR)	20-26
20.6.2.9	Control Command and Status Register (CCSR)	20-28
20.6.3	RapidIO Extended Features Space—Error Reporting Logical Registers	20-30
20.6.3.1	Error Reporting Block Header Register (ERBH)	20-30
20.6.3.2	Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)	20-30

Contents

Paragraph Number	Title	Page Number
20.6.3.3	Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)	20-32
20.6.3.4	Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)	20-33
20.6.3.5	Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)	20-34
20.6.3.6	Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)	20-35
20.6.4	RapidIO Extended Features Space—Error Reporting Physical Registers	20-36
20.6.4.1	Error Detect Command and Status Register (EDCSR)	20-36
20.6.4.2	Error Rate Enable Command and Status Register (ERECSR)	20-36
20.6.4.3	Error Capture Attributes Command and Status Register (ECACSR)	20-37
20.6.4.4	Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)	20-38
20.6.4.5	Packet Error Capture Command and Status Register 1 (PECCSR1)	20-39
20.6.4.6	Packet Error Capture Command and Status Register 2 (PECCSR2)	20-39
20.6.4.7	Packet Error Capture Command and Status Register 3 (PECCSR3)	20-40
20.6.4.8	Error Rate Command and Status Register (ERCSR)	20-40
20.6.4.9	Error Rate Threshold Command and Status Register (ERTCSR)	20-41
20.6.5	RapidIO Implementation Space Registers	20-42
20.6.5.1	Logical Layer Configuration Register (LLCR)	20-42
20.6.5.2	Error/Port-Write Interrupt Status Register (EPWISR)	20-43
20.6.5.3	Logical Retry Error Threshold Configuration Register (LRETCR)	20-43
20.6.5.4	Physical Retry Error Threshold Configuration Register (PRETCR)	20-44
20.6.5.5	Alternate Device ID Command and Status Register (ADIDCSR)	20-44
20.6.5.6	Accept-All Configuration Register (AACR)	20-45
20.6.5.7	Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)	20-45
20.6.5.8	Implementation Error Command and Status Register (IECSR)	20-46
20.6.5.9	Physical Configuration Register (PCR)	20-47
20.6.5.10	Serial Link Command and Status Register (SLCSR)	20-47
20.6.5.11	Serial Link Error Injection Configuration Register (SLEICR)	20-48
20.6.6	Revision Control Registers	20-49
20.6.6.1	IP Block Revision Register 1 (IPBRR1)	20-49
20.6.6.2	IP Block Revision Register 2 (IPBRR2)	20-49
20.6.7	RapidIO Implementation Space—ATMU Registers	20-50
20.6.7.1	Segmented Outbound Window Description	20-50
20.6.7.2	RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)	20-52
20.6.7.3	RapidIO Outbound Window Translation Extended Address Registers 0–8 (ROWTEAR*n*)	20-53
20.6.7.4	RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)	20-54

Contents

Paragraph Number	Title	Page Number
20.6.7.5	RapidIO Outbound Window Attributes Registers 0–8 (ROWAR*n*)	20-54
20.6.7.6	RapidIO Outbound Window Segment 1–3 Registers 1–8 (ROWS*n*R*n*)	20-56
20.6.7.7	RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)	20-58
20.6.7.8	RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)	20-58
20.6.7.9	RapidIO Inbound Window Attributes Registers 0–4 (RIWAR*n*)	20-59
20.7	RapidIO Message Unit Registers	20-61
20.7.1	RapidIO Outbound Message 0 Registers	20-61
20.7.1.1	Outbound Message *n* Mode Registers (OM*n*MR)	20-61
20.7.1.2	Outbound Message *n* Status Registers (OM*n*SR)	20-63
20.7.1.3	Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR) and Outbound Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)	20-64
20.7.1.4	Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (EOM*n*DQEPAR) and Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (OM*n*DQEPAR)	20-66
20.7.1.5	Extended Outbound Message *n* Source Address Registers (EOM*n*SAR) and Outbound Message *n* Source Address Registers (OM*n*SAR)	20-67
20.7.1.6	Outbound Message *n* Destination Port Registers (OM*n*DPR)	20-68
20.7.1.7	Outbound Message *n* Destination Attributes Registers (OM*n*DATR)	20-69
20.7.1.8	Outbound Message *n* Double-word Count Registers (OM*n*DCR)	20-69
20.7.1.9	Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)	20-70
20.7.1.10	Outbound Message *n* Multicast Group Registers (OM*n*MGR)	20-71
20.7.1.11	Outbound Message *n* Multicast List Registers (OM*n*MLR)	20-72
20.7.2	RapidIO Inbound Message Registers	20-72
20.7.2.1	Inbound Message *n* Mode Registers (IM*n*MR)	20-72
20.7.2.2	Inbound Message *n* Status Registers (IM*n*SR)	20-74
20.7.2.3	Extended Inbound Message Frame Queue Dequeue Pointer Address Registers (EIM*n*FQDPAR) and Inbound Message Frame Queue Dequeue Pointer Address Registers (IM*n*FQDPAR)	20-75
20.7.2.4	Extended Inbound Message Frame Queue Enqueue Pointer Address Registers (EIM*n*FQEPAR) and Inbound Message Frame Queue Enqueue Pointer Address Registers (IM*n*FQEPAR)	20-76
20.7.2.5	Inbound Message *n* Maximum Interrupt Report Interval Registers (IM*n*MIRIR)	20-77
20.7.3	Outbound RapidIO Doorbell Controller Registers	20-78
20.7.3.1	Outbound Doorbell Mode Register (ODMR)	20-78
20.7.3.2	Outbound Doorbell Status Register (ODSR)	20-79
20.7.3.3	Outbound Doorbell Destination Port Register (ODDPR)	20-79
20.7.3.4	Outbound Doorbell Destination Attributes Register (ODDATR)	20-80

Contents

Paragraph Number	Title	Page Number
20.7.3.5	Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)	20-81
20.7.4	Inbound RapidIO Doorbell Controller	20-81
20.7.4.1	Inbound Doorbell Mode Register (IDMR)	20-81
20.7.4.2	Inbound Doorbell Status Register (IDSR)	20-83
20.7.4.3	Extended Inbound Doorbell Queue Dequeue Pointer Address Register (EIDQDPAR) and Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR)	20-84
20.7.4.4	Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR) and Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)	20-85
20.7.4.5	Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)	20-86
20.7.5	RapidIO Port-Write Registers	20-87
20.7.5.1	Inbound Port-Write Mode Register (IPWMR)	20-87
20.7.5.2	Inbound Port-Write Status Register (IPWSR)	20-88
20.7.5.3	Extended Inbound Port-Write Queue Base Address Register (EIPWQBAR) and Inbound Port-Write Queue Base Address Register (IPWQBAR)	20-88
20.8	Functional Description	20-89
20.8.1	RapidIO Transaction Summary	20-89
20.8.2	RapidIO Packet Format Summary	20-91
20.8.3	RapidIO Control Symbol Summary	20-92
20.8.4	Accessing Configuration Registers via RapidIO Packets	20-94
20.8.4.1	Inbound Maintenance Accesses	20-94
20.8.4.1.1	Guidelines	20-94
20.8.4.2	Outbound Maintenance Accesses	20-94
20.8.5	RapidIO Outbound ATMU	20-95
20.8.5.1	Valid Hits to Multiple ATMU Windows	20-95
20.8.5.2	Window Boundary Crossing Errors	20-96
20.8.6	RapidIO Inbound ATMU	20-97
20.8.6.1	Hits to Multiple ATMU Windows	20-97
20.8.6.2	Window Boundary Crossing Errors	20-97
20.8.7	Generating Link-Request/Reset-Device	20-98
20.8.8	Outbound Drain Mode	20-98
20.8.9	Input Port Disable Mode	20-99
20.8.10	Software Assisted Error Recovery Register Support	20-99
20.8.11	Hot-Swap Support	20-100
20.8.11.1	Method 1	20-100
20.8.11.1.1	Extraction	20-100
20.8.11.1.2	Insertion	20-101
20.8.11.2	Method 2 with RapidIO Port Hot-Swapped	20-101

Contents

Paragraph Number	Title	Page Number
20.8.11.2.1	Extraction	20-101
20.8.11.2.2	Insertion	20-102
20.8.12	Errors and Error Handling	20-102
20.8.12.1	RapidIO Error Description	20-102
20.8.12.2	Physical Layer RapidIO Errors Detected	20-103
20.8.12.3	Logical Layer Errors and Error Handling	20-105
20.8.12.3.1	Logical Layer RapidIO Errors Detected	20-106
20.9	RapidIO Message Unit	20-139
20.9.1	Overview	20-139
20.9.2	Features	20-140
20.9.3	Outbound Modes of Operation	20-141
20.9.4	Outbound Message Controller Operation	20-141
20.9.4.1	Direct Mode Operation	20-141
20.9.4.1.1	Interrupts	20-142
20.9.4.1.2	Message Error Response Errors	20-143
20.9.4.1.3	Packet Response Time-out Errors	20-143
20.9.4.1.4	Retry Error Threshold Exceeded Errors	20-143
20.9.4.1.5	Transaction Errors	20-143
20.9.4.1.6	Error Handling	20-143
20.9.4.1.7	Disabling and Enabling the Message Controller	20-144
20.9.4.1.8	Hardware Error Handling	20-144
20.9.4.1.9	Programming Errors	20-149
20.9.4.2	Chaining Mode	20-149
20.9.4.2.1	Message Controller Initialization	20-150
20.9.4.2.2	Chaining Mode Operation	20-151
20.9.4.2.3	Changing Descriptor Queues in Chaining Mode	20-152
20.9.4.2.4	Preventing Queue Overflow in Chaining Mode	20-152
20.9.4.2.5	Switching Between Direct and Chaining Modes	20-152
20.9.4.2.6	Chaining Mode Descriptor Format	20-153
20.9.4.2.7	Chaining Mode Controller Interrupts	20-153
20.9.4.2.8	Message Error Response Errors	20-154
20.9.4.2.9	Packet Response Time-out Errors	20-154
20.9.4.2.10	Retry Error Threshold Exceeded Errors	20-154
20.9.4.2.11	Transaction Errors	20-155
20.9.4.2.12	Error Handling	20-155
20.9.4.2.13	Hardware Error Handling	20-155
20.9.4.2.14	Programming Errors	20-156
20.9.4.3	Message Controller Arbitration	20-156
20.9.5	Inbound Message Controller Operation	20-157
20.9.5.1	Inbound Message Controller Initialization	20-158
20.9.5.2	Inbound Controller Operation	20-158

Contents

Paragraph Number	Title	Page Number
20.9.5.3	Message Steering	20-159
20.9.5.4	Retry Response Conditions	20-159
20.9.5.5	Inbound Message Controller Interrupts	20-160
20.9.5.5.1	Message Request Time-out Errors	20-160
20.9.5.5.2	Transaction Errors	20-160
20.9.5.5.3	Error Handling	20-161
20.9.5.5.4	Hardware Error Handling	20-161
20.9.5.5.5	Programming Errors	20-166
20.9.5.5.6	Disabling and Enabling the Inbound Message Controller	20-167
20.9.6	RapidIO Message Passing Logical Specification Registers	20-168
20.10	RapidIO Doorbell and Port-Write Unit	20-168
20.10.1	Features	20-168
20.10.2	Doorbell Controller	20-169
20.10.2.1	Outbound Doorbell Controller	20-169
20.10.2.1.1	Interrupts	20-170
20.10.2.1.2	Error Response Errors	20-170
20.10.2.1.3	Packet Response Time-Out Errors	20-171
20.10.2.1.4	Retry Error Threshold Exceeded Errors	20-171
20.10.2.1.5	Error Handling	20-171
20.10.2.1.6	Disabling and Enabling the Doorbell Controller	20-171
20.10.2.1.7	Hardware Error Handling	20-171
20.10.2.1.8	Programming Errors	20-174
20.10.2.2	Inbound Doorbell Controller	20-175
20.10.2.2.1	Inbound Doorbell Controller Initialization	20-175
20.10.2.2.2	Inbound Doorbell Controller Operation	20-175
20.10.2.2.3	Inbound Doorbell Queue Entry Format	20-176
20.10.2.2.4	Retry Response Conditions	20-177
20.10.2.2.5	Doorbell Controller Interrupts	20-177
20.10.2.2.6	Transaction Errors	20-177
20.10.2.2.7	Error Handling	20-178
20.10.2.2.8	Hardware Error Handling	20-178
20.10.2.2.9	Programming Errors	20-181
20.10.2.2.10	Disabling and Enabling the Doorbell Controller	20-181
20.10.2.3	RapidIO Message Passing Logical Specification Registers	20-181
20.10.3	Port-Write Controller	20-182
20.10.3.1	Port-Write Controller Initialization	20-182
20.10.3.2	Port-Write Controller Operation	20-183
20.10.3.3	Port-Write Controller Interrupt	20-183
20.10.3.4	Discarding Port-Writes	20-184
20.10.3.5	Transaction Errors	20-184
20.10.3.5.1	Error Handling	20-184

Contents

Paragraph Number	Title	Page Number
20.10.3.6	Hardware Error Handling	20-184
20.10.3.6.1	Programming Errors	20-188
20.10.3.7	Disabling and Enabling the Port-Write Controller	20-188
20.10.3.8	RapidIO Message Passing Logical Specification Registers	20-188

Chapter 21
PCI Express Interface Controller

21.1	Introduction	21-1
21.1.1	Overview	21-1
21.1.1.1	Outbound Transactions	21-2
21.1.1.2	Inbound Transactions	21-3
21.1.2	Features	21-3
21.1.3	Modes of Operation	21-4
21.1.3.1	Root Complex/Endpoint Modes	21-4
21.1.3.2	Link Width	21-4
21.2	External Signal Descriptions	21-5
21.3	Memory Map/Register Definitions	21-5
21.3.1	PCI Express Memory Mapped Registers	21-6
21.3.2	PCI Express Configuration Access Registers	21-10
21.3.2.1	PCI Express Configuration Address Register (PEX_CONFIG_ADDR)	21-10
21.3.2.2	PCI Express Configuration Data Register (PEX_CONFIG_DATA)	21-11
21.3.2.3	PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)	21-11
21.3.2.4	PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)	21-12
21.3.2.5	PCI Express Configuration Register (PEX_CONFIG)	21-13
21.3.3	PCI Express Power Management Event and Message Registers	21-13
21.3.3.1	PCI Express PME and Message Detect Register (PEX_PME_MES_DR)	21-13
21.3.3.2	PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)	21-15
21.3.3.3	PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)	21-16
21.3.3.4	PCI Express Power Management Command Register (PEX_PMCR)	21-18
21.3.4	PCI Express IP Block Revision Registers	21-19
21.3.4.1	IP Block Revision Register 1 (PEX_IP_BLK_REV1)	21-19
21.3.4.2	IP Block Revision Register 2 (PEX_IP_BLK_REV2)	21-19
21.3.5	PCI Express ATMU Registers	21-20
21.3.5.1	PCI Express Outbound ATMU Registers	21-20
21.3.5.1.1	PCI Express Outbound Translation Address Registers (PEXOTAR*n*)	21-20

Contents

Paragraph Number	Title	Page Number
21.3.5.1.2	PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)	21-21
21.3.5.1.3	PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)	21-22
21.3.5.1.4	PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)	21-22
21.3.5.2	PCI Express Inbound ATMU Registers	21-25
21.3.5.2.1	EP Inbound ATMU Implementation	21-25
21.3.5.2.2	RC Inbound ATMU Implementation	21-25
21.3.5.2.3	PCI Express Inbound Translation Address Registers (PEXITAR*n*)	21-26
21.3.5.2.4	PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)	21-27
21.3.5.2.5	PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)	21-27
21.3.5.2.6	PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)	21-28
21.3.6	PCI Express Error Management Registers	21-30
21.3.6.1	PCI Express Error Detect Register (PEX_ERR_DR)	21-30
21.3.6.2	PCI Express Error Interrupt Enable Register (PEX_ERR_EN)	21-32
21.3.6.3	PCI Express Error Disable Register (PEX_ERR_DISR)	21-34
21.3.6.4	PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)	21-36
21.3.6.5	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0)	21-37
21.3.6.5.1	PEX_ERR_CAP_R0—Outbound Case	21-37
21.3.6.5.2	PEX_ERR_CAP_R0—Inbound Case	21-37
21.3.6.6	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)	21-38
21.3.6.6.1	PEX_ERR_CAP_R1—Outbound Case	21-38
21.3.6.6.2	PEX_ERR_CAP_R1—Inbound Case	21-39
21.3.6.7	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)	21-40
21.3.6.7.1	PEX_ERR_CAP_R2—Outbound Case	21-40
21.3.6.7.2	PEX_ERR_CAP_R2—Inbound Case	21-41
21.3.6.8	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)	21-42
21.3.6.8.1	PEX_ERR_CAP_R3—Outbound Case	21-42
21.3.6.8.2	PEX_ERR_CAP_R3—Inbound Case	21-42
21.3.7	PCI Express Configuration Space Access	21-43
21.3.7.1	RC Configuration Register Access	21-43
21.3.7.1.1	PCI Express Configuration Access Register Mechanism	21-43
21.3.7.1.2	Outbound ATMU Configuration Mechanism (RC-Only)	21-44
21.3.7.2	EP Configuration Register Access	21-45
21.3.8	PCI Compatible Configuration Headers	21-45
21.3.8.1	Common PCI Compatible Configuration Header Registers	21-46
21.3.8.1.1	PCI Express Vendor ID Register—Offset 0x00	21-46
21.3.8.1.2	PCI Express Device ID Register—Offset 0x02	21-46
21.3.8.1.3	PCI Express Command Register—Offset 0x04	21-46
21.3.8.1.4	PCI Express Status Register—Offset 0x06	21-48

Contents

Paragraph Number	Title	Page Number
21.3.8.1.5	PCI Express Revision ID Register—Offset 0x08	21-49
21.3.8.1.6	PCI Express Class Code Register—Offset 0x09	21-49
21.3.8.1.7	PCI Express Cache Line Size Register—Offset 0x0C	21-50
21.3.8.1.8	PCI Express Latency Timer Register—0x0D	21-51
21.3.8.1.9	PCI Express Header Type Register—0x0E	21-51
21.3.8.1.10	PCI Express BIST Register—0x0F	21-52
21.3.8.2	Type 0 Configuration Header	21-52
21.3.8.2.1	PCI Express Base Address Registers—0x10–0x27	21-52
21.3.8.2.2	PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C	21-55
21.3.8.2.3	PCI Express Subsystem ID Register (EP-Mode Only)—0x2E	21-55
21.3.8.2.4	Capabilities Pointer Register—0x34	21-56
21.3.8.2.5	PCI Express Interrupt Line Register (EP-Mode Only)—0x3C	21-56
21.3.8.2.6	PCI Express Interrupt Pin Register—0x3D	21-57
21.3.8.2.7	PCI Express Minimum Grant Register (EP-Mode Only)—0x3E	21-57
21.3.8.2.8	PCI Express Maximum Latency Register (EP-Mode Only)—0x3F	21-58
21.3.8.3	Type 1 Configuration Header	21-58
21.3.8.3.1	PCI Express Base Address Register 0—0x10	21-59
21.3.8.3.2	PCI Express Primary Bus Number Register—Offset 0x18	21-59
21.3.8.3.3	PCI Express Secondary Bus Number Register—Offset 0x19	21-60
21.3.8.3.4	PCI Express Subordinate Bus Number Register—Offset 0x1A	21-60
21.3.8.3.5	PCI Express Secondary Latency Timer Register—0x1B	21-61
21.3.8.3.6	PCI Express I/O Base Register—0x1C	21-61
21.3.8.3.7	PCI Express I/O Limit Register—0x1D	21-61
21.3.8.3.8	PCI Express Secondary Status Register—0x1E	21-62
21.3.8.3.9	PCI Express Memory Base Register—0x20	21-63
21.3.8.3.10	PCI Express Memory Limit Register—0x22	21-63
21.3.8.3.11	PCI Express Prefetchable Memory Base Register—0x24	21-64
21.3.8.3.12	PCI Express Prefetchable Memory Limit Register—0x26	21-64
21.3.8.3.13	PCI Express Prefetchable Base Upper 32 Bits Register—0x28	21-65
21.3.8.3.14	PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C	21-65
21.3.8.3.15	PCI Express I/O Base Upper 16 Bits Register—0x30	21-65
21.3.8.3.16	PCI Express I/O Limit Upper 16 Bits Register—0x32	21-66
21.3.8.3.17	Capabilities Pointer Register—0x34	21-66
21.3.8.3.18	PCI Express Interrupt Line Register—0x3C	21-67
21.3.8.3.19	PCI Express Interrupt Pin Register—0x3D	21-67
21.3.8.3.20	PCI Express Bridge Control Register—0x3E	21-68
21.3.9	PCI Compatible Device-Specific Configuration Space	21-69
21.3.9.1	PCI Express Power Management Capability ID Register—0x44	21-70
21.3.9.2	PCI Express Power Management Capabilities Register—0x46	21-70
21.3.9.3	PCI Express Power Management Status and Control Register—0x48	21-71
21.3.9.4	PCI Express Power Management Data Register—0x4B	21-71

Contents

Paragraph Number	Title	Page Number
21.3.9.5	PCI Express Capability ID Register—0x4C	21-72
21.3.9.6	PCI Express Capabilities Register—0x4E	21-72
21.3.9.7	PCI Express Device Capabilities Register—0x50	21-73
21.3.9.8	PCI Express Device Control Register—0x54	21-73
21.3.9.9	PCI Express Device Status Register—0x56	21-74
21.3.9.10	PCI Express Link Capabilities Register—0x58	21-75
21.3.9.11	PCI Express Link Control Register—0x5C	21-75
21.3.9.12	PCI Express Link Status Register—0x5E	21-76
21.3.9.13	PCI Express Slot Capabilities Register—0x60	21-77
21.3.9.14	PCI Express Slot Control Register—0x64	21-77
21.3.9.15	PCI Express Slot Status Register—0x66	21-78
21.3.9.16	PCI Express Root Control Register (RC Mode Only)—0x68	21-79
21.3.9.17	PCI Express Root Status Register (RC Mode Only)—0x6C	21-79
21.3.9.18	PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70	21-80
21.3.9.19	PCI Express MSI Message Control Register (EP Mode Only)—0x72	21-80
21.3.9.20	PCI Express MSI Message Address Register (EP Mode Only)—0x74	21-81
21.3.9.21	PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78	21-81
21.3.9.22	PCI Express MSI Message Data Register (EP Mode Only)—0x7C	21-82
21.3.10	PCI Express Extended Configuration Space	21-83
21.3.10.1	PCI Express Advanced Error Reporting Capability ID Register—0x100	21-84
21.3.10.2	PCI Express Uncorrectable Error Status Register—0x104	21-84
21.3.10.3	PCI Express Uncorrectable Error Mask Register—0x108	21-85
21.3.10.4	PCI Express Uncorrectable Error Severity Register—0x10C	21-86
21.3.10.5	PCI Express Correctable Error Status Register—0x110	21-87
21.3.10.6	PCI Express Correctable Error Mask Register—0x114	21-87
21.3.10.7	PCI Express Advanced Error Capabilities and Control Register—0x118	21-88
21.3.10.8	PCI Express Header Log Register—0x11C–0x12B	21-88
21.3.10.9	PCI Express Root Error Command Register—0x12C	21-89
21.3.10.10	PCI Express Root Error Status Register—0x130	21-90
21.3.10.11	PCI Express Correctable Error Source ID Register—0x134	21-91
21.3.10.12	PCI Express Error Source ID Register—0x136	21-91
21.3.10.13	LTSSM State Status Register—0x404	21-92
21.3.10.14	PCI Express Controller Core Clock Ratio Register—0x440	21-93
21.3.10.15	PCI Express Power Management Timer Register—0x450	21-94
21.3.10.16	PCI Express PME Time-Out Register (EP-Mode Only)—0x454	21-95
21.3.10.17	PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478	21-96
21.3.10.18	Configuration Ready Register—0x4B0	21-96
21.3.10.19	PME_To_Ack Timeout Register (RC-Mode Only)—0x590	21-97
21.3.10.20	Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0	21-97

Contents

Paragraph Number	Title	Page Number
21.4	Functional Description	21-98
21.4.1	Architecture	21-100
21.4.1.1	PCI Express Transactions	21-100
21.4.1.2	Byte Ordering	21-100
21.4.1.2.1	Byte Order for Configuration Transactions	21-102
21.4.1.3	Lane Reversal	21-102
21.4.1.4	Transaction Ordering Rules	21-103
21.4.1.5	Memory Space Addressing	21-103
21.4.1.6	I/O Space Addressing	21-103
21.4.1.7	Configuration Space Addressing	21-104
21.4.1.8	Serialization of Configuration and I/O Writes	21-104
21.4.1.9	Messages	21-104
21.4.1.9.1	Outbound ATMU Message Generation	21-104
21.4.1.9.2	Inbound Messages	21-106
21.4.1.10	Error Handling	21-108
21.4.1.10.1	PCI Express Error Logging and Signaling	21-108
21.4.1.10.2	PCI Express Controller Internal Interrupt Sources	21-110
21.4.1.10.3	Error Conditions	21-112
21.4.2	Interrupts	21-114
21.4.2.1	EP Interrupt Generation	21-114
21.4.2.1.1	Hardware INTx Message Generation	21-114
21.4.2.1.2	Hardware MSI Generation	21-114
21.4.2.1.3	Software INTx Message Generation	21-114
21.4.2.1.4	Software MSI Generation	21-114
21.4.2.2	RC Handling of INTx Message and MSI Interrupts	21-115
21.4.2.2.1	INTx Message Handling	21-115
21.4.2.2.2	MSI Handling	21-115
21.4.3	Initial Credit Advertisement	21-115
21.4.4	Power Management	21-116
21.4.4.1	L2/L3 Ready Link State	21-116
21.4.5	Hot Reset	21-117
21.4.6	Link Down	21-117
21.5	Initialization/Application Information	21-117
21.5.1	Boot Mode and Inbound Configuration Transactions	21-117

Chapter 22
General Purpose I/O (GPIO)

22.1	Introduction	22-1
22.1.1	Overview	22-1
22.1.2	Features	22-1

Contents

Paragraph Number	Title	Page Number
22.2	External Signal Description	22-2
22.2.1	Signals Overview	22-2
22.3	Memory Map/Register Definition	22-2
22.3.1	GPIO Direction Register (GPDIR)	22-2
22.3.2	GPIO Open Drain Register (GPODR)	22-3
22.3.3	GPIO Data Register (GPDAT)	22-3
22.3.4	GPIO Interrupt Event Register (GPIER)	22-4
22.3.5	GPIO Interrupt Mask Register (GPIMR)	22-4
22.3.6	GPIO Interrupt Control Register (GPICR)	22-5

Part IV
Global Functions and Debug

Chapter 23
Global Utilities

23.1	Overview	23-1
23.2	Global Utilities Features	23-1
23.2.1	Power Management and Block Disables	23-1
23.2.2	Accessing Current POR Configuration Settings	23-1
23.2.3	Clock Control	23-1
23.3	External Signal Description	23-1
23.3.1	Signals Overview	23-2
23.3.2	Detailed Signal Descriptions	23-2
23.4	Memory Map/Register Definition	23-3
23.4.1	Register Descriptions	23-4
23.4.1.1	POR PLL Status Register (PORPLLSR)	23-4
23.4.1.2	POR Boot Mode Status Register (PORBMSR)	23-6
23.4.1.3	POR I/O Impedance Status and Control Register (PORIMPSCR)	23-7
23.4.1.4	POR Device Status Register (PORDEVSR)	23-7
23.4.1.5	POR Debug Mode Status Register (PORDBGMSR)	23-11
23.4.1.6	POR Device Status Register 2 (PORDEVSR2)	23-12
23.4.1.7	General-Purpose POR Configuration Register (GPPORCR)	23-13
23.4.1.8	Alternate Function Signal Multiplex Control Register (PMUXCR)	23-13
23.4.1.9	Device Disable Register (DEVDISR)	23-14
23.4.1.10	Power Management Control and Status Register (POWMGTCSR)	23-16
23.4.1.11	Machine Check Summary Register (MCPSUMR)	23-18
23.4.1.12	Reset Request Status and Control Register (RSTRSCR)	23-20
23.4.1.13	Exception Reset Control Register (ECTRSTCR)	23-20
23.4.1.14	Automatic Reset Status Register (AUTORSTSR)	23-21
23.4.1.15	Processor Version Register (PVR)	23-22

Contents

Paragraph Number	Title	Page Number
23.4.1.16	System Version Register (SVR)	23-23
23.4.1.17	Reset Control Register (RSTCR)	23-23
23.4.1.18	LBC Voltage Select Control Register (LBCVSELCR)	23-24
23.4.1.19	DDR Clock Disable Register (DDRCLKDR)	23-24
23.4.1.20	Clock Out Control Register (CLKOCR)	23-26
23.4.1.21	SerDes1 Control Register (SRDS1CR)	23-27
23.4.1.22	SerDes2 Control Register (SRDS2CR)	23-28
23.5	Functional Description	23-30
23.5.1	Power Management	23-30
23.5.1.1	Relationship Between Both Cores and Device Power Management States	23-30
23.5.1.2	$\overline{\text{CKSTP\_IN0/1}}$ is Not Power Management	23-31
23.5.1.3	Dynamic Power Management	23-32
23.5.1.4	Shutting Down Unused Blocks	23-32
23.5.1.5	Software-Controlled Power-Down States	23-32
23.5.1.5.1	Doze Mode	23-32
23.5.1.5.2	Nap Mode	23-33
23.5.1.5.3	Sleep Mode	23-33
23.5.1.6	Power Management Control Fields	23-33
23.5.1.7	Power-Down Sequence Coordination	23-34
23.5.1.8	Interrupts and Power Management	23-36
23.5.1.8.1	Interrupts and Power Management Controlled by MSR[WE]	23-36
23.5.1.8.2	Interrupts and Power Management Controlled by POWMGTCSR	23-36
23.5.1.9	Snooping in Power-Down Modes	23-37
23.5.1.10	Software Considerations for Power Management	23-37
23.5.1.11	Requirements for Reaching and Recovering from Sleep State	23-37

Chapter 24
Device Performance Monitor

24.1	Introduction	24-1
24.1.1	Overview	24-2
24.1.2	Features	24-3
24.2	Signal Descriptions	24-3
24.3	Memory Map and Register Definition	24-3
24.3.1	Register Summary	24-4
24.3.2	Control Registers	24-6
24.3.2.1	Performance Monitor Global Control Register (PMGC0)	24-6
24.3.2.2	Performance Monitor Local Control Registers (PMLCA*n*, PMLCB*n*)	24-6
24.3.3	Counter Registers	24-10
24.3.3.1	Performance Monitor Counters (PMC0–PMC11)	24-11
24.4	Functional Description	24-12

Contents

Paragraph Number	Title	Page Number
24.4.1	Performance Monitor Interrupt	24-12
24.4.2	Event Counting	24-12
24.4.3	Threshold Events	24-12
24.4.4	Chaining	24-13
24.4.5	Triggering	24-14
24.4.6	Burstiness Counting	24-14
24.4.7	Performance Monitor Events	24-16
24.4.8	Performance Monitor Examples	24-35

Chapter 25
Debug Features and Watchpoint Facility

25.1	Introduction	25-1
25.1.1	Overview	25-1
25.1.2	Features	25-3
25.1.3	Modes of Operation	25-3
25.1.3.1	Memory Debug Mode (Local Bus and DDR)	25-4
25.1.3.2	DDR SDRAM Interface Debug Mode	25-4
25.1.3.3	Watchpoint Monitor Modes	25-5
25.1.3.4	Trace Buffer Modes	25-5
25.2	External Signal Description	25-5
25.2.1	Overview	25-6
25.2.2	Detailed Signal Descriptions	25-7
25.2.2.1	Debug Signals—Details	25-7
25.2.2.2	Watchpoint Monitor Trigger Signals—Details	25-8
25.2.2.3	Test Signals—Details	25-8
25.3	Memory Map/Register Definition	25-10
25.3.1	Watchpoint Monitor Register Descriptions	25-11
25.3.1.1	Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)	25-11
25.3.1.2	Watchpoint Monitor Address Register (WMAR)	25-13
25.3.1.3	Watchpoint Monitor Address Mask Register (WMAMR)	25-14
25.3.1.4	Watchpoint Monitor Transaction Mask Register (WMTMR)	25-14
25.3.1.5	Watchpoint Monitor Status Register (WMSR)	25-16
25.3.2	Trace Buffer Register Descriptions	25-16
25.3.2.1	Trace Buffer Control Registers (TBCR0, TBCR1)	25-16
25.3.2.2	Trace Buffer Address Register (TBAR)	25-19
25.3.2.3	Trace Buffer Address Mask Register (TBAMR)	25-19
25.3.2.4	Trace Buffer Transaction Mask Register (TBTMR)	25-20
25.3.2.5	Trace Buffer Status Register (TBSR)	25-20
25.3.2.6	Trace Buffer Access Control Register (TBACR)	25-21
25.3.2.7	Trace Buffer Access Data High Register (TBADHR)	25-22

Contents

Paragraph Number	Title	Page Number
25.3.2.8	Trace Buffer Access Data Register (TBADR)	25-22
25.3.3	Context ID Registers	25-23
25.3.3.1	Programmed Context ID Register (PCIDR)	25-23
25.3.3.2	Current Context ID Register (CCIDR)	25-23
25.3.4	Trigger Out Function	25-24
25.3.4.1	Trigger Out Source Register (TOSR)	25-24
25.4	Functional Description	25-25
25.4.1	Source and Target ID	25-25
25.4.2	DDR SDRAM Interface Debug	25-26
25.4.2.1	Debug Information on Debug Pins	25-27
25.4.2.2	Debug Information on ECC Pins	25-27
25.4.3	Local Bus Interface Debug	25-27
25.4.4	Watchpoint Monitor	25-27
25.4.4.1	Watchpoint Monitor Performance Monitor Events	25-28
25.4.5	Trace Buffer	25-28
25.4.5.1	Traced Data Formats (as a Function of TBCR1[IFSEL])	25-29
25.5	Initialization	25-30

Appendix A
Revision History

Glossary

Contents

Paragraph Number	Title	Page Number

Figures

Figure Number	Title	Page Number
1-1	MPC8572E Block Diagram	1-2
1-2	Processing Transactions Across the On-Chip Fabric	1-13
1-3	Two Processors Running in Symmetric Multiprocessor Mode	1-14
1-4	Integrated Dual Core Device Application	1-15
1-5	VPN Access Router Enabled by PCI Express and Ethernet	1-16
1-6	High-Performance Communication System Using Distributed Processing	1-16
1-7	High-Performance Communication System	1-17
1-8	RAID Controller Application Using MPC8572E	1-17
1-9	MPC8572E with SerDes	1-18
1-10	MPC8572E as a Control Plane Processor for a DSP Farm Interconnected with RapidIO	1-18
2-1	Local Memory Map Example	2-2
2-2	Local Access IP Block Revision Register 1 (LAIPBRR1)	2-6
2-3	Local Access IP Block Revision Register 2 (LAIPBRR2)	2-6
2-4	Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR11)	2-7
2-5	Local Access Window *n* Attributes Registers (LAWAR0–LAWAR11)	2-8
2-6	Top-Level Register Map Example	2-12
2-7	General Utilities Registers Mapping to Configuration, Control, and Status Memory Block	2-14
2-8	PIC Mapping to Configuration, Control, and Status Memory Block	2-15
2-9	RapidIO Mapping to Configuration, Control, and Status Memory Block	2-16
2-10	Device-Specific Register Mapping to Configuration, Control, and Status Memory Block	2-17
3-1	MPC8572E Signal Groupings (1/3)	3-3
3-2	MPC8572E Signal Groupings (2/3) (Continued)	3-4
3-3	MPC8572E Signal Groupings (3/3) (Continued)	3-5
4-1	Configuration, Control, and Status Register Base Address Register (CCSRBAR)	4-6
4-2	Alternate Configuration Base Address Register (ALTCBAR)	4-7
4-3	Alternate Configuration Attribute Register (ALTCAR)	4-7
4-4	Boot Page Translation Register (BPTR)	4-9
4-5	Power-On Reset Sequence	4-12
4-6	Clock Subsystem Block Diagram	4-31
4-7	RTC and Core Timer Facilities Clocking Options	4-33
5-1	e500 Core Complex Block Diagram	5-2
5-2	Vector and Floating-Point APUs	5-5
5-3	MU Pipeline, Showing Divide Bypass	5-7
5-4	Three-Stage Load/Store Unit	5-8
5-5	Instruction Pipeline Flow	5-14
5-6	GPR Issue Queue (GIQ)	5-15

Figures

Figure Number	Title	Page Number
5-7	e500 Core Programming Model	5-17
5-8	MMU Structure	5-23
5-9	Effective-to-Real Address Translation Flow	5-24
5-10	Effective-to-Real Address Translation Flow (e500v2)	5-25
6-1	Core Register Model	6-2
6-2	Integer Exception Register (XER)	6-8
6-3	Condition Register (CR)	6-9
6-4	Link Register (LR)	6-11
6-5	Count Register (CTR)	6-11
6-6	Machine State Register (MSR)	6-11
6-7	Processor ID Register (PIR)	6-13
6-8	Processor Version Register (PVR)	6-13
6-9	System Version Register (SVR)	6-14
6-10	Timer Control Register (TCR)	6-14
6-11	Timer Status Register (TSR)	6-15
6-12	Time Base Upper/Lower Registers (TBU/TBL)	6-16
6-13	Decrementer Register (DEC)	6-16
6-14	Decrementer Auto-Reload Register (DECAR)	6-17
6-15	Save/Restore Register 0 (SRR0)	6-17
6-16	Save/Restore Register 1 (SRR1)	6-17
6-17	Critical Save/Restore Register 0 (CSRR0)	6-17
6-18	Critical Save/Restore Register 1 (CSRR1)	6-18
6-19	Data Exception Address Register (DEAR)	6-18
6-20	Interrupt Vector Prefix Register (IVPR)	6-18
6-21	Interrupt Vector Offset Registers (IVOR*n*)	6-18
6-22	Exception Syndrome Register (ESR)	6-19
6-23	Machine Check Save/Restore Register 0 (MCSRR0)	6-20
6-24	Machine Check Save/Restore Register 1 (MCSRR1)	6-20
6-25	Machine Check Address Register (MCAR)	6-21
6-26	Machine Check Address Register Upper (MCARU)	6-21
6-27	Machine Check Syndrome Register (MCSR)	6-21
6-28	Software-Use SPRs (SPRG0–SPRG7 and USPRG0)	6-22
6-29	Branch Buffer Entry Address Register (BBEAR)	6-23
6-30	Branch Buffer Target Address Register (BBTAR)	6-23
6-31	Branch Unit Control and Status Register (BUCSR)	6-24
6-32	Hardware Implementation-Dependent Register 0 (HID0)	6-25
6-33	Hardware Implementation-Dependent Register 1 (HID1)	6-26
6-34	L1 Cache Control and Status Register 0 (L1CSR0)	6-28
6-35	L1 Cache Control and Status Register 1 (L1CSR1)	6-29
6-36	L1 Cache Configuration Register 0 (L1CFG0)	6-30
6-37	L1 Cache Configuration Register 1 (L1CFG1)	6-31

Figures

Figure Number	Title	Page Number
6-38	Process ID Registers (PID0–PID2)	6-32
6-39	MMU Control and Status Register 0 (MMUCSR0)	6-32
6-40	MMU Configuration Register (MMUCFG)	6-32
6-41	TLB Configuration Register 0 (TLB0CFG)	6-33
6-42	TLB Configuration Register 1 (TLB1CFG)	6-34
6-43	MAS Register 0 (MAS0)	6-34
6-44	MAS Register 1 (MAS1)	6-35
6-45	MAS Register 2 (MAS2)	6-36
6-46	MAS Register 3 (MAS3)	6-37
6-47	MAS Register 4 (MAS4)	6-38
6-48	MAS Register 6 (MAS6)	6-38
6-49	MAS Register 7 (MAS7)	6-39
6-50	Debug Control Register 0 (DBCR0)	6-39
6-51	Debug Control Register 1 (DBCR1)	6-41
6-52	Debug Control Register 2 (DBCR2)	6-42
6-53	Debug Status Register (DBSR)	6-43
6-54	Instruction Address Compare Registers (IAC1–IAC2)	6-45
6-55	Data Address Compare Registers (DAC1–DAC2)	6-45
6-56	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)	6-45
6-57	Accumulator (ACC)	6-47
6-58	Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0)	6-49
6-59	Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)	6-50
6-60	Local Control B Registers (PMLCb0–PMLCb3)/User Local Control B Registers (UPMLCb0–UPMLCb3)	6-51
6-61	Performance Monitor Counter Registers (PMC0–PMC3)/User Performance Monitor Counter Registers (UPMC0–UPMC3)	6-52
7-1	L2 Cache/SRAM Configuration	7-1
7-2	Cache Organization	7-4
7-3	Physical Address Usage for L2 Cache Accesses	7-5
7-4	Physical Address Usage for SRAM Accesses	7-6
7-5	Data Bus Connection of CCB	7-8
7-6	Address Bus Connection of CCB	7-8
7-7	L2 Control Register (L2CTL)	7-10
7-8	L2 Cache Way Allocation for Processors Register (L2CWAP)	7-13
7-9	Cache External Write Address Registers (L2CEWAR*n*)	7-15
7-10	Cache External Write Address Registers Extended Address (L2CEWAREA*n*)	7-16
7-11	Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)	7-16
7-12	L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)	7-18

Figures

Figure Number	Title	Page Number
7-13	L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA*n*)	7-19
7-14	L2 Error Injection Mask High Register (L2ERRINJHI)	7-20
7-15	L2 Error Injection Mask Low Register (L2ERRINJLO)	7-20
7-16	L2 Error Injection Mask Control Register (L2ERRINJCTL)	7-21
7-17	L2 Error Capture Data High Register (L2CAPTDATAHI)	7-22
7-18	L2 Error Capture Data Low Register (L2CAPTDATALO)	7-22
7-19	L2 Error Syndrome Register (L2CAPTECC)	7-22
7-20	L2 Error Detect Register (L2ERRDET)	7-23
7-21	L2 Error Disable Register (L2ERRDIS)	7-24
7-22	L2 Error Interrupt Enable Register (L2ERRINTEN)	7-24
7-23	L2 Error Attributes Capture Register (L2ERRATTR)	7-25
7-24	L2 Error Address Capture Register (L2ERRADDRL)	7-26
7-25	L2 Error Address Capture Register (L2ERRADDRH)	7-26
7-26	L2 Error Control Register (L2ERRCTL)	7-27
7-27	L2 Cache Line Replacement Algorithm	7-33
8-1	e500 Coherency Module Block Diagram	8-1
8-2	ECM CCB Address Configuration Register (EEBACR)	8-3
8-3	ECM CCB Port Configuration Register (EEBPCR)	8-4
8-4	ECM IP Block Revision Register 1 (EIPBRR1)	8-5
8-5	ECM IP Block Revision Register 2 (EIPBRR2)	8-6
8-6	ECM Error Detect Register (EEDR)	8-6
8-7	ECM Error Enable Register (EEER)	8-7
8-8	ECM Error Attributes Capture Register (EEATR)	8-8
8-9	ECM Error Low Address Capture Register (EELADR)	8-9
8-10	ECM Error High Address Capture Register (EEHADR)	8-9
9-1	DDR Memory Controller Simplified Block Diagram	9-2
9-2	Chip Select Bounds Registers (CS*n*_BNDS)	9-14
9-3	Chip Select Configuration Register (CS*n*_CONFIG)	9-15
9-4	Chip Select Configuration Register 2 (CSn_CONFIG_2)	9-17
9-5	DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)	9-17
9-6	DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)	9-19
9-7	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)	9-21
9-8	DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)	9-23
9-9	DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)	9-25
9-10	DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)	9-28
9-11	DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)	9-30
9-12	DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)	9-31
9-13	DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)	9-32
9-14	DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)	9-34
9-15	DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)	9-35

Figures

Figure Number	Title	Page Number
9-16	DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)	9-35
9-17	DDR Initialization Address Configuration Register (DDR_INIT_ADDR)	9-36
9-18	DDR Initialization Extended Address Configuration Register (DDR_INIT_EXT_ADDR)	9-36
9-19	DDR SDRAM Timing Configuration 4 Register (TIMING_CFG_4)	9-37
9-20	DDR SDRAM Timing Configuration 5 Register (TIMING_CFG_5)	9-39
9-21	DDR ZQ Calibration Control Register (DDR_ZQ_CNTL)	9-41
9-22	DDR Write Leveling Control Register (DDR_WRLVL_CNTL)	9-42
9-23	DDR Debug Status Register 1 (DDRDSR_1)	9-45
9-24	DDR Debug Status Register 2 (DDRDSR_2)	9-46
9-25	DDR Control Driver Register 1 (DDRCDR_1)	9-48
9-26	DDR Control Driver Register 2 (DDRCDR_2)	9-49
9-27	DDR IP Block Revision 1 (DDR_IP_REV1)	9-50
9-28	DDR IP Block Revision 2 (DDR_IP_REV2)	9-50
9-29	Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)	9-51
9-30	Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)	9-51
9-31	Memory Data Path Error Injection Mask ECC Register (ERR_INJECT)	9-52
9-32	Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)	9-52
9-33	Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)	9-53
9-34	Memory Data Path Read Capture ECC Register (CAPTURE_ECC)	9-53
9-35	Memory Error Detect Register (ERR_DETECT)	9-54
9-36	Memory Error Disable Register (ERR_DISABLE)	9-55
9-37	Memory Error Interrupt Enable Register (ERR_INT_EN)	9-56
9-38	Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)	9-57
9-39	Memory Error Address Capture Register (CAPTURE_ADDRESS)	9-58
9-40	Memory Error Extended Address Capture Register (CAPTURE_EXT_ADDRESS)	9-59
9-41	Single-Bit ECC Memory Error Management Register (ERR_SBE)	9-59
9-42	DDR Memory Controller Block Diagram	9-61
9-43	Typical Dual Data Rate SDRAM Internal Organization	9-62
9-44	Typical DDR SDRAM Interface Signals	9-62
9-45	Example 256-Mbyte DDR SDRAM Configuration With ECC	9-63
9-46	DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2	9-74
9-47	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTOR	9-74
9-48	DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3	9-74
9-49	DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4	9-75
9-50	DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs	9-76
9-51	DDR SDRAM Mode-Set Command Timing	9-76
9-52	Registered DDR SDRAM DIMM Burst Write Timing	9-77
9-53	Write Timing Adjustments Example for Write Latency = 1	9-78
9-54	DDR SDRAM Bank Staggered Auto Refresh Timing	9-79
9-55	DDR SDRAM Power-Down Mode	9-80

Figures

Figure Number	Title	Page Number
9-56	DDR SDRAM Self-Refresh Entry Timing	9-81
9-57	DDR SDRAM Self-Refresh Exit Timing	9-81
10-1	Interrupt Sources Block Diagram Features	10-2
10-2	Pass-Through Mode Example	10-6
10-3	Block Revision Register 1 (BRR1)	10-20
10-4	Block Revision Register 2 (BRR2)	10-21
10-5	Feature Reporting Register (FRR)	10-21
10-6	Global Configuration Register (GCR)	10-22
10-7	Vendor Identification Register (VIR)	10-23
10-8	Processor Core Initialization Register (PIR)	10-23
10-9	Interprocessor Interrupt Vector/Priority Register (IPIVPR*n*)	10-24
10-10	Spurious Vector Register (SVR)	10-25
10-11	Timer Frequency Reporting Registers (TFRR*x*)	10-25
10-12	Global Timer Current Count Registers (GTCCR*xn*)	10-26
10-13	Global Timer Base Count Register (GTBCR*xn*)	10-27
10-14	Global Timer Vector/Priority Register (GTVPR*xn*)	10-27
10-15	Global Timer Destination Registers (GTDR*xn*)	10-28
10-16	Example Calculation for Cascaded Timers	10-29
10-17	Timer Control Registers (TCR*x*)	10-29
10-18	External Interrupt Summary Register (ERQSR)	10-31
10-19	IRQ_OUT Summary Register 0 (IRQSR0)	10-31
10-20	IRQ_OUT Summary Register 1 (IRQSR1)	10-32
10-21	IRQ_OUT Summary Register 2 (IRQSR2)	10-33
10-22	Critical Interrupt Summary Register 0 (CISR0)	10-33
10-23	Critical Interrupt Summary Register 1 (CISR1)	10-34
10-24	Critical Interrupt Summary Register 2 (CISR2)	10-34
10-25	Performance Monitor Mask Registers 0 (PM*n*MR0)	10-35
10-26	Performance Monitor Mask Registers 1 (PM*n*MR1)	10-36
10-27	Performance Monitor Mask Registers 2 (PM*n*MR2)	10-36
10-28	Message Registers (MSGRs)	10-37
10-29	Message Enable Register (MER)	10-37
10-30	Message Status Register (MSR)	10-38
10-31	Message Shared Interrupt Registers (MSIR*n*)	10-39
10-32	Shared Message Signaled Interrupt Status Register (MSISR)	10-39
10-33	Shared Message Signaled Interrupt Index Register (MSIIR)	10-40
10-34	Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)	10-40
10-35	Shared Message Signaled Interrupt Destination Registers (MSIDR*n*)	10-41
10-36	Destination Register Summary	10-42
10-37	Vector/Priority Register Summary	10-43
10-38	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)	10-43
10-39	External Interrupt Destination Registers (EIDRs)	10-44

Figures

Figure Number	Title	Page Number
10-40	Internal Interrupt Vector/Priority Registers (IIVPRs)	10-45
10-41	Internal Interrupt Destination Registers (IIDRs)	10-46
10-42	Messaging Interrupt Vector/Priority Registers (MIVPR*n*)	10-47
10-43	Messaging Interrupt Destination Registers (MIDR*n*)	10-48
10-44	Per-CPU Register Address Decoding in a Four-Core Device	10-50
10-45	Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)	10-50
10-46	Processor Core Current Task Priority Registers (CTPR*n*)	10-51
10-47	Processor Core Who Am I Registers (WHOAMI*n*)	10-52
10-48	Processor Core Interrupt Acknowledge Registers (IACK*n*)	10-53
10-49	End of Interrupt Registers (EOI*n*)	10-53
10-50	PIC Interrupt Processing Flow Diagram for Each Core (*n*)	10-55
11-1	SEC Connected to MPC8572E System Bus	11-3
11-2	SEC Functional Modules	11-4
11-3	Descriptor Format	11-22
11-4	Header Dword	11-23
11-5	Pointer Dword	11-28
11-6	Link Table Entry	11-29
11-7	Descriptors, Link Tables, and Parcels	11-31
11-8	Fetch FIFO Enqueue Counter	11-38
11-9	Descriptor Finished Counter	11-38
11-10	Data Bytes In Counter	11-39
11-11	Descriptor Finished Counter	11-39
11-12	Channel Configuration Register	11-41
11-13	Channel Status Register	11-44
11-14	Current Descriptor Pointer Register	11-47
11-15	Fetch FIFO	11-48
11-16	Gather Link Table	11-48
11-17	Scatter Link Table	11-49
11-18	EU Assignment Status Register (EUASR)	11-53
11-19	Interrupt Enable Register	11-54
11-20	Interrupt Status Register	11-57
11-21	Interrupt Clear Register	11-58
11-22	ID Register	11-59
11-23	IP Block Revision Register	11-59
11-24	Master Control Register	11-60
11-25	AESU Mode Register	11-63
11-26	AESU Key Size Register	11-67
11-27	AESU Data Size Register	11-68
11-28	AESU Reset Control Register	11-68
11-29	AESU Status Register	11-70
11-30	AESU Interrupt Status Register	11-71

Figures

Figure Number	Title	Page Number
11-31	AESU Interrupt Mask Register	11-73
11-32	AESU ICV Size Register	11-75
11-33	AESU End_of_Message Register	11-75
11-34	AESU CCM Context Registers	11-81
11-35	AFEU Mode Register	11-93
11-36	AFEU Key Size Register	11-94
11-37	AFEU Context/Data Size Register	11-95
11-38	AFEU Reset Control Register	11-95
11-39	AFEU Status Register	11-96
11-40	AFEU Interrupt Status Register	11-97
11-41	AFEU Interrupt Mask Register	11-98
11-42	AFEU End of Message Register	11-100
11-43	CRCU Mode Register	11-103
11-44	CRCU Key Size Register	11-104
11-45	CRCU Data Size Register	11-104
11-46	CRCU Reset Control Register	11-104
11-47	CRCU Control Register	11-105
11-48	CRCU Status Register	11-106
11-49	CRCU Interrupt Status Register	11-107
11-50	CRCU Interrupt Mask Register	11-108
11-51	CRCU Context Register (Write)	11-110
11-52	CRCU Context Register (Read-Default Mode)	11-111
11-53	CRCU Context Register (Read-Raw Mode)	11-111
11-54	CRCU Key Register	11-112
11-55	DEU Mode Register	11-113
11-56	DEU Key Size Register	11-114
11-57	DEU Data Size Register	11-114
11-58	DEU Reset Control Register	11-115
11-59	DEU Status Register	11-116
11-60	DEU Interrupt Status Register	11-117
11-61	DEU Interrupt Mask Register	11-119
11-62	DEU End_of_Message Register	11-120
11-63	KEU Mode Register	11-122
11-64	KEU Key Size Register	11-124
11-65	KEU Data Size Register	11-124
11-66	KEU Reset Control Register	11-125
11-67	KEU Status Register	11-126
11-68	KEU Interrupt Status Register	11-128
11-69	KEU Interrupt Mask Register	11-129
11-70	KEU Data Out Register (F9 MAC)	11-131
11-71	KEU End of Message Register	11-131

Figures

Figure Number	Title	Page Number
11-72	KEU IV_1 Register	11-132
11-73	KEU IV_2 Register (Fresh)	11-133
11-74	KEU Key Data Register_1 (CK-High)	11-134
11-75	KEU Key Data Register_2 (CK-Low)	11-134
11-76	KEU Key Data Register_3 (IK-High)	11-134
11-77	KEU Key Data Register_4 (IK-Low)	11-134
11-78	MDEU Mode Register in Old Configuration	11-136
11-79	MDEU Mode Register in New Configuration	11-138
11-80	MDEU Key Size Register	11-140
11-81	MDEU Data Size Register	11-141
11-82	MDEU Reset Control Register	11-141
11-83	MDEU Status Register	11-142
11-84	MDEU Interrupt Status Register	11-143
11-85	MDEU Interrupt Mask Register	11-145
11-86	MDEU ICV Size Register	11-146
11-87	MDEU End_of_Message Register	11-147
11-88	MDEU Context Register	11-148
11-89	PKEU Mode Register	11-151
11-90	PKEU Key Size Register	11-151
11-91	PKEU AB Size Register	11-153
11-92	PKEU Data Size Register	11-153
11-93	PKEU Reset Control Register	11-154
11-94	PKEU Status Register	11-155
11-95	PKEU Interrupt Status Register	11-156
11-96	PKEU Interrupt Mask Register	11-157
11-97	PKEU End_of_Message Register	11-159
11-98	RNGU Mode Register	11-160
11-99	RNGU Data Size Register	11-161
11-100	RNGU Reset Control Register	11-161
11-101	RNGU Status Register	11-162
11-102	RNGU Interrupt Status Register	11-163
11-103	RNGU Interrupt Mask Register	11-164
11-104	RNGU End_of_Message Register	11-165
12-1	I^2C Block Diagram	12-1
12-2	I^2C Address Register (I2CADR)	12-6
12-3	I^2C Frequency Divider Register (I2CFDR)	12-6
12-4	I^2C Control Register (I2CCR)	12-7
12-5	I^2C Status Register (I2CSR)	12-9
12-6	I^2C Data Register (I2CDR)	12-10
12-7	I^2C Digital Filter Sampling Rate Register (I2CDFSRR)	12-11
12-8	I^2C Interface Transaction Protocol	12-12

Figures

Figure Number	Title	Page Number
12-9	EEPROM Data Format for One Register Preload Command	12-20
12-10	EEPROM Contents	12-20
12-11	Example I^2C Interrupt Service Routine Flowchart	12-25
13-1	UART Block Diagram	13-2
13-2	Receiver Buffer Registers (URBR*n*)	13-5
13-3	Transmitter Holding Registers (UTHR*n*)	13-6
13-4	Divisor Most Significant Byte Registers (UDMB0, UDMB1)	13-6
13-5	Divisor Least Significant Byte Registers (UDLB*n*)	13-7
13-6	Interrupt Enable Register (UIER)	13-8
13-7	Interrupt ID Registers (UIIR)	13-9
13-8	FIFO Control Registers (UFCR*n*)	13-10
13-9	Line Control Register (ULCR)	13-11
13-10	Modem Control Register (UMCR)	13-13
13-11	Line Status Register (ULSR)	13-14
13-12	Modem Status Register (UMSR)	13-15
13-13	Scratch Register (USCR)	13-16
13-14	Alternate Function Register (UAFR)	13-16
13-15	DMA Status Register (UDSR)	13-17
13-16	UART Bus Interface Transaction Protocol Example	13-19
14-1	Enhanced Local Bus Controller Block Diagram	14-1
14-2	Base Registers (BR*n*)	14-11
14-3	Option Registers (OR*n*) in GPCM Mode	14-14
14-4	Option Registers (OR*n*) in FCM Mode	14-16
14-5	Option Registers (OR*n*) in UPM Mode	14-19
14-6	UPM Memory Address Register (MAR)	14-20
14-7	UPM Mode Registers (M*x*MR)	14-21
14-8	Memory Refresh Timer Prescaler Register (MRTPR)	14-23
14-9	UPM Data Register in UPM Mode (MDR)	14-24
14-10	FCM Data Register in FCM Mode (MDR)	14-24
14-11	Special Operation Initiation Register (LSOR)	14-25
14-12	UPM Refresh Timer (LURT)	14-25
14-13	Transfer Error Status Register (LTESR)	14-26
14-14	Transfer Error Check Disable Register (LTEDR)	14-28
14-15	Transfer Error Interrupt Enable Register (LTEIR)	14-29
14-16	Transfer Error Attributes Register (LTEATR	14-30
14-17	Transfer Error Address Register (LTEAR)	14-31
14-18	Local Bus Configuration Register	14-32
14-19	Clock Ratio Register (LCRR)	14-33
14-20	Flash Mode Register	14-34
14-21	Flash Instruction Register	14-36
14-22	Flash Command Register	14-37

Figures

Figure Number	Title	Page Number
14-23	Flash Block Address Register	14-38
14-24	Flash Page Address Register, Small Page Device (ORx[PGS] = 0)	14-38
14-25	Flash Page Address Register, Large Page Device (ORx[PGS] = 1)	14-38
14-26	Flash Byte Count Register	14-40
14-27	Basic Operation of Memory Controllers in the eLBC	14-41
14-28	Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420 (LCRR[PBYP] = 0)	14-43
14-29	Basic eLBC Bus Cycle with LALE, TA, and $\overline{\text{LCSn}}$	14-44
14-30	eLBC Bus Cycles in PLL and PLL-bypassed Modes (GPCM and UPM only)	14-46
14-31	Enhanced Local Bus to GPCM Device Interface	14-47
14-32	GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0)	14-47
14-33	GPCM General Read Timing Parameters	14-48
14-34	GPCM General Write Timing Parameters	14-49
14-35	GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0)	14-51
14-36	GPCM Relaxed Timing Back-to-Back Reads (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, EHTR = 0)	14-52
14-37	GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1)	14-53
14-38	GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)	14-53
14-39	GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1)	14-54
14-40	GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)	14-55
14-41	GPCM Read Followed by Write (TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)	14-55
14-42	External Termination of GPCM Access	14-56
14-43	Local Bus to 8-bit FCM Device Interface	14-57
14-44	FCM Basic Page Read Timing (PGS = 1, CSCT = 0, CST = 0, CHT = 1, RST = 1, SCY = 0, TRLX = 0, EHTR = 1)	14-58
14-45	FCM Buffer RAM Memory Map for Small-Page (512-byte page) NAND Flash Devices	14-60
14-46	FCM Buffer RAM Memory Map for Large-Page (2-Kbyte page) NAND Flash Devices	14-61
14-47	FCM ECC Calculation	14-62
14-48	ECC Layout for LBCR[EPAR] = 0 (~ represents logical negation)	14-62
14-49	ECC Placement in NAND Flash Spare Regions in Relation to FMR[ECCM]	14-63
14-50	FCM Instruction Sequencer Mechanism	14-63
14-51	Timing of FCM Command/Address and Write Data Cycles (for TRLX = 0, CHT = 0, CST = 1, SCY = 1)	14-66
14-52	Example of FCM Command and Address Timing with Minimum Delay Parameters (for TRLX = 0, CHT = 0, CST = 0, SCY = 0)	14-67
14-53	Example of FCM Command and Address Timing with Relaxed Parameters (for TRLX = 1, CHT = 0, CST = 1, SCY = 2)	14-68

Figures

Figure Number	Title	Page Number
14-54	FCM Delay Prior to Sampling LFRB State	14-68
14-55	FCM Read Data Timing (for TRLX = 0, RST = 0, SCY = 1)	14-69
14-56	FCM Read Data Timing with Extended Hold Time (for TRLX = 0, EHTR = 1, RST = 1, SCY = 1)	14-70
14-57	FCM Buffer RAM Memory Map During Boot Loading	14-71
14-58	User-Programmable Machine Functional Block Diagram	14-73
14-59	RAM Array Indexing	14-74
14-60	Memory Refresh Timer Request Block Diagram	14-75
14-61	UPM Clock Scheme	14-78
14-62	RAM Array and Signal Generation	14-78
14-63	LCSn Signal Selection	14-82
14-64	LBS Signal Selection	14-82
14-65	UPM Read Access Data Sampling	14-86
14-66	Effect of LUPWAIT Signal	14-87
14-67	Multiplexed Address/Data Bus for 32-Bit Addressing	14-88
14-68	Local Bus Peripheral Hierarchy for High Bus Speeds	14-89
14-69	GPCM Address Timings	14-89
14-70	GPCM Data Timings	14-90
14-71	Interface to Different Port-Size Devices	14-92
14-72	Single-Beat Read Access to FPM DRAM	14-98
14-73	Single-Beat Write Access to FPM DRAM	14-99
14-74	Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)	14-100
14-75	Refresh Cycle (CBR) to FPM DRAM	14-101
14-76	Exception Cycle	14-102
14-77	Interface to ZBT SRAM	14-103
15-1	eTSEC Block Diagram	15-2
15-2	TSEC_ID Register	15-27
15-3	TSEC_ID2 Register	15-27
15-4	IEVENT Register Definition	15-29
15-5	IMASK Register Definition	15-32
15-6	EDIS Register Definition	15-34
15-7	ECNTRL Register Definition	15-35
15-8	PTV Register Definition	15-38
15-9	DMACTRL Register	15-39
15-10	TBIPA Register Definition	15-40
15-11	TCTRL Register Definition	15-41
15-12	TSTAT Register Definition	15-43
15-13	DFVLAN Register Definition	15-46
15-14	TXIC Register Definition	15-47
15-15	TQUEUE Register Definition	15-48

Figures

Figure Number	Title	Page Number
15-16	TR03WT Register Definition	15-49
15-17	TR47WT Register Definition	15-49
15-18	TBDBPH Register Definition	15-50
15-19	TBPTR0–TBPTR7 Register Definition	15-51
15-20	TBASEH Register Definition	15-51
15-21	TBASE Register Definition	15-52
15-22	TMR_TXTS*n*_ID Register Definition	15-52
15-23	TMR_TXTS*n*_H/L Register Definition	15-53
15-24	RCTRL Register Definition	15-54
15-25	RSTAT Register Definition	15-57
15-26	RXIC Register Definition	15-59
15-27	RQUEUE Register Definition	15-60
15-28	RBIFX Register Definition	15-62
15-29	Receive Queue Filer Table Address Register Definition	15-63
15-30	Receive Queue Filer Table Control Register Definition	15-64
15-31	Receive Queue Filer Table Property IDs 0, 2–15 Register Definition	15-65
15-32	Receive Queue Filer Table Property ID1 Register Definition	15-65
15-33	MRBLR Register Definition	15-68
15-34	RBDBPH Register Definition	15-69
15-35	RBPTR0–RBPTR7 Register Definition	15-69
15-36	RBASEH Register Definition	15-70
15-37	RBASE Register Definition	15-71
15-38	TMR_RXTS_H/L Register Definition	15-71
15-39	MACCFG1 Register Definition	15-75
15-40	MACCFG2 Register Definition	15-76
15-41	IPGIFG Register Definition	15-78
15-42	Half-Duplex Register Definition	15-79
15-43	Maximum Frame Length Register Definition	15-80
15-44	MII Management Configuration Register Definition	15-81
15-45	MIIMCOM Register Definition	15-82
15-46	MIIMADD Register Definition	15-82
15-47	MII Mgmt Control Register Definition	15-83
15-48	MIIMSTAT Register Definition	15-83
15-49	MII Mgmt Indicator Register Definition	15-84
15-50	Interface Status Register Definition	15-84
15-51	MAC Station Address Part 1 Register Definition	15-85
15-52	MAC Station Address Part 2 Register Definition	15-86
15-53	MAC Exact Match Address *n* Part 1 Register Definition	15-86
15-54	MAC Exact Match Address *x* Part 2 Register Definition	15-87
15-55	Transmit and Receive 64-Byte Frame Register Definition	15-88
15-56	Transmit and Receive 65- to 127-Byte Frame Register Definition	15-89

Figures

Figure Number	Title	Page Number
15-57	Transmit and Received 128- to 255-Byte Frame Register Definition	15-89
15-58	Transmit and Received 256- to 511-Byte Frame Register Definition	15-90
15-59	Transmit and Received 512- to 1023-Byte Frame Register Definition	15-90
15-60	Transmit and Received 1024- to 1518-Byte Frame Register Definition	15-91
15-61	Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition	15-91
15-62	Receive Byte Counter Register Definition	15-92
15-63	Receive Packet Counter Register Definition	15-92
15-64	Receive FCS Error Counter Register Definition	15-93
15-65	Receive Multicast Packet Counter Register Definition	15-93
15-66	Receive Broadcast Packet Counter Register Definition	15-94
15-67	Receive Control Frame Packet Counter Register Definition	15-94
15-68	Receive Pause Frame Packet Counter Register Definition	15-95
15-69	Receive Unknown OPCode Packet Counter Register Definition	15-95
15-70	Receive Alignment Error Counter Register Definition	15-96
15-71	Receive Frame Length Error Counter Register Definition	15-96
15-72	Receive Code Error Counter Register Definition	15-97
15-73	Receive Carrier Sense Error Counter Register Definition	15-97
15-74	Receive Undersize Packet Counter Register Definition	15-98
15-75	Receive Oversize Packet Counter Register Definition	15-98
15-76	Receive Fragments Counter Register Definition	15-99
15-77	Receive Jabber Counter Register Definition	15-99
15-78	Receive Dropped Packet Counter Register Definition	15-100
15-79	Transmit Byte Counter Register Definition	15-100
15-80	Transmit Packet Counter Register Definition	15-101
15-81	Transmit Multicast Packet Counter Register Definition	15-101
15-82	Transmit Broadcast Packet Counter Register Definition	15-102
15-83	Transmit Pause Control Frame Counter Register Definition	15-102
15-84	Transmit Deferral Packet Counter Register Definition	15-103
15-85	Transmit Excessive Deferral Packet Counter Register Definition	15-103
15-86	Transmit Single Collision Packet Counter Register Definition	15-104
15-87	Transmit Multiple Collision Packet Counter Register Definition	15-104
15-88	Transmit Late Collision Packet Counter Register Definition	15-105
15-89	Transmit Excessive Collision Packet Counter Register Definition	15-105
15-90	Transmit Total Collision Counter Register Definition	15-106
15-91	Transmit Drop Frame Counter Register Definition	15-106
15-92	Transmit Jabber Frame Counter Register Definition	15-107
15-93	Transmit FCS Error Counter Register Definition	15-107
15-94	Transmit Control Frame Counter Register Definition	15-108
15-95	Transmit Oversized Frame Counter Register Definition	15-108
15-96	Transmit Undersize Frame Counter Register Definition	15-109
15-97	Transmit Fragment Counter Register Definition	15-109

Figures

Figure Number	Title	Page Number
15-98	Carry Register 1 (CAR1) Register Definition	15-110
15-99	Carry Register 2 (CAR2) Register Definition	15-111
15-100	Carry Mask Register 1 (CAM1) Register Definition	15-112
15-101	Carry Mask Register 2 (CAM2) Register Definition	15-114
15-102	Receive Filer Rejected Packet Counter Register Definition	15-115
15-103	IGADDR*n* Register Definition	15-116
15-104	GADDR*n* Register Definition	15-116
15-105	FIFOCFG Register Definition	15-117
15-106	ATTR Register Definition	15-119
15-107	ATTRELI Register Definition	15-120
15-108	RQPRM Register Definition	15-121
15-109	RFBPTR0–RFBPTR7 Register Definition	15-122
15-110	TMR_CTRL Register Definition	15-122
15-111	TMR_TEVENT Register Definition	15-124
15-112	TMR_PEVENT Register Definition	15-126
15-113	TMR_PEMASK Register Definition	15-127
15-114	TMR_CNT_H Register Definition	15-128
15-115	TMR_ADD Register Definition	15-129
15-116	TMR_ACC Register Definition	15-130
15-117	TMR_PRSC Register Definition	15-130
15-118	TMROFF_H/L Register Definition	15-131
15-119	TMR_ALARM1-2_H/L Register Definition	15-131
15-120	TMR_FIPER*n* Register Definition	15-133
15-121	TMR_ETTS1-2_H/L Register Definition	15-133
15-122	Control Register Definition	15-136
15-123	Status Register Definition	15-137
15-124	AN Advertisement Register Definition	15-138
15-125	AN Link Partner Base Page Ability Register Definition	15-140
15-126	AN Expansion Register Definition	15-141
15-127	AN Next Page Transmit Register Definition	15-142
15-128	AN Link Partner Ability Next Page Register Definition	15-143
15-129	Extended Status Register Definition	15-144
15-130	Jitter Diagnostics Register Definition	15-144
15-131	TBI Control Register Definition	15-145
15-132	eTSEC-MII Connection	15-147
15-133	eTSEC-RMII Connection	15-148
15-134	eTSEC-GMII Connection	15-149
15-135	eTSEC-RGMII Connection	15-150
15-136	eTSEC-TBI Connection	15-151
15-137	eTSEC-RTBI Connection	15-152
15-138	eTSEC-FIFO (8-Bit) Connection	15-160

Figures

Figure Number	Title	Page Number
15-139	8-Bit GMII-Style Packet FIFO Timing	15-160
15-140	8-Bit Encoded Packet FIFO Timing	15-161
15-141	eTSEC-FIFO (16-Bit) Connection	15-162
15-142	16-Bit GMII-Style Packet FIFO Timing	15-163
15-143	16-Bit Encoded Packet FIFO Timing	15-164
15-144	Definition of Custom Preamble Sequence	15-170
15-145	Definition of Received Preamble Sequence	15-171
15-146	Ethernet Address Recognition Flowchart	15-172
15-147	Sample C Code for Computing eTSEC Hash Table Indices	15-174
15-148	Location of Frame Control Blocks for TOE Parameters	15-182
15-149	Transmit Frame Control Block	15-182
15-150	Receive Frame Control Block	15-184
15-151	Structure of the Receive Queue Filer Table	15-187
15-152	1588 Timer Design Partition	15-198
15-153	Ethernet Sampling Points for 1588	15-199
15-154	PTP Packet Format	15-200
15-155	Example of eTSEC Memory Structure for BDs	15-201
15-156	Buffer Descriptor Ring	15-202
15-157	Transmit Buffer Descriptor	15-202
15-158	Mapping of TxBDs to a C Data Structure	15-203
15-159	Receive Buffer Descriptor	15-205
15-160	Mapping of RxBDs to a C Data Structure	15-206
16-1	Ethernet Protocol in Relation to the OSI Protocol Stack	16-1
16-2	Ethernet/IEEE 802.3 Frame Structure	16-2
16-3	Ethernet/IEEE 802.3 Frame Structure With More Details	16-3
16-4	FEC Block Diagram	16-4
16-5	IEVENT Register Definition	16-13
16-6	IMASK Register Definition	16-16
16-7	Error Disabled Register (EDIS)	16-17
16-8	MINFLR Register Definition	16-18
16-9	PTV Register Definition	16-18
16-10	DMACTRL Register Definition	16-19
16-11	FIFO_PAUSE_CTRL Register Definition	16-21
16-12	FIFO_TX_THR Register Definition	16-21
16-13	FIFO_TX_STARVE Register Definition	16-22
16-14	FIFO_TX_STARVE_SHUTOFF Register Definition	16-22
16-15	TCTRL Register Definition	16-23
16-16	TSTAT Register Definition	16-24
16-17	TBDLEN Register Definition	16-25
16-18	CTBPTR Register Definition	16-26
16-19	TBPTR Register Definition	16-26

Figures

Figure Number	Title	Page Number
16-20	TBASE Register Definition	16-27
16-21	OSTBD Register Definition	16-27
16-22	OSTBDP Register Definition	16-29
16-23	RCTRL Register Definition	16-30
16-24	RSTAT Register Definition	16-31
16-25	RBDLEN Register Definition	16-31
16-26	CRBPTR Register Definition	16-32
16-27	MRBL Register Definition	16-32
16-28	RBPTR Register Definition	16-33
16-29	RBASE Register Definition	16-33
16-30	MACCFG1 Register Definition	16-36
16-31	MACCFG2 Register Definition	16-38
16-32	IPGIFG Register Definition	16-39
16-33	Half-Duplex Register Definition	16-40
16-34	Maximum Frame Length Register Definition	16-41
16-35	MII Management Configuration Register Definition	16-41
16-36	MIIMCOM Register Definition	16-42
16-37	MIIMADD Register Definition	16-43
16-38	MII Management Control Register Definition	16-44
16-39	MIIMSTAT Register Definition	16-44
16-40	MII Management Indicator Register Definition	16-45
16-41	Interface Control Register Definition	16-45
16-42	Interface Status Register Definition	16-47
16-43	Station Address Part 1 Register Definition	16-47
16-44	Station Address Part 2 Register Definition	16-48
16-45	IADDR*n* Register Definition	16-49
16-46	GADDR*n* Register Definition	16-50
16-47	ATTR Register Definition	16-50
16-48	ATTRELI Register Definition	16-51
16-49	FEC-MII Connection	16-53
16-50	Ethernet Address Recognition Flowchart	16-60
16-51	Example of FEC Memory Structure for BD	16-67
16-52	Buffer Descriptor Ring	16-67
16-53	Transmit Buffer Descriptor	16-68
16-54	Receive Buffer Descriptor	16-70
17-1	High Level Block Diagram of the PM	17-4
17-2	Interrupt Status Register Common Control Format	17-15
17-3	Interrupt Enable Register Common Control Format	17-16
17-4	Interrupt Status Disable Register Common Control Format	17-17
17-5	Interrupt Inhibit Register Common Control Format	17-18
17-6	Statistics Counters Overflow Status Register (SCOS)	17-19

Figures

Figure Number	Title	Page Number
17-7	Performance Monitor Threshold Format	17-20
17-8	Channel Scheduling Control Register Format	17-21
17-9	STNIB Register Format	17-22
17-10	STNIS Register Format	17-23
17-11	STNTH1 Register Format	17-23
17-12	STNTH2 Register Format	17-24
17-13	STNTHL Register Format	17-24
17-14	STNTHS Register Format	17-25
17-15	STNCH Register Format	17-25
17-16	SWDB Register Format	17-26
17-17	KES Variable Length Trigger Size Register Format	17-26
17-18	KES Error Configuration Register Format	17-27
17-19	PME Error Handling Disables Register Format	17-28
17-20	STNPM Register Format	17-29
17-21	STNS1M Register Format	17-30
17-22	DXE Pattern Range Counter Configuration Register Format	17-30
17-23	STNPMR Register Format	17-31
17-24	Pattern Description and Stateful Rule Base Address High Register Format	17-31
17-25	Pattern Description and Stateful Rule Base Address Low Register Format	17-32
17-26	DXE Memory Control Register Format	17-32
17-27	DXE Error Configuration Register Format	17-33
17-28	STNDSR Register Format	17-34
17-29	STNESR Register Format	17-34
17-30	STNS1R Register Format	17-35
17-31	STNOB Register Format	17-35
17-32	SRE Context Base Address High Register Format	17-36
17-33	SRE Context Base Address Low Register Format	17-36
17-34	SRE Memory Control Register Format	17-37
17-35	SRE Configuration Register Format	17-38
17-36	SRE Rule Reset Vector 0–7 Register Format	17-39
17-37	SRE Rule Reset First Index Register Format	17-39
17-38	SRE Rule Reset 32-Byte Increment Register Format	17-40
17-39	SRE Rule Reset Repetitions Register Format	17-41
17-40	SRE Rule Reset Work Configuration Register Format	17-41
17-41	SRE Free Running Counter Configuration Register Format	17-42
17-42	SRE Error Configuration 1 Register Format	17-43
17-43	SRE Error Configuration 2 Register Format	17-43
17-44	SRE Error Configuration 3 Register Format	17-44
17-45	STDBC Register Format	17-44
17-46	STDBP Register Format	17-45
17-47	STDWC Register Format	17-45

Figures

Figure Number	Title	Page Number
17-48	DBPPWL Register Format	17-46
17-49	Free Buffer List Buffer Size Register Format	17-47
17-50	Free Buffer List Assignment A Register Format	17-47
17-51	Free Buffer List Assignment B Register Format	17-49
17-52	Free Buffer Manager Control Register Format	17-50
17-53	MIA Byte Count Register (MIA_BYC)	17-51
17-54	MIA Block Count Register (MIA_BLC)	17-51
17-55	MIA Count Enable Register (MIA_CE)	17-52
17-56	MIA Control Register (MIA_CR)	17-55
17-57	MIA Arbitration Control Register (MIA_ACR)	17-57
17-58	MIA Local Snoop Hit Count Registers (MIA_LSHC0/1/2)	17-58
17-59	MIA Cancelled Write Count Registers (MIA_CWC0/1/2)	17-58
17-60	PM IP Block Revision 1Register (PM_IP_REV_1)	17-59
17-61	PM IP Block Revision 2 Register (PM_IP_REV_2)	17-59
17-62	Interrupt Status Register 0 Format	17-60
17-63	Interrupt Enable Register 0 Format	17-62
17-64	Interrupt Status Disable Register 0 Format	17-63
17-65	Interrupt Inhibit Register 0 Format	17-64
17-66	Interrupt Interval Control Register 0 Format	17-65
17-67	Notifications Produced Index Register Channel 0 Format	17-66
17-68	Commands Consumed Index Register Channel 0 Format	17-67
17-69	Free Buffer Deallocate FIFO Consumer Index Register Channel 0 Format	17-67
17-70	Channel Error Status Register Channel 0 Format	17-68
17-71	Notifications Consumed Index Register Channel 0 Format	17-69
17-72	Command Produced Index Register Channel 0 Format	17-69
17-73	Commands Expired Index Register Channel 0 Format	17-70
17-74	Free Buffer Deallocate FIFO Producer Index Register Channel 0 Format	17-71
17-75	Output Truncated Incidence Counter Channel 0 Format	17-71
17-76	Read Byte Counter Channel 0 Format	17-72
17-77	Statistics Counters Overflow Status 0 Register (SCOS0)	17-72
17-78	Channel Reset and Control Register Channel 0 Format	17-73
17-79	Command FIFO Base Address High Register Channel 0 Format	17-74
17-80	Command FIFO Base Address Low Register Channel 0 Format	17-75
17-81	Command FIFO Depth Register Channel 0 Format	17-75
17-82	Command FIFO Threshold Register Channel 0 Format	17-76
17-83	Notification FIFO Base Address High Register Channel 0 Format	17-77
17-84	Notification FIFO Base Address Low Register Channel 0 Format	17-77
17-85	Notification FIFO Depth Register Channel 0 Format	17-78
17-86	Notification FIFO Threshold Register Channel 0 Format	17-78
17-87	Notification Deflate Length Limit Register Channel 0 Format	17-79
17-88	Notification Result Length Limit Register Channel 0 Format	17-80

Figures

Figure Number	Title	Page Number
17-89	Free Buffer Deallocate FIFO Base Address High Register Channel 0 Format	17-80
17-90	Free Buffer Deallocate FIFO Base Address Low Register Channel 0 Format	17-81
17-91	Free Buffer Deallocate FIFO Depth Register Channel 0 Format	17-81
17-92	Free Buffer Deallocate FIFO Threshold Register Channel 0 Format	17-82
17-93	Stream Context Configuration Register Channel 0 Format	17-82
17-94	Stream Context Base Address High Register Channel 0 Format	17-83
17-95	Stream Context Base Address Low Register Channel 0 Format	17-84
17-96	Residue Configuration Register Channel 0 Format	17-84
17-97	Residue Base Address High Register Channel 0 Format	17-85
17-98	Residue Base Address Low Register Channel 0 Format	17-86
17-99	Cache Awareness Control Register Channel 0 Format	17-86
17-100	Memory Transaction Priority Control Register Channel 0 Format	17-88
17-101	Free Buffer List Threshold Register Channel 0 Format	17-89
17-102	Free Buffer List High and Low Watermark A Register Channel 0 Format	17-90
17-103	Free Buffer List Allocation and Deallocation Counter A Register Channel 0 Format	17-91
17-104	Free Buffer List On Deck Disable A Register Channel 0 Format	17-91
17-105	Free Buffer List Physical Head Address High A Register Channel 0 Format	17-92
17-106	Free Buffer List Physical Head Address Low A Register Channel 0 Format	17-93
17-107	Free Buffer List Virtual Head Address High A Register Channel 0 Format	17-93
17-108	Free Buffer List Virtual Head Address Low A Register Channel 0 Format	17-94
17-109	Free Buffer List Physical Tail Address High A Register Channel 0 Format	17-94
17-110	Free Buffer List Physical Tail Address Low A Register Channel 0 Format	17-95
17-111	Free Buffer List Number of Buffers A Register Channel 0 Format	17-95
17-112	Free Buffer List Buffer Size A Register Channel 0 Format	17-96
17-113	Free Buffer List Virtual On Deck Pointer A High Register Channel 0 Format	17-97
17-114	Free Buffer List Virtual On Deck Pointer A Low Register Channel 0 Format	17-97
17-115	Trigger and Confidence Stage	17-104
17-116	Variable Length Trigger Table	17-105
17-117	Variable Length Trigger (VLT) Table Hash	17-107
17-118	Variable Length Trigger Table Row Structure	17-107
17-119	2-Byte Trigger (2BT) Table Hash	17-109
17-120	2-Byte Trigger Table Row Structure	17-109
17-121	1-Byte Trigger (1BT)	17-110
17-122	Special Trigger (ST)	17-111
17-123	Confidence Table	17-112
17-124	Confidence Table Row Structure	17-113
17-125	Confidence Hash	17-114
17-126	2-Byte and Variable Length Trigger Collision Linking Algorithm	17-117
17-127	1-Byte Trigger Collision Linking Algorithm	17-118
17-128	Confidence Mask Codes	17-119
17-129	DXE NFA Operation	17-121

Figures

Figure Number	Title	Page Number
17-130	DXE 128 byte SUI Window Formats	17-122
17-131	Test Line First Block (128-Byte) Format	17-128
17-132	Instruction Test Line Format	17-130
17-133	Pattern Description and Stateful Rule memory space	17-136
17-134	Reaction Linked List Structure	17-139
17-135	1 Reaction Head 128-Byte Block Format	17-140
17-136	Reaction Extension 128-Byte Block Format	17-140
17-137	SRE Instructions Format	17-143
17-138	Operand Register Select Table (for Accumulator Load Instruction)	17-148
17-139	SRE Session Context Entries	17-151
17-140	DMA Engine Memory Structures	17-159
17-141	Input Data Linked List Structure	17-165
17-142	Input Data Scatter/Gather Structure	17-167
17-143	FBM Linked List Structure	17-205
18-1	Table Lookup Unit Block Diagram	18-1
18-2	TLU_ID1 Register Format	18-8
18-3	TLU_ID2 Register Format	18-9
18-4	IEVENT Register Format	18-10
18-5	IMASK Register Format	18-11
18-6	IEATR Register Format	18-11
18-7	IEADD Register Format	18-12
18-8	IEDIS Register Format	18-13
18-9	MBANK*n* Register Format	18-14
18-10	PTBL*n* Register Format	18-15
18-11	CRAMR Register Format	18-16
18-12	CRAMW Register Format	18-17
18-13	CFIND Register Format	18-17
18-14	CTHTK Register Format	18-18
18-15	CTCRT Register Format	18-18
18-16	CTCHS Register Format	18-19
18-17	CTDAT Register Format	18-19
18-18	CHITS Register Format	18-20
18-19	CMISS Register Format	18-20
18-20	CHCOL Register Format	18-21
18-21	CCRTL Register Format	18-21
18-22	CARO Register Format	18-22
18-23	CARM Register Format	18-23
18-24	CMDOP Register Format	18-24
18-25	CMDIX Register Format	18-25
18-26	CSTAT Register Format	18-25
18-27	KD*n*B Register Format	18-27

Figures

Figure Number	Title	Page Number
18-28	Table Lookup Operation	18-28
18-29	Write Command Format	18-30
18-30	Add Command Format	18-31
18-31	Read Command Format	18-32
18-32	Acchash Command Format	18-34
18-33	Find Command Format	18-35
18-34	Findr Command Format	18-36
18-35	Findw Command Format	18-38
18-36	Data Simple Table Entry Format	18-41
18-37	Hash Simple Table Entry Format	18-41
18-38	Trie Simple Table Entry Format	18-42
18-39	Use of COUNT Fields to Resolve Collisions with Trie Entries	18-44
18-40	Key Simple Table Entry Format for 32-Byte/64-Byte Keys	18-44
18-41	Key Simple Table Entry Format for 96-Byte/128-Byte Keys	18-45
18-42	IPTD Simple Table Entry Format	18-46
18-43	ENTROPY Format for HASH ETYPE	18-48
18-44	ENTROPY Format for RUNDELTA ETYPE	18-48
18-45	Flat Data Recommended Memory Layout	18-51
18-46	Hash-Trie-Key Data Structure	18-52
18-47	Typical Memory Layout of Hash-Trie-Key Tables	18-53
18-48	Hash-Trie-Key State Transitions	18-54
18-49	Chained-Hash Data Structure	18-55
18-50	Typical Memory Layout of Chained-Hash Tables	18-56
18-51	Chained-Hash State Transitions	18-57
18-52	CRT Data Structure	18-59
18-53	Typical Memory Layout of CRT Tables	18-60
18-54	CRT State Transitions	18-61
18-55	TLU Hash Function in C	18-62
19-1	DMA Block Diagram	19-1
19-2	DMA Operational Flow Chart	19-4
19-3	DMA Signal Summary	19-5
19-4	DMA Mode Registers (MR*n*)	19-9
19-5	Status Registers (SR*n*)	19-12
19-6	Basic Chaining Mode Flow Chart	19-13
19-7	Extended Current Link Descriptor Address Registers (ECLNDARn)	19-14
19-8	Current Link Descriptor Address Registers (CLNDAR*n*)	19-14
19-9	Source Attributes Registers (SATR*n*)	19-15
19-10	Source Address Registers (SAR*n*)	19-16
19-11	Source Address Registers for RapidIO Maintenance Reads (SAR*n*)	19-17
19-12	Destination Attributes Registers (DATR*n*)	19-18
19-13	Destination Address Registers (DAR*n*)	19-19

Figures

Figure Number	Title	Page Number
19-14	Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)	19-20
19-15	Byte Count Registers (BCR*n*)	19-20
19-16	Next Link Descriptor Address Registers (NLNDAR*n*)	19-21
19-17	Extended Next Link Descriptor Address Registers (ENLNDAR*n*)	19-21
19-18	Extended Current List Descriptor Address Registers (ECLSDAR*n*)	19-22
19-19	Current List Descriptor Address Registers (CLSDAR*n*)	19-22
19-20	Extended Next List Descriptor Address Registers (ENLSDAR*n*)	19-23
19-21	Next List Descriptor Address Registers (NLSDAR*n*)	19-23
19-22	Source Stride Registers (SSR*n*)	19-24
19-23	Destination Stride Registers (DSR*n*)	19-25
19-24	DMA General Status Register (DGSR)	19-25
19-25	External Control Interface Timing	19-32
19-26	Stride Size and Stride Distance	19-34
19-27	DMA Transaction Flow with DMA Descriptors	19-37
19-28	List Descriptor Format	19-38
19-29	Link Descriptor Format	19-38
19-30	DMA Data Paths	19-40
20-1	RapidIO Endpoint and RMU	20-1
20-2	Device Identity Capability Register (DIDCAR)	20-11
20-3	Device Information Capability Register (DICAR)	20-11
20-4	Assembly Identity Capability Register (AIDCAR)	20-12
20-5	Assembly Information Capability Register (AICAR)	20-12
20-6	Processing Element Features Capability Register (PEFCAR)	20-13
20-7	Source Operations Capability Register (SOCAR)	20-14
20-8	Destination Operations Capability Register (DOCAR)	20-15
20-9	Mailbox Command and Status Register (MCSR)	20-16
20-10	Port-Write and Doorbell Command and Status Register (PWDCSR)	20-18
20-11	Processing Element Logic Layer Control Command and Status Register (PELLCCSR)	20-19
20-12	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	20-20
20-13	Base Device ID Command and Status Register (BDIDCSR)	20-20
20-14	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	20-21
20-15	Component Tag Command and Status Register (CTCSR)	20-21
20-16	Port Maintenance Block Header 0 (PMBH0)	20-22
20-17	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	20-22
20-18	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	20-23
20-19	General Control Command and Status Register (GCCSR)	20-23
20-20	Link Maintenance Request Command and Status Register (LMREQCSR)	20-24
20-21	Link Maintenance Response Command and Status Register (LMRESPCSR)	20-25
20-22	Local ackID Status Command and Status Register (LASCSR)	20-26

Figures

Figure Number	Title	Page Number
20-23	Error and Status Command and Status Register (ESCSR)	20-26
20-24	Control Command and Status Register (CCSR)	20-28
20-25	Error Reporting Block Header (ERBH)	20-30
20-26	Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)	20-31
20-27	Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)	20-32
20-28	Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)	20-34
20-29	Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)	20-34
20-30	Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)	20-35
20-31	Error Detect Command and Status Register (EDCSR)	20-36
20-32	Error Rate Enable Command and Status Register (ERECSR)	20-37
20-33	Error Capture Attributes Command and Status Register (ECACSR)	20-37
20-34	Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)	20-38
20-35	Packet Error Capture Command and Status Register 1 (PECCSR1)	20-39
20-36	Packet Error Capture Command and Status Register 2 (PECCSR2)	20-39
20-37	Packet Error Capture Command and Status Register 3 (PECCSR3)	20-40
20-38	Error Rate Command and Status Register (ERCSR)	20-40
20-39	Error Rate Threshold Command and Status Register (ERTCSR)	20-41
20-40	Logical Layer Configuration Register (LLCR)	20-42
20-41	Error/Port-Write Interrupt Status Register (EPWISR)	20-43
20-42	Logical Retry Error Threshold Configuration Register (LRETCR)	20-43
20-43	Physical Retry Error Threshold Configuration Register (PRETCR)	20-44
20-44	Alternate Device ID Command and Status Register (ADIDCSR)	20-45
20-45	Accept-All Configuration Register (AACR)	20-45
20-46	Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)	20-46
20-47	Implementation Error Command and Status Register (IECSR)	20-46
20-48	Physical Configuration Register (PCR)	20-47
20-49	Serial Link Command and Status Register (SLCSR)	20-47
20-50	Serial Link Error Injection Configuration Register (SLEICR)	20-48
20-51	IP Block Revision Register 1 (IPBRR1)	20-49
20-52	IP Block Revision Register 2 (IPBRR2)	20-49
20-53	Example of Attribute Aliasing	20-51
20-54	Example of Multi-Targeting	20-52
20-55	RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)	20-52
20-56	RapidIO Outbound Window Translation Extended Address Registers 0–8	20-53
20-57	RapidIO Outbound Window Base Address Registers 1–8	20-54
20-58	RapidIO Outbound Window Attributes Registers 0–8	20-54
20-59	RapidIO Outbound Window Segment 1–3 Registers 1–8 (ROWS*n*R*n*)	20-56

Figures

Figure Number	Title	Page Number
20-60	RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)	20-58
20-61	RapidIO Inbound Window Base Address Registers1–4	20-58
20-62	Outbound Message *n* Mode Registers (OM*n*MR)	20-61
20-63	Outbound Message *n* Status Registers (OM*n*SR)	20-63
20-64	Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR)	20-65
20-65	Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)	20-65
20-66	Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (EOM*n*DQEPAR)	20-66
20-67	Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (OM*n*DQEPAR)	20-67
20-68	Extended Outbound Message *n* Source Address Registers (EOM*n*SAR)	20-67
20-69	Outbound Message *n* Source Address Registers (OM*n*SAR)	20-67
20-70	Outbound Message *n* Destination Port Registers (OM*n*DPR)	20-68
20-71	Outbound Message *n* Destination Attributes Registers (OM*n*DATR)	20-69
20-72	Outbound Message *n* Double Word Count Registers (OM*n*DCR)	20-70
20-73	Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)	20-70
20-74	Outbound Message *n* Multicast Group Registers (OM*n*MGR)	20-71
20-75	Outbound Message *n* Multicast List Registers (OM*n*MLR)	20-72
20-76	Inbound Message *n* Mode Registers (IMnMR)	20-72
20-77	Inbound Message *n* Status Registers (IMnSR)	20-74
20-78	Extended Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (EIMnFQDPAR)	20-75
20-79	Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (IMnFQDPAR)	20-76
20-80	Extended Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (EIMnFQEPAR)	20-77
20-81	Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (IMnFQEPAR)	20-77
20-82	Inbound Message *n* Maximum Interrupt Report Interval Registers (IMnMIRIR)	20-78
20-83	Outbound Mode Register (ODMR)	20-78
20-84	Outbound Doorbell Status Register (ODSR)	20-79
20-85	Outbound Doorbell Destination Port Registers (ODDPR)	20-79
20-86	Outbound Doorbell Destination Attributes Register (ODDATR)	20-80
20-87	Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)	20-81
20-88	Inbound Doorbell Mode Register (IDMR)	20-82
20-89	Inbound Doorbell Status Register (IDSR)	20-83
20-90	Extended Inbound Doorbell Queue Dequeue Pointer Address Registers (EIDQDPAR)	20-84

Figures

Figure Number	Title	Page Number
20-91	Inbound Doorbell Queue Dequeue Pointer Address Registers (IDQDPAR)	20-85
20-92	Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR)	20-86
20-93	Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)	20-86
20-94	Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)	20-86
20-95	Inbound Port-Write Mode Register (IPWMR)	20-87
20-96	Inbound Port-Write Status Register (IPWSR)	20-88
20-97	Extended Port-Write Queue Base Address Register (EIPWQBAR)	20-88
20-98	Inbound Port-Write Queue Base Address Register (IPWQBAR)	20-89
20-99	Outbound Frame Queue Structure	20-150
20-100	Descriptor Dequeue Pointer and Descriptor	20-153
20-101	Inbound Message Structure	20-157
20-102	Inbound Doorbell Queue and Pointer Structure	20-169
20-103	Doorbell Entry Format	20-176
20-104	Inbound Port-Write Structure	20-182
21-1	PCI Express Controller Block Diagram	21-2
21-2	PCI Express Configuration Address Register (PEX_CONFIG_ADDR)	21-10
21-3	PCI Express Configuration Data Register (PEX_CONFIG_DATA)	21-11
21-4	PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)	21-11
21-5	PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)	21-12
21-6	PCI Express Configuration Register (PEX_CONFIG)	21-13
21-7	PCI Express PME and Message Detect Register (PEX_PME_MES_DR)	21-13
21-8	PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)	21-15
21-9	PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)	21-17
21-10	PCI Express Power Management Command Register (PEX_PMCR)	21-18
21-11	IP Block Revision Register 1	21-19
21-12	IP Block Revision Register 2	21-19
21-13	RC Outbound Transaction Flow	21-20
21-14	PCI Express Outbound Translation Address Registers (PEXOTAR*n*)	21-21
21-15	PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)	21-21
21-16	PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)	21-22
21-17	PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)	21-22
21-18	PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR*n*)	21-23
21-19	RC Inbound Transaction Flow	21-26
21-20	PCI Express Inbound Translation Address Registers (PEXITAR*n*)	21-26
21-21	PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)	21-27
21-22	PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)	21-27
21-23	PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)	21-28
21-24	PCI Express Error Detect Register (PEX_ERR_DR)	21-30
21-25	PCI Express Error Interrupt Enable Register (PEX_ERR_EN)	21-32
21-26	PCI Express Error Disable Register (PEX_ERR_DISR)	21-34

Figures

Figure Number	Title	Page Number
21-27	PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)	21-36
21-28	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction	21-37
21-29	PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction	21-38
21-30	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) Internal Source, Outbound Transaction	21-39
21-31	PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) External Source, Inbound Transaction	21-39
21-32	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source, Outbound Transaction	21-40
21-33	PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source, Inbound Transaction	21-41
21-34	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) Internal Source, Outbound Transaction	21-42
21-35	PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) External Source, Inbound Transaction	21-42
21-36	PCI Express PCI-Compatible Configuration Header Common Registers	21-45
21-37	PCI Express Vendor ID Register	21-46
21-38	PCI Express Device ID Register	21-46
21-39	PCI Express Command Register	21-47
21-40	PCI Express Status Register	21-48
21-41	PCI Express Revision ID Register	21-49
21-42	PCI Express Class Code Register	21-50
21-43	PCI Express Bus Cache Line Size Register	21-50
21-44	PCI Express Bus Latency Timer Register	21-51
21-45	PCI Express Bus Latency Timer Register	21-51
21-46	PCI Express PCI-Compatible Configuration Header—Type 0	21-52
21-47	PCI Express Base Address Register 0 (PEXCSRBAR)	21-53
21-48	32-Bit Memory Base Address Register (BAR1)	21-53
21-49	64-Bit Low Memory Base Address Register	21-54
21-50	64-Bit High Memory Base Address Register	21-54
21-51	PCI Express Subsystem Vendor ID Register	21-55
21-52	PCI Express Subsystem ID Register	21-55
21-53	Capabilities Pointer Register	21-56
21-54	PCI Express Interrupt Line Register	21-56
21-55	PCI Express Interrupt Pin Register	21-57
21-56	PCI Express Maximum Grant Register (MAX_GNT)	21-57
21-57	PCI Express Maximum Latency Register (MAX_LAT)	21-58
21-58	PCI Express PCI-Compatible Configuration Header—Type 1	21-58
21-59	PCI Express Base Address Register 0 (PEXCSRBAR)	21-59

Figures

Figure Number	Title	Page Number
21-60	PCI Express Primary Bus Number Register	21-59
21-61	PCI Express Secondary Bus Number Register	21-60
21-62	PCI Express Subordinate Bus Number Register	21-60
21-63	PCI Express I/O Base Register	21-61
21-64	PCI Express I/O Limit Register	21-61
21-65	PCI Express Secondary Status Register	21-62
21-66	PCI Express Memory Base Register	21-63
21-67	PCI Express Memory Limit Register	21-63
21-68	PCI Express Prefetchable Memory Base Register	21-64
21-69	PCI Express Prefetchable Memory Limit Register	21-64
21-70	PCI Express Prefetchable Base Upper 32 Bits Register	21-65
21-71	PCI Express Prefetchable Limit Upper 32 Bits Register	21-65
21-72	PCI Express I/O Base Upper 16 Bits Register	21-65
21-73	PCI Express I/O Limit Upper 16 Bits Register	21-66
21-74	Capabilities Pointer Register	21-66
21-75	PCI Express Interrupt Line Register	21-67
21-76	PCI Express Interrupt Pin Register	21-67
21-77	PCI Express Bridge Control Register	21-68
21-78	PCI Compatible Device-Specific Configuration Space	21-69
21-79	PCI Express Power Management Capability ID Register	21-70
21-80	PCI Express Power Management Capabilities Register	21-70
21-81	PCI Express Power Management Status and Control Register	21-71
21-82	PCI Express Power Management Data Register	21-71
21-83	PCI Express Capability ID Register	21-72
21-84	PCI Express Capabilities Register	21-72
21-85	PCI Express Device Capabilities Register	21-73
21-86	PCI Express Device Control Register	21-73
21-87	PCI Express Device Status Register	21-74
21-88	PCI Express Link Capabilities Register	21-75
21-89	PCI Express Link Control Register	21-75
21-90	PCI Express Link Status Register	21-76
21-91	PCI Express Slot Capabilities Register	21-77
21-92	PCI Express Slot Control Register	21-78
21-93	PCI Express Slot Status Register	21-78
21-94	PCI Express Root Control Register	21-79
21-95	PCI Express Root Status Register	21-79
21-96	PCI Express Capability ID Register	21-80
21-97	PCI Express MSI Message Control Register	21-80
21-98	PCI Express MSI Message Address Register	21-81
21-99	PCI Express MSI Message Upper Address Register	21-81
21-100	PCI Express MSI Message Data Register	21-82

Figures

Figure Number	Title	Page Number
21-101	PCI Express Extended Configuration Space	21-83
21-102	PCI Express Advanced Error Reporting Capability ID Register	21-84
21-103	PCI Express Uncorrectable Error Status Register	21-84
21-104	PCI Express Uncorrectable Error Mask Register	21-85
21-105	PCI Express Uncorrectable Error Severity Register	21-86
21-106	PCI Express Correctable Error Status Register	21-87
21-107	PCI Express Correctable Error Mask Register	21-87
21-108	PCI Express Advanced Error Capabilities and Control Register	21-88
21-109	PCI Express Header Log Register	21-89
21-110	PCI Express Root Error Command Register	21-90
21-111	PCI Express Root Error Status Register	21-90
21-112	PCI Express Correctable Error Source ID Register	21-91
21-113	PCI Express Correctable Error Source ID Register	21-91
21-114	PCI Express LTSSM State Status Register (PEX_LTSSM_STAT)	21-92
21-115	PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO)	21-94
21-116	PCI Express Power Management Timer Register (PEX_PM_TIMER)	21-94
21-117	PCI Express PME Time-Out Register (PEX_PME_TIMEOUT)	21-95
21-118	PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE)	21-96
21-119	PCI Express Configuration Ready Register (PEX_CFG_READY)	21-96
21-120	PCI Express PME_To_Ack Timeout Register (PEX_PME_TO_ACK_TOR)	21-97
21-121	PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK)	21-98
21-122	Requestor/Completer Relationship	21-98
21-123	PCI Express High-Level Layering	21-99
21-124	PCI Express Packet Flow	21-99
21-125	Address Invariant Byte Ordering—4 Bytes Outbound	21-101
21-126	Address Invariant Byte Ordering—4 Bytes Inbound	21-101
21-127	Address Invariant Byte Ordering—8 Bytes Outbound	21-101
21-128	Address Invariant Byte Ordering—2 bytes Inbound	21-102
21-129	PEX_CONFIG_DATA Byte Ordering	21-102
21-130	PCI Express Error Classification	21-108
21-131	PCI Express Device Error Signaling Flowchart	21-109
21-132	$\overline{\text{WAKE}}$ Generation Example	21-116
22-1	GPIO Module Block Diagram	22-1
22-2	GPIO Direction Register (GPDIR)	22-2
22-3	GPIO Open Drain Register (GPODR)	22-3
22-4	GPIO Data Register (GPDAT)	22-3
22-5	GPIO Interrupt Event Register (GPIER)	22-4
22-6	GPIO Interrupt Mask Register (GPIMR)	22-4
22-7	GPIO Interrupt Control Register (GPICR)	22-5
23-1	POR PLL Status Register (PORPLLSR)	23-5
23-2	POR Boot Mode Status Register (PORBMSR)	23-6

Figures

Figure Number	Title	Page Number
23-3	POR I/O Impedance Status and Control Register (PORIMPSCR)	23-7
23-4	POR Device Status Register (PORDEVSR)	23-8
23-5	POR Debug Mode Status Register (PORDBGMSR)	23-11
23-6	POR Device Status Register 2 (PORDEVSR2)	23-12
23-7	POR Configuration Register (GPPORCR)	23-13
23-8	Alternate Function Pin Multiplex Control Register (PMUXCR)	23-13
23-9	Device Disable Register (DEVDISR)	23-14
23-10	Power Management Control & Status Register (POWMGTCSR)	23-17
23-11	Machine Check Summary Register (MCPSUMR)	23-19
23-12	Reset Request Status and Control Register (RSTRSCR)	23-20
23-13	Checkstop Status and Control Register (ECTRSTCR)	23-20
23-14	Checkstop Status and Control Register (AUTORSTSR)	23-21
23-15	Processor Version Register (PVR)	23-22
23-16	System Version Register (SVR)	23-23
23-17	Reset Control Register (RSTCR)	23-23
23-18	LBC Voltage Select Control Register (LBCVSELCR)	23-24
23-19	DDR Clock Disable Register (DDRCLKDR)	23-25
23-20	Clock Out Control Register (CLKOCR)	23-26
23-21	SerDes1 Control Register (SRDS1CR)	23-27
23-22	SerDes2 Control Register (SRDS2CR)	23-29
23-23	e500 Cores Power Management State Diagram	23-31
23-24	MPC8572E Power Management Handshaking Signals	23-35
24-1	Performance Monitor Block Diagram	24-2
24-2	Performance Monitor Global Control Register (PMGC0)	24-6
24-3	Performance Monitor Local Control Register A0 (PMLCA0)	24-7
24-4	Performance Monitor Local Control A Registers (PMLCA1–PMLCA11)	24-7
24-5	Performance Monitor Local Control Register B0 (PMLCB0)	24-8
24-6	Performance Monitor Local Control Register B (PMLCB1–PMLCB11)	24-9
24-7	Performance Monitor Counter Register 0 (PMC0)	24-11
24-8	Performance Monitor Counter Register (PMC1–PMC11)	24-11
24-9	Duration Threshold Event Sequence Timing Diagram	24-13
24-10	Burst Size, Distance, Granularity, and Burstiness Counting	24-14
24-11	Burstiness Counting Timing Diagram	24-16
25-1	Debug and Watchpoint Monitor Block Diagram	25-2
25-2	Watchpoint Monitor Control Register 0 (WMCR0)	25-11
25-3	Watchpoint Monitor Control Register 1 (WMCR1)	25-12
25-4	Watchpoint Monitor Address Register (WMAR)	25-13
25-5	Watchpoint Monitor Address Mask Register (WMAMR)	25-14
25-6	Watchpoint Monitor Transaction Mask Register (WMTMR)	25-14
25-7	Watchpoint Monitor Status Register (WMSR)	25-16
25-8	Trace Buffer Control Register 0 (TBCR0)	25-16

Figures

Figure Number	Title	Page Number
25-9	Trace Buffer Control Register 1 (TBCR1)	25-18
25-10	Trace Buffer Address Register (TBAR)	25-19
25-11	Trace Buffer Address Mask Register (TBAMR)	25-19
25-12	Trace Buffer Transaction Mask Register (TBTMR)	25-20
25-13	Trace Buffer Status Register (TBSR)	25-20
25-14	Trace Buffer Access Control Register (TBACR))	25-21
25-15	Trace Buffer Read High Register (TBADHR)	25-22
25-16	Trace Buffer Access Data Register (TBADR)	25-23
25-17	Programmed Context ID Register (PCIDR)	25-23
25-18	Current Context ID Register (CCIDR)	25-24
25-19	Trigger Out Source Register (TOSR)	25-24
25-20	e500 Coherency Module Dispatch (CMD) Trace Buffer Entry	25-29
25-21	PCI Express Trace Buffer Entry	25-29

Figures

Figure Number	Title	Page Number

Tables

Table Number	Title	Page Number
i	Acronyms and Abbreviated Terms	cxxxii
1-1	Supported eTSEC1 and eTSEC2 Configurations	1-8
1-2	Supported eTSEC3 and eTSEC4 Configurations	1-8
2-1	Target Interface Codes	2-1
2-2	Local Access Windows Example	2-2
2-3	Format of ATMU Window Definitions	2-4
2-4	Local Access Register Memory Map	2-5
2-5	LAIPBRR1 Field Descriptions	2-6
2-6	LAIPBRR2 Field Descriptions	2-7
2-7	LAWBAR*n* Field Descriptions	2-7
2-8	LAWAR*n* Field Descriptions	2-8
2-9	Overlapping Local Access Windows	2-9
2-10	Local Memory Configuration, Control, and Status Register Summary	2-13
2-11	Memory Map	2-18
3-1	MPC8572E Signal Reference by Functional Block	3-6
3-2	MPC8572E Signal Reference by Signal Name	3-14
3-3	MPC8572E Reset Configuration Signals	3-23
4-1	Signal Summary	4-1
4-2	System Control Signals—Detailed Signal Descriptions	4-2
4-3	Clock Signals—Detailed Signal Descriptions	4-3
4-4	Local Configuration Control Register Map	4-4
4-5	CCSRBAR Bit Settings	4-6
4-6	ALTCBAR Bit Settings	4-7
4-7	ALTCAR Bit Settings	4-7
4-8	BPTR Bit Settings	4-9
4-9	CCB Clock PLL Ratio	4-13
4-10	DDR Complex Clock PLL Ratio	4-14
4-11	e500 Core0 Clock PLL Ratios	4-14
4-12	e500 Core1 Clock PLL Ratios	4-15
4-13	Boot ROM Location	4-15
4-14	Host/Agent Configuration	4-17
4-15	I/O Port Selection	4-18
4-16	CPU Boot Configuration	4-20
4-17	Boot Sequencer Configuration	4-20
4-18	DDR DRAM Type	4-21
4-19	FEC Configuration	4-21
4-20	eTSEC1 SGMII Mode Configuration	4-22
4-21	eTSEC2 SGMII Mode Configuration	4-22

Tables

Table Number	Title	Page Number
4-22	eTSEC3 SGMII Mode Configuration	4-22
4-23	eTSEC4 SGMII Mode Configuration	4-22
4-24	SGMII SerDes Reference Clock Configuration	4-23
4-25	eTSEC1/eTSEC2 Width Configuration	4-23
4-26	eTSEC3/eTSEC4 Width Configuration	4-24
4-27	eTSEC1 Protocol Configuration	4-25
4-28	eTSEC2 Protocol Configuration	4-25
4-29	eTSEC3 Protocol Configuration	4-26
4-30	eTSEC4 Protocol Configuration	4-26
4-31	RapidIO Device ID	4-27
4-32	RapidIO System Size	4-27
4-33	Memory Debug Configuration	4-28
4-34	DDR Debug Configuration	4-28
4-35	General-Purpose POR Configuration	4-29
4-36	eLBC FCM ECC Configuration	4-29
4-37	SerDes1 Enable POR Configuration	4-29
4-38	SGMII SerDes Enable POR Configuration	4-30
4-39	Engineering Use POR Config Signal	4-30
4-40	High Speed Interface 1 Clocking	4-31
5-1	Device Revision Level Cross-Reference	5-4
5-2	Performance Monitor Instructions	5-11
5-3	Cache Locking Instructions	5-11
5-4	Scalar and Vector Embedded Floating-Point Instructions	5-12
5-5	BTB Locking Instructions	5-13
5-6	Interrupt Registers	5-20
5-7	Interrupt Vector Registers and Exception Conditions	5-21
5-8	Differences Between the e500 Core and the PowerQUICC III Core Implementation	5-31
6-1	Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation)	6-4
6-2	Additional SPRs (by SPR Abbreviation)	6-7
6-3	XER Field Description	6-9
6-4	BI Operand Settings for CR Fields	6-9
6-5	CR0 Bit Descriptions	6-11
6-6	MSR Field Descriptions	6-12
6-7	PVR Field Descriptions	6-14
6-8	SVR Field Descriptions	6-14
6-9	TCR Field Descriptions	6-15
6-10	TSR Field Descriptions	6-16
6-11	IVOR Assignments	6-18
6-12	ESR Field Descriptions	6-19
6-13	MCSR Field Descriptions	6-21
6-14	SPR Assignments	6-22

Tables

Table Number	Title	Page Number
6-15	BBEAR Field Descriptions	6-23
6-16	BBTAR Field Descriptions	6-23
6-17	BUCSR Field Descriptions	6-24
6-18	HID0 Field Descriptions	6-25
6-19	HID1 Field Descriptions	6-26
6-20	L1CSR0 Field Descriptions	6-28
6-21	L1CSR1 Field Descriptions	6-29
6-22	L1CFG0 Field Descriptions	6-30
6-23	L1CFG1 Field Descriptions	6-31
6-24	MMUCSR0 Field Descriptions	6-32
6-25	MMUCFG Field Descriptions	6-33
6-26	TLB0CFG Field Descriptions	6-33
6-27	TLB1CFG Field Descriptions	6-34
6-28	MAS0 Field Descriptions—MMU Read/Write and Replacement Control	6-35
6-29	MAS1 Field Descriptions—Descriptor Context and Configuration Control	6-36
6-30	MAS2 Field Descriptions—EPN and Page Attributes	6-36
6-31	MAS3 Field Descriptions–RPN and Access Control	6-37
6-32	MAS4 Field Descriptions—Hardware Replacement Assist Configuration	6-38
6-33	MAS6—TLB Search Context Register 0	6-39
6-34	MAS 7 Field Descriptions—High Order RPN	6-39
6-35	DBCR0 Field Descriptions	6-40
6-36	DBCR1 Field Descriptions	6-41
6-37	DBCR2 Field Descriptions	6-42
6-38	DBSR Field Descriptions	6-44
6-39	SPEFSCR Field Descriptions	6-45
6-40	ACC FIeld Descriptions	6-47
6-41	Supervisor-Level PMRs (PMR[5] = 1)	6-48
6-42	User-Level PMRs (PMR[5] = 0) (Read Only)	6-48
6-43	PMGC0 Field Descriptions	6-49
6-44	PMLCa0–PMLCa3 Field Descriptions	6-50
6-45	PMLCb0–PMLCb3 Field Descriptions	6-51
6-46	PMC0–PMC3 Field Descriptions	6-52
7-1	Available L2 Cache/SRAM Configurations	7-3
7-2	Way Selection for SRAM Accesses	7-6
7-3	L2/SRAM Memory-Mapped Registers	7-9
7-4	L2CTL Field Descriptions	7-10
7-5	L2CWAP Field Descriptions	7-14
7-6	L2CEWAR*n* Field Descriptions	7-15
7-7	L2CEWAREA*n* Field Descriptions	7-16
7-8	L2CEWCR*n* Field Descriptions	7-17
7-9	L2SRBAR*n* Field Descriptions	7-18

Tables

Table Number	Title	Page Number
7-10	L2SRBAREA*n* Field Descriptions	7-19
7-11	L2ERRINJHI Field Description	7-20
7-12	L2ERRINJLO Field Description	7-21
7-13	L2ERRINJCTL Field Descriptions	7-21
7-14	L2CAPTDATAHI Field Description	7-22
7-15	L2CAPTDATALO Field Description	7-22
7-16	L2CAPTECC Field Descriptions	7-23
7-17	L2ERRDET Field Descriptions	7-23
7-18	L2ERRDIS Field Descriptions	7-24
7-19	L2ERRINTEN Field Descriptions	7-25
7-20	L2ERRATTR Field Descriptions	7-25
7-21	L2ERRADDRL Field Description	7-26
7-22	L2ERRADDRH Field Description	7-27
7-23	L2ERRCTL Field Descriptions	7-27
7-24	Fastest Read Timing—Hit in L2	7-29
7-25	PLRU Bit Update Algorithm	7-34
7-26	PLRU-Based Victim Selection Mechanism	7-35
7-27	L2 Cache States	7-37
7-28	State Transitions Due to Core-Initiated Transactions	7-37
7-29	State Transitions Due to System-Initiated Transactions	7-40
7-30	L2 Cache ECC Syndrome Encoding	7-42
7-31	L2 Cache ECC Syndrome Encoding (Check Bits)	7-43
8-1	ECM Memory Map	8-3
8-2	EEBACR Field Descriptions	8-4
8-3	EEBPCR Field Descriptions	8-4
8-4	EIPBRR1 Field Descriptions	8-6
8-5	EIPBRR2 Field Descriptions	8-6
8-6	EEDR Field Descriptions	8-7
8-7	EEER Field Descriptions	8-7
8-8	EEATR Field Descriptions	8-8
8-9	EELADR Field Descriptions	8-9
8-10	EEHADR Field Descriptions	8-9
9-1	DDR Memory Interface Signal Summary	9-4
9-2	Memory Address Signal Mappings	9-6
9-3	Memory Interface Signals—Detailed Signal Descriptions	9-7
9-4	Clock Signals—Detailed Signal Descriptions	9-11
9-5	DDR Memory Controller Memory Map	9-12
9-6	CS*n*_BNDS Field Descriptions	9-14
9-7	CS*n*_CONFIG Field Descriptions	9-15
9-8	CSn_CONFIG_2 Field Descriptions	9-17
9-9	TIMING_CFG_3 Field Descriptions	9-18

Tables

Table Number	Title	Page Number
9-10	TIMING_CFG_0 Field Descriptions	9-19
9-11	TIMING_CFG_1 Field Descriptions	9-21
9-12	TIMING_CFG_2 Field Descriptions	9-23
9-13	DDR_SDRAM_CFG Field Descriptions	9-25
9-14	DDR_SDRAM_CFG_2 Field Descriptions	9-29
9-15	DDR_SDRAM_MODE Field Descriptions	9-31
9-16	DDR_SDRAM_MODE_2 Field Descriptions	9-31
9-17	DDR_SDRAM_MD_CNTL Field Descriptions	9-32
9-18	Settings of DDR_SDRAM_MD_CNTL Fields	9-33
9-19	DDR_SDRAM_INTERVAL Field Descriptions	9-34
9-20	DDR_DATA_INIT Field Descriptions	9-35
9-21	DDR_SDRAM_CLK_CNTL Field Descriptions	9-35
9-22	DDR_INIT_ADDR Field Descriptions	9-36
9-23	DDR_INIT_EXT_ADDR Field Descriptions	9-37
9-24	TIMING_CFG_4 Field Descriptions	9-37
9-25	TIMING_CFG_5 Field Descriptions	9-39
9-26	DDR_ZQ_CNTL Field Descriptions	9-41
9-27	DDR_WRLVL_CNTL Field Descriptions	9-42
9-28	DDRDSR_1 Field Descriptions	9-45
9-29	DDRDSR_2 Field Descriptions	9-46
9-30	DDRCDR_1 Field Descriptions	9-48
9-31	DDRCDR_2 Field Descriptions	9-49
9-32	DDR_IP_REV1 Field Descriptions	9-50
9-33	DDR_IP_REV2 Field Descriptions	9-50
9-34	DATA_ERR_INJECT_HI Field Descriptions	9-51
9-35	DATA_ERR_INJECT_LO Field Descriptions	9-51
9-36	ERR_INJECT Field Descriptions	9-52
9-37	CAPTURE_DATA_HI Field Descriptions	9-53
9-38	CAPTURE_DATA_LO Field Descriptions	9-53
9-39	CAPTURE_ECC Field Descriptions	9-54
9-40	ERR_DETECT Field Descriptions	9-54
9-41	ERR_DISABLE Field Descriptions	9-55
9-42	ERR_INT_EN Field Descriptions	9-56
9-43	CAPTURE_ATTRIBUTES Field Descriptions	9-57
9-44	CAPTURE_ADDRESS Field Descriptions	9-59
9-45	CAPTURE_EXT_ADDRESS Field Descriptions	9-59
9-46	ERR_SBE Field Descriptions	9-60
9-47	Byte Lane to Data Relationship	9-64
9-48	Supported DDR1 SDRAM Device Configurations	9-65
9-49	Supported DDR2 SDRAM Device Configurations	9-65
9-50	Supported DDR3 SDRAM Device Configurations	9-66

Tables

Table Number	Title	Page Number
9-51	DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled	9-66
9-52	DDR2 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled	9-67
9-53	Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Disabled	9-68
9-54	Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks with Partial Array Self Refresh Disabled	9-69
9-55	DDR2 Address Multiplexing with Partial Array Self Refresh Enabled	9-69
9-56	Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled	9-70
9-57	DDR SDRAM Command Table	9-72
9-58	DDR SDRAM Interface Timing Intervals	9-72
9-59	DDR SDRAM Power-Saving Modes Refresh Configuration	9-80
9-60	Memory Controller–Data Beat Ordering	9-82
9-61	DDR SDRAM ECC Syndrome Encoding	9-83
9-62	DDR SDRAM ECC Syndrome Encoding (Check Bits)	9-85
9-63	Memory Controller Errors	9-86
9-64	Memory Interface Configuration Register Initialization Parameters	9-86
9-65	Programming Differences Between Memory Types	9-88
10-1	Processor Interrupts Generated Outside the Core—Types and Sources	10-4
10-2	Internal Interrupt Assignments	10-7
10-3	Interrupt Signals—Detailed Signal Descriptions	10-9
10-4	PIC Register Address Map	10-10
10-5	BRR1 Field Descriptions	10-21
10-6	BRR2 Field Descriptions	10-21
10-7	FRR Field Descriptions	10-22
10-8	GCR Field Descriptions	10-22
10-9	VIR Field Descriptions	10-23
10-10	PIR Field Descriptions	10-24
10-11	IPIVPR*n* Field Descriptions	10-24
10-12	SVR Field Descriptions	10-25
10-13	TFRR*x* Field Descriptions	10-26
10-14	GTCCR*xn* Field Descriptions	10-26
10-15	GTBCR*xn* Field Descriptions	10-27
10-16	GTVPR*xn* Field Descriptions	10-27
10-17	GTDR*xn* Field Descriptions	10-28
10-18	Parameters for Hourly Interrupt Timer Cascade Example	10-29
10-19	TCR*x* Field Descriptions	10-30
10-20	ERQSR Field Descriptions	10-31
10-21	IRQSR0 Field Descriptions	10-32

Tables

Table Number	Title	Page Number
10-22	IRQSR1 Field Descriptions	10-32
10-23	IRQSR2 Field Descriptions	10-33
10-24	CISR0 Field Descriptions	10-33
10-25	CISR1 Field Descriptions	10-34
10-26	CISR2 Field Descriptions	10-34
10-27	PM*n*MR0 Field Descriptions	10-35
10-28	PM*n*MR1 Field Descriptions	10-36
10-29	PM*n*MR2 Field Descriptions	10-36
10-30	MSGR*n* Field Descriptions	10-37
10-31	MER Field Descriptions	10-38
10-32	MSR Field Descriptions	10-38
10-33	MSIR*n* Field Descriptions	10-39
10-34	MSISR Field Descriptions	10-39
10-35	MSIIR Field Descriptions	10-40
10-36	MSIVPRn Field Descriptions	10-41
10-37	MSIDR*n* Field Descriptions	10-41
10-38	EIVPR*n* Field Descriptions	10-43
10-39	EIDR*n* Field Descriptions	10-44
10-40	IIVPR*n* Field Descriptions	10-46
10-41	IIDR*n* Field Descriptions	10-47
10-42	MIVPR*n* Field Descriptions	10-47
10-43	MIDR*n* Field Descriptions	10-48
10-44	Per-CPU Registers—Private Access Address Offsets	10-49
10-45	IPIDR*n* Field Descriptions	10-51
10-46	CTPRn Field Descriptions	10-51
10-47	WHOAMIn Field Descriptions	10-52
10-48	IACKn Field Descriptions	10-53
10-49	EOIn Field Descriptions	10-53
10-50	PCI Express INTx/IRQ*n* Sharing	10-58
11-1	Example Descriptor	11-5
11-2	SEC Address Map	11-14
11-3	SEC Detailed Address Map	11-15
11-4	Header Dword Bit Definitions	11-24
11-5	Header Dword Writeback Bit Definitions	11-25
11-6	EU_SEL0 and EU_SEL1 Values	11-25
11-7	Descriptor Types	11-26
11-8	Pointer Dword Field Definitions	11-28
11-9	Link Table Field Definitions	11-29
11-10	Descriptor Format Summary	11-32
11-11	Channel Configuration Register Signals	11-41
11-12	Writeback Options	11-43

Tables

Table Number	Title	Page Number
11-14	Channel Status Register Signals	11-44
11-13	Done Interrupt Options	11-44
11-15	Channel Status Register Error Field Definitions	11-46
11-16	Current Descriptor Pointer Register Signals	11-47
11-17	Fetch FIFO Field Descriptions	11-48
11-18	Channel Assignment Value	11-53
11-19	Field Names in Interrupt Enable, Interrupt Status, and Interrupt Clear Registers	11-54
11-20	IP Block Revision Register Fields	11-59
11-21	Master Control Register Signals	11-60
11-22	AESU Mode Register Field Descriptions	11-63
11-23	AES Cipher Modes	11-66
11-24	Use of Data Size Register	11-68
11-25	AESU Reset Control Register Field Descriptions	11-69
11-26	AESU Status Register Field Descriptions	11-70
11-27	AESU Interrupt Status Register Field Descriptions	11-71
11-28	AESU Interrupt Mask Register Field Descriptions	11-73
11-29	AESU Context Registers for Confidentiality Modes	11-76
11-30	AESU Context Registers for Integrity Modes	11-77
11-31	AESU Context Registers for Modes Providing Confidentiality and Integrity	11-80
11-32	GCM Cipher Mode Auxiliary Bit Definitions	11-85
11-33	GCM Encryption Context	11-88
11-34	GCM Decryption Context	11-89
11-35	GCM w/ICV Context	11-90
11-36	GCM-GHASH Context	11-91
11-37	AFEU Mode Register Field Descriptions	11-93
11-38	AFEU Reset Control Register Field Descriptions	11-95
11-39	AFEU Status Register Field Descriptions	11-96
11-40	AFEU Interrupt Status Register	11-97
11-41	AFEU Interrupt Mask Register	11-99
11-42	CRCU Mode Register Bit Definitions	11-103
11-43	CRCU Reset Control Register Field Descriptions	11-105
11-44	CRCU Status Register Bit Definitions	11-106
11-45	CRCU Interrupt Status Register Bit Definitions	11-107
11-46	CRCU Interrupt Mask Register Bit Definitions	11-109
11-47	DEU Mode Register Field Descriptions	11-113
11-48	DEU Cipher Modes	11-113
11-49	DEU Key Size Register Field Descriptions	11-114
11-50	DEU Reset Control Register Field Descriptions	11-115
11-51	DEU Status Register	11-116
11-52	DEU Interrupt Status Register Field Descriptions	11-117

Tables

Table Number	Title	Page Number
11-53	DEU Interrupt Mask Register Field Descriptions	11-119
11-54	KEU Mode Register Field Descriptions	11-123
11-55	KEU Reset Control Register Field Descriptions	11-126
11-56	KEU Status Register Fields Description	11-126
11-57	KEU Interrupt Status Register Signals Description	11-128
11-58	KEU Interrupt Mask Register Fields Description	11-129
11-59	KEU IV_1 Register Fields Description	11-132
11-60	MDEU Mode Register in Old Configuration	11-136
11-61	MDEU Mode Register in New Configuration	11-138
11-62	Mode Register—HMAC or SSL-MAC Generated by Single Descriptor	11-139
11-63	Mode Register—HMAC Generated Across a Sequence of Descriptors	11-140
11-64	MDEU Reset Control Register Field Descriptions	11-141
11-65	MDEU Status Register Field Descriptions	11-142
11-66	MDEU Interrupt Status Register Field Descriptions	11-144
11-67	MDEU Interrupt Mask Register Field Descriptions	11-145
11-68	ROUTINE Field Description	11-151
11-69	PKEU Reset Control Register Field Descriptions	11-154
11-70	PKEU Status Register Field Descriptions	11-155
11-71	PKEU Interrupt Status Register Field Descriptions	11-156
11-72	PKEU Interrupt Mask Register Field Descriptions	11-158
11-73	RNGU Reset Control Register Field Descriptions	11-161
11-74	RNGU Status Register Field Descriptions	11-162
11-75	RNGU Interrupt Status Register Field Descriptions	11-164
11-76	RNGU Interrupt Mask Register Field Descriptions	11-165
12-1	I^2C Interface Signal Descriptions	12-3
12-2	I^2C Interface Signal—Detailed Signal Descriptions	12-4
12-3	I^2C Memory Map	12-5
12-4	I2CADR Field Descriptions	12-6
12-5	I2CFDR Field Descriptions	12-7
12-6	I2CCR Field Descriptions	12-8
12-7	I2CSR Field Descriptions	12-9
12-8	I2CDR Field Descriptions	12-10
12-9	I2CDFSRR Field Descriptions	12-11
13-1	DUART Signals—Detailed Signal Descriptions	13-3
13-2	DUART Register Summary	13-4
13-3	URBR Field Descriptions	13-5
13-4	UTHR Field Descriptions	13-6
13-5	UDMB Field Descriptions	13-6
13-6	UDLB Field Descriptions	13-7
13-7	Baud Rate Examples	13-7
13-8	UIER Field Descriptions	13-8

Tables

Table Number	Title	Page Number
13-9	UIIR Field Descriptions	13-9
13-10	UIIR IID Bits Summary	13-9
13-11	UFCR Field Descriptions	13-10
13-12	ULCR Field Descriptions	13-12
13-13	Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]	13-13
13-14	UMCR Field Descriptions	13-13
13-15	ULSR Field Descriptions	13-14
13-16	UMSR Field Descriptions	13-15
13-17	USCR Field Descriptions	13-16
13-18	UAFR Field Descriptions	13-17
13-19	UDSR Field Descriptions	13-17
13-20	UDSR[TXRDY] Set Conditions	13-18
13-21	UDSR[TXRDY] Cleared Conditions	13-18
13-22	UDSR[RXRDY] Set Conditions	13-18
13-23	UDSR[RXRDY] Cleared Conditions	13-18
14-1	Signal Properties—Summary	14-4
14-2	Enhanced Local Bus Controller Detailed Signal Descriptions	14-5
14-3	Enhanced Local Bus Controller Registers	14-8
14-4	BR*n* Field Descriptions	14-11
14-5	Reset value of OR0 Register	14-12
14-6	Memory Bank Sizes in Relation to Address Mask	14-13
14-7	OR*n*—GPCM Field Descriptions	14-14
14-8	OR*n*—FCM Field Descriptions	14-17
14-9	OR*n*—UPM Field Descriptions	14-20
14-10	MAR Field Descriptions	14-21
14-11	M*x*MR Field Descriptions	14-21
14-12	MRTPR Field Descriptions	14-23
14-13	MDR Field Description	14-24
14-14	LSOR Field Description	14-25
14-15	LURT Field Descriptions	14-26
14-16	LTESR Field Descriptions	14-27
14-17	LTEDR Field Descriptions	14-28
14-18	LTEIR Field Descriptions	14-29
14-19	LTEATR Field Descriptions	14-30
14-20	LTEAR Field Descriptions	14-31
14-21	LBCR Field Descriptions	14-32
14-22	LCRR Field Descriptions	14-34
14-23	FMR Field Descriptions	14-35
14-24	FIR Field Descriptions	14-37
14-25	FCR Field Descriptions	14-37
14-26	FBAR Field Descriptions	14-38

Tables

Table Number	Title	Page Number
14-27	FPAR Field Descriptions, Small Page Device (ORx[PGS] = 0)	14-39
14-28	FPAR Field Descriptions, Large Page Device (ORx[PGS] = 01)	14-39
14-29	FBCR Field Descriptions	14-40
14-30	GPCM Read Control Signal Timing	14-48
14-31	GPCM Write Control Signal Timing	14-50
14-32	Boot Bank Field Values after Reset for GPCM as Boot Controller	14-56
14-33	FCM Chip-Select to First Command Timing	14-66
14-34	FCM Command, Address, and Write Data Timing Parameters	14-67
14-35	FCM Read Data Timing Parameters	14-69
14-36	Boot Bank Field Values after Reset for FCM as Boot Controller	14-70
14-37	UPM Routines Start Addresses	14-74
14-38	RAM Word Field Descriptions	14-79
14-39	RAM Word Field Descriptions	14-79
14-40	M*x*MR Loop Field Use	14-83
14-41	UPM Address Multiplexing	14-84
14-42	Data Bus Drive Requirements For Read Cycles	14-92
14-43	FCM Register Settings for Soft Reset (ORn[PGS] = 1)	14-93
14-44	FCM Register Settings for Status Read (ORn[PGS] = 1)	14-94
14-45	FCM Register Settings for ID Read (ORn[PGS] = 1)	14-94
14-46	FCM Register Settings for Page Read (ORn[PGS] = 1)	14-95
14-47	FCM Register Settings for Block Erase (ORn[PGS] = 1)	14-96
14-48	FCM Register Settings for Page Program (ORn[PGS] = 1)	14-96
15-1	eTSEC*n* Network Interface Signal Properties	15-7
15-2	eTSEC Signals—Detailed Signal Descriptions	15-10
15-3	Module Memory Map Summary	15-15
15-4	Module Memory Map	15-16
15-5	TSEC_ID Field Descriptions	15-27
15-6	TSEC_ID2 Field Descriptions	15-28
15-7	IEVENT Field Descriptions	15-29
15-8	IMASK Field Descriptions	15-33
15-9	EDIS Field Descriptions	15-34
15-10	ECNTRL Field Descriptions	15-36
15-11	eTSEC Interface Configurations	15-37
15-12	PTV Field Descriptions	15-38
15-13	DMACTRL Field Descriptions	15-39
15-14	TBIPA Field Descriptions	15-41
15-15	TCTRL Field Descriptions	15-41
15-16	TSTAT Field Descriptions	15-43
15-17	DFVLAN Field Descriptions	15-46
15-18	TXIC Field Descriptions	15-47
15-19	TQUEUE Field Descriptions	15-48

Tables

Table Number	Title	Page Number
15-20	TR03WT Field Descriptions	15-49
15-21	TR47WT Field Descriptions	15-50
15-22	TBDBPH Field Descriptions	15-50
15-23	TBPTR*n* Field Descriptions	15-51
15-24	TBASEH Field Descriptions	15-51
15-25	TBASE0–TBASE7 Field Descriptions	15-52
15-26	TMR_TXTS*n*_ID Register Field Descriptions	15-53
15-27	TMR_TXTS*n*_H/L Register Field Descriptions	15-53
15-28	RCTRL Field Descriptions	15-54
15-29	RSTAT Field Descriptions	15-57
15-30	RXIC Field Descriptions	15-59
15-31	RQUEUE Field Descriptions	15-60
15-32	RBIFX Field Descriptions	15-62
15-33	RQFAR Field Descriptions	15-63
15-34	RQFCR Field Descriptions	15-64
15-35	RQFPR Field Descriptions	15-66
15-36	MRBLR Field Descriptions	15-68
15-37	RBDBPH Field Descriptions	15-69
15-38	RBPTR*n* Field Descriptions	15-70
15-39	RBASEH Field Descriptions	15-70
15-40	RBASE0–RBASE7 Field Descriptions	15-71
15-41	TMR_RXTS_H/L Register Field Descriptions	15-72
15-42	MACCFG1 Field Descriptions	15-75
15-43	MACCFG2 Field Descriptions	15-77
15-44	IPGIFG Field Descriptions	15-78
15-45	HAFDUP Field Descriptions	15-79
15-46	MAXFRM Descriptions	15-80
15-47	MIIMCFG Field Descriptions	15-81
15-48	MIIMCOM Descriptions	15-82
15-49	MIIMADD Field Descriptions	15-82
15-50	MIIMCON Field Descriptions	15-83
15-51	MIIMSTAT Field Descriptions	15-83
15-52	MIIMIND Field Descriptions	15-84
15-53	IFSTAT Field Descriptions	15-84
15-54	MACSTNADDR1 Field Descriptions	15-85
15-55	MACSTNADDR2 Field Descriptions	15-86
15-56	MAC*n*ADDR1 Field Descriptions	15-87
15-57	MAC01ADDR2–MAC15ADDR2 Field Descriptions	15-87
15-58	TR64 Field Descriptions	15-88
15-59	TR127 Field Descriptions	15-89
15-60	TR255 Field Descriptions	15-89

Tables

Table Number	Title	Page Number
15-61	TR511 Field Descriptions	15-90
15-62	TR1K Field Descriptions	15-90
15-63	TRMAX Field Descriptions	15-91
15-64	TRMGV Field Descriptions	15-91
15-65	RBYT Field Descriptions	15-92
15-66	RPKT Field Descriptions	15-92
15-67	RFCS Field Descriptions	15-93
15-68	RMCA Field Descriptions	15-93
15-69	RBCA Field Descriptions	15-94
15-70	RXCF Field Descriptions	15-94
15-71	RXPF Field Descriptions	15-95
15-72	RXUO Field Descriptions	15-95
15-73	RALN Field Descriptions	15-96
15-74	RFLR Field Descriptions	15-96
15-75	RCDE Field Descriptions	15-97
15-76	RCSE Field Descriptions	15-97
15-77	RUND Field Descriptions	15-98
15-78	ROVR Field Descriptions	15-98
15-79	RFRG Field Descriptions	15-99
15-80	RJBR Field Descriptions	15-99
15-81	RDRP Field Descriptions	15-100
15-82	TBYT Field Descriptions	15-100
15-83	TPKT Field Descriptions	15-101
15-84	TMCA Field Descriptions	15-101
15-85	TBCA Field Descriptions	15-102
15-86	TXPF Field Descriptions	15-102
15-87	TDFR Field Descriptions	15-103
15-88	TEDF Field Descriptions	15-103
15-89	TSCL Field Descriptions	15-104
15-90	TMCL Field Descriptions	15-104
15-91	TLCL Field Descriptions	15-105
15-92	TXCL Field Descriptions	15-105
15-93	TNCL Field Descriptions	15-106
15-94	TDRP Field Descriptions	15-106
15-95	TJBR Field Descriptions	15-107
15-96	TFCS Field Descriptions	15-107
15-97	TXCF Field Descriptions	15-108
15-98	TOVR Field Descriptions	15-108
15-99	TUND Field Descriptions	15-109
15-100	TFRG Field Descriptions	15-109
15-101	CAR1 Field Descriptions	15-110

Tables

Table Number	Title	Page Number
15-102	CAR2 Field Descriptions	15-111
15-103	CAM1 Field Descriptions	15-112
15-104	CAM2 Field Descriptions	15-114
15-105	RREJ Field Descriptions	15-115
15-106	IGADDR*n* Field Descriptions	15-116
15-107	GADDR*n* Field Descriptions	15-117
15-108	FIFOCFG Field Descriptions	15-117
15-109	ATTR Field Descriptions	15-119
15-110	ATTRELI Field Descriptions	15-120
15-111	RQPRM Field Descriptions	15-121
15-112	RFBPTR0–RFBPTR7 Field Descriptions	15-122
15-113	TMR_CTRL Register Field Descriptions	15-123
15-114	TMR_TEVENT Register Field Descriptions	15-124
15-115	TMR_TEMASK Register Definition	15-125
15-116	TMR_TEMASK Register Field Descriptions	15-125
15-117	TMR_PEVENT Register Field Descriptions	15-126
15-118	TMR_PEMASK Register Field Descriptions	15-127
15-119	TMR_STAT Register Definition	15-128
15-120	TMR_STAT Register Field Descriptions	15-128
15-121	TMR_CNT_H/L Register Field Descriptions	15-129
15-122	TMR_ADD Register Field Descriptions	15-129
15-123	TMR_ACC Register Field Descriptions	15-130
15-124	TMR_PRSC Register Field Descriptions	15-130
15-125	TMROFF_H/L Register Field Descriptions	15-131
15-126	TMR_ALARM*n*_H/L Register Field Descriptions	15-132
15-127	TMR_FIPER Register Field Descriptions	15-133
15-128	TMR_ETTS1-2_H Register Field Descriptions	15-134
15-129	TBI MII Register Set	15-135
15-130	CR Field Descriptions	15-136
15-131	SR Descriptions	15-137
15-132	ANA Field Descriptions	15-138
15-133	PAUSE Priority Resolution	15-139
15-134	ANLPBPA Field Descriptions	15-140
15-135	ANEX Field Descriptions	15-142
15-136	ANNPT Field Descriptions	15-142
15-137	ANLPANP Field Descriptions	15-143
15-138	EXST Field Descriptions	15-144
15-139	JD Field Descriptions	15-145
15-140	TBICON Field Descriptions	15-146
15-141	Ethernet Modes versus eTSEC (V = Available)	15-147
15-142	GMII, MII, and RMII Signals Multiplexing	15-153

Tables

Table Number	Title	Page Number
15-143	RGMII, TBI, and RTBI Signals Multiplexing	15-154
15-144	RGMII and RTBI Signals Multiplexing	15-155
15-145	RGMII Signals Multiplexing	15-156
15-146	Shared Signals	15-157
15-147	Valid Combinations of eTSEC Signals and Interface Modes	15-158
15-148	Signal Encoding for GMII-Style 8-Bit FIFO	15-161
15-149	Signal Encoding for Encoded 8-Bit FIFO	15-161
15-150	Signal Encoding for GMII-Style 16-Bit FIFO	15-163
15-151	Signal Encoding for Encoded 16-Bit FIFO	15-164
15-152	Steps for Minimum Register Initialization	15-165
15-153	Custom Preamble Field Descriptions	15-170
15-154	Received Preamble Field Descriptions	15-171
15-155	Flow Control Frame Structure	15-175
15-156	Non-Error Transmit Interrupts	15-177
15-157	Non-Error Receive Interrupts	15-177
15-158	Interrupt Coalescing Timing Threshold Ranges	15-178
15-159	Transmission Errors	15-179
15-160	Reception Errors	15-180
15-161	Tx Frame Control Block Description	15-183
15-162	Rx Frame Control Block Descriptions	15-185
15-163	Special Filer Rules	15-189
15-164	Receive Queue Filer Interrupt Events	15-189
15-165	Filer Table Example—802.1p Priority Filing	15-190
15-166	Filer Table Example—IP Diff-Serv Code Points Filing	15-191
15-167	Filer Table Example—TCP and UDP Port Filing	15-192
15-168	PTP Payload Special Fields	15-199
15-169	Transmit Data Buffer Descriptor (TxBD) Field Descriptions	15-203
15-170	Receive Buffer Descriptor Field Descriptions	15-206
15-171	MII Interface Mode Signal Configuration	15-208
15-172	Shared MII Signals	15-209
15-173	MII Mode Register Initialization Steps	15-209
15-174	GMII Interface Mode Signal Configuration	15-212
15-175	Shared GMII Signals	15-213
15-176	GMII Mode Register Initialization Steps	15-213
15-177	TBI Interface Mode Signal Configuration	15-216
15-178	Shared TBI Signals	15-217
15-179	TBI Mode Register Initialization Steps	15-217
15-180	RGMII Interface Mode Signal Configuration	15-220
15-181	Shared RGMII Signals	15-221
15-182	RGMII Mode Register Initialization Steps	15-221
15-183	RMII Interface Mode Signal Configuration	15-224

Tables

Table Number	Title	Page Number
15-184	Shared RMII Signals	15-225
15-185	RMII Mode Register Initialization Steps	15-225
15-186	RTBI Interface Mode Signal Configuration	15-228
15-187	Shared RTBI Signals	15-229
15-188	RTBI Mode Register Initialization Steps	15-229
15-189	8-Bit FIFO Interface Mode Signal Configurations, eTSEC1/2	15-232
15-190	8-Bit FIFO Interface Mode Signal Configurations, eTSEC3/4	15-233
15-191	8-Bit FIFO Mode Register Initialization Steps	15-234
15-192	16-Bit FIFO Interface Mode Signal Configuration (eTSECs1 and 2)	15-235
15-193	16-Bit FIFO Mode Register Initialization Steps	15-237
15-194	SGMII Interface Signal Configuration (4-Wire)	15-238
15-195	SGMII Mode Register Initialization Steps	15-238
16-1	FEC Signals—Detailed Signal Descriptions	16-7
16-2	Module Memory Map Summary	16-8
16-3	Module Memory Map	16-9
16-4	IEVENT Field Descriptions	16-13
16-5	IMASK Field Descriptions	16-16
16-6	EDIS Field Descriptions	16-17
16-7	MINFLR Field Descriptions	16-18
16-8	PTV Field Descriptions	16-18
16-9	DMACTRL Field Descriptions	16-19
16-10	FIFO_PAUSE_CTRL Field Descriptions	16-21
16-11	FIFO_TX_THR Field Descriptions	16-22
16-12	FIFO_TX_STARVE Field Descriptions	16-22
16-13	FIFO_TX_STARVE_SHUTOFF Field Descriptions	16-23
16-14	TCTRL Field Descriptions	16-24
16-15	TSTAT Field Descriptions	16-24
16-16	TBDLEN Field Descriptions	16-25
16-17	CTBPTR Field Descriptions	16-26
16-18	TBPTR Field Descriptions	16-26
16-19	TBASE Field Descriptions	16-27
16-20	OSTBD Field Descriptions	16-28
16-21	OSTBDP Field Descriptions	16-29
16-22	RCTRL Field Descriptions	16-30
16-23	RSTAT Field Descriptions	16-31
16-24	RBDLEN Field Descriptions	16-31
16-25	CRBPTR Field Descriptions	16-32
16-26	MRBLR Field Descriptions	16-32
16-27	RBPTR Field Descriptions	16-33
16-28	RBASE Field Descriptions	16-33
16-29	MACCFG1 Field Descriptions	16-36

Tables

Table Number	Title	Page Number
16-30	MACCFG2 Field Descriptions	16-38
16-31	IPGIFG Field Descriptions	16-39
16-32	HAFDUP Field Descriptions	16-40
16-33	MAXFRM Field Descriptions	16-41
16-34	MIIMCFG Field Descriptions	16-42
16-35	MIIMCOM Field Descriptions	16-43
16-36	MIIMADD Field Descriptions	16-43
16-37	MIIMCON Field Descriptions	16-44
16-38	MIIMSTAT Field Descriptions	16-44
16-39	MIIMIND Field Descriptions	16-45
16-40	Interface Control Register Field Descriptions	16-46
16-41	IFSTAT Field Descriptions	16-47
16-42	MACSTNADDR1 Field Descriptions	16-48
16-43	MACSTNADDR2 Field Descriptions	16-48
16-44	IADDR*n* Field Description	16-49
16-45	GADDR*n* Field Descriptions	16-50
16-46	ATTR Field Descriptions	16-51
16-47	ATTRELI Field Descriptions	16-52
16-48	MII Interface Signals	16-53
16-49	Steps of Minimum Register Initialization	16-55
16-50	Flow Control Frame Structure	16-62
16-51	Non-Error Transmit Interrupts	16-63
16-52	Non-Error Receive Interrupts	16-64
16-53	Transmission Errors	16-65
16-54	Reception Errors	16-65
16-55	Transmit Data Buffer Descriptor (TxBD) Field Descriptions	16-68
16-56	Receive Buffer Descriptor Field Descriptions	16-70
16-57	MII Interface Mode Signal Configuration	16-72
16-58	MII Mode Register Initialization Steps	16-73
17-1	Register Conventions	17-1
17-2	Glossary	17-2
17-3	Pattern Matcher Overall Register Memory Map	17-6
17-4	PM Individual Register Memory Map	17-7
17-5	Interrupt Status Register Common Control Field Descriptions	17-15
17-6	Interrupt Enable Register Common Control Field Descriptions	17-16
17-7	Interrupt Status Disable Register Common Control Field Descriptions	17-17
17-8	Interrupt Inhibit Register Common Control Field Descriptions	17-18
17-9	SCOS Field Descriptions	17-19
17-10	Performance Monitor Threshold Field Descriptions	17-20
17-11	Channel Scheduling Control Register Field Descriptions	17-21
17-12	STNIB Register Field Descriptions	17-22

Tables

Table Number	Title	Page Number
17-13	STNIS Register Field Descriptions	17-23
17-14	STNTH1 Register Field Descriptions	17-23
17-15	STNTH2 Register Field Descriptions	17-24
17-16	STNTHL Register Field Descriptions	17-24
17-17	STNTHS Register Field Descriptions	17-25
17-18	STNCH Register Field Descriptions	17-25
17-19	SWDB Register Field Descriptions	17-26
17-20	KES Variable Length Trigger Size Register Field Descriptions	17-26
17-21	KES Error Configuration Register Field Descriptions	17-27
17-22	PME Error Handling Disables Register Field Descriptions	17-28
17-23	STNPM Register Field Descriptions	17-29
17-24	STNS1M Register Field Descriptions	17-30
17-25	DXE pattern range counter configuration Register Field Descriptions	17-30
17-26	STNPMR Register Field Descriptions	17-31
17-27	Pattern Description and Stateful Rule Base Address High Register Field Descriptions	17-32
17-28	Pattern Description and Stateful Rule Base Address Low Register Field Descriptions	17-32
17-29	DXE Memory Control Register Field Descriptions	17-33
17-30	DXE Error Configuration Register Field Descriptions	17-33
17-31	STNDSR Register Field Descriptions	17-34
17-32	STNESR Register Field Descriptions	17-35
17-33	STNS1R Register Field Descriptions	17-35
17-34	STNOB Register Field Descriptions	17-36
17-35	SRE Context Base Address High Register Field Descriptions	17-36
17-36	SRE Context Base Address Low Register Field Descriptions	17-37
17-37	SRE Memory Control Register Field Descriptions	17-37
17-38	SRE Configuration Register Field Descriptions	17-38
17-39	SRE Rule Reset Vector 0–7 Register Field Descriptions	17-39
17-40	SRE Rule Reset First Index Field Descriptions	17-40
17-41	SRE Rule Reset 32-Byte Increment Register Field Descriptions	17-40
17-42	SRE Rule Reset Repetitions Register Field Descriptions	17-41
17-43	SRE Rule Reset Work Configuration Register Field Descriptions	17-41
17-44	SRE Free Running Counter Configuration Register Field Descriptions	17-42
17-45	SRE Error Configuration 1 Register Field Descriptions	17-43
17-46	SRE Error Configuration 2 Register Field Descriptions	17-43
17-47	SRE Error Configuration 3 Register Field Descriptions	17-44
17-48	STDBC Register Field Descriptions	17-44
17-49	STDBP Register Field Descriptions	17-45
17-50	STDWC Register Field Descriptions	17-45
17-51	DBPPWL Register Field Descriptions	17-46
17-52	Free Buffer List Buffer Size Register Field Descriptions	17-47
17-53	Free Buffer List Assignment A Register Field Descriptions	17-48

Tables

Table Number	Title	Page Number
17-54	Free Buffer List Assignment B Register Field Descriptions	17-49
17-55	Free Buffer Manager Control Register Field Descriptions	17-50
17-56	MIA_BYC Field Descriptions	17-51
17-57	MIA_BLC Field Descriptions	17-52
17-58	MIA_CE Field Descriptions	17-52
17-59	MIA_CR Field Descriptions	17-55
17-60	MIA_ACR Field Descriptions	17-57
17-61	MIA_LSHC0/1/2 Field Descriptions	17-58
17-62	MIA_CWC0/1/2 Field Descriptions	17-59
17-63	PM_IP_REV_1 Field Descriptions	17-59
17-64	PM_IP_REV_2 Field Descriptions	17-60
17-65	Interrupt Status Register 0 Field Descriptions	17-60
17-66	Interrupt Enable Register 0 Field Descriptions	17-63
17-67	Interrupt Status Disable Register 0 Field Descriptions	17-64
17-68	Interrupt Inhibit Register 0 Field Descriptions	17-65
17-69	Interrupt Interval Control Register 0 Field Descriptions	17-65
17-70	Notifications Produced Index Register Channel 0 Field Descriptions	17-66
17-71	Commands Consumed Index Register Channel 0 Field Descriptions	17-67
17-72	Free Buffer Deallocate FIFO Consumer Index Register Channel 0 Field Descriptions	17-68
17-73	Channel Error Status Register Channel 0 Field Descriptions	17-68
17-74	Notifications Consumed Index Register Channel 0 Field Descriptions	17-69
17-75	Command Produced Index Register Channel 0 Field Descriptions	17-70
17-76	Commands Expired Index Register Channel 0 Field Descriptions	17-70
17-77	Free Buffer Deallocate FIFO Producer Index Register Channel 0 Field Descriptions	17-71
17-78	Output Truncated Incidence Counter Channel 0 Field Descriptions	17-72
17-79	Read Byte Counter Channel 0 Field Descriptions	17-72
17-80	SCOS0 Field Descriptions	17-73
17-81	Channel Reset and Control Register Channel 0 Field Descriptions	17-73
17-82	Command FIFO Base Address High Register Channel 0 Field Descriptions	17-75
17-83	Command FIFO Base Address Low Register Channel 0 Field Descriptions	17-75
17-84	Command FIFO Depth Register Channel 0 Field Descriptions	17-76
17-85	Command FIFO Threshold Register Channel 0 Field Descriptions	17-76
17-86	Notification FIFO Base Address High Register Channel 0 Field Descriptions	17-77
17-87	Notification FIFO Base Address Low Register Channel 0 Field Descriptions	17-77
17-88	Notification FIFO Depth Register Channel 0 Field Descriptions	17-78
17-89	Notification FIFO Threshold Register Channel 0 Field Descriptions	17-79
17-90	Notification Deflate Length Limit Register Channel 0 Field Descriptions	17-79
17-91	Notification Result Length Limit Register Channel 0 Field Descriptions	17-80
17-92	Free Buffer Deallocate FIFO Base Address High Register Channel 0 Field Descriptions	17-80
17-93	Free Buffer Deallocate FIFO Base Address Low Register Channel 0 Field Descriptions	17-81
17-94	Free Buffer Deallocate FIFO Depth Register Channel 0 Field Descriptions	17-81

Tables

Table Number	Title	Page Number
17-95	Free Buffer Deallocate FIFO Threshold Register Channel 0 Field Descriptions	17-82
17-96	Stream Context Configuration Register Channel 0 Field Descriptions	17-83
17-97	Stream Context Base Address High Register Channel 0 Field Descriptions	17-83
17-98	Stream Context Base Address Low Register Channel 0 Field Descriptions	17-84
17-99	Residue Configuration Register Channel 0 Field Descriptions	17-85
17-100	Residue Base Address High Register Channel 0 Field Descriptions	17-86
17-101	Residue Base Address Low Register Channel 0 Field Descriptions	17-86
17-102	Cache Awareness Control Register Channel 0 Field Descriptions	17-87
17-103	Memory Transaction Priority Control Register Channel 0 Field Descriptions	17-88
17-104	Free Buffer List Threshold Register Channel 0 Field Descriptions	17-90
17-105	Free Buffer List High and Low Watermark A Register Channel 0 Field Descriptions	17-90
17-106	Free Buffer List Allocation and Deallocation Counter A Register Channel 0 Field Descriptions	17-91
17-107	Free Buffer List On Deck Disable A Register Channel 0 Field Descriptions	17-92
17-108	Free Buffer List Physical Head Address High A Register Channel 0 Field Descriptions	17-92
17-109	Free Buffer List Physical Head Address Low A Register Channel 0 Field Descriptions	17-93
17-110	Free Buffer List Virtual Head Address High A Register Channel 0 Field Descriptions	17-93
17-111	Free Buffer List Virtual Head Address Low A Register Channel 0 Field Descriptions	17-94
17-112	Free Buffer List Physical Tail Address High A Register Channel 0 Field Descriptions	17-94
17-113	Free Buffer List Physical Tail Address Low A Register Channel 0 Field Descriptions	17-95
17-114	Free Buffer List Number of Buffers A Register Channel 0 Field Descriptions	17-95
17-115	Free Buffer List Buffer Size A Register Channel 0 Field Descriptions	17-96
17-116	Free Buffer List Virtual On Deck Pointer A High Register Channel 0 Field Descriptions	17-97
17-117	Free Buffer List Virtual On Deck Pointer A Low Register Channel 0 Field Descriptions	17-97
17-118	Pattern Matcher Functional Units	17-99
17-119	Pre-Defined Group	17-101
17-120	Special Triggers Description	17-111
17-121	DXE Pattern Match Reporting—“Default” Inconclusive Mode	17-126
17-122	DXE Pattern Match Reporting—“Alternate” Inconclusive Mode	17-126
17-123	Test Line First Block Field Description	17-128
17-124	Test Line Extension Block (128-byte) Format	17-129
17-125	Instruction Test Line Field Description	17-130
17-126	Capture Codes Description	17-132
17-127	Element Codes	17-134
17-128	Example 1 Head Test Lines	17-136
17-129	Example 1 Test Lines	17-136
17-130	Example 2 Head Test Lines	17-137
17-131	Example 2 Test Lines	17-137
17-132	SRE Instruction Descriptions	17-144
17-133	SRE Instruction Field Descriptions	17-146
17-134	Pattern Match Event Meta Data Registers	17-148

Tables

Table Number	Title	Page Number
17-135	Simple Match Report	17-153
17-136	Simple End of SUI Report	17-153
17-137	Verbose Mode 1 Report	17-154
17-138	Verbose Mode 2 Report—Stateless Rule	17-154
17-139	Verbose Mode 2 Report—Dualstate Without Context Rule	17-155
17-140	Verbose Mode 2 Report—Dualstate With Context Rule	17-155
17-141	Verbose Mode 2 Report—Multistate Rule	17-155
17-142	Verbose Mode 3 Report—Dualstate with Context Rule	17-155
17-143	Verbose Mode 3 Report—Multistate Rule	17-156
17-144	Linked List Next Buffer Descriptor Format	17-178
17-145	Scatter/Gather Block Descriptor Format	17-179
17-146	Command FIFO Entry Fields	17-179
17-147	Command Code Enumeration	17-184
17-148	Stream Context Update Type and Value Descriptions	17-184
17-149	Stream Context Activity Code Field Definitions	17-185
17-150	Read Table Entry Pattern Matcher Control Format	17-186
17-151	Read Table Entry Response Format	17-187
17-152	Write Table Entry Pattern Matcher Control Format	17-187
17-153	Clear Session Context by Session ID Pattern Matcher Control Format	17-188
17-154	Notification FIFO Entry Fields (Format = Block Vector)	17-189
17-155	Notification FIFO Entry Fields (Format = Linked List)	17-191
17-156	Notification FIFO Exception Code Enumeration	17-192
17-157	DMA Engine Channel Scheduling Algorithms	17-202
17-158	Free Buffer Deallocate FIFO Linked List Entry Format	17-202
17-159	Free Buffer Deallocate FIFO Scatter/Gather Block Vector Entry Format	17-203
17-160	MIA Read and Write Port Assignments	17-207
17-161	Pattern Matcher Performance Monitor Events	17-209
18-1	Module Memory Map Summary	18-4
18-2	Module Memory Map	18-5
18-3	TLU_ID1 Field Descriptions	18-8
18-4	TLU_ID2 Field Descriptions	18-9
18-5	IEVENT Field Descriptions	18-10
18-6	IMASK Field Descriptions	18-11
18-7	IEATR Field Descriptions	18-12
18-8	IEADD Field Descriptions	18-12
18-9	IEDIS Field Descriptions	18-13
18-10	MBANK*n* Field Descriptions	18-14
18-11	PTBL*n* Field Descriptions	18-15
18-12	CRAMR Field Descriptions	18-16
18-13	CRAMW Field Descriptions	18-17
18-14	CFIND Field Descriptions	18-17

Tables

Table Number	Title	Page Number
18-15	CTHTK Field Descriptions	18-18
18-16	CTCRT Field Descriptions	18-18
18-17	CTCHS Field Descriptions	18-19
18-18	CTDAT Field Descriptions	18-19
18-19	CHITS Field Descriptions	18-20
18-20	CMISS Field Descriptions	18-20
18-21	CHCOL Field Descriptions	18-21
18-22	CCRTL Field Descriptions	18-21
18-23	CARO Field Descriptions	18-22
18-24	CARM Field Descriptions	18-23
18-25	CMDOP Field Descriptions	18-24
18-26	CMDIX Field Descriptions	18-25
18-27	CSTAT Field Descriptions	18-26
18-28	KD0B–KD7B Field Descriptions	18-27
18-29	Summary of TLU Commands	18-29
18-30	Write Command Field Descriptions	18-30
18-31	Add Command Field Descriptions	18-32
18-32	Read Command Field Descriptions	18-33
18-33	Acchash Command Field Descriptions	18-34
18-34	Find Command Field Descriptions	18-35
18-35	Findr Command Field Descriptions	18-36
18-36	Findw Command Field Descriptions	18-38
18-37	Command Status in Relation to Errors	18-39
18-38	TLU Error Conditions	18-39
18-39	Allowed TLU State Transitions	18-40
18-40	TLU Table State versus Table Type	18-41
18-41	Hash Entry Field Descriptions	18-42
18-42	Trie Entry Field Descriptions	18-43
18-43	Key Entry Field Descriptions	18-45
18-44	IPTD Entry Field Descriptions	18-46
19-1	Relationship of Modes and Features	19-3
19-2	DMA Mode Bit Settings	19-3
19-3	DMA Signals—Detailed Signal Descriptions	19-5
19-4	DMA Register Summary	19-6
19-5	MR*n* Field Descriptions	19-10
19-6	SR*n* Field Descriptions	19-12
19-7	ECLNDARn Field Descriptions	19-14
19-8	CLNDAR*n* Field Descriptions	19-15
19-9	SATR*n* Field Descriptions	19-15
19-10	SAR*n* Field Descriptions	19-17
19-11	SAR*n* Field Descriptions	19-17

Tables

Table Number	Title	Page Number
19-12	DATR*n* Field Descriptions	19-18
19-13	DAR*n* Field Descriptions	19-19
19-14	DAR*n* Field Descriptions	19-20
19-15	BCR*n* Field Descriptions	19-20
19-16	NLNDAR*n* Field Descriptions	19-21
19-17	ENLNDAR*n* Field Descriptions	19-22
19-18	ECLSDAR*n* Field Descriptions	19-22
19-19	CLSDAR*n* Field Descriptions	19-23
19-20	ENLSDARn Field Descriptions	19-23
19-21	NLSDAR*n* Field Descriptions	19-24
19-22	SSR*n* Field Descriptions	19-24
19-23	DSR*n* Field Descriptions	19-25
19-24	DGSR Field Descriptions	19-25
19-25	Channel State Table	19-34
19-26	List DMA Descriptor Summary	19-35
19-27	Link DMA Descriptor Summary	19-36
20-1	RapidIO Memory Map	20-5
20-2	DIDCAR Field Descriptions	20-11
20-3	DICAR Field Descriptions	20-12
20-4	AIDCAR Field Descriptions	20-12
20-5	AICAR Field Descriptions	20-13
20-6	PEFCAR Field Descriptions	20-13
20-7	SOCAR Field Descriptions	20-14
20-8	DOCAR Field Descriptions	20-15
20-9	MCSR Field Definitions	20-17
20-10	PWDCSR Field Descriptions	20-18
20-11	PELLCCSR Field Descriptions	20-19
20-12	LCSBA1CSR Field Descriptions	20-20
20-13	BDIDCSR Field Descriptions	20-20
20-14	HBDIDLCSR Field Descriptions	20-21
20-15	CTCSR Field Descriptions	20-21
20-16	PMBH0 Field Descriptions	20-22
20-17	PLTOCCSR Field Descriptions	20-23
20-18	PRTOCCSR Field Descriptions	20-23
20-19	GCCSR Field Descriptions	20-24
20-20	LMREQCSR Field Descriptions	20-25
20-21	LMRESPCSR Field Descriptions	20-25
20-22	LASCSR Field Descriptions	20-26
20-23	ESCSR Field Descriptions	20-27
20-24	CCSR Field Descriptions	20-28
20-25	ERBH Field Descriptions	20-30

Tables

Table Number	Title	Page Number
20-26	LTLEDCSR Field Descriptions	20-31
20-27	LTLEECSR Field Descriptions	20-32
20-28	LTLACCSR Field Descriptions	20-34
20-29	LTLDIDCCSR Field Descriptions	20-35
20-30	LTLCCCSR Field Descriptions	20-35
20-31	EDCSR Field Descriptions	20-36
20-32	ERECSR Field Descriptions	20-37
20-33	ECACSR Field Descriptions	20-38
20-34	PCSECCSR0 Field Descriptions	20-39
20-35	PECCSR1 Field Descriptions	20-39
20-36	PECCSR2 Field Descriptions	20-39
20-37	PECCSR3 Field Descriptions	20-40
20-38	ERCSR Field Descriptions	20-40
20-39	ERTCSR Field Descriptions	20-42
20-40	LLCR Field Descriptions	20-42
20-41	EPWISR Field Descriptions	20-43
20-42	LRETCR Field Descriptions	20-44
20-43	PRETCR Field Descriptions	20-44
20-44	ADIDCSR Field Descriptions	20-45
20-45	AACR Field Descriptions	20-45
20-46	LOPTTLCR Field Descriptions	20-46
20-47	IECSR Field Descriptions	20-46
20-48	PCR Field Descriptions	20-47
20-49	SLCSR Field Descriptions	20-48
20-50	SLEICR Field Descriptions	20-48
20-51	IPBRR1 Field Descriptions	20-49
20-52	IPBRR2 Field Descriptions	20-50
20-53	ROWTAR*n* Field Descriptions	20-53
20-54	ROWTEAR*n* Field Descriptions	20-53
20-55	ROWBAR*n* Descriptions	20-54
20-56	ROWAR*n* Field Descriptions	20-55
20-57	ROWS*n*R*n* Field Descriptions	20-57
20-58	RIWTAR*n* Field Descriptions	20-58
20-59	RIWBAR*n* Field Descriptions	20-59
20-60	RapidIO Inbound Window Attributes Register 1–4	20-59
20-61	RapidIO Inbound Window Attributes Register 0	20-59
20-62	RIWAR*n* Field Descriptions	20-59
20-63	OM*n*MR Field Descriptions	20-62
20-64	OM*n*SR Field Descriptions	20-63
20-65	EOM*n*DQDPAR Field Descriptions	20-65
20-66	OM*n*DQDPAR Field Descriptions	20-65

Tables

Table Number	Title	Page Number
20-67	EOM*n*DQEPAR Field Descriptions	20-66
20-68	OM*n*DQEPAR Field Descriptions	20-67
20-69	EOM*n*SAR Field Descriptions	20-67
20-70	OM*n*SAR Field Descriptions	20-68
20-71	OM*n*DPR Field Descriptions	20-68
20-72	OM*n*DATR Field Descriptions	20-69
20-73	OM*n*DCR Field Descriptions	20-70
20-74	OM*n*RETCR Field Descriptions	20-71
20-75	OM*n*MGR Field Descriptions	20-72
20-76	OM*n*MLR Field Descriptions	20-72
20-77	IMnMR Field Descriptions	20-73
20-78	IMnSR Field Descriptions	20-74
20-79	EIMnFQDPAR Field Descriptions	20-76
20-80	IMnFQDPAR Field Descriptions	20-76
20-81	EIMnFQEPAR Field Descriptions	20-77
20-82	IMnFQEPAR Field Descriptions	20-77
20-83	IMnMIRIR Field Descriptions	20-78
20-84	ODMR Field Descriptions	20-78
20-85	ODSR Field Descriptions	20-79
20-86	ODDPR Field Descriptions	20-80
20-87	ODDATR Field Descriptions	20-80
20-88	ODRETCR Field Descriptions	20-81
20-89	IDMR Field Descriptions	20-82
20-90	IDSR Field Descriptions	20-83
20-91	EIDQDPAR Field Descriptions	20-85
20-92	IDQDPAR Field Descriptions	20-85
20-93	EIDQEPAR Field Descriptions	20-86
20-94	IDQEPAR Field Descriptions	20-86
20-95	IDMIRIR Field Descriptions	20-87
20-96	IPWMR Field Descriptions	20-87
20-97	IPWSR Field Descriptions	20-88
20-98	EIPWQBAR Field Descriptions	20-89
20-99	IPWQBAR Field Descriptions	20-89
20-100	RapidIO I/O Transactions	20-90
20-101	RapidIO Message Passing Transactions	20-90
20-102	RapidIO GSM Transactions	20-91
20-103	RapidIO Small Transport Field Packet Format	20-91
20-104	1x/4x LP-Serial Control Symbol Format	20-92
20-105	Physical RapidIO Errors Detected	20-104
20-106	Physical RapidIO Threshold Response	20-105
20-107	Hardware Errors for NRead Transaction	20-106

Tables

Table Number	Title	Page Number
20-108	Hardware Errors for Maintenance Read/Write Req Transaction	20-108
20-109	Hardware Errors for Atomic (inc, dec, set, or clr) Read Transaction	20-110
20-110	Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions	20-112
20-111	Hardware Errors For SWrite Transactions	20-115
20-112	Hardware Errors For Maintenance Response Transactions	20-116
20-113	Hardware Errors for IO/GSM Response Transactions (Not Maintenance)	20-120
20-114	Hardware Errors For DMA Message Response Transactions	20-125
20-115	Hardware Errors For Message Request Transactions	20-127
20-116	Hardware Errors For Message Response Transactions	20-130
20-117	Hardware Errors for Doorbell Request Transaction	20-131
20-118	Hardware Errors For Doorbell Response Transactions	20-133
20-119	Hardware Errors for PortWrite Transaction	20-134
20-120	Hardware Errors for Reserved Ftype	20-137
20-121	Hardware Errors for Outbound Transaction Crossed ATMU Boundary	20-138
20-122	Hardware Errors for Outbound Packet Time-to-live Errors	20-139
20-123	Outbound Message Direct Mode Hardware Errors	20-145
20-124	Outbound Message Direct Mode Programming Errors	20-149
20-125	Outbound Message Unit Descriptor Summary	20-153
20-126	Outbound Message Chaining Mode Hardware Errors	20-156
20-127	Outbound Message Chaining Mode Programming Errors	20-156
20-128	Inbound Message Hardware Errors	20-162
20-129	Inbound Message Programming Errors	20-166
20-130	Outbound Doorbell Hardware Errors	20-172
20-131	Outbound Doorbell Programming Errors	20-174
20-132	Inbound Doorbell Target Info Definition	20-176
20-133	Source Info Definition	20-176
20-134	Inbound Doorbell Hardware Errors	20-178
20-135	Inbound Doorbell Programming Errors	20-181
20-136	Inbound Port-Write Hardware Errors	20-185
20-137	Inbound Port-Write Programming Errors	20-188
21-1	POR Parameters for PCI Express Controller	21-4
21-2	PCI Express Interface Signals—Detailed Signal Descriptions	21-5
21-3	PCI Express Memory-Mapped Register Map	21-6
21-4	PEX_CONFIG_ADDR Field Descriptions	21-10
21-5	PEX_CONFIG_DATA Field Descriptions	21-11
21-6	PEX_OTB_CPL_TOR Field Descriptions	21-12
21-7	PEX_CONF_RTY_TOR Field Descriptions	21-12
21-8	PEX_CONFIG Field Descriptions	21-13
21-9	PEX_PME_MES_DR Field Descriptions	21-14
21-10	PEX_PME_MES_DISR Field Descriptions	21-15

Tables

Table Number	Title	Page Number
21-11	PEX_PME_MES_IER Field Descriptions	21-17
21-12	PEX_PMCR Field Descriptions	21-18
21-13	PCI Express IP Block Revision Register 1 Field Descriptions	21-19
21-14	PCI Express IP Block Revision Register 2 Field Descriptions	21-20
21-15	PEXOTAR*n* Field Descriptions	21-21
21-16	PCI Express Outbound Extended Address Translation Register *n* Field Descriptions	21-21
21-17	PCI Express Outbound Window Base Address Register *n* Field Descriptions	21-22
21-18	PEXOWAR*n* Field Descriptions	21-23
21-19	PCI Express Inbound Translation Address Registers Field Descriptions	21-26
21-20	PCI Express Inbound Window Base Address Register Field Descriptions	21-27
21-21	PCI Express Inbound Window Base Extended Address Register Field Descriptions	21-28
21-22	PCI Express Inbound Window Attributes Registers Field Descriptions	21-28
21-23	PCI Express Error Detect Register Field Descriptions	21-31
21-24	PCI Express Error Interrupt Enable Register Field Descriptions	21-33
21-25	PCI Express Error Disable Register Field Descriptions	21-34
21-26	PCI Express Error Capture Status Register Field Descriptions	21-36
21-27	PCI Express Error Capture Register 0 Field Descriptions Internal Source, Outbound Transaction	21-37
21-28	PCI Express Error Capture Register 0 Field Descriptions External Source, Inbound Transaction	21-38
21-29	PCI Express Error Capture Register 1 Field Descriptions Internal Source, Outbound Transaction	21-39
21-30	PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound CompletionTransaction	21-39
21-31	PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound Memory Request Transaction	21-40
21-32	PCI Express Error Capture Register 2 Field Descriptions Internal Source, Outbound Transaction	21-40
21-33	PCI Express Error Capture Register 2 Field Descriptions External Source, Inbound Completion Transaction	21-41
21-34	PCI Express Error Capture Register 2 Field Descriptions External Source, Inbound Memory Request Transaction	21-41
21-35	PCI Express Error Capture Register 3 Field Descriptions Internal Source, Outbound Transaction	21-42
21-36	PEX Error Capture Register 3 Field Descriptions External Source, Inbound Memory Request Transaction	21-43
21-37	PCI Express Vendor ID Register Field Description	21-46
21-38	PCI Express Device ID Register Field Description	21-46
21-39	PCI Express Command Register Field Descriptions	21-47
21-40	PCI Express Status Register Field Descriptions	21-48
21-41	PCI Express Revision ID Register Field Descriptions	21-49

Tables

Table Number	Title	Page Number
21-42	PCI Express Class Code Register Field Descriptions	21-50
21-43	PCI Express Bus Cache Line Size Register Field Descriptions	21-50
21-44	PCI Express Bus Latency Timer Register Field Descriptions	21-51
21-45	PCI Express Bus Latency Timer Register Field Descriptions	21-51
21-46	PEXCSRBAR Field Descriptions	21-53
21-47	32-Bit Memory Base Address Register (BAR1) Field Descriptions	21-53
21-48	64-Bit Low Memory Base Address Register Field Descriptions	21-54
21-49	Bit Setting for 64-Bit High Memory Base Address Register	21-54
21-50	PCI Express Subsystem Vendor ID Register Field Description	21-55
21-51	PCI Express Subsystem ID Register Field Description	21-55
21-52	Capabilities Pointer Register Field Description	21-56
21-53	PCI Express Interrupt Line Register Field Description	21-56
21-54	PCI Express Interrupt Pin Register Field Description	21-57
21-55	PCI Express Maximum Grant Register Field Description	21-57
21-56	PCI Express Maximum Latency Register Field Description	21-58
21-57	PEXCSRBAR Field Descriptions	21-59
21-58	PCI Express Primary Bus Number Register Field Description	21-59
21-59	PCI Express Secondary Bus Number Register Field Description	21-60
21-60	PCI Express Subordinate Bus Number Register Field Description	21-60
21-61	PCI Express I/O Base Register Field Description	21-61
21-62	PCI Express I/O Limit Register Field Description	21-62
21-63	PCI Express Secondary Status Register Field Description	21-62
21-64	PCI Express Memory Base Register Field Description	21-63
21-65	PCI Express Memory Limit Register Field Description	21-63
21-66	PCI Express Prefetchable Memory Base Register Field Description	21-64
21-67	PCI Express Prefetchable Memory Limit Register Field Description	21-64
21-68	PCI Express Prefetchable Base Upper 32 Bits Register	21-65
21-69	PCI Express Prefetchable Limit Upper 32 Bits Register	21-65
21-70	PCI Express I/O Base Upper 16 Bits Register Field Description	21-66
21-71	PCI Express I/O Limit Upper 16 Bits Register Field Description	21-66
21-72	Capabilities Pointer Register Field Description	21-66
21-73	PCI Express Interrupt Line Register Field Description	21-67
21-74	PCI Express Interrupt Pin Register Field Description	21-67
21-75	PCI Express Bridge Control Register Field Description	21-68
21-76	PCI Express Power Management Capability ID Register Field Description	21-70
21-77	PCI Express Power Management Capabilities Register Field Description	21-70
21-78	PCI Express Status and Control Register Field Description	21-71
21-79	PCI Express Power Management Data Register Field Description	21-71
21-80	PCI Express Capability ID Register Field Description	21-72
21-81	PCI Express Capabilities Register Field Description	21-72
21-82	PCI Express Device Capabilities Register Field Description	21-73

Tables

Table Number	Title	Page Number
21-83	PCI Express Device Control Register Field Description	21-74
21-84	PCI Express Device Status Register Field Description	21-74
21-85	PCI Express Link Capabilities Register Field Description	21-75
21-86	PCI Express Link Control Register Field Description	21-75
21-87	PCI Express Link Status Register Field Description	21-76
21-88	PCI Express Slot Capabilities Register Field Description	21-77
21-89	PCI Express Slot Control Register Field Description	21-78
21-90	PCI Express Slot Status Register Field Descriptions	21-78
21-91	PCI Express Root Control Register Field Description	21-79
21-92	PCI Express Root Status Register Field Description	21-80
21-93	PCI Express Capability ID Register Field Description	21-80
21-94	PCI Express MSI Message Control Register Field Description	21-80
21-95	PCI Express MSI Message Address Register Field Description	21-81
21-96	PCI Express MSI Message Upper Address Register Field Description	21-81
21-97	PCI Express MSI Message Data Register Field Description	21-82
21-98	PCI Express Advanced Error Reporting Capability ID Register Field Description	21-84
21-99	PCI Express Uncorrectable Error Status Register Field Description	21-84
21-100	PCI Express Uncorrectable Error Mask Register Field Description	21-85
21-101	PCI Express Uncorrectable Error Severity Register Field Description	21-86
21-102	PCI Express Correctable Error Status Register Field Description	21-87
21-103	PCI Express Correctable Error Mask Register Field Description	21-88
21-104	PCI Express Advanced Error Capabilities and Control Register Field Description	21-88
21-105	PCI Express Header Log Register Field Description	21-89
21-106	PCI Express Root Error Command Register Field Description	21-90
21-107	PCI Express Root Error Command Status Field Description	21-90
21-108	PCI Express Correctable Error Source ID Register Field Description	21-91
21-109	PCI Express Correctable Error Source ID Register Field Description	21-91
21-110	PEX_LTSSM_STAT Field Descriptions	21-92
21-111	PEX_LTSSM_STAT Status Codes	21-92
21-112	PEX_GCLK_RATIO Field Descriptions	21-94
21-113	PEX_PM_TIMER Field Descriptions	21-95
21-114	PEX_PME_TIMEOUT Field Descriptions	21-95
21-115	PEX_SSVID_UPDATE Field Descriptions	21-96
21-116	PEX_CFG_READY Field Descriptions	21-97
21-117	PEX_PME_TO_ACK_TOR Field Descriptions	21-97
21-118	PEX_SS_INTR_MASK Field Descriptions	21-98
21-119	PCI Express Transactions	21-100
21-120	Lane Assignment With and Without Lane Reversal	21-102
21-121	Internal Platform (OCeaN) Message Data Format	21-105
21-122	PCI Express ATMU Outbound Messages	21-105
21-123	PCI Express RC Inbound Message Handling	21-106

Tables

Table Number	Title	Page Number
21-124	PCI Express EP Inbound Message Handling	21-107
21-125	PCI Express Internal Controller Interrupt Sources	21-110
21-126	Error Conditions	21-112
21-127	Initial credit advertisement	21-115
21-128	Power Management State Supported	21-116
22-1	IPIC External Signals—Detailed Signal Descriptions	22-2
22-2	GPIO Register Address Map	22-2
22-3	GPDIR Bit Settings	22-3
22-4	GPODR Bit Settings	22-3
22-5	GP*n*DAT Bit Settings	22-4
22-6	GPIER Bit Settings	22-4
22-7	GPIMR Bit Settings	22-5
22-8	GPICR Bit Settings	22-5
23-1	External Signal Summary	23-2
23-2	Detailed Signal Descriptions	23-2
23-3	Global Utilities Block Register Summary	23-3
23-4	PORPLLSR Field Descriptions	23-5
23-5	PORBMSR Field Descriptions	23-6
23-6	PORIMPSCR Field Descriptions	23-7
23-7	PORDEVSR Field Descriptions	23-8
23-8	PORDBGMSR Field Descriptions	23-11
23-9	PORDEVSR2 Field Descriptions	23-12
23-10	GPPORCR Field Descriptions	23-13
23-11	PMUXCR Field Descriptions	23-14
23-12	DEVDISR Field Descriptions	23-14
23-13	POWMGTCSR Field Descriptions	23-17
23-14	MCPSUMR Field Descriptions	23-19
23-15	RSTRSCR Field Descriptions	23-20
23-16	ECTRSTCR Field Descriptions	23-21
23-17	AUTORSTSR Field Descriptions	23-21
23-18	PVR Field Descriptions	23-23
23-19	SVR Field Descriptions	23-23
23-20	RSTCR Field Descriptions	23-24
23-21	LBCVSELCR Field Descriptions	23-24
23-22	DDRCLKDR Field Descriptions	23-25
23-23	CLKOCR Field Descriptions	23-26
23-24	SRDS1CR Field Descriptions	23-28
23-25	SRDS2CR Field Descriptions	23-29
23-26	MPC8572E Power Management Modes—Basic Description	23-31
23-27	Power Management Entry Protocol and Initiating Functional Units	23-34
24-1	Control Register Memory Map	24-4

Tables

Table Number	Title	Page Number
24-2	PMGC0 Field Descriptions	24-6
24-3	PMLCA0 Field Descriptions	24-7
24-4	PMLCA1–PMLCA11 Field Descriptions	24-8
24-5	PMLCB0 Field Descriptions	24-8
24-6	PMLCB*n* Field Descriptions	24-9
24-7	PMC0 Field Descriptions	24-11
24-8	PMC[1–9] Field Descriptions	24-11
24-9	Burst Definition	24-14
24-10	Performance Monitor Events	24-16
24-11	PMGC0 and PMLCAn Settings	24-36
24-12	Register Settings for Counting Examples	24-36
25-1	POR Configuration Settings and Debug Modes	25-3
25-2	Debug, Watchpoint and Test Signal Summary	25-6
25-3	Debug Signals—Detailed Signal Descriptions	25-7
25-4	Watchpoint and Trigger Signals—Detailed Signal Descriptions	25-8
25-5	JTAG Test and Other Signals—Detailed Signal Descriptions	25-8
25-6	Debug and Watchpoint Monitor Memory Map	25-10
25-7	WMCR0 Field Descriptions	25-11
25-8	WMCR1 Field Descriptions	25-13
25-9	WMAR Field Descriptions	25-13
25-10	WMAMR Field Descriptions	25-14
25-11	WMTMR Field Descriptions	25-14
25-12	Transaction Types By Interface	25-15
25-13	WMSR Field Descriptions	25-16
25-14	TBCR0 Field Descriptions	25-17
25-15	TBCR1 Field Descriptions	25-18
25-16	TBAR Field Descriptions	25-19
25-17	TBAMR Field Descriptions	25-19
25-18	TBTMR Field Descriptions	25-20
25-19	TBSR Field Descriptions	25-21
25-20	TBACR Field Descriptions	25-22
25-21	TBADHR Field Descriptions	25-22
25-22	TBADR Field Descriptions	25-23
25-23	PCIDR Field Descriptions	25-23
25-24	CCIDR Field Descriptions	25-24
25-25	TOSR Field Descriptions	25-25
25-26	Source and Target ID Values	25-25
25-27	CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)	25-29
25-28	PCI Express Trace Buffer Entry Field Descriptions	25-30

Tables

Table Number	Title	Page Number

About This Book

This reference manual defines the functionality of the MPC8572E. This device integrates two PowerPC™ processor cores, based on Power Architecture™ technology, with system logic required for networking, telecommunications, and wireless infrastructure applications. The e500v2 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the Book E definition of the Power Architecture. This book is intended as a companion to the *PowerPC e500 Core Complex Reference Manual*.

Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

Organization

Following is a summary and a brief description of the major parts of this reference manual:

Part I, "Overview," describes the many features of the MPC8572E integrated host processor at an overview level. The following chapters are included:

- Chapter 1, "Overview," provides a high-level description of features and functionality of the MPC8572E integrated host processor. It describes the MPC8572E, its interfaces, and programming model. The functional operation of the MPC8572E, with emphasis on peripheral functions, is also described.
- Chapter 2, "Memory Map," describes the memory map of the MPC8572E. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.
- Chapter 3, "Signal Descriptions," provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).
- Chapter 4, "Reset, Clocking, and Initialization," describes the hard and soft resets, the power-on reset (POR) sequence, power-on reset configuration, clocking, and initialization of the MPC8572E.

Part II, "e500 Core Complex and L2 Cache," describes the many features of the MPC8572E core processors at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, “Core Complex Overview,” provides an overview of the e500v2 core processor and the L1 caches and MMU that, together with the cores, comprise the core complex.
- Chapter 6, “Core Register Summary,” provides a listing of the e500v2 registers in reference form.
- Chapter 7, “L2 Look-Aside Cache/SRAM,” describes the L2 cache of the MPC8572E. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

Part III, “Memory, Security, and I/O Interfaces,” defines the memory, security, and I/O interfaces of the MPC8572E and it describes how these blocks interact with other blocks on the device. The following chapters are included:

- Chapter 8, “e500 Coherency Module,” defines the e500 coherency module and how it facilitates communication between the e500v2 cores, the L2 cache, and the other blocks that comprise the coherent memory domain of the MPC8572E.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500v2 cores in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500v2- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8572E.
- Chapter 9, “DDR Memory Controllers,” describes the two DDR2/DDR3 SDRAM memory controllers of the MPC8572E. These fully programmable controllers support most DDR memories available today, including both buffered and unbuffered devices. The 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 like crawl mode and ECC error injection support rapid system debug.
- Chapter 10, “Programmable Interrupt Controller (PIC),” describes the embedded multiprocessor programmable interrupt controller (PIC) of the MPC8572E. The PIC is an OpenPIC-compliant interrupt controller that provides multiprocessor interrupt management and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to a CPU for servicing.
- Chapter 11, “Security Engine (SEC) 3.0,” describes the security controller of the MPC8572E. The SEC 3.0 off-loads computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor cores of the MPC8572E. It is optimized to process all cryptographic algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, 802.11i, 3G, A5/3 for GSM and EDGE, and GEA3 for GPRS.
- Chapter 12, “I^2C Interfaces,” describes the inter-IC (IIC or I^2C) bus controllers of the MPC8572E. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters and LCDs. The MPC8572E powers up in boot sequencer mode which allows the I^2C1 controller to initialize configuration registers.
- Chapter 13, “DUART,” describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- Chapter 14, “Enhanced Local Bus Controller,” describes the enhanced local bus controller (eLBC) of the MPC8572E. The main component of the enhanced local bus controller is its memory controller which provides a seamless interface to many types of memory devices and peripherals.

The memory controller is responsible for controlling eight memory banks shared by a general-purpose chip-select machine (GPCM), a NAND flash control machine (FCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to SRAM, EPROM, Flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- Chapter 15, "Enhanced Three-Speed Ethernet Controllers," describes the four enhanced three-speed Ethernet controllers (eTSECs) on the MPC8572E. These controllers provide 10/100/1Gb Ethernet support with a complete set of media-independent interface options including MII, RMII, GMII, RGMII, SGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8572E memory coherency module. The controllers provide full-duplex FIFO interface modes, hardware offload acceleration capabilities, and quality of service support. They are backward compatible with PowerQUICC™ III TSEC controllers.
- Chapter 16, "10/100 Fast Ethernet Controller," describes the fast Ethernet controller (FEC) port on the MPC8572E. This controller provides 10/100 fast Ethernet support suitable for diagnostic and maintenance interface with a multicore system.
- Chapter 17, "Pattern Matcher (PM) 1.1," describes the pattern matching engine (PME) of the MPC8572E. The PME assists in data organization based on regular expression pattern matching or state rule pattern matching. The PME also includes a DEFLATE engine to decompress DEFLATE compress format data, include zlib and gzip.
- Chapter 18, "Table Lookup Unit," describes the two table lookup units (TLUs) of the MPC8572E. The TLUs give access to application-defined routing topology, control, and statistics tables in external memory. The TLUs support several types of table lookup algorithms and provide resources for efficient generation of table entry addresses in memory, hash generation of addresses, and binary table searching algorithms for both exact-match and longest-prefix match strategies.
- Chapter 19, "DMA Controllers," describes both of the four-channel general-purpose DMA controllers of the MPC8572E. The DMA controllers transfer blocks of data independent of the e500v2 cores or external hosts. Data movement occurs among the local address space. Each DMA controller has four high-speed channels. Both the e500 cores and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 20, "Serial RapidIO Interface," describes the Serial RapidIO™ interface of the MPC8572E. The Serial RapidIO interface conforms to the *RapidIO™ Interconnect Specification, Revision 1.2*. Both 1x and 4x LP-serial link interfaces are possible, with transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps) per lane. Also included is a RapidIO messaging unit (RMU) which manages two inbox/outbox mailboxes for data and one doorbell message structure.The message unit can also multicast a single-segment 156-byte message to up to 32 different destination DevIDs.
- Chapter 21, "PCI Express Interface Controller," describes the three PCI-Express controllers of the MPC8572E. Each controller is compliant with the *PCI Express Base Specification Revision 1.0a*. The physical layer of these controllers operate at 2.5 Gbaud per lane. Configuration options allow multiple width configurations among the three controllers.

- Chapter 22, “General Purpose I/O (GPIO),” describes the general-purpose input and output signals of the MPC8572E.

Part IV, “Global Functions and Debug,” defines other global blocks of the MPC8572E. The following chapters are included:

- Chapter 23, “Global Utilities,” defines the global utilities of the MPC8572E. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and signal multiplexing capabilities.
- Chapter 24, “Device Performance Monitor,” describes the performance monitor of the MPC8572E. Note that the MPC8572E performance monitor is similar to but separate from the performance monitor implemented on the e500v2 core.
- Chapter 25, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8572E.

Suggested Reading

This section lists additional reading that provides background for the information in this manual as well as general information about the architecture.

General Information

The following documentation, published by Morgan-Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA, provides useful information about the PowerPC architecture and computer architecture in general:

- *The PowerPC Architecture: A Specification for a New Family of RISC Processors*, Second Edition, by International Business Machines, Inc.
- *Computer Architecture: A Quantitative Approach*, Third Edition, by John L. Hennessy and David A. Patterson
- *Computer Organization and Design: The Hardware/Software Interface*, Second Edition, by David A. Patterson and John L. Hennessy

Related Documentation

Freescale documentation is available from the sources listed on the back cover of this manual; the document order numbers are included in parentheses for ease in ordering:

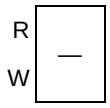
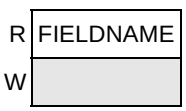
- *EREF: A Reference for Freescale Book E and the e500 Core*—This book provides a higher-level view of the programming model as it is defined by Book E, the Freescale Book E implementation standards, and the e500 microprocessor.
- Reference manuals (formerly called user’s manuals)—These books provide details about individual implementations.
- Addenda/errata to reference or user’s manuals—Because some processors have follow-on parts an addendum is provided that describes the additional features and functionality changes. These addenda are intended for use with the corresponding reference or user’s manuals.

- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Product Briefs—Each device has a technical summary that provides an overview of its features.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale processors.

Additional literature is published as new processors become available. For a current list of documentation, refer to http://www.freescale.com.

Conventions

This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, **bcctr***x*. Book titles in text are set in italics Internal signals are set in lowercase italics, for example, $\overline{core\_int}$
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, **r**B	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REG[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 a numeric variable.
¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example TCR[WP]||TCR[WPEXT]
	Indicates a reserved bit field in a memory-mapped or an e500 register.
	Indicates a read-only bit field in a memory-mapped register.

R	
W	FIELDNAME

Indicates a write-only bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.

Signal Conventions

$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.
lowercase_italics	Lowercase italics is used to indicate internal signals.
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration. For more information, see Section 3.2, “Configuration Signals Sampled at Reset.”

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document.

Table i. Acronyms and Abbreviated Terms

Term	Meaning
ADB	Allowable disconnect boundary
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CEPT	Conférence des administrations européenes des postes et télécommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
ECM	e500 coherency module
EHPI	Enhanced host port interface

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
GUI	Graphical user interface
I^2C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
IrDA	Infrared Data Association
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group
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
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OSI	Open systems interconnection

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
PCI	Peripheral component interconnect bus
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
POR	Power-on reset
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCC	Serial communication controller
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
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
TSEC	Three-speed Ethernet controller
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
UTP	Unshielded twisted pair

Table i. Acronyms and Abbreviated Terms (continued)

Term	Meaning
VA	Virtual address
ZBT	Zero bus turnaround

Part I
Overview

Part I describes the many features of the MPC8572E integrated host processor at an overview level. The following chapters are included:

Chapter 1, “Overview,” provides a high-level description of features and functionality of the MPC8572E integrated host processor. It describes the MPC8572E, its interfaces, and programming model. The functional operation of the MPC8572E, with emphasis on peripheral functions, is also described.

Chapter 2, “Memory Map,” describes the MPC8572E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of each.

Chapter 3, “Signal Descriptions,” provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).

Chapter 4, “Reset, Clocking, and Initialization,” describes the hard and soft resets, power-on reset (POR) sequence, power-on reset configuration, clocking, and initialization of the MPC8572E.

Chapter 1
Overview

The MPC8572E integrates two PowerPC™ processor cores with system logic required for networking, telecommunications, and wireless infrastructure applications. The MPC8572E is a member of the PowerQUICC III™ family of devices that combine system-level support for industry-standard interfaces with processors that implement the PowerPC architecture. This chapter provides a high-level description of features and functionality of the MPC8572E integrated processor.

1.1 Introduction

The MPC8572E uses e500v2 cores and high-speed interconnect technology to balance processor performance with I/O system throughput. The e500v2 core implements the enhanced Book E instruction set architecture and provides unprecedented levels of hardware and software debugging support.

In addition, the MPC8572E offers a double-precision floating-point auxiliary processing unit (APU), 1024 Kbytes of level-2 cache, four integrated 10/100/1Gb enhanced three-speed Ethernet controllers (eTSECs) with TCP/IP acceleration, classification capabilities, and SGMII interface capabilities, a 10/100 fast Ethernet controller (FEC) maintenance port, two table lookup units (TLUs), two DDR2/DDR3 SDRAM memory controllers, a multiprocessor programmable interrupt controller, two I^2C controllers, two four-channel direct memory access (DMA) controllers, an integrated security engine (SEC) with XOR acceleration, an enhanced local bus controller (eLBC), a pattern matching engine (PME) with DEFLATE capabilities, a general-purpose I/O port, and dual universal asynchronous receiver/transmitters (DUART). For high speed interconnect, the MPC8572E provides a set of multiplexed pins that support two high-speed interface standards: 1x/4x serial RapidIO (with message unit) and up to x8 PCI Express (with the capability to offer three independent PCI Express links). The high level of integration in the MPC8572E helps simplify board design and offers significant bandwidth and performance.

The MPC8572E is also available without a security engine, in a configuration known as the MPC8572. All specifications other than those relating to security apply to the MPC8572 exactly as described in this document.

1.2 MPC8572E Overview

This section provides a high-level overview of MPC8572E features. Figure 1-1 shows the major functional units within the MPC8572E.

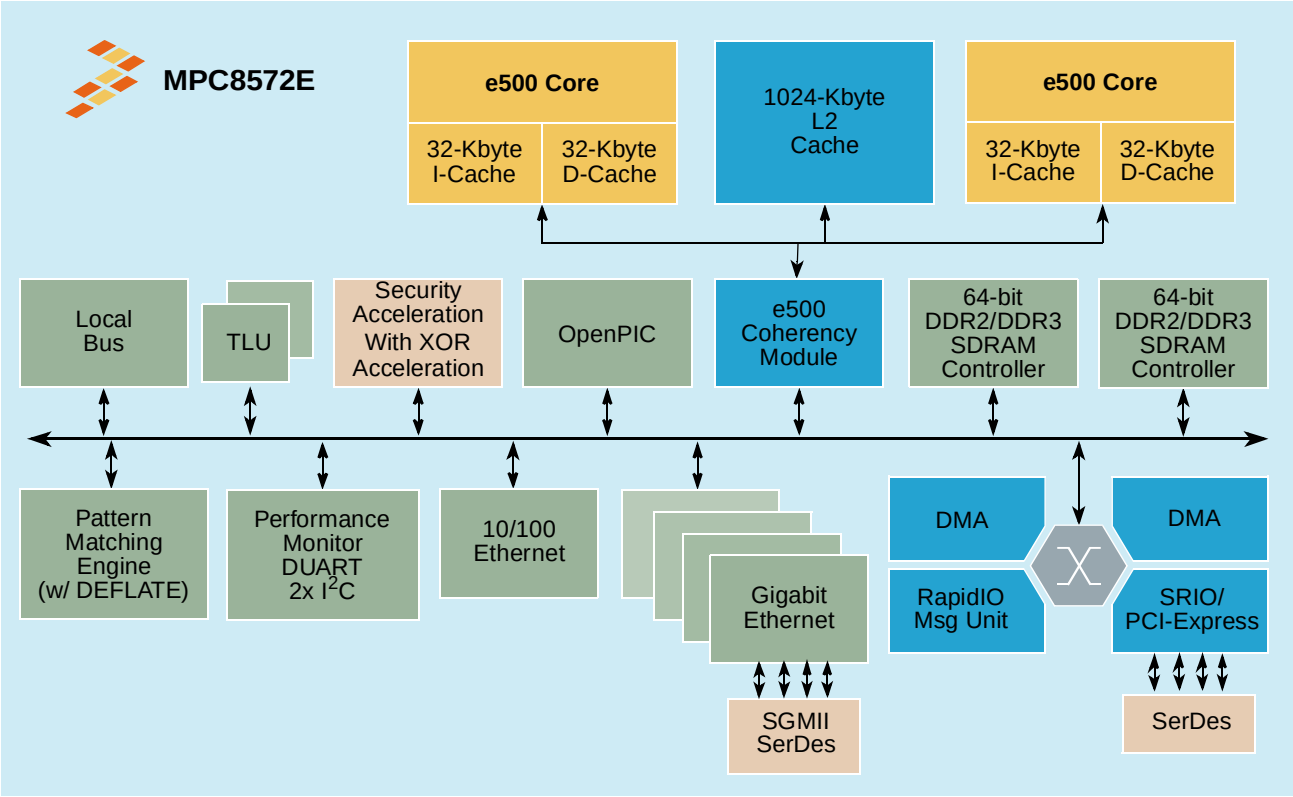


Figure 1-1. MPC8572E Block Diagram

1.2.1 Key Features

Key features of the MPC8572E device include:

- Two high-performance PowerPC e500v2 cores with 36-bit physical addressing
- 1024-Kbyte level 2 cache
- Integrated security engine (SEC) with XOR acceleration
- Four integrated 10/100/1Gb enhanced three-speed Ethernet controllers (eTSECs) with TCP/IP acceleration, classification capabilities, and SGMII interface capabilities
- 10/100 Fast Ethernet controller (FEC) maintenance interface
- Two DDR2/DDR3 SDRAM memory controllers
- High-speed interfaces:
  - Three PCI Express controllers
  - One serial RapidIO (SRIO) controller with RapidIO messaging unit
- Pattern matching engine (PME) with DEFLATE engine
- Two table lookup units (TLUs)

- Programmable interrupt controller (PIC)
- Two four-channel DMA controllers
- Two I^2C controllers
- DUART
- Enhanced local bus controller (eLBC)
- Eight general-purpose I/O signals
- Power management (power saving modes: doze, nap, and sleep)
- System performance monitor
- System access port (SAP)
- IEEE Std 1149.1™ compatible, JTAG boundary scan
- 1023 FC-PBGA package

1.3 MPC8572E Architecture Overview

The following sections describe the major functional units of the MPC8572E.

1.3.1 e500 Core Overview

This device uses the e500v2 microprocessor core complex. The e500v2 cores have internal PLLs that allow independent optimization of the operating frequencies. The core frequencies are derived from an external oscillator. For background information regarding the e500 core refer to the following documents:

- *EREF: A Reference for Freescale Book E and the e500 Core*
- *PowerPC e500 Core Family Reference Manual*
- *PowerPC e500 Application Binary Interface User's Guide*

NOTE

The e500 defines features that are not implemented on this device. It also generally defines some features that this device implements more specifically. An understanding of these differences can be critical to ensure proper operations.

Two high-performance, 32-bit, Book E–enhanced, e500v2 cores implement the PowerPC architecture. In addition to 36-bit physical addressing, this version of the e500 core includes:

- Double-precision floating-point APU with an instruction set for double-precision (64-bit) floating-point instructions that use the 64-bit general-purpose registers.
- Embedded vector and scalar single-precision floating-point APUs with an instruction set for single-precision (32-bit) floating-point instructions.

1.3.2 On-Chip Memory Unit

The MPC8572E also contains 1024 Kbytes of L2 cache/SRAM shared by both cores, as follows:

- Eight-way set-associative cache organization with 32-byte cache lines.
- Per-way allocation of cache region to a given processor
- Flexible configuration (1, 2, 4, or 8 ways can be configured as SRAM).
- External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing).
- SRAM:
  - I/O devices access SRAM regions by marking transactions as snoopable (global).
  - Regions can reside at any aligned location in the memory map.
  - Byte-accessible ECC uses read-modify-write accesses for smaller-than-cache-line accesses.

1.3.3 e500 Coherency Module (ECM)

To maintain coherency across local cacheable memory, the e500 coherency module (ECM) provides I/O-initiated transactions to snoop the bus between the e500 cores or between either core and the integrated L2 cache. Also, a flexible switch-type structure routes or dispatches core- and I/O-initiated transactions to to target modules on the device.

The flexible MPC8572E 36-bit physical address map consists of local space and external address space. For the local address map, twelve local access windows define mapping within the local 36-bit (64-Gbyte) address space. Inbound and outbound translation windows can map the MPC8572E 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).

1.3.4 DDR SDRAM Controllers

The two DDR memory controllers support DDR2 and DDR3 SDRAM. The memory interface controls main memory accesses and provides a maximum of 32 Gbytes of main memory for a total of 64 Gbytes with two memory controllers. The MPC8572E also supports chip-select interleaving and controller interleaving. There is a variety of MPC8572E SDRAM configurations. SDRAM banks can be built using DIMMs or directly-attached memory devices. Sixteen multiplexed address signals provide for device densities of 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbit, and 4 Gbit. Four chip-select signals per controller support up to eight total banks of memory. Bank sizes range from 64 Mbyte to 4 Gbyte. Nine column address strobes (D*n*_MDM[0:8]) provide byte selection for memory bank writes. There is cache line and page interleaving between memory controllers.

The MPC8572E can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 32 simultaneously open pages per controller can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, page mode can save 3 to 4 clock cycles for subsequent burst accesses that hit in an active page.

Using ECC, the MPC8572E detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

The MPC8572E can invoke a level of system power management by asserting the D*n*_MCKE SDRAM signal on-the-fly to put the memory into a low-power sleep mode.

The MPC8572E offers the following options to support battery-backed main memory:

- Hardware based

  An external voltage sense device is connected to the MPC8572E via an interrupt line. The external interrupt from this device is steered through the MPC8572E interrupt controller to the IRQ_OUT signal. The IRQ_OUT signal from the interrupt controller is steered to an enable bit in the DDR controller which immediately causes main memory to enter self-refresh mode.

  This proposal precludes any other simultaneous use of IRQ_OUT.
- Software based

  The DDR controller also has a software-programmable bit that immediately puts main memory into self-refresh mode.

  It is expected that a critical interrupt routine triggered by an external voltage sense device has time to set this bit.

The DDR controllers offer both hardware and software options for battery-backed main memory. 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.

1.3.5 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) implements the logic and programming structures of the OpenPIC architecture, providing for external interrupts (with fully nested interrupt delivery), message interrupts, internal-logic driven interrupts, and global high-resolution timers. Up to 16 programmable interrupt priority levels are supported. Inter-processor interrupt (IPI) communication allows one core to interrupt the other or either core to interrupt itself. The PIC can be bypassed in favor of an external interrupt controller.

1.3.6 Integrated Security Engine (SEC)

NOTE

The features described in this section are only available on the MPC8572E.
The MPC8572 does not include an integrated security engine.

The SEC 3.0 off-loads computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor cores of the MPC8572E. It is optimized to process all cryptographic algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, 802.11i, 3G, A5/3 for GSM and EDGE, and GEA3 for GPRS. The SEC 3.0 derives from integrated security cores in other members of the PowerQUICC family, including the SEC 2.1 Rev2 in the MPC8548E and the SEC 1.0 in the MPC8272E. Components of the SEC are as follows:

- XOR engine for parity checking in RAID storage applications. Also, the exclusive OR (XOR) operation to generate parity data in RAID applications can be accelerated. XOR operations use SEC descriptors and offload both parity generation and data movement from the e500 core.

- Four crypto-channels, each supporting multi-command descriptor chains.
- Eight cryptographic execution units:
  - Advanced Encryption Standard unit (AESU)
  - ARC four execution unit (AFEU)
  - Cyclic Redundancy Check Accelerator (CRCA)
  - Data Encryption Standard execution unit (DEU)
  - Kasumi execution unit (KEU)
  - Message digest execution unit (MDEU)
  - Public key execution unit (PKEU)
  - Random number generator (RNGB)

1.3.7 I^2C Controllers

The MPC8572E provides two inter-IC (IIC or I^2C) interfaces. The I^2C bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I^2C allows the MPC8572E to exchange data with other I^2C devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus (serial data SDA and serial clock SCL) minimizes the interconnections between devices. I^2C allows the connection of additional devices to the bus for expansion and system development.

The I^2C controller is a true multimaster bus which includes collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. This feature allows for complex applications with multiprocessor control. The I^2C controller consists of a transmitter/receiver unit, a clocking unit, and a control unit. The dual I^2C units support general broadcast mode, and on-chip filtering rejects spikes on the bus.

1.3.8 Boot Sequencer

The MPC8572E provides a boot sequencer that uses the I^2C1 interface to access an external serial ROM and loads the data into the MPC8572E configuration registers. The boot sequencer is enabled by a configuration pin sampled at the negation of the MPC8572E hardware reset signal. If enabled, the boot sequencer holds the MPC8572E processor cores in reset until the boot sequence is complete. If the boot sequencer is not enabled, the processor cores exit reset and fetches boot code in default configurations.

1.3.9 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8572E includes a DUART intended for use in maintenance, bringing-up, and debugging of systems. The MPC8572E provides a standard four-wire handshake (SIN, SOUT, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$) for each port. The DUART is a slave interface. An interrupt is provided to the interrupt controller or optionally steered externally to allow device handshakes. Interrupts are generated for transmit, receive, line status, and MODEM status.

The MPC8572E DUART supports full-duplex operation. It is compatible with the PC16450 and PC16550 programming models. Also, 16-byte FIFOs are supported for both the transmitter and the receiver.

Software programmable baud generators divide the system clock to generate a 16x clock. Serial interface data formats (data length, parity, 1/1.5/2 stop bits, baud rate) are also software selectable.

1.3.10 Enhanced Local Bus Controller

The enhanced local bus controller (eLBC) port connects to a wide 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 (GPCM) 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 GPCM, UPM, or FCM controller. All controllers can reside in the same system. Features of the local bus controller are as follows:

- Multiplexed 32-bit address and data bus
- Eight chip selects for eight external slaves
- Up to eight-beat burst transfers
- 32-, 16-, and 8-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, 16, or 32 bits)
- Zero-bus-turnaround (ZBT) RAM

1.3.11 Enhanced Three-Speed Ethernet Controllers (eTSECs)

Four MPC8572E on-chip enhanced three-speed Ethernet controllers (eTSECs) incorporate a media access control (MAC) sublayer that supports 10 and 100 Mbps and 1 Gbps Ethernet/802.3 networks with MII, RMII, GMII, RGMII, TBI, and RTBI physical interfaces as well as SGMII interfaces through a dedicated SerDes. The eTSECs include 2-Kbyte receive and 10-Kbyte transmit FIFOs and DMA functions.

The MPC8572E eTSECs support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. Frame headers and buffer descriptors (BDs) can be forced into the L2 cache to speed classification or other frame processing. They are designed to comply with IEEE Std 802.3™, 802.3u™, 802.3x™, 802.3z™, 802.3ac™, 802.3ab™. The BDs are based on the MPC8260 and MPC860T 10/100 Ethernet programming models. Each eTSEC provides hardware support for accelerating TCP/IP packet transmission and reception. By default, TCP/IP acceleration is not enabled and the eTSEC processes frames as pure Ethernet frames, emulating a PowerQUICC III TSEC and allowing existing driver software to be re-used with minimal change. Key features of these controllers include:

- Flexible configuration for multiple PHY interface configurations. The SGMII interface is available for any combination of eTSECs, regardless of the configuration of any other eTSEC.

NOTE

When enabled, the FEC occupies eTSEC3 and eTSEC4 parallel interface signals. In such a mode, eTSEC3 and eTSEC4 are *only* available via SGMII interfaces.

Table 1-1 lists available configurations for eTSECs 1 and 2.

Table 1-1. Supported eTSEC1 and eTSEC2 Configurations[1]

Mode Option	eTSEC1	eTSEC2
Ethernet standard interfaces	TBI, GMII, or MII	TBI, GMII, or MII
Ethernet reduced interfaces	RTBI, RGMII, or RMII	RTBI, RGMII, or RMII
FIFO and mixed interfaces	8-bit FIFO	RTBI, RGMII, RMII, or 8-bit FIFO
	TBI, GMII, MII, RTBI, RGMII, RMII, or 8-bit FIFO	8-bit FIFO
	16-bit FIFO	Not used/not available

1 Both interfaces must use the same voltage (2.5 or 3.3 V).

Table 1-2 lists the available configurations for eTSECs 3 and 4.

Table 1-2. Supported eTSEC3 and eTSEC4 Configurations[1]

Mode Option	eTSEC3	eTSEC4
Ethernet standard interfaces	TBI, GMII, or MII	Not used/not available
Ethernet reduced interfaces	RTBI, RGMII, or RMII	RTBI, RGMII, or RMII
FIFO interface	8-bit FIFO	Not used/not available

1 Both interfaces must use the same voltage.

- TCP/IP acceleration and QoS features:
  - IP v4 and IP v6 header recognition on receive
  - IP v4 header checksum verification and generation
  - TCP and UDP checksum verification and generation
  - Per-packet configurable acceleration
  - Recognition of VLAN, stacked (Q-in-Q) VLAN, 802.2, PPPoE session, MPLS stacks, and ESP/AH IP-security headers
  - All FIFO modes
  - Transmission from up to eight physical queues
  - Reception to up to eight physical queues
- Full- and half-duplex Ethernet support (1000 Mbps supports only full duplex): IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software-programmed PAUSE frame generation and recognition)
- IEEE Std 802.1™ virtual local area network (VLAN) tags and priority
- VLAN insertion and deletion
  - Per-frame VLAN control word or default VLAN for each eTSEC
  - Extracted VLAN control word passed to software separately
- Programmable Ethernet preamble insertion and extraction of up to 7 bytes
- MAC address recognition
- Can force allocation of header information and buffer descriptors into L2 cache

1.3.12 10/100 Fast Ethernet Maintenance Interface (FEC)

The MPC8572E offers a fast Ethernet controller (FEC) for use as a system maintenance interface. The FEC incorporates a media access control (MAC) sublayer that supports 10- and 100-Mbps Ethernet/802.3 networks with an MII physical interface.

As in the eTSEC controllers, the FEC buffer descriptors are based on the MPC8260 and MPC860T 10/100 Ethernet programming models, allowing existing driver software to be re-used with minimal change.

NOTE

Note that when enabled, the FEC occupies eTSEC3 and eTSEC4 parallel interface signals. In such a mode, eTSEC3 and eTSEC4 are only available via SGMII interfaces.

1.3.13 Pattern Matching Engine (PME) with DEFLATE Engine

The Pattern Matching Engine (PME) is capable of performing unanchored regular expression searches for up to 16,000 patterns. It offers support for built-in case insensitivity, repetition, nested rules, and both predefined and user-defined character classes. Additionally, the PME provides the ability to create stateful relationships between data examination events such as positive matches so that more complex regular expressions as well as certain other complex multi-pattern rules can be implemented. These stateful relationships allow the implementation, directly in silicon, of advanced searching such as contextual searches, where a positive match is reported only if the pattern appears in, for example, the header portion of the content, and ignored if it appears in the data portion; or the implementation of multi-pattern signatures, where a positive match is reported only if all patterns that constitute that signature are detected; or for protocol snooping, such as in tracking a SIP (session information protocol) session, where stateful relationships can be used to monitor the SIP exchange between a client and a server by searching for and correlating detected strings in the client-to-server direction (for example, INVITE) with detected strings in the server-to-client direction (for example, ACK), and then capturing the negotiated media channel.

The Deflate Engine can be used to expand data that has previously been compressed in the DEFLATE format, including gzip, zlib, and raw DEFLATE. It can be used in a standalone fashion, independent of the PME, or used in conjunction with the PME. In this latter case, software invokes the Deflate Engine and the PME through a single multifunction API call, instructing the hardware to DMA the compressed data to the Deflate Engine first, prior to performing pattern matching. After the Deflate Engine is finished processing, the resulting decompressed data is passed to the PME directly, without further software involvement or any additional memory access.

1.3.14 Table Lookup Unit (TLU)

The table lookup units (TLU) give access to application-defined routing topology, control, and statistics tables in external memory. Each TLU accesses external memory arrays attached either to the device DDR2/DDR3 memory controllers or to the enhanced local bus controller (eLBC). The cores and the TLUs communicate through messages passed among the memory-mapped configuration and status registers of each TLU. Each TLU uses a 64-bit wide data path for such register accesses.

The TLUs support several types of table lookup algorithms and provide resources for efficient generation of table entry addresses in memory, hash generation of addresses, and binary table searching algorithms for both exact-match and longest-prefix match strategies. The blocks of the TLU allow the implementation of a variety of table lookup algorithms to meet different application needs. In general, its functional blocks are organized in a basic loop that performs the following functions:

1. Parses a TLU command issued by software via registers
2. Calculates the initial index based on a key (for example, an initial hash value)
3. Evaluates the current table node as follows:
   — Fetches memory data at the current index
   — Fetches a portion of the key
   — If the lookup algorithm is complete, goes to step 5; otherwise goes to step 4.
4. Calculates a new index, and then goes back to step 3
5. Fetches the data at the current index
6. Returns the data to the CPU via key/data buffer registers

Each TLU has 32 physical tables, PTBL[0–31], each with an associated configuration register containing a physical table with a base table address configured by software. Also, in the configuration registers is the initial table format to expect, the method of generating the initial index, and how an index value maps to a physical memory address by selection of one of four external memory banks. This means that all table entries can be accessed by an index value, and individual bytes by an offset value. The entries can range from 8 bytes to 64 bytes in length.

1.3.15 OceaN Switch Fabric

In order to reduce the strain on core interconnects with the addition of new functional blocks in this generation of the PowerQUICC family, a multi-port, on-chip, non-blocking crossbar switch fabric called OCeaN (on-chip network) has been provided. The switch fabric serves to decrease contention and increase bandwidth. This non-blocking crossbar fabric allows full-duplex port communication with independent per-port transaction queuing and flow control.

1.3.16 Integrated DMA

Two MPC8572E DMA engines are capable of transferring blocks of data from any legal address range to any other legal address range. Therefore, they can perform DMA transfers between any I/O or memory ports or even between two devices or locations on the same port.

The four-channel DMA controllers allow chaining (both extended and direct) through local memory-mapped chain descriptors. Misaligned transfers are supported. In addition, advanced capabilities such as stride transfers and complex transaction chaining are supported.

DMA transfers can be initiated by a single write to a configuration register. There is also support for external control of transfers using $\overline{\text{DMA}n\text{\_DREQ}}$, $\overline{\text{DMA}n\text{\_DACK}}$, and $\overline{\text{DMA}n\text{\_DDONE}}$ handshake signals.

DMA descriptors encompass a rich set of attributes that allow DMA transfers to bypass outbound address translation and supply external addresses and attributes directly to the RapidIO port. Local attributes such as snoop and L2 write stashing can be specified by descriptors.

Interrupts are provided on a completed segment, link, list, chain, or on an error condition. Coherency is selectable and hardware enforced (snoop/no snoop).

1.3.17 High Speed I/O Interfaces

The MPC8572E supports two high-speed I/O interface standards: Serial RapidIO and PCI Express. Due to pin multiplexing, one of the following configurations must be selected at power-on reset:

- Single x4/x1 serial RapidIO interface
- Single x4/x1 serial RapidIO interface AND single x4/x2/x1 PCI Express interface
- Single x8/x4/x2/x1 PCI Express interface
- Dual x4/x2/x1 PCI Express interfaces
- Single x4/x2/x1 AND dual x2/x1 PCI Express interfaces

1.3.18 Serial RapidIO Interface

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 RapidIO architecture provides a rich variety of features including high data bandwidth, low-latency capability, and support for high-performance I/O devices, as well as providing message-passing and software-managed programming models.

The RapidIO unit supports the I/O and message-passing logical specifications, both 8- and 16-bit common transport specifications, and the 1x/4x LP-Serial physical layer specification of the *RapidIO Interconnect Specification, Revision 1.2*. It does not support the globally shared memory logical specification.

Highlights of the implementation include: four priority levels and ordering within a priority level, CRC error management, and 32- to 256-byte transactions.

The physical layer of the RapidIO unit can operate at 1.25-, 2.5-, and 3.125-Gbaud data rates over four lanes. The theoretical unidirectional peak bandwidth is 10 Gbps. Receive and transmit ports operate independently, resulting in an aggregate theoretical bandwidth of 20 Gbps.

1.3.19 RapidIO Message Unit

The RapidIO messaging unit supports two inbox/outbox mailboxes (queues) for data and one doorbell message structure. Both chaining and direct modes are provided for the outbox, and messages can hold up to 16 packets of 256 bytes, or a total of 4 Kbytes.

The message unit supports up to three outstanding letters from one or more of the four mailboxes. The message unit pipelines letters to improve messaging performance.

The message unit also supports the ability to multicast a single-segment 256-byte message to up to 32 different destination DevIDs.

1.3.20 PCI Express Interfaces

Each of up to three PCI Express interfaces is compliant with the *PCI Express Base Specification Revision 1.0a*. Power-on reset configuration options allow root complex or endpoint functionality.

Each interface is selectable at boot time to support either 32 or 64-bit addressing. The maximum supported packet payload size is 256 bytes.

The physical layer supports x8, x4, x2, and x1 lane widths with each lane running at the specified data rate of 2.5 Gbaud.

Each interface supports virtual channel 0 (VC0) and traffic class 0 (TC0) only.

Inbound INTx transactions are supported and change the state of one of four level-sensitive interrupts presented to the PIC. Outbound INTx transactions are not supported.

Message signaled interrupt (MSI) transactions are supported and control up to 256 interrupt sources within the PIC. Inbound transactions cause specific edge-triggered interrupt sources to be controlled within the PIC. Outbound MSI transactions are created by software using the MSI Capability Register Sets.

The physical layer of the PCI Express interface operates at a 2.5-Gbaud data rate per lane. The theoretical unidirectional peak bandwidth is 16 Gbps. Receive and transmit ports operate independently, resulting in an aggregate theoretical bandwidth of 32 Gbps.

1.3.21 Power Management

In addition to low-voltage operation and dynamic power management in its execution units, the MPC8572E supports four power consumption modes: full on, doze, nap, and sleep. The three low-power modes, doze, nap, and sleep, can be entered under software control in the e500 cores or by external masters accessing a configuration register. Note that doze and nap modes are controlled per core and sleep mode is a device-wide low-power state.

Doze mode suspends execution of instructions in an e500 core. The core is left in a standby mode in which cache snooping and time base interrupts are still enabled. Device logic external to the processor core is fully functional in this mode.

Nap mode shuts down clocks to all the e500 functional units except the time base, which can be disabled separately. No snooping is performed in nap mode, but the device logic external to the processor cores is fully functional.

Sleep mode shuts down not only the e500 cores, but also all of the MPC8572E I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the device can be awakened.

1.3.22 Clocking

The MPC8572E takes in the SYSCLK signal as an input to the platform PLL and multiplies it to generate the platform clock, which operates at the same frequency as the SDRAM data rate when the DDR controllers are configured to operate in synchronous mode (for example, 600 MHz). The L2 cache also operates at this frequency. Note that the DDRCLK input sources the DDR complex PLL for use when operating the DDR complex in asynchronous mode. The e500 cores use the platform clock as an input to

their PLLs, which multiply it again to generate the core clocks. Each core's multiplier may be set independently such that each core may operate at a different frequency.

Although the DDR controller clocking is source synchronous, a PLL is used in the enhanced local bus memory controller (eLBC) to generate memory clocks. Six differential clock pairs are generated per DDR controller for DDR SDRAMs. Three clock outputs are generated for the enhanced local bus controller.

1.3.23 Address Map

The MPC8572E supports a flexible 36-bit physical address map. Conceptually, the address map consists of local space and external address space. The local address map is 64 Gbytes. The MPC8572E can be made part of a larger system address space through the mapping of translation windows. This functionality is included in the address translation and mapping units (ATMUs). Both inbound and outbound translation windows are provided. The ATMUs allows the MPC8572E to be part of larger address maps such as the PCI Express 64-bit address environment and the RapidIO environment.

1.3.24 Processing Across the On-Chip Fabric

When processing across the on-chip fabric, the ATMUs at each fabric port are used to determine the flow of data across the MPC8572E. The ATMUs at each fabric port are responsible for generating a fabric port destination ID as well as a new local device address. The port ID and local address are based on the programmed destination of the transaction. The following is a general overview of how the ATMUs process transactions over the on-chip fabric (refer to Figure 1-2).

1. When a transaction on one of the fabric ports begins, the ATMU on the origination port translates the programmed destination address into both a destination fabric port ID and a local device address.
2. The data is then processed across the on-chip fabric from the origination port to the destination port.
3. If the destination port connects off chip (for example, to a PCI Express or RapidIO device), the local device address is translated by the destination port ATMU to an outbound address with respect to the destination port's memory map, and the data is processed accordingly.

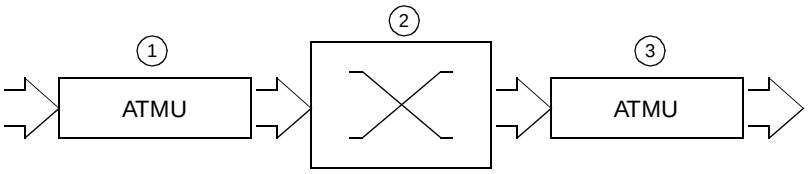


Figure 1-2. Processing Transactions Across the On-Chip Fabric

1.3.25 Data Processing with the e500 Coherency Module

Processing via the ECM is similar to processing across the on-chip fabric (in the sense of how data is received and transmitted) with the exception that the transaction passes through the ECM. The purpose of the ECM is to provide a means for any I/O transaction to maintain coherency with the cacheable DDR SDRAM and the local bus memory. However, simply because the ECM is used does not make transactions across it coherent. The e500 cores and L2 cache are snooped to maintain coherency only if the transaction

across the ECM is designated as global from the perspective of the cores or snoopable from the perspective of platform I/O masters (GBL bit set). Otherwise, the transaction passes through the ECM using the ECM as a simple conduit to get to its destination. In essence, only global or snoopable transactions across the ECM are coherent transactions; all others (across the on-chip fabric) are non-coherent.

1.4 MPC8572E Application Examples

The MPC8572E is a very flexible device that can be configured to meet many system application needs.

Both MPC8572E cores can operate in a symmetric multiprocessing mode to achieve higher performance, or they can perform separate tasks, potentially running independent operating systems. This flexibility enables application developers to assign distinct processing resources to distinct tasks that need guaranteed performance. For example, one core can manage a data plane and the other a control plane.

The value proposition of a dual-core device is further enhanced by a high degree of peripheral integration on system controllers such as DDR controllers. A device with faster internal buses can entirely replace the system controller where a discrete processor without integration was used previously.

Many applications use two processors running in symmetric multiprocessor mode connected to a system controller by a system bus, as shown in Figure 1-3. A single dual-core integrated device can replace both processors, saving board space and significantly enhancing performance.

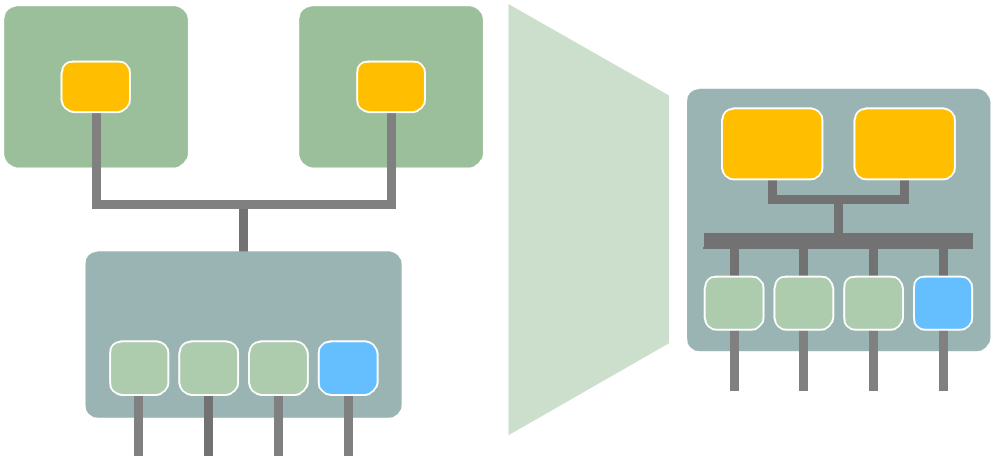


Figure 1-3. Two Processors Running in Symmetric Multiprocessor Mode

1.4.1 Dual-Core Device Application

There are two main ways to map operating systems to the two MPC8572E cores, as follows:

- Symmetric multiprocessing
- Cooperative asymmetric multiprocessing
  - Two copies of the same OS that are non-SMP enabled
  - Two separate operating systems

Figure 1-4 shows how to use an integrated dual-core device such as the MPC8572E. A high-end line card uses an ASIC or network processing unit (NPU) for the data path. The ASIC/NPU manages user interfaces on the faceplate on the left as well as the interface to the backplane on the right. The dual-core device is responsible for the control plane. The two cores can operate in an SMP configuration, or two separate operating systems can be used for separate control plane tasks.

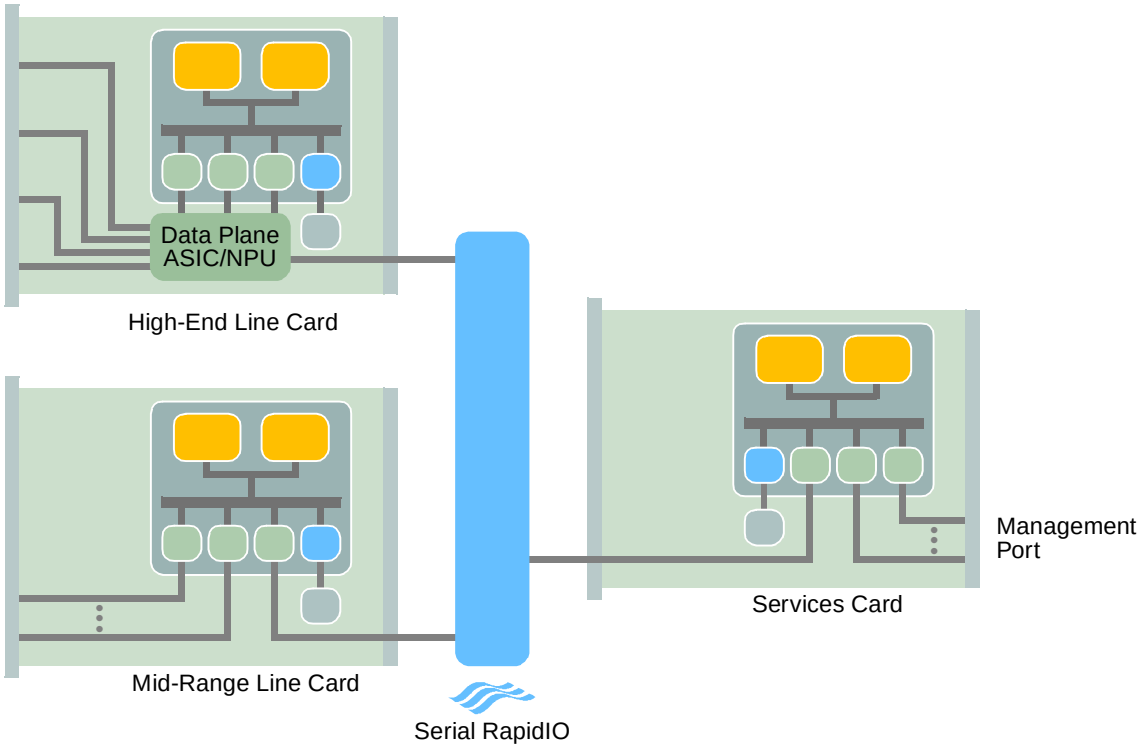


Figure 1-4. Integrated Dual Core Device Application

The lower line card shows a mid-range system. Here, the eTSEC interfaces are the faceplate interfaces, and the serial RapidIO® block connects to the backplane switch fabric. The CPUs handle both the control and data plane in a variety of configurations. Two popular alternatives split functionality directionally (one core per direction) or split functionality vertically (one core handling the data plane, one core handling the control plane).

At the right of the switch fabric is an example service card, which is easily added to a system to add new features without replacing all the line cards with upgraded ASICs. One service card supports a new feature set in a centralized scheme, receiving traffic from all line cards, so the high performance of a dual-core device is required. The serial RapidIO port connects to the fabric, and the eTSECs implement a management interface.

1.4.2 Virtual Private Network (VPN) Access Router

Figure 1-5 shows a virtual private network (VPN) access router enabled through PCI Express and Ethernet.

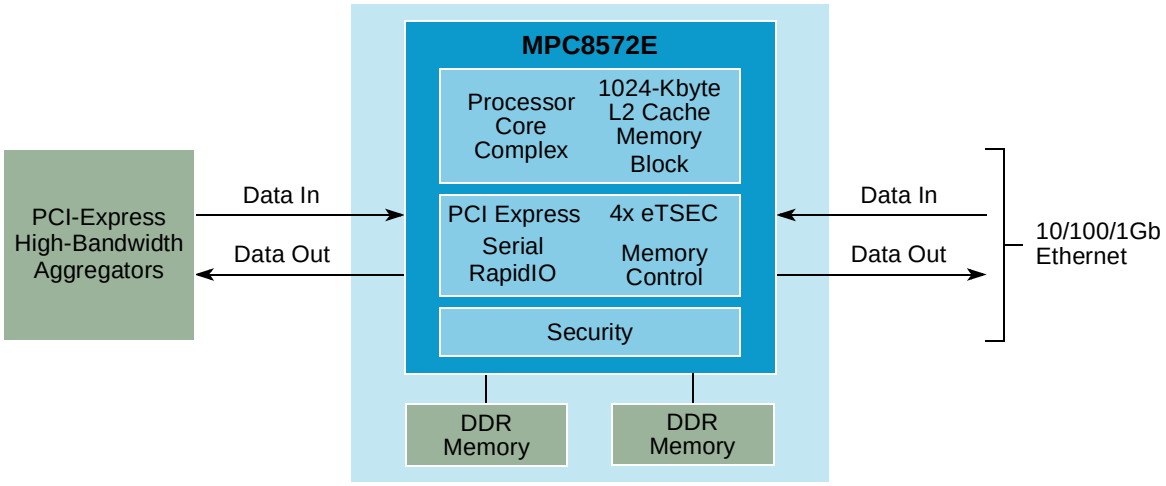


Figure 1-5. VPN Access Router Enabled by PCI Express and Ethernet

1.4.3 High-Performance Communication System Using Distributed Processing

Figure 1-6 shows an MPC8572E as a control plane processor in a high-performance communication system.

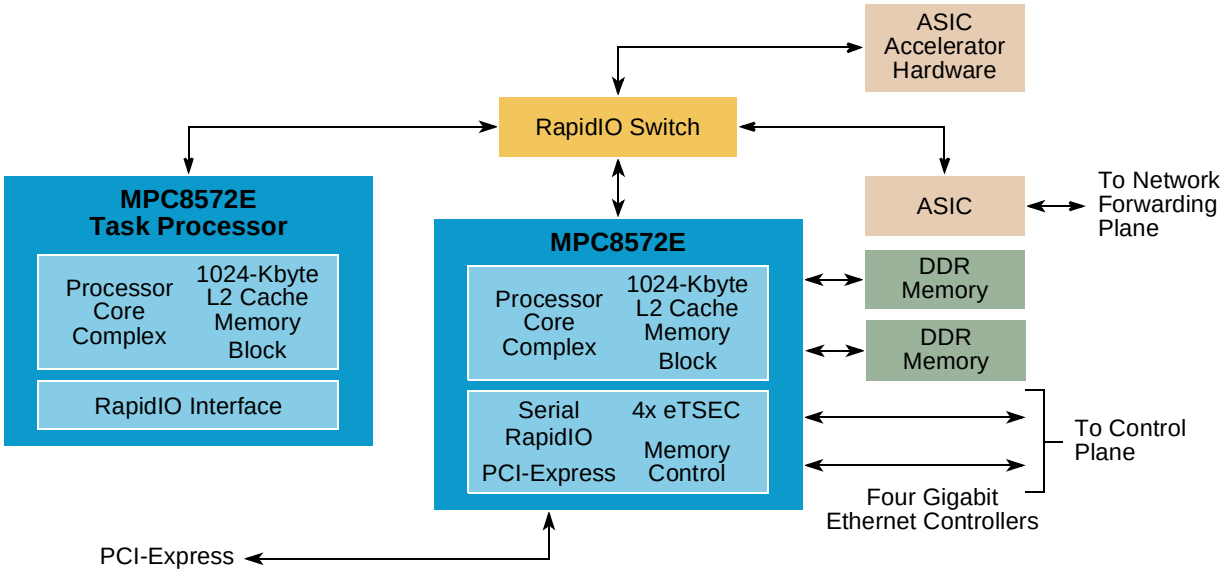


Figure 1-6. High-Performance Communication System Using Distributed Processing

1.4.4 High-Performance Communication System

Figure 1-7 shows the MPC8572E as part of a high-end network card used in a system area network that is enabled by PCI Express.

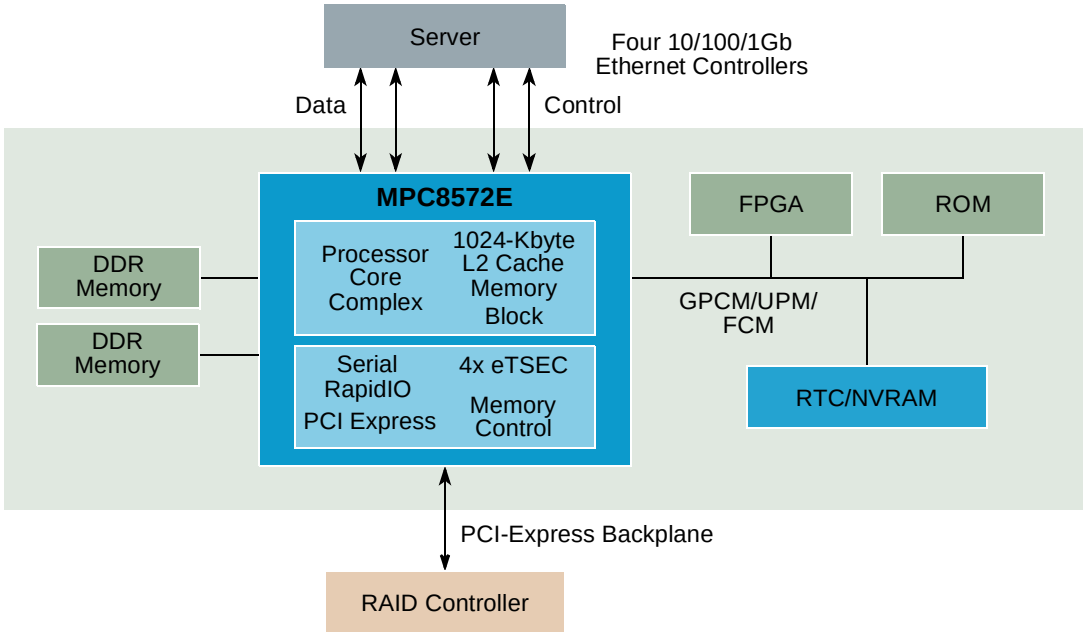


Figure 1-7. High-Performance Communication System

1.4.5 RAID Controller Application

Figure 1-8 shows the MPC8572E in a redundant array of independent disks (RAID) controller application.

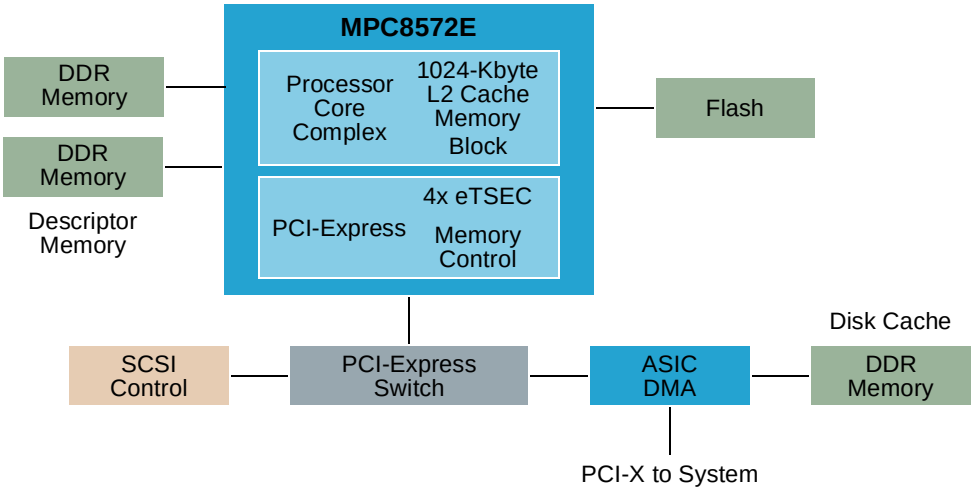


Figure 1-8. RAID Controller Application Using MPC8572E

1.4.6 SerDes Application

Figure 1-9 shows the MPC8572E in a serializer/deserializer (SerDes) data transfer mode application enabled by Ethernet.

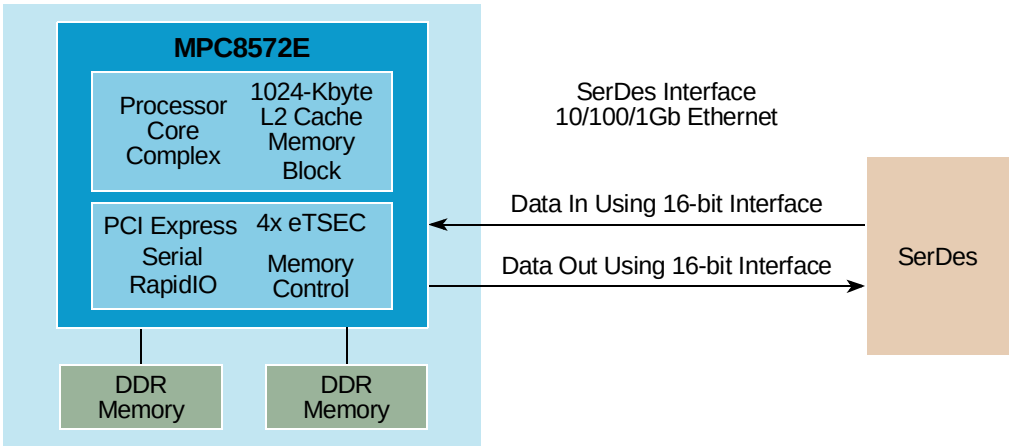


Figure 1-9. MPC8572E with SerDes

1.4.7 DSP Farm Application

Figure 1-10 shows a DSP farm enabled by the MPC8572E.

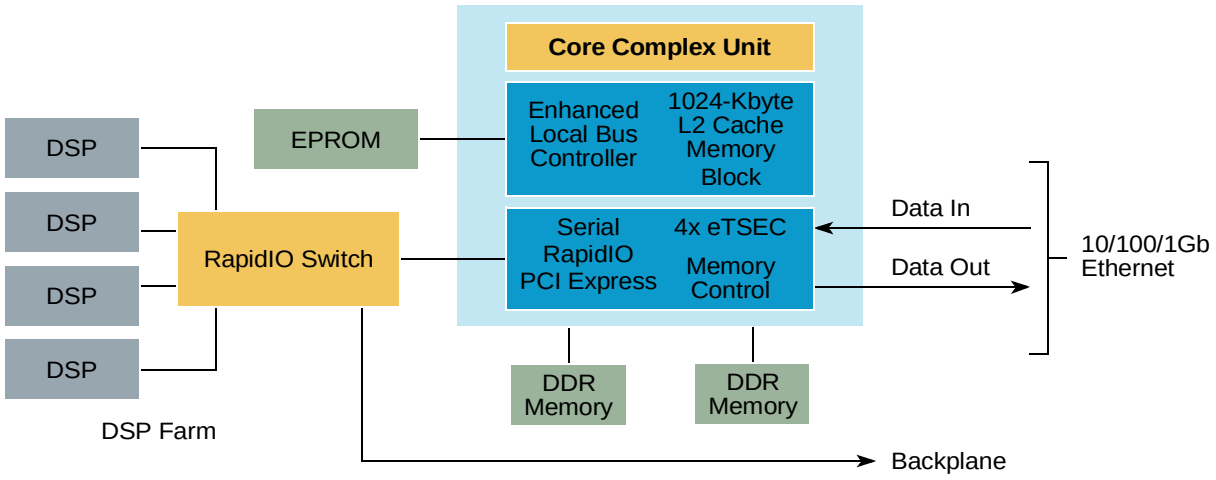


Figure 1-10. MPC8572E as a Control Plane Processor for a DSP Farm Interconnected with RapidIO

Chapter 2
Memory Map

This chapter describes the MPC8572E memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.

2.1 Local Memory Map Overview and Example

The MPC8572E provides an extremely flexible local memory map. The local memory map refers to the 36-bit address space seen by the processors as it accesses memory and I/O space. DMA engines also see this same local memory map. All memory accessed by the MPC8572E DDR SDRAM and local bus memory controllers exists in this memory map, as do all memory-mapped configuration, control, and status registers.

The local memory map is defined by a set of twelve local access windows. Each of these windows maps a region of memory to a particular target interface, such as one of the DDR SDRAM controllers or PCI Express controllers. Note that the local access windows do not perform any address translation. The size of each window can be configured from 4 Kbytes to 32 Gbytes. The target interface is specified using the codes shown in Table 2-1.

Table 2-1. Target Interface Codes

Target Interface	Target Code
PCI Express 3	00000
PCI Express 2	00001
PCI Express 1	00010
Local bus	00100
Interleaved DDR SDRAM	01011
Serial RapidIO	01100
DDR SDRAM 1	01111
DDR SDRAM 2	10110

Figure 2-1 shows an example memory map.

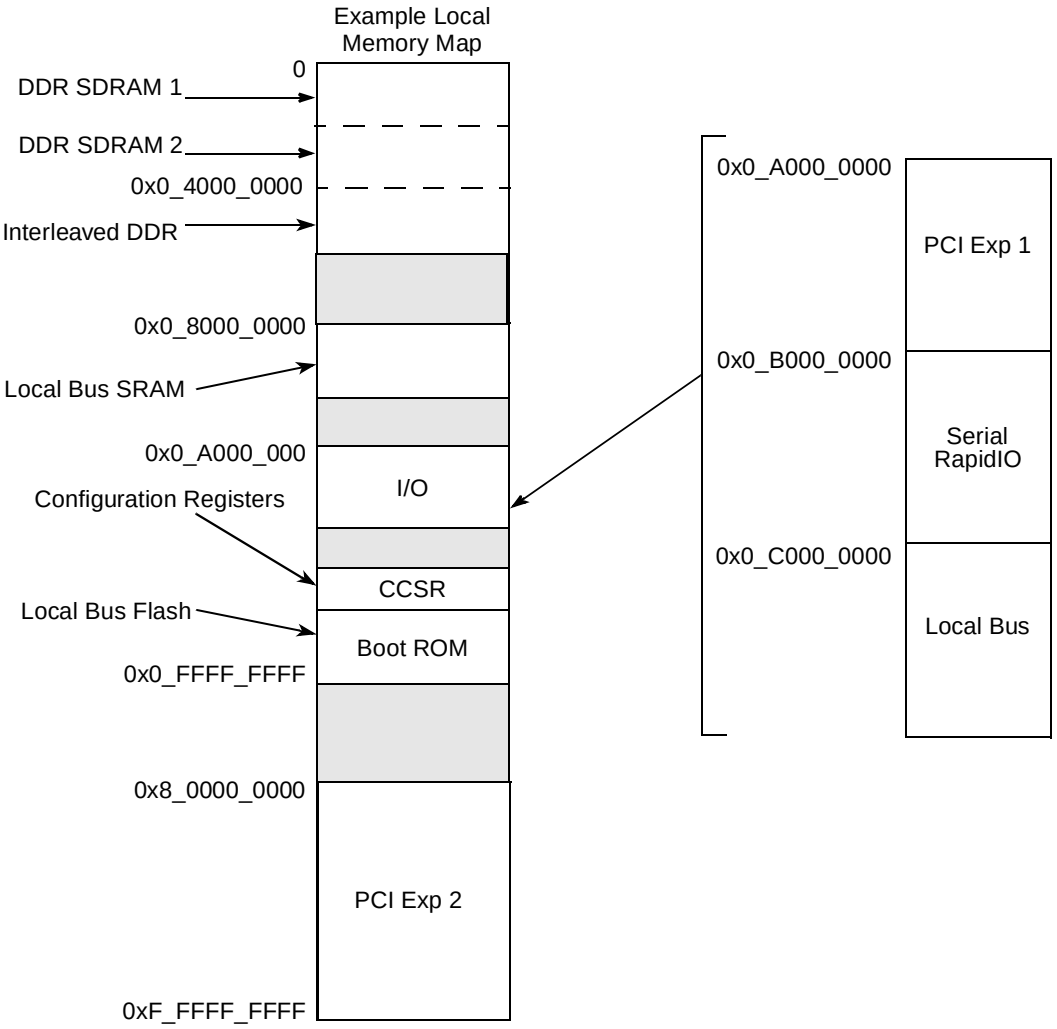


Figure 2-1. Local Memory Map Example

Table 2-2 shows one corresponding set of local access window settings.

Table 2-2. Local Access Windows Example

Window	Base Address	Size	Target Interface
0	0x0_0000_0000	512 Mbytes	0b01111 (DDR SDRAM 1)
1	0x0_2000_0000	512 Mbytes	0b10110 (DDR SDRAM 2)
2	0x0_4000_0000	512 Mbytes	0b01011 (Interleaved DDR)
3	0x0_8000_0000	512 Mbytes	0b00100 (Local Bus)
4	0x0_A000_0000	256 Mbytes	0b00010 (PCI Express 1)
5	0x0_B000_0000	256 Mbytes	0b01100 (Serial RapidIO)
6	0x0_C000_0000	256 Mbytes	0b00100 (Local Bus)

Table 2-2. Local Access Windows Example (continued)

Window	Base Address	Size	Target Interface
7	0x8_0000_0000	32 Gbytes	0b0001 (PCI Express 2)
8–11	Unused		

In this example, it is not necessary to use a local access window to specify the range of memory used for memory-mapped registers because this is a fixed 1-Mbyte space pointed to by CCSRBAR. See Section 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)." Neither is it required to define a local access window to describe the location of the boot ROM because it is in the default location (see Section 4.4.3.4, "Boot ROM Location"). However, note that the e500 cores only provides one default TLB entry for each core to access boot code and it allows for accesses within the highest 4 Kbytes of the low 4 Gbytes of memory. In order for the e500 cores to access the full 8 Mbytes of default boot space (and the 1 Mbyte of CCSR space), additional TLB entries must be set up within the e500 cores' MMU for mapping these regions.

2.2 Address Translation and Mapping

Four distinct types of translation and mapping operations are performed on transactions in the MPC8572E. These are as follows:

- Mapping a local address to a target interface
- Assigning attributes to transactions
- Translating the local 36-bit address to an external address space
- Translating external addresses to the local 36-bit address space

The local access windows perform target mapping for transactions within the local address space. No address translation is performed by the local access windows.

Outbound ATMU windows perform the mapping from the local 36-bit address space to the address spaces of RapidIO or PCI Express, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

Inbound ATMU windows perform the address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a similar function as the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the local access windows. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a local access window must also map that same local address to the same target.

All of the configuration registers that define translation and mapping functions use the concept of translation or mapping windows, and all follow the same register format. Table 2-3 summarizes the general format of these window definitions.

Table 2-3. Format of ATMU Window Definitions

Register	Function
Translation address	High-order address bits defining location of the window in the target address space
Base address	High-order address bits defining location of the window in the initial address space
Window size/attributes	Window enable, window size, target interface, and transaction attributes

Windows must be a power-of-two size. 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.

2.2.1 SRAM Windows

The on-chip memory array of the MPC8572E can be configured as a memory-mapped SRAM ranging from 128 Kbytes to 1 Mbyte. Configuration registers in the L2 cache controller set the base addresses and sizes for these windows. When enabled, these windows supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR. It is possible to have SRAM windows overlap local access windows, but this is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if the SRAM window overlaps a local access window.

See Section 7.3.1.4.1, "L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBARn)," for information about configuring SRAM windows.

2.2.2 Window into Configuration Space

CCSRBAR defines a window used to access all memory-mapped configuration, control, and status registers. No address translation is done, so there are no associated translation address registers. The window is always enabled with a fixed size of 1 Mbyte; no other attributes are attached, so there is no associated size/attribute register. This window always takes precedence over all local access windows. See Section 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)," and Section 2.3, "Configuration, Control, and Status Register Map."

2.2.3 Local Access Windows

As demonstrated in the address map overview in Section 2.1, "Local Memory Map Overview and Example," local access windows associate a range of the local 36-bit address space with a particular target interface. This allows the internal interconnections of the MPC8572E to route a transaction from its source to the proper target. No address translation is performed. The base address defines the high order address bits that give the location of the window in the local address space. The window attributes enable the window, define its size, and specify the target interface.

With the exception of configuration space (mapped by CCSRBAR), on-chip SRAM regions (mapped by L2SRBAR registers), and default boot ROM, all addresses used by the system must be mapped by a local access window. This includes addresses that are mapped by inbound ATMU windows; target mappings of inbound ATMU windows and local access windows must be consistent.

The local access window registers exist as part of the local access block in the general utilities registers. See Section 2.3.4, "General Utilities Registers." A detailed description of the local access window registers is given in the following sections. Note that the minimum size of a window is 4 Kbytes, so the low order 12 bits of the base address cannot be specified.

2.2.3.1 Local Access Register Memory Map

Table 2-4 shows the memory map for the local access registers. In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 2-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0BF8	LAIPBRR1—Local access IP block revision register 1	R	0x0000_0000	2.2.3.2/2-6
0x0_0BFC	LAIPBRR2—Local access IP block revision register 2	R	0x0000_0000	2.2.3.3/2-6
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.4/2-7

Table 2-4. Local Access Register Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D28	LAWBAR9—Local access window 9 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D30	LAWAR9—Local access window 9 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D48	LAWBAR10—Local access window 10 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D50	LAWAR10—Local access window 10 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D68	LAWBAR11—Local access window 11 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D70	LAWAR11—Local access window 11 attribute register	R/W	0x0000_0000	2.2.3.5/2-8

2.2.3.2 Local Access IP Block Revision Register 1 (LAIPBRR1)

The local access IP block revision register 1 is shown in Figure 2-2.

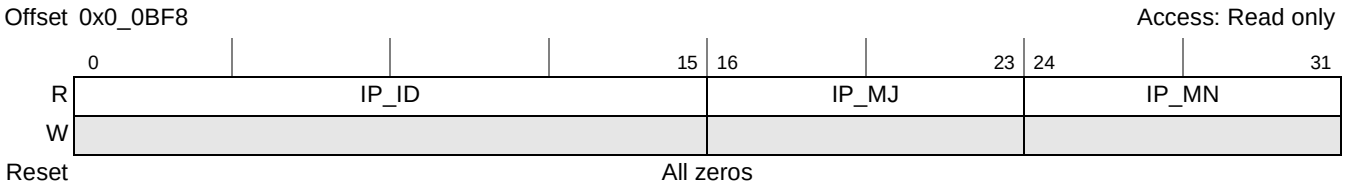


Figure 2-2. Local Access IP Block Revision Register 1 (LAIPBRR1)

Table 2-5 describes LAIPBRR1 fields.

Table 2-5. LAIPBRR1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID
16–23	IP_MJ	Major revision
24–31	IP_MN	Minor revision

2.2.3.3 Local Access IP Block Revision Register 2 (LAIPBRR2)

The local access IP block revision register 2 is shown in Figure 2-3.

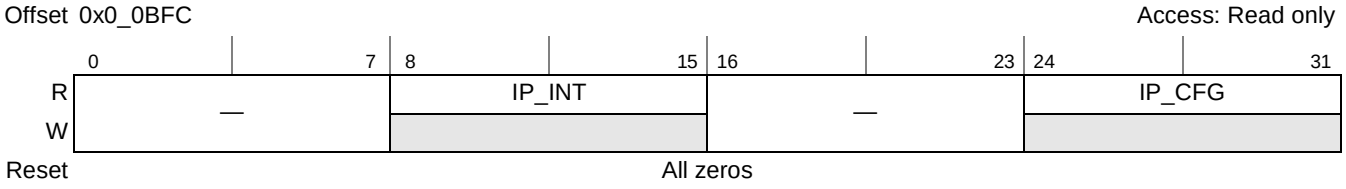


Figure 2-3. Local Access IP Block Revision Register 2 (LAIPBRR2)

Table 2-6 describes LAIPBRR2 fields.

Table 2-6. LAIPBRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

2.2.3.4 Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR11)

Figure 2-4 shows the bit fields of the LAWBAR*n* registers.

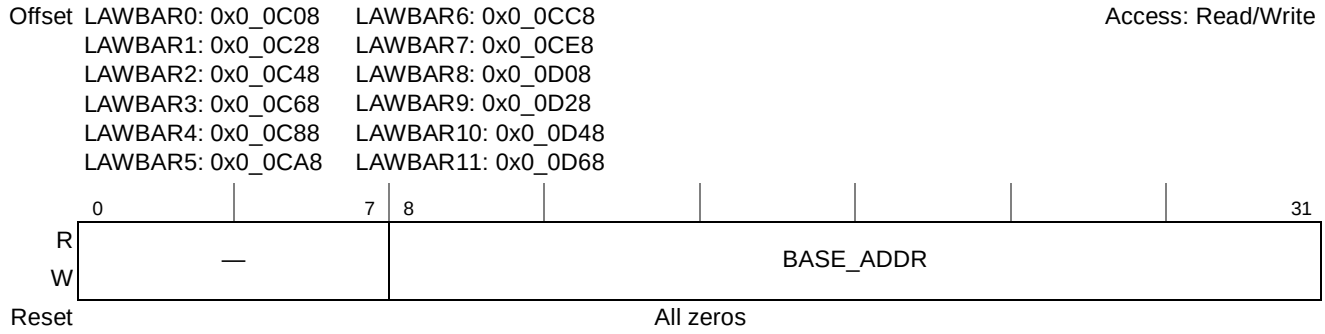


Figure 2-4. Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR11)

Table 2-7 describes LAWBAR*n* fields.

Table 2-7. LAWBAR*n* Field Descriptions

Bits	Name	Description
0–7	—	Write reserved, read = 0
8–31	BASE_ADDR	Identifies the 24 most significant address bits of the base of local access window *n*. The specified base address should be aligned to the window size, as defined by LAWAR*n*[SIZE].

2.2.3.5 Local Access Window *n* Attributes Registers (LAWAR0–LAWAR11)

Figure 2-5 shows the bit fields of the LAWAR*n* registers.

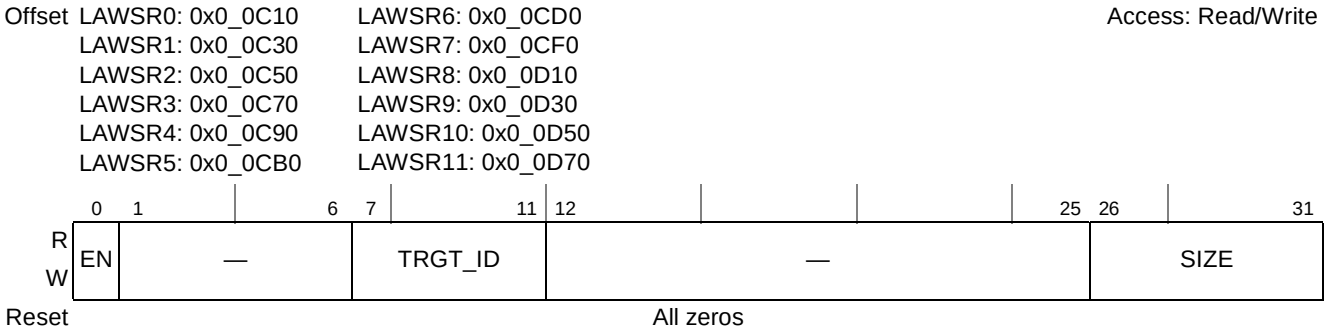


Figure 2-5. Local Access Window *n* Attributes Registers (LAWAR0–LAWAR11)

Table 2-8 describes LAWAR*n* fields.

Table 2-8. LAWAR*n* Field Descriptions

Bits	Name	Description
0	EN	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.
1–6	—	Write reserved, read = 0
7–11	TRGT_IF	Identifies the target interface ID when a transaction hits in the address range defined by this window. Note that configuration registers and SRAM regions are mapped by the windows defined by CCSRBAR and L2SRBAR. These mappings supersede local access window mappings, so configuration registers and SRAM do not appear as a target for local access windows. 00000 PCI Express 3 00001 PCI Express 2 00010 PCI Express 1 00011 Reserved 00100 Local bus memory controller 00101–01010 Reserved 01011 Interleaved DDR SDRAM 01100 Serial RapidIO 01101–01110 Reserved 01111 DDR SDRAM 1 10110 DDR SDRAM 2
12–25	—	Write reserved, read = 0
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 100010 32 Gbytes 100011–111111 Reserved

2.2.3.6 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence. For instance, if two windows are set up as shown in Table 2-9, local access window 1 governs the mapping of the 1-Mbyte region from 0x0_7FF0_0000 to 0x0_7FFF_FFF, even though the window described in local access window 2 also encompasses that memory region.

Table 2-9. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
1	0x0_7FF0_0000	1 Mbyte	0b00100 (Local bus controller)
2	0x0_0000_0000	2 Gbytes	0b01111 (DDR SDRAM 1)

2.2.3.7 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 local access window configuration register before enabling any other devices to use the window. For instance, if local access windows 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 done by one of the local e500 processors, the read of LAWAR3 should be followed by an **isync** instruction.

2.2.3.8 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the local access windows and the additional mapping functions that happen at the target interface. The local access windows define how a transaction is routed through the MPC8572E internal interconnects from the transactions source to its target. After the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR SDRAM controllers have chip select registers that map a memory request to a particular external device. Similarly, the local bus controller has base registers that perform a similar function. The RapidIO and PCI Express interfaces have outbound address translation and mapping units that map the local address into an external address space.

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

2.2.3.9 Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects

If a local access window maps an address to an interface other than the DDR SDRAM controllers, then there should not be a valid chip select configured for the same address in either of the DDR SDRAM

controllers. Because DDR SDRAM chip selects boundaries are defined by a beginning and ending address, it is easy to define them so that they do not overlap with local access windows that map to other interfaces.

2.2.4 Outbound Address Translation and Mapping Windows

Outbound address translation and mapping refers to the translation of addresses from the local 36-bit address space to the external address space and attributes of a particular I/O interface. On the MPC8572E, the following blocks have outbound address translation and mapping units (ATMUs):

- Serial RapidIO
- PCI Express

The serial RapidIO controller has eight outbound ATMU windows plus a default window. If a transaction's address does not hit any of the eight outbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. See Section 20.6.7, "RapidIO Implementation Space—ATMU Registers," for a detailed description of the serial RapidIO outbound ATMU windows.

The PCI Express interfaces each have four outbound ATMU windows plus a default window. The PCI Express outbound ATMU registers include an extended translation address register so that up to 64 bits of external address space can be supported. See Section 21.3.5.1, "PCI Express Outbound ATMU Registers," for a detailed description of the PCI Express outbound ATMU windows.

2.2.5 Inbound Address Translation and Mapping Windows

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface (such as PCI Express or RapidIO address space) to the local 36-bit address space understood by the internal interfaces of the MPC8572E. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. On the MPC8572E, the following blocks have inbound address translation and mapping units (ATMUs):

- Serial RapidIO
- PCI Express

2.2.5.1 Serial RapidIO Inbound ATMU

The serial RapidIO controller has four inbound ATMU windows plus a default. If the inbound transaction's address does not hit any of the four inbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. See Section 20.8.6, "RapidIO Inbound ATMU," for a detailed description of the serial RapidIO inbound ATMU windows.

2.2.5.2 PCI Express Inbound ATMU

Each of the three PCI Express controllers have three inbound ATMU windows plus a default. See Section 21.3.5.2, "PCI Express Inbound ATMU Registers," for a detailed description of the PCI Express outbound ATMU windows.

2.2.5.3 Illegal Interaction Between Inbound ATMUs and Local Access Windows

Because both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another. For instance, it is a programming error to have an inbound ATMU map a transaction to local memory space (target interface 0b1111) if the resulting translated local address is mapped to PCI Express 1 (target interface 0b00010) by a local access window. Such a programming error may result in unpredictable system deadlocks.

2.3 Configuration, Control, and Status Register Map

All of the memory mapped configuration, control, and status registers in the MPC8572E are contained within a 1-Mbyte address region. To allow for flexibility, the configuration, control, and status block 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 Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register (CCSRBAR).” The default value for CCSRBAR is 4 Gbytes–9 Mbytes, or 0x0_FF70_0000.

NOTE

The configuration, control, and status window must not overlap a local access window that maps to either of the DDR controllers. Otherwise, undefined behavior occurs.

An example of a top-level memory map with the default location of the configuration, control, and status registers is shown in Figure 2-6.

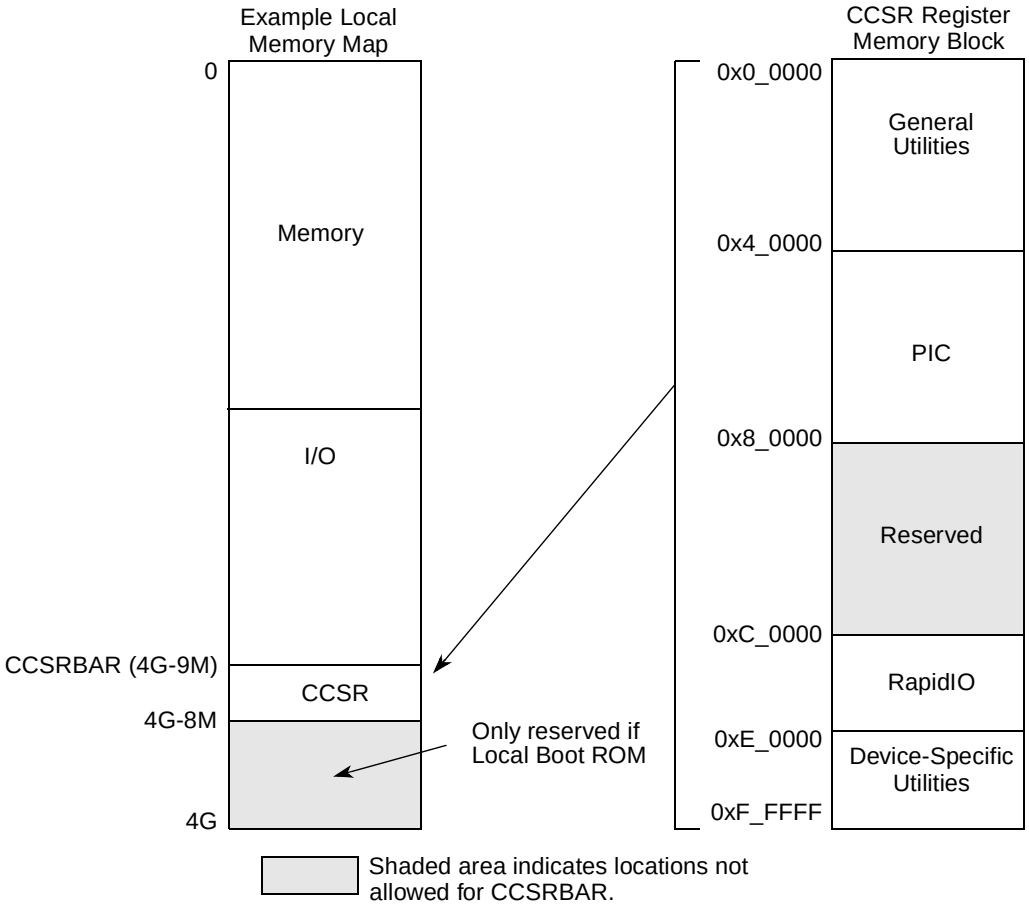


Figure 2-6. Top-Level Register Map Example

2.3.1 Accessing CCSR Memory from the Local Processor

When the local e500 processors 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 chased 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.

2.3.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the e500 processors, the configuration, control, and status registers are accessible from external interfaces. This allows external masters on the I/O ports to configure the MPC8572E.

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 interfaces' base address for accessing the local CCSR memory is selectable through each of the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10, described in Section 21.3.8.2.1, "PCI Express Base Address Registers—0x10–0x27." An external PCI Express master sets this register by running a PCI Express configuration transaction to the MPC8572E. 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 Section 20.6.1.11, "Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)." 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.

2.3.3 Organization of CCSR Memory

The configuration, control, and status registers of the MPC8572E are grouped according to functional units. Most functional blocks are allocated a 4-Kbyte address space for registers. Registers that fall into this category are referred to as general utilities registers. These registers occupy the first 256 Kbytes of CCSR memory.

Registers controlling functions that are not particular to a functional unit but to the device as a whole occupy the highest 256 Kbytes of CCSR memory and are referred to as device-specific registers.

Some functional units, such as RapidIO and the OpenPIC-based interrupt controller have larger address spaces as defined by their programming models. The registers for these blocks are given their own large regions of CCSR memory.

Table 2-10. Local Memory Configuration, Control, and Status Register Summary

Offset from CCSRBAR	Register Grouping
0x0_0000–0x3_FFFF	General utilities
0x4_0000–0x7_FFFF	Programmable interrupt controller (PIC)
0x8_0000–0xB_FFFF	Reserved
0xC_0000–0xD_FFFF	RapidIO
0xE_0000–0xF_FFFF	Device-specific utilities

2.3.4 General Utilities Registers

Figure 2-7 provides an overview of the general utilities registers.

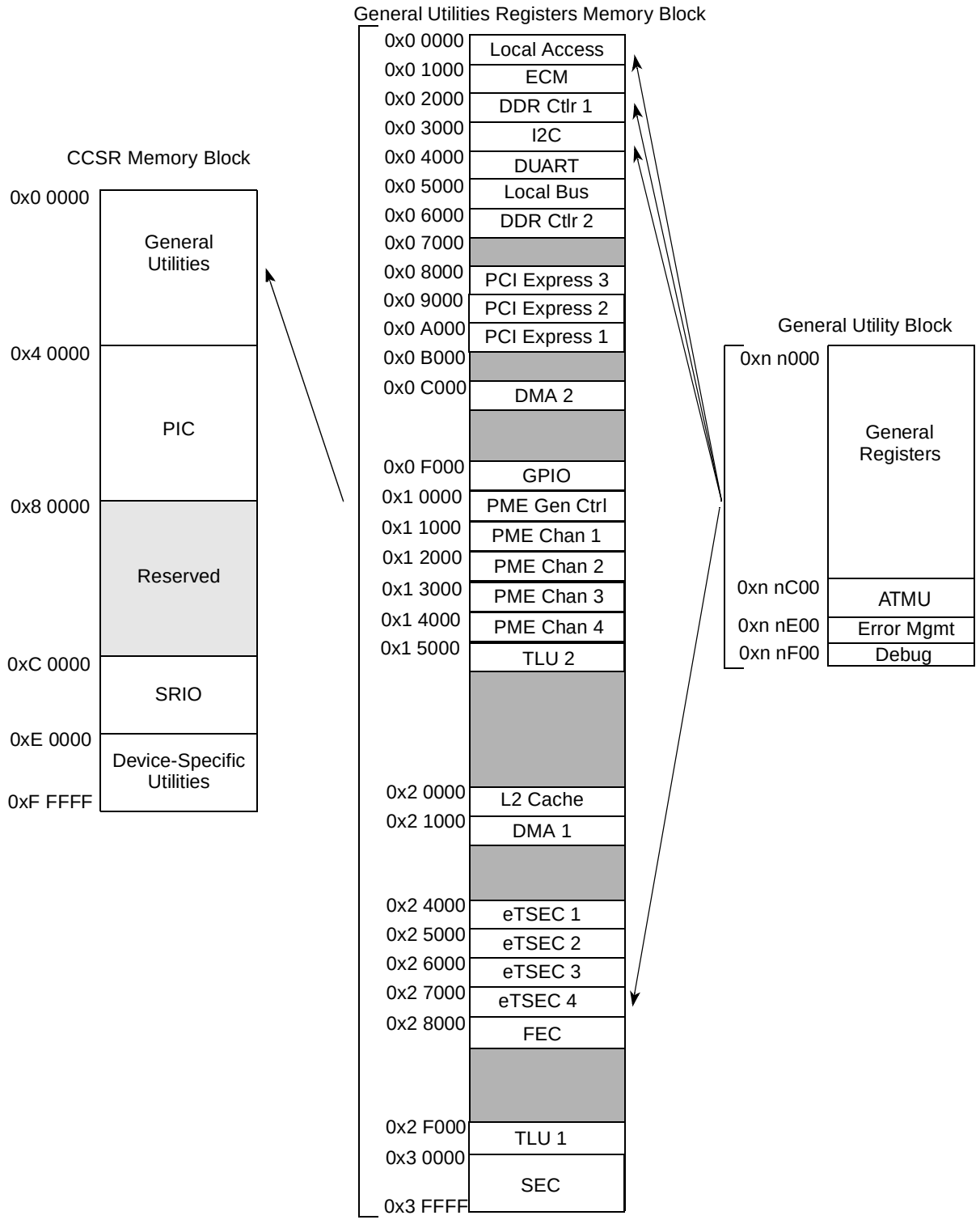


Figure 2-7. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

Figure 2-7 also shows the organization of registers inside the 4-Kbyte register space allocated to an individual functional block. The first 3 Kbytes are available for general registers. The next 512 bytes are dedicated to address translation and mapping registers, if applicable to that particular functional unit (for example, PCI Express 1). If a unit has error management registers, they are typically placed starting at offset 0xE00 from the beginning of the block's 4-Kbyte space, and any debug registers are typically placed in the final 256 bytes of the unit's register space starting at offset 0xF00.

General utilities registers are accessed as 32-bit quantities except for both sets of the TLU registers and all SEC registers, which are accessed as 64-bit quantities, and the DUART and I^2C registers, which are accessed as bytes.

NOTE

Refer to detailed register descriptions for each functional unit for exact locations, sizes, and access requirements. Some blocks may have exceptions to the above guidelines.

2.3.5 Interrupt Controller and CCSR

The programmable interrupt controller (PIC) registers are at offset 0x4_0000 from CCSRBAR, see Figure 2-8. Its programming model follows the OpenPIC architecture. The interrupt controller registers should only be accessed with 32-bit accesses.

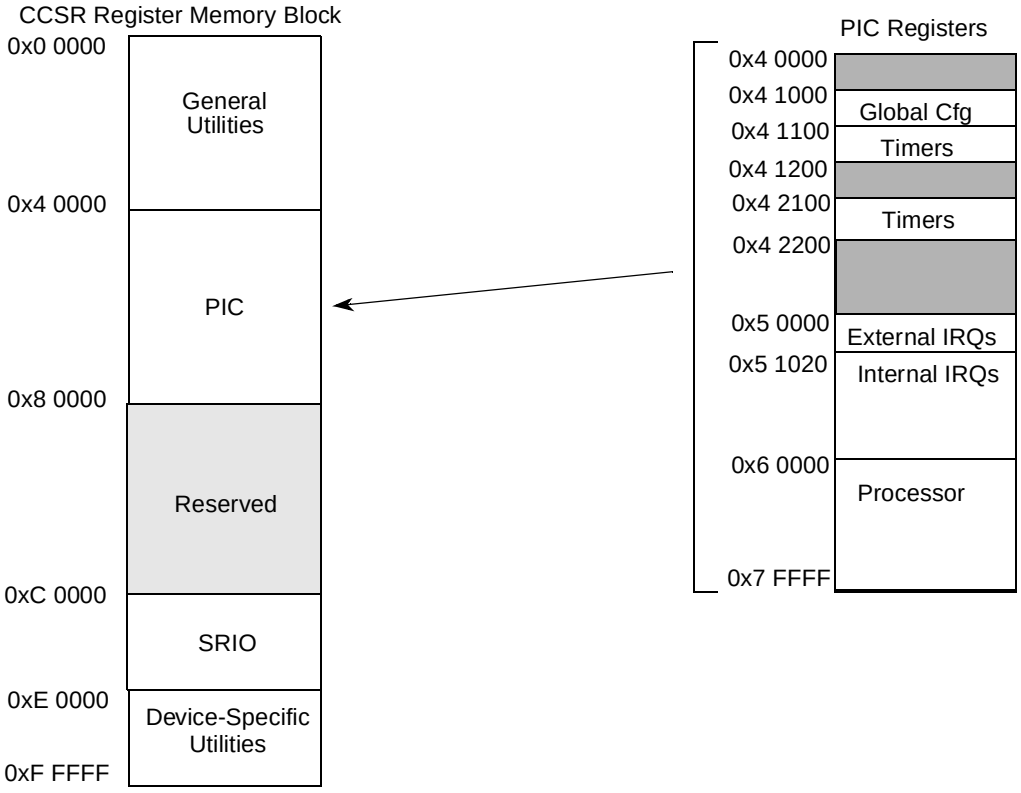


Figure 2-8. PIC Mapping to Configuration, Control, and Status Memory Block

2.3.6 Serial RapidIO and CCSR

The serial RapidIO module uses 128 Kbytes of CCSR memory; refer to Figure 2-9. All registers are 32-bits wide and should only be accessed with 32-bit accesses. The 4-Kbyte RapidIO implementation block has the same internal organization as those defined for the general utilities described above.

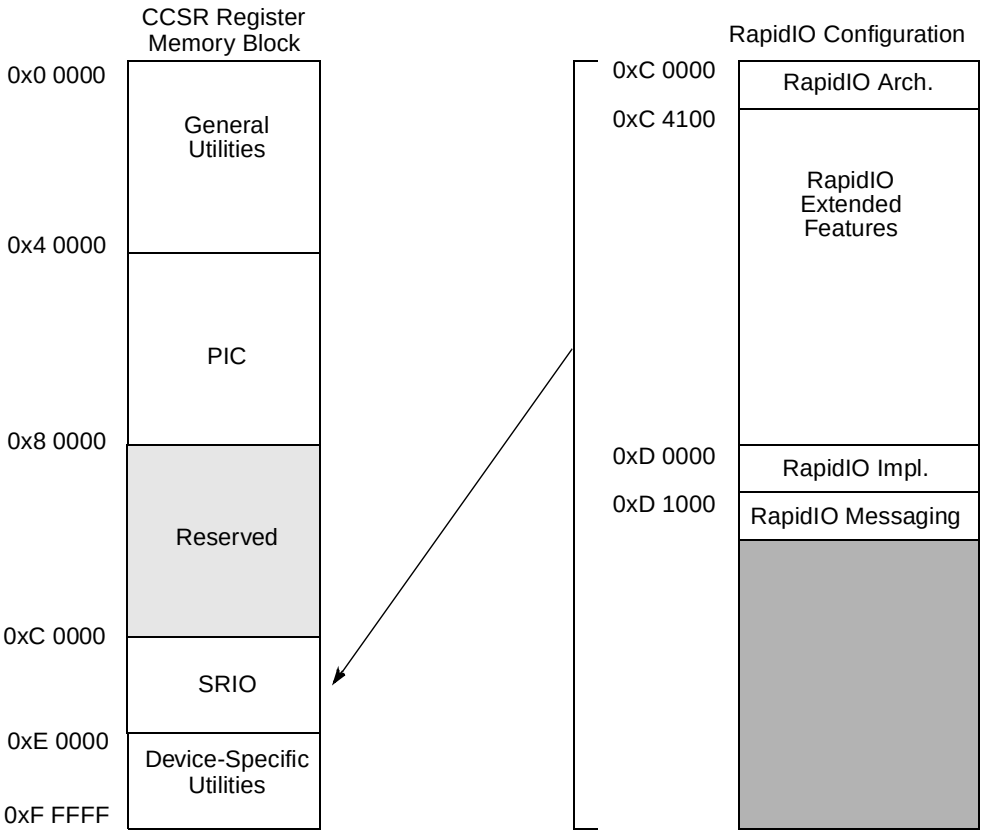


Figure 2-9. RapidIO Mapping to Configuration, Control, and Status Memory Block

2.3.7 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities (refer to Figure 2-10). These registers are accessible with 32-bit accesses only. Transactions of other than 32-bit are considered a programming error and operation is undefined.

Reserved bits in the following register descriptions are not guaranteed to have predictable values. Software must preserve the values of reserved bits when writing to a register. Also, when reading from a register, software should not rely on the value of any reserved bit remaining consistent.

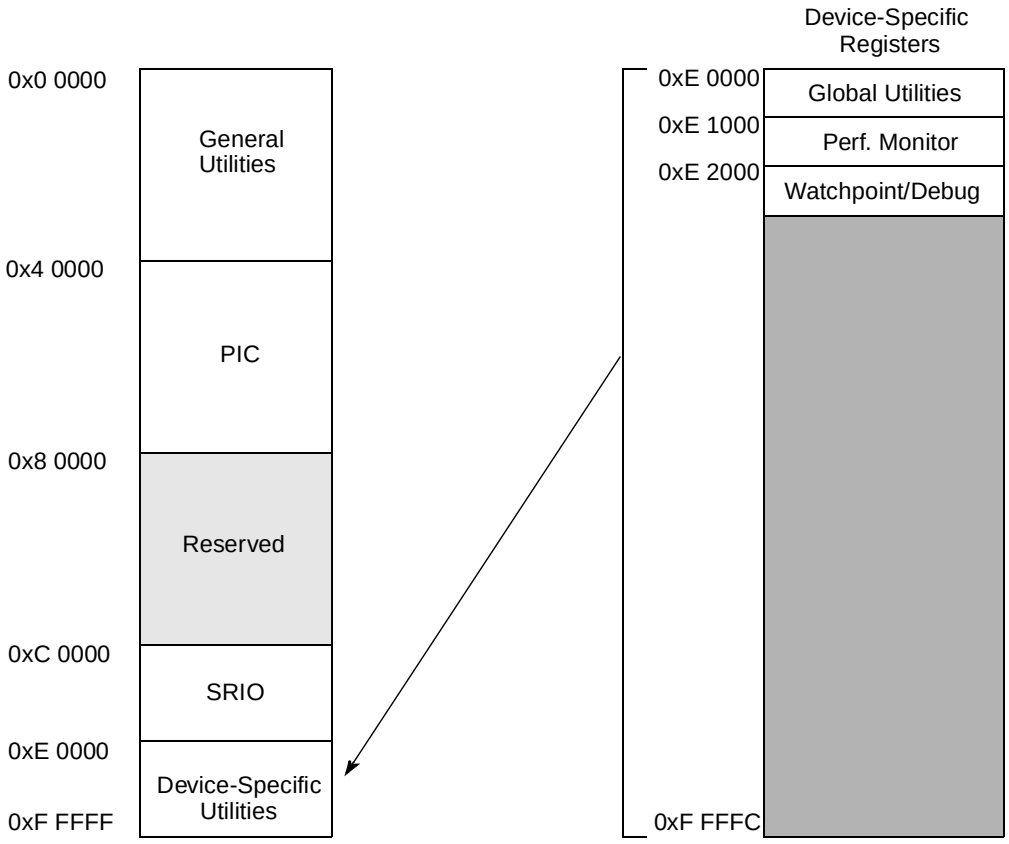


Figure 2-10. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.3.8 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.

2.4 Complete CCSR Map

Table 2-11 lists the MPC8572E memory-mapped registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.

- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

NOTE

In the eTSEC (three-speed Ethernet controller) section of the following table, registers denoted with an asterisk (*) are new to the enhanced TSEC (eTSEC) and are not supported by PowerQUICC III TSECs.

Table 2-11. Memory Map

Offset	Register	Access	Reset	Section/Page
Local-Access Registers—Configuration, Control, and Status Registers				
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-6
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-7
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-7
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-9
Local-Access Registers—Local-Access Window Base and Size Registers				
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D08	LAWBAR8—Local access window 8 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D10	LAWAR8—Local access window 8 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D28	LAWBAR9—Local access window 9 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D30	LAWAR9—Local access window 9 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
0x0_0D48	LAWBAR10—Local access window 10 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D50	LAWAR10—Local access window 10 attribute register	R/W	0x0000_0000	2.2.3.5/2-8

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_0D68	LAWBAR11—Local access window 11 base address register	R/W	0x0000_0000	2.2.3.4/2-7
0x0_0D70	LAWAR11—Local access window 11 attribute register	R/W	0x0000_0000	2.2.3.5/2-8
e500 Coherency Module				
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*00_0000	8.2.1.2/8-4
0x0_1BF8	ECM IP Block Revision Register 1	R	0x0001_0000	8.2.1.3/8-5
0x0_1BFC	ECM IP Block Revision Register 2	R	0x0000_0000	8.2.1.4/8-6
0x0_1E00	EEDR—ECM error detect register	w1c	0x0000_0000	8.2.1.5/8-6
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.6/8-7
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.7/8-8
0x0_1E10	EELADR—ECM error low address capture register	R	0x0000_0000	8.2.1.8/8-9
0x0_1E14	EEHADR—ECM error high address capture register	R	0x0000_0000	8.2.1.9/8-9
DDR Memory Controller 1 Registers				
0x0_2000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-13
0x0_2008	CS1_BNDS—Chip select 1 memory bounds			
0x0_2010	CS2_BNDS—Chip select 2 memory bounds			
0x0_2018	CS3_BNDS—Chip select 3 memory bounds			
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-14
0x0_2084	CS1_CONFIG—Chip select 1 configuration			
0x0_2088	CS2_CONFIG—Chip select 2 configuration			
0x0_208C	CS3_CONFIG—Chip select 3 configuration			
0x0_20C0	CS0_CONFIG_2—Chip select 0 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0_20C4	CS1_CONFIG_2—Chip select 1 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0_20C8	CS2_CONFIG_2—Chip select 2 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0_20CC	CS3_CONFIG_2—Chip select 3 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0_2100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.7/9-23
0x0_2104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.4/9-17
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.5/9-19
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.6/9-21
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.8/9-25
0x0_2114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.9/9-28
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.10/9-30
0x0_211C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.11/9-31
0x0_2120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.12/9-32
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.13/9-34
0x0_2128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.14/9-35

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.15/9-35
0x0_2148	DDR_INIT_ADDRESS—DDR training initialization address	R/W	0x0000_0000	9.4.1.16/9-36
0x0_214C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	9.4.1.17/9-36
0x0_2160	TIMING_CFG_4— DDR SDRAM timing configuration 4	R/W	0x0000_0000	9.4.1.18/9-37
0x0_2164	TIMING_CFG_5— DDR SDRAM timing configuration 5	R/W	0x0000_0000	9.4.1.19/9-39
0x0_2170	DDR_ZQ_CNTL— DDR ZQ calibration control	R/W	0x0000_0000	9.4.1.20/9-40
0x0_2174	DDR_WRLVL_CNTL— DDR write leveling control	R/W	0x0000_0000	9.4.1.21/9-42
0x0_2B20	DDRDSR_1—DDR Debug Status Register 1	R	0x0000_0000	9.4.1.22/9-45
0x0_2B24	DDRDSR_2—DDR Debug Status Register 2	R	0x0000_0000	9.4.1.23/9-46
0x0_2B28	DDRCDR_1—DDR Control Driver Register 1	R/W	0x0000_0000	9.4.1.24/9-46
0x0_2B2C	DDRCDR_2—DDR Control Driver Register 2	R/W	0x0000_0000	9.4.1.25/9-49
0x0_2BF8	DDR_IP_REV1—DDR IP Block Revision 1	R/W	0x*nnnn_nnnn*[1]	9.4.1.26/9-50
0x0_2BFC	DDR_IP_REV2—DDR IP Block Revision 2	R/W	0x0000_0000	9.4.1.27/9-50
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.28/9-51
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.29/9-51
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.30/9-52
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.31/9-52
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.32/9-53
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.33/9-53
0x0_2E40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.34/9-54
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.35/9-55
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.36/9-56
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.37/9-57
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.38/9-58
0x0_2E54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	9.4.1.39/9-59
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.40/9-59
I^2C				
I^2C1 Registers				
0x0_3000	I2CADR—I^2C address register	R/W	0x00	12.3.1.1/12-6
0x0_3004	I2CFDR—I^2C frequency divider register	R/W	0x00	12.3.1.2/12-6
0x0_3008	I2CCR—I^2C control register	Mixed	0x00	12.3.1.3/12-7
0x0_300C	I2CSR—I^2C status register	Mixed	0x81	12.3.1.4/12-9

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_3010	I2CDR—I^{2}C data register	R/W	0x00	12.3.1.5/12-10
0x0_3014	I2CDFSRR—I^{2}C digital filter sampling rate register	R/W	0x10	12.3.1.6/12-11
I^{2}C2 Registers				
0x0_3100–0x0_3114	I^{2}C2 Registers[2]			
DUART Registers				
0x0_4500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	13.3.1/13-5
0x0_4500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	13.3.1/13-5
0x0_4500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	13.3.1/13-5
0x0_4501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	13.3.1/13-5
0x0_4501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	13.3.1/13-5
0x0_4502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	13.3.1/13-5
0x0_4502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	13.3.1/13-5
0x0_4502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	13.3.1/13-5
0x0_4503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	13.3.1/13-5
0x0_4504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	13.3.1/13-5
0x0_4505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	13.3.1/13-5
0x0_4506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	13.3.1/13-5
0x0_4507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	13.3.1/13-5
0x0_4510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	13.3.1/13-5
0x0_4600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	13.3.1/13-5
0x0_4600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	13.3.1/13-5
0x0_4600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	13.3.1/13-5
0x0_4601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	13.3.1/13-5
0x0_4601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	13.3.1/13-5
0x0_4602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	13.3.1/13-5
0x0_4602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	13.3.1/13-5
0x0_4602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	13.3.1/13-5
0x0_4603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	13.3.1/13-5
0x0_4604	UMCR—ULCR[DLAB] = x UART1 modem control register	R/W	0x00	13.3.1/13-5
0x0_4605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	13.3.1/13-5
0x0_4606	UMSR—ULCR[DLAB] = x UART1 modem status register	R	0x00	13.3.1/13-5
0x0_4607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	13.3.1/13-5

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	13.3.1/13-5
Enhanced Local Bus Controller Registers				
0x0_5000	BR0—Base register 0	R/W	0x0000_*nn*01[3]	14.3.1/14-10
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	14.3.1/14-10
0x0_500C	OR1—Options register 1		0x0000_0000	
0x0_5014	OR2—Options register 2			
0x0_501C	OR3—Options register 3			
0x0_5024	OR4—Options register 4			
0x0_502C	OR5—Options register 5			
0x0_5034	OR6—Options register 6			
0x0_503C	OR7—Options register 7			
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	14.3.1/14-10
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	14.3.1/14-10
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	14.3.1/14-10
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	14.3.1/14-10
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	14.3.1/14-10
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	14.3.1/14-10
0x0_5090	LSOR—Special operation initiation register	R/W	0x0000_0000	14.3.1/14-10
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	14.3.1/14-10
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	14.3.1/14-10
0x0_50B0	LTESR—Transfer error status register	w1c	0x0000_0000	14.3.1/14-10
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	14.3.1/14-10
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	14.3.1/14-10
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	14.3.1/14-10
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	14.3.1/14-10
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	14.3.1/14-10
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008[4]	14.3.1/14-10
0x0_50E0	FMR—Flash mode register	R/W	0x0000_0800	14.3.1/14-10
0x0_50E4	FIR—Flash instruction register	R/W	0x0000_0000	14.3.1/14-10

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_50E8	FCR—Flash command register	R/W	0x0000_0000	14.3.1/14-10
0x0_50EC	FBAR—Flash block address register	R/W	0x0000_0000	14.3.1/14-10
0x0_50F0	FPAR—Flash page address register	R/W	0x0000_0000	14.3.1/14-10
0x0_50F4	FBCR—Flash byte count register	R/W	0x0000_0000	14.3.1/14-10
DDR Memory Controller 2 Registers				
0x0_6000–0x0_6FFF	DDR controller 2 registers[5]			
PCI Express Registers				
0x0_8000–0x0_8FFF	PCI Express controller 3 registers[6]			
0x0_9000–0x0_9FFF	PCI Express controller 2 registers[7]			
PCI Express Controller 1				
PCI Express Configuration Access Registers				
0x0_A000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	21.3.2/21-10
0x0_A004	PEX _CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	21.3.2/21-10
0x0_A00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	21.3.2/21-10
0x0_A010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	21.3.2/21-10
0x0_A014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	21.3.2/21-10
PCI Express Power Management Event & Message Registers				
0x0_A020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	21.3.3/21-13
0x0_A024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	21.3.3/21-13
0x0_A028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	21.3.3/21-13
0x0_A02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	21.3.3/21-13
PCI Express Block ID Registers				
0x0_ABF8	Block revision register 1	R	0x0208_0100	21.3.4/21-19
0x0_ABFC	Block revision register 2	R	0x0000_0000	21.3.4/21-19
PCI Express ATMU Registers				
0x0_AC00–0x0_AC1C–Outbound Window 0 (default)				
0x0_AC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	21.3.5/21-20

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_AC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	21.3.5/21-20
0x0_AC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	21.3.5/21-20
0x0_AC20–0x0_AC3C—Outbound Window 1				
0x0_AC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	21.3.5/21-20
0x0_AC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	21.3.5/21-20
0x0_AC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	21.3.5/21-20
0x0_AC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	21.3.5/21-20
0x0_AC40–0x0_AC5C—Outbound Window 2				
0x0_AC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_AC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_AC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_AC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	21.3.5/21-20
0x0_AC60–0x0_AC7C—Outbound Window 3				
0x0_AC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	21.3.5/21-20
0x0_AC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	21.3.5/21-20
0x0_AC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	21.3.5/21-20
0x0_AC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023	21.3.5/21-20
0x0_AC80–0x0_AC9C—Outbound Window 4				
0x0_AC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	21.3.5/21-20
0x0_AC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	21.3.5/21-20
0x0_AC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	21.3.5/21-20
0x0_AC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	21.3.5/21-20
0x0_ADA0–0x0_ADBC–Inbound Window 3				
0x0_ADA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	21.3.5/21-20

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_ADA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	21.3.5/21-20
0x0_ADAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	21.3.5/21-20
0x0_ADB0	PEXIWAR3—PCI Express inbound window attributes register 3	Mixed	0x20F4_4023	21.3.5/21-20
0x0_ADC0–0x0_ADDC–Inbound Window 2				
0x0_ADC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_ADC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_ADCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	21.3.5/21-20
0x0_ADD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	21.3.5/21-20
0x0_ADE0–0x0_ADFC–Inbound Window 1				
0x0_ADE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	21.3.5/21-20
0x0_ADE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	21.3.5/21-20
0x0_ADF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x0000_0000	21.3.5/21-20
PCI Express Error Management Registers				
0x0_AE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	21.3.6/21-30
0x0_AE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	21.3.6/21-30
0x0_AE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	21.3.6/21-30
0x0_AE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	21.3.6/21-30
0x0_AE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	21.3.6/21-30
0x0_AE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	21.3.6/21-30
0x0_AE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	21.3.6/21-30
0x0_AE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	21.3.6/21-30
DMA Controller 2 Registers				
0x0_C000–0x0_CFFC	DMA controller 2 registers[8]			
GPIO Configuration Registers				
0x0_FC00	GPIO direction register (GPDIR)	R/W	0x0000_0000	22.3/22-2
0x0_FC04	GPIO open drain register (GPODR)	R/W	0x0000_0000	22.3/22-2
0x0_FC08	GPIO data register (GPDAT)	R/W	0x0000_0000	22.3/22-2
0x0_FC0C	GPIO interrupt event register (GPIER)	R/W	Undefined	22.3/22-2
0x0_FC10	GPIO interrupt mask register (GPIMR)	R/W	0x0000_0000	22.3/22-2
0x0_FC14	GPIO external interrupt control register (GPICR)	R/W	0x0000_0000	22.3/22-2

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Pattern Matching Engine Registers				
Pattern Matcher DMA Engine - Common Control Registers				
0x1_0000	ISRCC—Interrupt Status Register Common Control	w1c	0x0000_0000	17.3.1.1/17-14
0x1_0004	IERCC—Interrupt Enable Register Common Control	R/W	0x0000_0000	17.3.1.2/17-16
0x1_0008	ISDRCC—Interrupt Status Disable Register Common Control	R/W	0x0000_0000	17.3.1.3/17-17
0x1_000C	IIRCC—Interrupt Inhibit Register Common Control	R/W	0x0000_0000	17.3.1.4/17-18
0x1_0014	SCOS—Statistics Counters Overflow Status Register	w1c	0x0000_0000	17.3.1.5/17-19
0x1_001C	PERFT—Performance Monitor Threshold Register	R/W	0x0000_0000	17.3.1.6/17-20
0x1_0024	CSCR—Channel Scheduling Control Register	R/W	0x0000_0000	17.3.1.7/17-21
KES Control Registers				
0x1_0080	STNIB-—Statistic: Number of Input Bytes Scanned for Pattern Matching	RR	0x0000_0000	17.3.2/17-22
0x1_0084	STNIS-—Statistic: Number of Input SUIs	RR	0x0000_0000	17.3.2/17-22
0x1_0088	STNTH1-—Statistic: Number of Trigger Hits Against 1-Byte Trigger	RR	0x0000_0000	17.3.2/17-22
0x1_008C	STNTH2-—Statistic: Number of Trigger Hits Against 2-Byte Trigger	RR	0x0000_0000	17.3.2/17-22
0x1_0090	STNTHL-—Statistic: Number of Trigger Hits Against Variable Length Trigger	RR	0x0000	17.3.2/17-22
0x1_0094	STNTHS-—Statistic: Number of Trigger Hits Against Special Triggers	RR	0x0000	17.3.2/17-22
0x1_0098	STNCH-—Statistic: Number of Confidence Stage Hits	RR	0x0000	17.3.2/17-22
0x1_009C	SWDB-—Software Database Version Reg/Scratch Pad	R/W	0x0000_0000	17.3.2/17-22
0x1_00A0	KVLTS-—KES Variable Length Trigger Size	R/W	0x0000_0007	17.3.2/17-22
0x1_00A4	KEC-—KES Error Configuration	R/W	0x0000_0000	17.3.2/17-22
0x1_00A8	PEHE-—PME Error Handling Enables	R/W	0x0000_0000	17.3.2/17-22
DXE Control Registers				
0x1_0100	STNPM-—Statistic: Number of Pattern Matches	RR	0x0000	17.3.3/17-29
0x1_0104	STNS1M-—Statistic: Number of SUI with at Least 1 Pattern Match	RR	0x0000	17.3.3/17-29
0x1_0108	DRCC—DXE Pattern Range Counter Configuration	R/W	0x0000_0000	17.3.3/17-29
0x1_010C	STNPMR-—Statistic: Number of Pattern Matches Within Range of DRCC	RR	0x0000	17.3.3/17-29
0x1_0120	PDSRBAH—Pattern Description and Stateful Rule Base Address Register High	R/W	0x0000_0000	17.3.3/17-29
0x1_0124	PDSRBAL—Pattern Description and Stateful Rule Base Address Register Low	R/W	0x0000_0000	17.3.3/17-29

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1_0128	DMCR—DXE Memory Control Register	R/W	0x0000_0000	17.3.3/17-29
0x1_012C	DEC—DXE Error Configuration	R/W	0x0000_0000	17.3.3/17-29
SRE Control Registers				
0x1_0180	STNDSR-—Statistic: Number of DXE Generated SR Executions	RR	0x0000	17.3.4.1/17-34
0x1_0184	STNESR-—Statistic: Number of End of SUI generated SR executions	RR	0x0000	17.3.4.2/17-34
0x1_0188	STNS1R-—Statistic: Number of SUI with at Least 1 Report	RR	0x0000	17.3.4.3/17-35
0x1_018C	STNOB-—Statistic: Number of Output Bytes Produced in Reports	RR	0x0000_0000	17.3.4.4/17-35
0x1_01A8	SCBARH—SRE Context Base Address Register High	R/W	0x0000_0000	17.3.4.5/17-36
0x1_01AC	SCBARL—SRE Context Base Address Register Low	R/W	0x0000_0000	17.3.4.5/17-36
0x1_01B0	SMCR—SRE Memory Control Register	R/W	0x0000_0000	17.3.4.6/17-37
0x1_01B4	SREC—SRE Configuration	R/W	0x0000_0000	17.3.4.7/17-38
0x1_01B8	SRRV0—SRE Rule Reset Vector 0	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01BC	SRRV1—SRE Rule Reset Vector 1	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C0	SRRV2—SRE Rule Reset Vector 2	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C4	SRRV3—SRE Rule Reset Vector 3	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C8	SRRV4—SRE Rule Reset Vector 4	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01CC	SRRV5—SRE Rule Reset Vector 5	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D0	SRRV6—SRE Rule Reset Vector 6	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D4	SRRV7—SRE Rule Reset Vector 7	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D8	SRRFI—SRE Rule Reset First Index	R/W	0x0000_0000	17.3.4.9/17-39
0x1_01DC	SRRI—SRE Rule Reset 32-byte Increment	R/W	0x0000_0000	17.3.4.10/17-40
0x1_01E0	SRRR—SRE Rule Reset Repetitions	R/W	0x0000_0000	17.3.4.11/17-40
0x1_01E4	SRRWC—SRE Rule Reset Work Configuration	R/W	0x0000_0000	17.3.4.12/17-41
0x1_01E8	SFRCC—SRE Free Running Counter Configuration	R/W	0x0000_0000	17.3.4.13/17-42
0x1_01EC	SEC1—SRE Error Configuration 1	R/W	0x0000_0000	17.3.4.14/17-43
0x1_01F0	SEC2—SRE Error Configuration 2	R/W	0x0000_0000	17.3.4.15/17-43
0x1_01F4	SEC3—SRE Error Configuration 3	R/W	0x0000_0000	17.3.4.16/17-44
Deflate Control Registers				
0x1_0200	STDBC-—Statistic: Deflate Bytes Consumed	RR	0x0000_0000	17.3.5.1/17-44

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1_0204	STDBP-—Statistic: Deflate Bytes Produced	RR	0x0000_0000	17.3.5.2/17-45
0x1_0208	STDWC-—Statistic: Deflate Work Units Consumed	RR	0x0000	17.3.5.3/17-45
0x1_0224	DBPPWL-— Configuration: Deflate Bytes Produced Per Work Unit Limit	R/W	0x0000_0000	17.3.5.4/17-46
FBM Common Control Registers				
0x1_02A0	FBL0SIZE—Free Buffer List 0 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02A4	FBL1SIZE—Free Buffer List 1 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02A8	FBL2SIZE—Free Buffer List 2 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02AC	FBL3SIZE—Free Buffer List 3 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B0	FBL4SIZE—Free Buffer List 4 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B4	FBL5SIZE—Free Buffer List 5 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B8	FBL6SIZE—Free Buffer List 6 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02BC	FBL7SIZE—Free Buffer List 7 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02C0	FBLAAR—Free Buffer List Assignment A Register	R/W	0x0000_0000	17.3.6.2/17-47
0x1_02C4	FBLABR—Free Buffer List Assignment B Register	R/W	0x0000_0000	17.3.6.3/17-48
0x1_02E0	FBM_CR	R/W	0x0000_0000	17.3.6.4/17-50
Memory Interface Arbiter Registers				
0x1_0B80	MIA_BYC—MIA Byte Count Register	RR	0x0000_0000	17.3.7.1/17-51
0x1_0B84	MIA_BLC—MIA Block Count Register	RR	0x0000_0000	17.3.7.2/17-51
0x1_0B88	MIA_CE—MIA Count Enable Register	R/W	0x0000_0000	17.3.7.3/17-52
0x1_0BA0	MIA_CR—MIA Control Register	R/W	0x0000_0000	17.3.7.4/17-55
0x1_0BA4	MIA_ACR—MIA Arbitration Control Register	R/W	0x0000_0000	17.3.7.5/17-56
0x1_0BB0	MIA_LSHC0—MIA Local Snoop Hit Count 0 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BB4	MIA_LSHC1—MIA Local Snoop Hit Count 1 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BB8	MIA_LSHC2—MIA Local Snoop Hit Count 2 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BC0	MIA_CWC0—MIA Cancelled Write Count 0 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BC4	MIA_CWC1—MIA Cancelled Write Count 1 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BC8	MIA_CWC2—MIA Cancelled Write Count 2 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BF8	PM_IP_REV1—PM IP Block Revision 1 Register	R	0x0010_0000	17.3.7.8/17-59
0x1_0BFC	PM_IP_REV2—PM IP Block Revision 2 Register	R	0x0000_0000	17.3.7.9/17-59
PM DMA Engine - Channel 0 Control Registers				
0x1_1000	ISR0—Interrupt Status Register 0	w1c	0x0000_0000	17.3.8.1/17-60
0x1_1004	IER0—Interrupt Enable Register 0	R/W	0x0000_0000	17.3.8.2/17-62
0x1_1008	ISDR0—Interrupt Status Disable Register 0	R/W	0x0000_0000	17.3.8.3/17-63
0x1_100C	IIR0—Interrupt Inhibit Register 0	R/W	0x0000_0000	17.3.8.4/17-64

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1_1020	IICR0—Interrupt Interval Control Register 0	R/W	0x0000_0000	17.3.8.5/17-65
0x1_1060	NPIR0—Notifications Produced Index Register Channel 0	R	0x0000_0000	17.3.8.6/17-65
0x1_1064	CCIR0—Commands Consumed Index Register Channel 0	R	0x0000_0000	17.3.8.7/17-66
0x1_1070	FBCIR0—Free Buffer Deallocate FIFO Consumer Index Channel 0	R	0x0000_0000	17.3.8.8/17-67
0x1_1074	CHERR0—Channel Error Status Channel 0	R	0x0000_0000	17.3.8.9/17-68
0x1_1080	NCIR0—Notifications Consumed Index Register Channel 0	R/W	0x0000_0000	17.3.8.10/17-68
0x1_1084	CPIR0—Commands Produced Index Register Channel 0	R/W	0x0000_0000	17.3.8.11/17-69
0x1_1088	CEIR0—Commands Expired Index Register Channel 0	R/W	0x0000_0000	17.3.8.12/17-70
0x1_1090	FBPIR0—Free Buffer Deallocate FIFO Producer Index Channel 0	R/W	0x0000_0000	17.3.8.13/17-70
0x1_10A0	TRUNCIC0—Output Truncated Incidence Counter Channel 0	RR	0x0000_0000	17.3.8.14/17-71
0x1_10A4	RBC0—Read Byte Counter Channel 0	RR	0x0000_0000	17.3.8.15/17-72
0x1_10A8	SCOS0—Statistics Counters Overflow Status Register Channel 0	w1c	0x0000_0000	17.3.8.16/17-72
0x1_1100	CRCR0—Channel Reset and Control Register Channel 0	R/W	0x0000_0000	17.3.8.17/17-73
0x1_1120	CFIFO_BH0—Command FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.18/17-74
0x1_1124	CFIFO_BL0—Command FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.18/17-74
0x1_1128	CFIFO_DEPTH0—Command FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.19/17-75
0x1_112C	CFIFO_THRES0—Command FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.20/17-76
0x1_1140	NFIFO_BH0—Notification FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.21/17-76
0x1_1144	NFIFO_BL0—Notification FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.21/17-76
0x1_1148	NFIFO_DEPTH0—Notification FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.22/17-78
0x1_114C	NFIFO_THRES0—Notification FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.23/17-78
0x1_1150	NDLL0—Notification Deflate Length Limit Register Channel 0	R/W	0x0000_0000	17.3.8.24/17-79
0x1_1154	NRLL0—Notification Result Length Limit Register Channel 0	R/W	0x0000_0000	17.3.8.25/17-79
0x1_1160	FBFIFO_BH0—Free Buffer Deallocate FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.26/17-80
0x1_1164	FBFIFO_BL0—Free Buffer Deallocate FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.26/17-80
0x1_1168	FBFIFO_DEPTH0—Free Buffer Deallocate FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.27/17-81
0x1_116C	FBFIFO_THRES0—Free Buffer Deallocate FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.28/17-82

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1_1180	SC_CONFIG0—Stream Context Configuration Register Channel 0	R/W	0x0000_0000	17.3.8.29/17-82
0x1_1184	SC_BH0—Stream Context Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.30/17-83
0x1_1188	SC_BL0—Stream Context Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.30/17-83
0x1_11A0	RES_CONFIG0—Residue Configuration Register Channel 0	R/W	0x0000_0000	17.3.8.31/17-84
0x1_11A4	RES_BH0—Residue Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.32/17-85
0x1_11A8	RES_BL0—Residue Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.32/17-85
0x1_11C0	CACR0—Cache Awareness Control Register Channel 0	R/W	0x0000_0000	17.3.8.33/17-86
0x1_11C4	MTPCR0—Memory Transaction Priority Control Register Channel 0	R/W	0x0000_0000	17.3.8.34/17-88
FBM - Channel 0 Control Registers				
0x1_1400	FBL_THRESH_A0—Free Buffer List Threshold A Register Channel 0	R/W	0x0000_0000	17.3.9.1/17-89
0x1_1404	FBL_HLWM_A0—Free Buffer List High/Low Watermark A Register Channel 0	WR	0x0000_0000	17.3.9.2/17-90
0x1_1408	FBL_AD_CA0—Free Buffer List Allocation and Deallocation Counter Register A Channel 0	RR	0x0000_0000	17.3.9.3/17-91
0x1_140C	FBL_ODD_A0—Free Buffer List On Deck Disable A Register Channel 0	R/W	0x0000_0000	17.3.9.4/17-91
0x1_1410	FBL_PH_AH0—Free Buffer List Physical Head Address A High Register Channel 0	R	0x0000_0000	17.3.9.5/17-92
0x1_1414	FBL_PH_AL0—Free Buffer List Physical Head Address A Low Register Channel 0	R	0x0000_0000	17.3.9.5/17-92
0x1_1418	FBL_VH_AH0—Free Buffer List Virtual Head Address A High Register Channel 0	R	0x0000_0000	17.3.9.6/17-93
0x1_141C	FBL_VH_AL0—Free Buffer List Virtual Head Address A Low Register Channel 0	R	0x0000_0000	17.3.9.6/17-93
0x1_1420	FBL_PT_AH0—Free Buffer List Physical Tail Address A High Register Channel 0	R	0x0000_0000	17.3.9.7/17-94
0x1_1424	FBL_PT_AL0—Free Buffer List Physical Tail Address A Low Register Channel 0	R	0x0000_0000	17.3.9.7/17-94
0x1_1430	FBL_NB_A0—Free Buffer List Number of Buffers A Register Channel 0	R	0x0000_0000	17.3.9.8/17-95
0x1_1434	FBL_BS_A0—Free Buffer List Buffer Size A Register Channel 0	R	0x0000_0000	17.3.9.9/17-96
0x1_1438	FBL_VD_AH0—Free Buffer List Virtual On Deck Pointer A High Register Channel 0	R	0x0000_0000	17.3.9.10/17-96
0x1_143C	FBL_VD_AL0—Free Buffer List Virtual On Deck Pointer A Low Register Channel 0	R	0x0000_0000	17.3.9.10/17-96
0x1_1440	FBL_THRESH_B0—Free Buffer List Threshold B Register Channel 0	R/W	0x0000_0000	17.3.9.11/17-97

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1_1444	FBL_HLWM_B0—Free Buffer List High/Low Watermark B Register Channel 0	WR	0x0000_0000	17.3.9.12/17-97
0x1_1448	FBL_AD_CB0—Free Buffer List Allocation and Deallocation Counter Register B Channel 0	RR	0x0000_0000	17.3.9.13/17-98
0x1_144C	FBL_ODD_B0—Free Buffer List On Deck Disable B Register Channel 0	R/W	0x0000_0000	17.3.9.14/17-98
0x1_1450	FBL_PH_BH0—Free Buffer List Physical Head Address B High Register Channel 0	R	0x0000_0000	17.3.9.15/17-98
0x1_1454	FBL_PH_BL0—Free Buffer List Physical Head Address B Low Register Channel 0	R	0x0000_0000	17.3.9.15/17-98
0x1_1458	FBL_VH_BH0—Free Buffer List Virtual Head Address B High Register Channel 0	R	0x0000_0000	17.3.9.16/17-98
0x1_145C	FBL_VH_BL0—Free Buffer List Virtual Head Address B Low Register Channel 0	R	0x0000_0000	17.3.9.16/17-98
0x1_1460	FBL_PT_BH0—Free Buffer List Physical Tail Address B High Register Channel 0	R	0x0000_0000	17.3.9.17/17-98
0x1_1464	FBL_PT_BL0—Free Buffer List Physical Tail Address B Low Register Channel 0	R	0x0000_0000	17.3.9.17/17-98
0x1_1470	FBL_NB_B0—Free Buffer List Number of Buffers B Register Channel 0	R	0x0000_0000	17.3.9.18/17-98
0x1_1474	FBL_BS_B0—Free Buffer List Buffer Size B Register Channel 0	R	0x0000_0000	17.3.9.19/17-99
0x1_1478	FBL_PD_BH0—Free Buffer List Virtual On Deck Pointer B High Register Channel 0	R	0x0000_0000	17.3.9.20/17-99
0x1_147C	FBL_PD_BL0—Free Buffer List Virtual On Deck Pointer B Low Register Channel 0	R	0x0000_0000	17.3.9.20/17-99
Pattern Matcher DMA Engine - Channel 1 Control Registers				
0x1_2000-0x1_27FF	Pattern matcher DMA Engine and FBM Channel 1 has the same memory mapped registers that are described for channel 0 above, but at the offset listed to the left.			
Pattern Matcher DMA Engine - Channel 2 Control Registers				
0x1_3000-0x1_37FF	Pattern matcher DMA Engine and FBM Channel 2 has the same memory mapped registers that are described for channel 0 above, but at the offset listed to the left.			
Pattern Matcher DMA Engine - Channel 3 Control Registers				
0x1_4000-0x1_47FF	Pattern matcher DMA Engine and FBM Channel 3 has the same memory mapped registers that are described for channel 0 above, but at the offset listed to the left.			
Table Lookup Unit 2 Registers				
0x1_5000–0x1_5FFC	TLU 2 registers[9]			
L2/SRAM Memory-Mapped Configuration Registers				
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-9
0x2_0004	L2CWAP—L2 cache way allocation for processors	R/W	0x0000_0000	7.3.1.2/7-13

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3/7-15
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-17
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-17
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5/7-19
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5/7-19
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5/7-19
0x2_0E40	L2ERRDET—L2 error detect register	w1c	0x0000_0000	7.3.1.5/7-19
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5/7-19
0x2_0E50	L2ERRADDRH—L2 error address capture register high	R	0x0000_0000	7.3.1.5/7-19
0x2_0E54	L2ERRADDRL—L2 error address capture register low	R	0x0000_0000	7.3.1.5/7-19
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5/7-19
DMA Registers				
0x2_1100	MR*n*—DMA 0 mode register	R/W	0x0000_0000	19.3.1/19-9
0x2_1104	SR*n*—DMA 0 status register	Mixed	0x0000_0000	19.3.1/19-9
0x2_1108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_110C	CLNDAR*n*—DMA 0 current link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1110	SATR*n*—DMA 0 source attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_1114	SAR*n*—DMA 0 source address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1118	DATR*n*—DMA 0 destination attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_111C	DAR*n*—DMA 0 destination address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1120	BCR*n*—DMA 0 byte count register	R/W	0x0000_0000	19.3.1/19-9

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1128	NLNDAR*n*—DMA 0 next link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1134	CLSDAR*n*—DMA 0 current list alternate base descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_113C	NLSDAR*n*—DMA 0 next list descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1140	SSR*n*—DMA 0 source stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_1144	DSR*n*—DMA 0 destination stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_1180	MR*n*—DMA 1 mode register	R/W	0x0000_0000	19.3.1/19-9
0x2_1184	SR*n*—DMA 1 status register	Mixed	0x0000_0000	19.3.1/19-9
0x2_1188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_118C	CLNDAR*n*—DMA 1 current link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1190	SATR*n*—DMA 1 source attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_1194	SAR*n*—DMA 1 source address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1198	DATR*n*—DMA 1 destination attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_119C	DAR*n*—DMA 1 destination address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11A0	BCR*n*—DMA 1 byte count register	R/W	0x0000_0000	19.3.1/19-9
0x2_11A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11A8	NLNDAR*n*—DMA 1 next link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11B4	CLSDAR*n*—DMA 1 current list alternate base descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11BC	NLSDAR*n*—DMA 1 next list descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_11C0	SSR*n*—DMA 1 source stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_11C4	DSR*n*—DMA 1 destination stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_1200	MR*n*—DMA 2 mode register	R/W	0x0000_0000	19.3.1/19-9
0x2_1204	SR*n*—DMA 2 status register	Mixed	0x0000_0000	19.3.1/19-9
0x2_1208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_120C	CLNDAR*n*—DMA 2 current link descriptor address register	R/W	0x0000_0000	19.3.1/19-9

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1210	SATR*n*—DMA 2 source attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_1214	SAR*n*—DMA 2 source address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1218	DATR*n*—DMA 2 destination attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_121C	DAR*n*—DMA 2 destination address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1220	BCR*n*—DMA 2 byte count register	R/W	0x0000_0000	19.3.1/19-9
0x2_1224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1228	NLNDAR*n*—DMA 2 next link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1234	CLSDAR*n*—DMA 2 current list alternate base descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_123C	NLSDAR*n*—DMA 2 next list descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1240	SSR*n*—DMA 2 source stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_1244	DSR*n*—DMA 2 destination stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_1280	MR*n*—DMA 3 mode register	R/W	0x0000_0000	19.3.1/19-9
0x2_1284	SR*n*—DMA 3 status register	Mixed	0x0000_0000	19.3.1/19-9
0x2_1288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_128C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_1294	SAR3—DMA 3 source address register	R/W	0x0000_0000	19.3.1/19-9
0x2_1298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	19.3.1/19-9
0x2_129C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	19.3.1/19-9
0x2_12A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12A8	NLNDAR*n*—DMA 3 next link descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12B4	CLSDAR*3*—DMA 3 current list alternate base descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	19.3.1/19-9
0x2_12C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	19.3.1/19-9
0x2_12C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	19.3.1/19-9

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	19.3.1/19-9
eTSEC Registers				
eTSEC1 General Control and Status Registers				
0x2_4000	TSEC_ID*—Controller ID register	R	0x0124_0000	15.5.3.1/15-26
0x2_4004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	15.5.3.1/15-26
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	15.5.3.1/15-26
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	15.5.3.1/15-26
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	15.5.3.1/15-26
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	15.5.3.1/15-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	15.5.3.1/15-26
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0002	15.5.3.1/15-26
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	15.5.3.1/15-26
eTSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	15.5.3.2/15-41
0x2_4110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	15.5.3.2/15-41
0x2_4140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.2/15-41
0x2_418C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.2/15-41
0x2_419C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.2/15-41
0x2_41A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.2/15-41
0x2_41AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.2/15-41
0x2_41B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.2/15-41
0x2_41BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4200	TBASEH—TxBD base address high bits	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	15.5.3.2/15-41
0x2_420C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	15.5.3.2/15-41
0x2_421C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	15.5.3.2/15-41

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_422C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	15.5.3.2/15-41
0x2_423C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4280	TMR_TXTS1_ID*—Tx time stamp identification tag (set 1)	R/W	0x0000_0000	15.5.3.2/15-41
0x2_4284	TMR_TXTS2_ID*—Tx time stamp identification tag (set 2)	R/W	0x0000_0000	15.5.3.2/15-41
0x2_42C0	TMR_TXTS1_H*—Tx time stamp high (set 1)	R/W	0x0000_0000	15.5.3.2/15-41
0x2_42C4	TMR_TXTS1_L*—Tx time stamp high (set 1)	R/W	0x0000_0000	15.5.3.2/15-41
0x2_42C8	TMR_TXTS2_H*—Tx time stamp high (set 2)	R/W	0x0000_0000	15.5.3.2/15-41
0x2_42CC	TMR_TXTS2_L*—Tx time stamp high (set 2)	R/W	0x0000_0000	15.5.3.2/15-41
eTSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	15.5.3.3/15-53
0x2_4330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_433C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4384	RBPTR0—RxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.3/15-53
0x2_438C	RBPTR1*—RxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4394	RBPTR2*—RxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.3/15-53
0x2_439C	RBPTR3*—RxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.3/15-53
0x2_43A4	RBPTR4*—RxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.3/15-53
0x2_43AC	RBPTR5*—RxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.3/15-53
0x2_43B4	RBPTR6*—RxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.3/15-53
0x2_43BC	RBPTR7*—RxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4400	RBASEH—RxBD base address high bits	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4404	RBASE0—RxBD base address of ring 0	R/W	0x0000_0000	15.5.3.3/15-53
0x2_440C	RBASE1*—RxBD base address of ring 1	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4414	RBASE2*—RxBD base address of ring 2	R/W	0x0000_0000	15.5.3.3/15-53
0x2_441C	RBASE3*—RxBD base address of ring 3	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4424	RBASE4*—RxBD base address of ring 4	R/W	0x0000_0000	15.5.3.3/15-53
0x2_442C	RBASE5*—RxBD base address of ring 5	R/W	0x0000_0000	15.5.3.3/15-53
0x2_4434	RBASE6*—RxBD base address of ring 6	R/W	0x0000_0000	15.5.3.3/15-53

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_443C	RBASE7*—RxBD base address of ring 7	R/W	0x0000_0000	15.5.3.3/15-53
0x2_44C0	TMR_RXTS_H*—Rx timer time stamp register high	R/W	0x0000_0000	15.5.3.3/15-53
0X_44C4	TMR_RXTS_L* —Rx timer time stamp register low	R/W	0x0000_0000	15.5.3.3/15-53
eTSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	15.5.3.5/15-75
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	15.5.3.5/15-75
0x2_4508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	15.5.3.5/15-75
0x2_450C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	15.5.3.5/15-75
0x2_4510	MAXFRM—Maximum frame size	R/W	0x0000_0600	15.5.3.5/15-75
0x2_4520	MIIMCFG—MII management configuration	R/W	0x0000_0007	15.5.3.5/15-75
0x2_4524	MIIMCOM—MII management command	R/W	0x0000_0000	15.5.3.5/15-75
0x2_4528	MIIMADD—MII management address	R/W	0x0000_0000	15.5.3.5/15-75
0x2_452C	MIIMCON—MII management control	WO	0x0000_0000	15.5.3.5/15-75
0x2_4530	MIIMSTAT—MII management status	R	0x0000_0000	15.5.3.5/15-75
0x2_4534	MIIMIND—MII management indicator	R	0x0000_0000	15.5.3.5/15-75
0x2_453C	IFSTAT—Interface status	R	0x0000_0000	15.5.3.5/15-75
0x2_4540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	15.5.3.5/15-75
0x2_4544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	15.5.3.5/15-75

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	15.5.3.5/15-75
0x2_454C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x2_4550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x2_4554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x2_4558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x2_455C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x2_4560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x2_4564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x2_4568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x2_456C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x2_4570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x2_4574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x2_4578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x2_457C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x2_4580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x2_4584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x2_4588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x2_458C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x2_4590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	
0x2_4594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	
0x2_4598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	15.5.3.5/15-75
0x2_459C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x2_45A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x2_45A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x2_45A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x2_45AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x2_45B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x2_45B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x2_45B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x2_45BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
eTSEC1 Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	15.5.3.6/15-87
eTSEC1 Receive Counters				
0x2_469C	RBYT—Receive byte counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46A0	RPKT—Receive packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46BC	RALN—Receive alignment error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46C4	RCDE—Receive code error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46D4	RFRG—Receive fragments counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46D8	RJBR—Receive jabber counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46DC	RDRP—Receive drop counter	R/W	0x0000_0000	15.5.3.6/15-87
eTSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46E4	TPKT—Transmit packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_46FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_470C	TNCL—Transmit total collision counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	15.5.3.6/15-87

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_471C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4720	TXCF—Transmit control frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_4728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	15.5.3.6/15-87
0x2_472C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	15.5.3.6/15-87
eTSEC1 Counter Control and TOE Statistics Registers				
0x2_4730	CAR1—Carry register one register[10]	R	0x0000_0000	15.5.3.6/15-87
0x2_4734	CAR2—Carry register two register [10]	R	0x0000_0000	15.5.3.6/15-87
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	15.5.3.6/15-87
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	15.5.3.6/15-87
0x2_4740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	15.5.3.6/15-87
Hash Function Registers				
0x2_4800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	15.5.3.7/15-115
0x2_4804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x2_4808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x2_480C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x2_4810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x2_4814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x2_4818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x2_481C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	15.5.3.7/15-115
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
eTSEC1 FIFO Control Registers				
0x2_4A00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_0000	15.5.3.8/15-117
eTSEC1 DMA Attribute Registers				
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	15.5.3.9/15-119
0x2_4BFC	ATTRELI—Attribute extract length and extract index register	R/W	0x0000_0000	15.5.3.9/15-119
eTSEC1 1588 Registers				

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4E00	TMR_CTRL* - Timer control register	R/W	0x0001_0000	15.5.3.11/15-122
0x2_4E04	TMR_TEVENT* - time stamp event register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E08	TMR_TEMASK* - Timer event mask register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E0C	TMR_PEVENT* - time stamp event register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E10	TMR_PEMASK* - Timer event mask register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E14	TMR_STAT* - time stamp status register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E18	TMR_CNT_H* - timer counter high register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E1C	TMR_CNT_L* - timer counter low register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E20	TMR_ADD* - Timer drift compensation addend register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E24	TMR_ACC* - Timer accumulator register	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E28	TMR_PRSC* - timer prescale	R/W	0x0000_0002	15.5.3.11/15-122
0x2_4E30	TMR_OFF_H* - Timer offset high	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E34	TMR_OFF_L* - Timer offset low	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4E40	TMR_ALARM1_H* - Timer alarm 1 high register	R/W	0xFFFF_FFFF	15.5.3.11/15-122
0x2_4E44	TMR_ALARM1_L* - Timer alarm 1 high register	R/W	0xFFFF_FFFF	
0x2_4E48	TMR_ALARM2_H* - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x2_4E4C	TMR_ALARM2_L* - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x2_4E80	TMR_FIPER1* - Timer fixed period interval	R/W	0xFFFF_FFFF	15.5.3.11/15-122
0x2_4E84	TMR_FIPER2* - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x2_4E88	TMR_FIPER*3 - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x2_4EA0	TMR_ETTS1_H* - Time stamp of general purpose external trigger	R/W	0x0000_0000	15.5.3.11/15-122
0x2_4EA4	TMR_ETTS1_L* - Time stamp of general purpose external trigger	R/W	0x0000_0000	
0x2_4EA8	TMR_ETTS2_H* - Time stamp of general purpose external trigger	R/W	0x0000_0000	
0x2_4EAC	TMR_ETTS2_L* - Time stamp of general purpose external trigger	R/W	0x0000_0000	

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Other eTSECs				
0x2_5000–0x2_5FFC	eTSEC2 REGISTERS[11]			
0x2_6000–0x2_6FFC	eTSEC3 REGISTERS[12]			
0x2_7000–0x2_7FFC	eTSEC4 REGISTERS[13]			
Fast Ethernet Controller (FEC) Maintenance Controller Registers				
0x2_8010	IEVENT—Interrupt event register	R/W	0x0000_0000	16.5.3.1/16-12
0x2_8014	IMASK—Interrupt mask register	R/W	0x0000_0000	16.5.3.1/16-12
0x2_8018	EDIS—Error disabled register	R/W	0x0000_0000	16.5.3.1/16-12
0x2_8024	MINFLR—Minimum frame length register	R/W	0x0000_0040	16.5.3.1/16-12
0x2_8028	PTV—Pause time value register	R/W	0x0000_0000	16.5.3.1/16-12
0x2_802C	DMACTRL—DMA control register	R/W	0x0000_0000	16.5.3.1/16-12
FEC FIFO Control and Status Registers				
0x2_804C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	16.5.3.2/16-20
0x2_808C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	16.5.3.2/16-20
0x2_8098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	16.5.3.2/16-20
0x2_809C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	16.5.3.2/16-20
FEC Transmit Control and Status Registers				
0x2_8100	TCTRL—Transmit control register	R/W	0x0000_0000	16.5.3.3/16-23
0x2_8104	TSTAT—Transmit status register	R/W	0x0000_0000	16.5.3.3/16-23
0x2_810C	TBDLEN—TxBD data length register	R	0x0000_0000	16.5.3.3/16-23
0x2_8124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	16.5.3.3/16-23
0x2_8184	TBPTR—TxBD pointer register	R/W	0x0000_0000	16.5.3.3/16-23
0x2_8204	TBASE—TxBD base address register	R/W	0x0000_0000	16.5.3.3/16-23
0x2_82B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	16.5.3.3/16-23
0x2_82B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	16.5.3.3/16-23
FEC Receive Control and Status Registers				
0x2_8300	RCTRL—Receive control register	R/W	0x0000_0000	16.5.3.4/16-30
0x2_8304	RSTAT—Receive status register	R/W	0x0000_0000	16.5.3.4/16-30
0x2_830C	RBDLEN—RxBD data length register	R	0x0000_0000	16.5.3.4/16-30
0x2_8324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	16.5.3.4/16-30
0x2_8340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	16.5.3.4/16-30
0x2_8384	RBPTR—RxBD pointer register	R/W	0x0000_0000	16.5.3.4/16-30

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_8404	RBASE—RxBD base address register	R/W	0x0000_0000	16.5.3.4/16-30
FEC MAC Registers				
0x2_8500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	16.5.3.6/16-36
0x2_8504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	16.5.3.6/16-36
0x2_8508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	16.5.3.6/16-36
0x2_850C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	16.5.3.6/16-36
0x2_8510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	16.5.3.6/16-36
0x2_8520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	16.5.3.6/16-36
0x2_8524	MIIMCOM—MII management command register	R/W	0x0000_0000	16.5.3.6/16-36
0x2_8528	MIIMADD—MII management address register	R/W	0x0000_0000	16.5.3.6/16-36
0x2_852C	MIIMCON—MII management control register	W	0x0000_0000	16.5.3.6/16-36
0x2_8530	MIIMSTAT—MII management status register	R	0x0000_0000	16.5.3.6/16-36
0x2_8534	MIIMIND—MII management indicator register	R	0x0000_0000	16.5.3.6/16-36
0x2_853C	IFSTAT—Interface status register	R/W	0x0000_0000	16.5.3.6/16-36
0x2_8540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	16.5.3.6/16-36
0x2_8544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	16.5.3.6/16-36
FEC Hash Function Registers				
0x2_8800	IADDR0—Individual address register 0	R/W	0x0000_0000	16.5.3.7/16-48
0x2_8804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_8808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_880C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_8810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_8814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_8818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_881C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_8880	GADDR0—Group address register 0	R/W	0x0000_0000	16.5.3.7/16-48
0x2_8884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_8888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_888C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_8890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_8894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_8898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_889C	GADDR7—Group address register 7	R/W	0x0000_0000	
FEC Attribute Registers				
0x2_8BF8	ATTR—Attribute register	R/W	0x0000_0000	16.5.3.8/16-50

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_8BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	16.5.3.8/16-50
Table Lookup Unit 1 Registers				
TLU General Control/status Registers				
0x2_F000	TLU_ID1—TLU Identifier1 register	R	0x002F_0100	18.4.3.1/18-8
0x2_F004	TLU_ID2—TLU Identifier2 register	R	0x0000_0000	18.4.3.1/18-8
0x2_F010	IEVENT—Interrupt event register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F014	IMASK—Interrupt mask register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F018	IEATR—Interrupt error attributes register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F01C	IEADD—Interrupt error address register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F020	IEDIS—Interrupt error disable register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F040	MBANK0—Memory bank 0 base register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F044	MBANK1—Memory bank 1 base register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F048	MBANK2—Memory bank 2 base register	R/W	0x0000_0000	18.4.3.1/18-8
0x2_F04C	MBANK3—Memory bank 3 base register	R/W	0x0000_0000	18.4.3.1/18-8
TLU Physical Table Configuration Registers				
0x2_F100	PTBL0—Physical table 0 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F104	PTBL1—Physical table 1 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F108	PTBL2—Physical table 2 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F10C	PTBL3—Physical table 3 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F110	PTBL4—Physical table 4 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F114	PTBL5—Physical table 5 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F118	PTBL6—Physical table 6 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F11C	PTBL7—Physical table 7 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F120	PTBL8—Physical table 8 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F124	PTBL9—Physical table 9 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F128	PTBL10—Physical table 10 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F12C	PTBL11—Physical table 11 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F130	PTBL12—Physical table 12 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F134	PTBL13—Physical table 13 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F138	PTBL14—Physical table 14 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F13C	PTBL15—Physical table 15 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F140	PTBL16—Physical table 16 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F144	PTBL17—Physical table 17 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F148	PTBL18—Physical table 18 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F14C	PTBL19—Physical table 19 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F150	PTBL20—Physical table 20 configuration register	R/W	0x0000_0000	18.4.3.2/18-14

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_F154	PTBL21—Physical table 21 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F158	PTBL22—Physical table 22 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F15C	PTBL23—Physical table 23 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F160	PTBL24—Physical table 24 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F164	PTBL25—Physical table 25 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F168	PTBL26—Physical table 26 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F16C	PTBL27—Physical table 27 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F170	PTBL28—Physical table 28 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F174	PTBL29—Physical table 29 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F178	PTBL30—Physical table 30 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
0x2_F17C	PTBL31—Physical table 31 configuration register	R/W	0x0000_0000	18.4.3.2/18-14
TLU Statistics Counters				
0x2_F500	CRAMR—Memory reads count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F504	CRAMW—Memory writes count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F508	CFIND—Total find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F510	CTHTK—Hash/index-trie-key table find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F514	CTCRT—CRT table find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F518	CTCHS—Chained hash table find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F51C	CTDAT—Flat data table lookup count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F530	CHITS—Successful find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F534	CMISS—Failed find count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F538	CHCOL—Hash collision count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F53C	CCRTL—CRT level count register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F5F0	CARO—Counter carries-out register	R/W	0x0000_0000	18.4.3.3/18-16
0x2_F5F4	CARM—Counter carry mask register	R/W	0xFFF0_0000	18.4.3.3/18-16
TLU Command/Response Registers				
0x2_F600	CMDOP—TLU command operation register	R/W	0x0000_0000	18.4.3.4/18-23
0x2_F604	CMDIX—TLU command index register	R/W	0x0000_0000	18.4.3.4/18-23
0x2_F60C	CSTAT—TLU command status and response register	R/W	0x8000_0000	18.4.3.4/18-23
TLU Key/Data Registers				
0x2_F800	KD0B—Key/data word 0 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F804	KD1B—Key/data word 1 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F808	KD2B—Key/data word 2 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F80C	KD3B—Key/data word 3 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F810	KD4B—Key/data word 4 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F814	KD5B—Key/data word 5 buffer register	R/W	0x0000_0000	18.4.3.5/18-26

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_F818	KD6B—Key/data word 6 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
0x2_F81C	KD7B—Key/data word 7 buffer register	R/W	0x0000_0000	18.4.3.5/18-26
Integrated Security Engine (SEC) 3.0 Registers				
Controller				
0x3_1008	Interrupt enable	R/W	0x0000_0000	11.6.3/11-52
0x3_1010	Interrupt status	R	0x0000_0000	11.6.3/11-52
0x3_1018	Interrupt clear	R/W	0x0000_0000	11.6.3/11-52
0x3_1020	Identification	R	0x0000_0000	11.6.3/11-52
0x3_1028	EU assignment status	R	0x0000_0000	11.6.3/11-52
0x3_1030	Master control	R/W	0x0000_0000	11.6.3/11-52
Channel 1				
0x3_1108	Configuration register	R/W	0x0000_0000	11.5.4/11-40
0x3_1110	Pointer status	R/W	0x0000_0000	11.5.4/11-40
0x3_1140	Current descriptor pointer	R	0x0000_0000	11.5.4/11-40
0x3_1148	Fetch FIFO	W	0x0000_0000	11.5.4/11-40
0x3_1180 - 0x3_11BF	Descriptor buffer	R	0x0000_0000	11.5.4/11-40
0x3_11C0 - 0x3_11DF	Gather link tables	W	0x0000_0000	11.5.4/11-40
0x3_11E0 - 0x3_11FF	Scatter link tables	W	0x0000_0000	11.5.4/11-40
Channel 2				
0x3_1208	Configuration register	R/W	0x0000_0000	11.5.4/11-40
0x3_1210	Pointer status	R/W	0x0000_0000	11.5.4/11-40
0x3_1240	Current descriptor pointer	R	0x0000_0000	11.5.4/11-40
0x3_1248	Fetch FIFO	W	0x0000_0000	11.5.4/11-40
0x3_1280 - 0x3_12BF	Descriptor buffer	R	0x0000_0000	11.5.4/11-40
0x3_12C0 - 0x3_12DF	Gather link tables	W	0x0000_0000	11.5.4/11-40
0x3_12E0 - 0x3_12FF	Scatter link tables	W	0x0000_0000	11.5.4/11-40
Channel 3				
0x3_1308	Configuration register	R/W	0x0000_0000	11.5.4/11-40
0x3_1310	Pointer status	R/W	0x0000_0000	11.5.4/11-40
0x3_1340	Current descriptor pointer	R	0x0000_0000	11.5.4/11-40
0x3_1348	Fetch FIFO	W	0x0000_0000	11.5.4/11-40

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_1380 - 0x3_13BF	Descriptor buffer	R	0x0000_0000	11.5.4/11-40
0x3_13C0 - 0x3_13DF	Gather link tables	W	0x0000_0000	11.5.4/11-40
0x3_13E0 - 0x3_13FF	Scatter link tables	W	0x0000_0000	11.5.4/11-40
Channel 4				
0x3_1408	Configuration register	R/W	0x0000_0000	11.5.4/11-40
0x3_1410	Pointer status	R/W	0x0000_0000	11.5.4/11-40
0x3_1440	Current descriptor pointer	R	0x0000_0000	11.5.4/11-40
0x3_1448	Fetch FIFO	W	0x0000_0000	11.5.4/11-40
0x3_1480 - 0x3_14BF	Descriptor buffer	R	0x0000_0000	11.5.4/11-40
0x3_14C0 - 0x3_14DF	Gather link tables	W	0x0000_0000	11.5.4/11-40
0x3_14E0 - 0x3_14FF	Scatter link tables	W	0x0000_0000	11.5.4/11-40
Controller				
0x3_1BF8	IP block revision	R	0x0000_0000	11.6.3/11-52
DEU				
0x3_2000	Mode register	R/W	0x0000_0000	11.7.4/11-112
0x3_2008	Key size register	R/W	0x0000_0000	11.7.4/11-112
0x3_2010	Data size register	R/W	0x0000_0000	11.7.4/11-112
0x3_2018	Reset control register	R/W	0x0000_0000	11.7.4/11-112
0x3_2028	Status register	R	0x0000_0000	11.7.4/11-112
0x3_2030	Interrupt status register	R/W	0x0000_0000	11.7.4/11-112
0x3_2038	Interrupt mask register	R/W	0x0000_0000	11.7.4/11-112
0x3_2050	EU-Go	W	0x0000_0000	11.7.4/11-112
0x3_2100	IV register	R/W	0x0000_0000	11.7.4/11-112
0x3_2400	Key 1 register	W	0x0000_0000	11.7.4/11-112
0x3_2408	Key 2 register	W	0x0000_0000	11.7.4/11-112
0x3_2410	Key 3 register	W	0x0000_0000	11.7.4/11-112
0x3_2800 - 0x3_2FFF	Input FIFO / Output FIFO	R/W	0x0000_0000	11.7.4/11-112
AESU				
0x3_4000	Mode register	R/W	0x0000_0000	11.7.1/11-62
0x3_4008	Key size register	R/W	0x0000_0000	11.7.1/11-62

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_4010	Data size register	R/W	0x0000_0000	11.7.1/11-62
0x3_4018	Reset control register	R/W	0x0000_0000	11.7.1/11-62
0x3_4028	Status register	R	0x0000_0000	11.7.1/11-62
0x3_4030	Interrupt status register	R/W	0x0000_0000	11.7.1/11-62
0x3_4038	Interrupt mask register	R/W	0x0000_0000	11.7.1/11-62
0x3_4050	End of message register	W	0x0000_0000	11.7.1/11-62
0x3_4100 - 0x3_415F	Context	R/W	0x0000_0000	11.7.1/11-62
0x3_4400 - 0x3_441F	Key memory	R/W	0x0000_0000	11.7.1/11-62
0x3_4800 - 0x3_4FFF	Input FIFO / Output FIFO	R/W	0x0000_0000	11.7.1/11-62
MDEU				
0x3_6000	Mode register	R/W	0x0000_0000	11.7.6/11-135
0x3_6008	Key size register	R/W	0x0000_0000	11.7.6/11-135
0x3_6010	Data size register	R/W	0x0000_0000	11.7.6/11-135
0x3_6018	Reset control register	R/W	0x0000_0000	11.7.6/11-135
0x3_6028	Status register	R	0x0000_0000	11.7.6/11-135
0x3_6030	Interrupt status register	R/W	0x0000_0000	11.7.6/11-135
0x3_6038	Interrupt mask register	R/W	0x0000_0000	11.7.6/11-135
0x3_6040	ICV size register	W	0x0000_0000	11.7.6/11-135
0x3_6050	End_of_message	W	0x0000_0000	11.7.6/11-135
0x3_6100 - 0x3_6140	Context registers	R/W	0x0000_0000	11.7.6/11-135
0x3_6400 - 0x3_647F	Key registers	W	0x0000_0000	11.7.6/11-135
0x3_6800 - 0x3_6FFF	Input FIFO	W	0x0000_0000	11.7.6/11-135
AFEU				
0x3_8000	Mode register	R/W	0x0000_0000	11.7.2/11-92
0x3_8008	Key size register	R/W	0x0000_0000	11.7.2/11-92
0x3_8010	Data size register	R/W	0x0000_0000	11.7.2/11-92
0x3_8018	Reset control register	R/W	0x0000_0000	11.7.2/11-92
0x3_8028	Status register	R	0x0000_0000	11.7.2/11-92
0x3_8030	Interrupt status register	R/W	0x0000_0000	11.7.2/11-92
0x3_8038	Interrupt mask register	R/W	0x0000_0000	11.7.2/11-92
0x3_8050	End of message register	W	0x0000_0000	11.7.2/11-92

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_8100 - 0x3_81FF	Context memory	R/W	0x0000_0000	11.7.2/11-92
0x3_8200	Context memory pointers	R/W	0x0000_0000	11.7.2/11-92
0x3_8400 - 0x3_840F	Key registers	W	0x0000_0000	11.7.2/11-92
0x3_8800 - 0x3_8FFF (0x3_8E00)	Input FIFO / Output FIFO (special context address)	R/W	0x0000_0000	11.7.2/11-92
RNGU				
0x3_A000	Mode register	R/W	0x0000_0000	11.7.8/11-159
0x3_A010	Data size register	R/W	0x0000_0000	11.7.8/11-159
0x3_A018	Reset control register	R/W	0x0000_0000	11.7.8/11-159
0x3_A028	Status register	R	0x0000_0000	11.7.8/11-159
0x3_A030	Interrupt status register	R/W	0x0000_0000	11.7.8/11-159
0x3_A038	Interrupt mask register	R/W	0x0000_0000	11.7.8/11-159
0x3_A050	End_of_message	W	0x0000_0000	11.7.8/11-159
0x3_A400 - 0x3_A43F	Entropy Registers	W	0x0000_0000	11.7.8/11-159
0x3_A800 - 0x3_AFFF	Output FIFO	R	0x0000_0000	11.7.8/11-159
PKEU				
0x3_C000	Mode register	R/W	0x0000_0000	11.7.7/11-150
0x3_C008	Key size register	R/W	0x0000_0000	11.7.7/11-150
0x3_C010	Data size register	R/W	0x0000_0000	11.7.7/11-150
0x3_C018	Reset control register	R/W	0x0000_0000	11.7.7/11-150
0x3_C028	Status register	R	0x0000_0000	11.7.7/11-150
0x3_C030	Interrupt status register	R/W	0x0000_0000	11.7.7/11-150
0x3_C038	Interrupt mask register	R/W	0x0000_0000	11.7.7/11-150
0x3_C040	ABSize	R/W	0x0000_0000	11.7.7/11-150
0x3_C050	End_of_message	W	0x0000_0000	11.7.7/11-150
0x3_C200 - 0x3_C27F	Parameter memory A0	R/W	0x0000_0000	11.7.7/11-150
0x3_C280 - 0x3_C2FF	Parameter memory A1	R/W	0x0000_0000	11.7.7/11-150
0x3_C300 - 0x3_C37F	Parameter memory A2	R/W	0x0000_0000	11.7.7/11-150
0x3_C380 - 0x3_C3FF	Parameter memory A3	R/W	0x0000_0000	11.7.7/11-150

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x3_C400 - 0x3_C47F	Parameter memory B0	R/W	0x0000_0000	11.7.7/11-150
0x3_C480 - 0x3_C4FF	Parameter memory B1	R/W	0x0000_0000	11.7.7/11-150
0x3_C500 - 0x3_C57F	Parameter memory B2	R/W	0x0000_0000	11.7.7/11-150
0x3_C580 - 0x3_C5FF	Parameter memory B3	R/W	0x0000_0000	11.7.7/11-150
0x3_C800 - 0x3_C9FF	Parameter memory N	R/W	0x0000_0000	11.7.7/11-150
0x3_CA00 - 0x3_CBFF	Parameter memory E	W	0x0000_0000	11.7.7/11-150
KEU				
0x3_E000	Mode register	R/W	0x0000_0000	11.7.5/11-121
0x3_E008	Key size register	R/W	0x0000_0000	11.7.5/11-121
0x3_E010	Data size register	R/W	0x0000_0000	11.7.5/11-121
0x3_E018	Reset control register	R/W	0x0000_0000	11.7.5/11-121
0x3_E028	Status register	R	0x0000_0000	11.7.5/11-121
0x3_E030	Interrupt Status register	R/W	0x0000_0000	11.7.5/11-121
0x3_E038	Interrupt Mask register	R/W	0x0000_0000	11.7.5/11-121
0x3_E048	Data out register (F9 MAC)	R	0x0000_0000	11.7.5/11-121
0x3_E050	End of message register	W	0x0000_0000	11.7.5/11-121
0x3_E100	IV_1 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E108	ICV_In register	R/W	0x0000_0000	11.7.5/11-121
0x3_E110	IV_2 register (Fresh)	R/W	0x0000_0000	11.7.5/11-121
0x3_E118	Context_1 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E120	Context_2 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E128	Context_3 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E130	Context_4 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E138	Context_5 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E140	Context_6 register	R/W	0x0000_0000	11.7.5/11-121
0x3_E400	Key data register_1 (CK-high)	R/W	0x0000_0000	11.7.5/11-121
0x3_E408	Key data register_2 (CK-low)	R/W	0x0000_0000	11.7.5/11-121
0x3_E410	Key dataregister_3 (IK-high)	R/W	0x0000_0000	11.7.5/11-121
0x3_E418	Key data register_4 (IK-low)	R/W	0x0000_0000	11.7.5/11-121
0x3_E800 - 0x3_EFFF	Input FIFO / Output FIFO	R/W	0x0000_0000	11.7.5/11-121

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
CRCU				
0x3_F000	Mode register	R/W	0x0000_0000	11.7.3/11-102
0x3_F008	Key size register	R/W	0x0000_0000	11.7.3/11-102
0x3_F010	Data size register	R/W	0x0000_0000	11.7.3/11-102
0x3_F018	Reset control register	R/W	0x0000_0000	11.7.3/11-102
0x3_F020	Control	R/W	0x0000_0000	11.7.3/11-102
0x3_F028	Status register	R	0x0000_0000	11.7.3/11-102
0x3_F030	Interrupt status register	R/W	0x0000_0000	11.7.3/11-102
0x3_F038	Interrupt mask register	R/W	0x0000_0000	11.7.3/11-102
0x3_F048	Data out register	R	0x0000_0000	11.7.3/11-102
0x3_F050	End of message register	W	0x0000_0000	11.7.3/11-102
0x3_F100	Received ICV register	R/W	0x0000_0000	11.7.3/11-102
0x3_F108	Context register	R/W	0x0000_0000	11.7.3/11-102
0x3_F400	Key register	R/W	0x0000_0000	11.7.3/11-102
0x3_F800 - 0x3_FFFF	Input FIFO	W	0x0000_0000	11.7.3/11-102
Programmable Interrupt Controller Register Address Map				
Global Registers				
0x4_0000	BRR1—Block revision register 1	R	0x0040_0301	10.3.1/10-20
0x4_0010	BRR2—Block revision register 2	R	0x0000_0001	10.3.1/10-20
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.1/10-20
0x4_0050	IPIDR1—IPI 1 dispatch register			10.3.1/10-20
0x4_0060	IPIDR2—IPI 2 dispatch register			10.3.1/10-20
0x4_0070	IPIDR3—IPI 3 dispatch register			10.3.1/10-20
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.1/10-20
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.1/10-20
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.1/10-20
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.1/10-20
0x4_1000	FRR—Feature reporting register	R	0x006B_0102	10.3.1/10-20
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1/10-20
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1/10-20
0x4_1090	PIR—Processor core initialization register	R/W	0x0000_0000	10.3.1/10-20

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1/10-20
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			10.3.1/10-20
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			10.3.1/10-20
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			10.3.1/10-20
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1/10-20
Global Timer Group A Registers				
0x4_10F0	TFRRA—Timer frequency reporting register (Group A)	R/W	0x0000_0000	10.3.2/10-25
0x4_1100	GTCCRA0—Global timer 0 current count register (Group A)	R	0x0000_0000	10.3.2/10-25
0x4_1110	GTBCRA0—Global timer 0 base count register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_1120	GTVPRA0—Global timer 0 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_1130	GTDRA0—Global timer 0 destination register (Group A)	R/W	0x0000_0001	10.3.2/10-25
0x4_1140	GTCCRA1—Global timer 1 current count register (Group A)	R	0x0000_0000	10.3.2/10-25
0x4_1150	GTBCRA1—Global timer 1 base count register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_1160	GTVPRA1—Global timer 1 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_1170	GTDRA1—Global timer 1 destination register (Group A)	R/W	0x0000_0001	10.3.2/10-25
0x4_1180	GTCCRA2—Global timer 2 current count register (Group A)	R	0x0000_0000	10.3.2/10-25
0x4_1190	GTBCRA2—Global timer 2 base count register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_11A0	GTVPRA2—Global timer 2 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_11B0	GTDRA2—Global timer 2 destination register (Group A)	R/W	0x0000_0001	10.3.2/10-25
0x4_11C0	GTCCRA3—Global timer 3 current count register (Group A)	R	0x0000_0000	10.3.2/10-25
0x4_11D0	GTBCRA3—Global timer 3 base count register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_11E0	GTVPRA3—Global timer 3 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2/10-25
0x4_11F0	GTDRA3—Global timer 3 destination register (Group A)	R/W	0x0000_0001	10.3.2/10-25
0x4_1300	TCRA—Timer control register (Group A)	R/W	0x0000_0000	10.3.2/10-25
0x4_1308	ERQSR—External interrupt summary register	R	0x0000_0000	10.3.3/10-30
0x4_1310	IRQSR0—IRQ_OUT summary register 0	R	0x0000_0000	10.3.3/10-30
0x4_1320	IRQSR1—IRQ_OUT summary register 1	R	0x0000_0000	10.3.3/10-30
0x4_1324	IRQSR2—IRQ_OUT summary register 2	R	0x0000_0000	10.3.3/10-30
0x4_1330	CISR0—Critical interrupt summary register 0	R	0x0000_0000	10.3.3/10-30
0x4_1340	CISR1—Critical interrupt summary register 1	R	0x0000_0000	10.3.3/10-30
0x4_1344	CISR2—Critical interrupt summary register 2	R	0x0000_0000	10.3.3/10-30
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_0000	10.3.4/10-34
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4/10-34

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_0000	10.3.4/10-34
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_0000	10.3.4/10-34
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4/10-34
0x4_13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_0000	10.3.4/10-34
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5/10-36
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5/10-36
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5/10-36
0x4_1600	MSIR0—Message-shared interrupt register 0	RC	0x0000_0000	10.3.6/10-38
0x4_1610	MSIR1—Message-shared interrupt register 1	RC	0x0000_0000	10.3.6/10-38
0x4_1620	MSIR2—Message-shared interrupt register 2	RC	0x0000_0000	10.3.6/10-38
0x4_1630	MSIR3—Message-shared interrupt register 3	RC	0x0000_0000	10.3.6/10-38
0x4_1640	MSIR4—Message-shared interrupt register 4	RC	0x0000_0000	10.3.6/10-38
0x4_1650	MSIR5—Message-shared interrupt register 5	RC	0x0000_0000	10.3.6/10-38
0x4_1660	MSIR6—Message-shared interrupt register 6	RC	0x0000_0000	10.3.6/10-38
0x4_1670	MSIR7—Message-shared interrupt register 7	RC	0x0000_0000	10.3.6/10-38
0x4_1700	MSIMR—Message-shared interrupt mask register	R/W	0x0000_0000	10.3.6/10-38
0x4_1720	MSISR—Message-shared interrupt status register	R	0x0000_0000	10.3.6/10-38
0x4_1740	MSIIR—Message-shared interrupt index register	W	0x0000_0000	10.3.6/10-38
Global Timer Group B Registers				
0x4_20F0	TFRRB—Timer frequency reporting register group B	R/W	0x0000_0000	10.3.2/10-25
0x4_2100	GTCCRB0—Global timer current count register group B 0	R	0x0000_0000	10.3.2/10-25
0x4_2110	GTBCRB0—Global timer base count register group B 0	R/W	0x8000_0000	10.3.2/10-25
0x4_2120	GTVPRB0—Global timer vector/priority register group B 0	R/W	0x8000_0000	10.3.2/10-25
0x4_2130	GTDRB0—Global timer destination register group B 0	R/W	0x0000_0001	10.3.2/10-25
0x4_2140	GTCCRB1—Global timer current count register group B 1	R	0x0000_0000	10.3.2/10-25
0x4_2150	GTBCRB1—Global timer base count register group B 1	R/W	0x8000_0000	10.3.2/10-25
0x4_2160	GTVPRB1—Global timer vector/priority register group B 1	R/W	0x8000_0000	10.3.2/10-25
0x4_2170	GTDRB1—Global timer destination register group B 1	R/W	0x0000_0001	10.3.2/10-25
0x4_2180	GTCCRB2—Global timer current count register group B 2	R	0x0000_0000	10.3.2/10-25
0x4_2190	GTBCRB2—Global timer base count register group B 2	R/W	0x8000_0000	10.3.2/10-25

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_21A0	GTVPRB2—Global timer vector/priority register group B 2	R/W	0x8000_0000	10.3.2/10-25
0x4_21B0	GTDRB2—Global timer destination register group B 2	R/W	0x0000_0001	10.3.2/10-25
0x4_21C0	GTCCRB3—Global timer current count register group B 3	R	0x0000_0000	10.3.2/10-25
0x4_21D0	GTBCRB3—Global timer base count register group B 3	R/W	0x8000_0000	10.3.2/10-25
0x4_21E0	GTVPRB3—Global timer vector/priority register group B 3	R/W	0x8000_0000	10.3.2/10-25
0x4_21F0	GTDRB3—Global timer destination register group B 3	R/W	0x0000_0001	10.3.2/10-25
0x4_2300	TCRB—Timer control register (Group B)	R/W	0x0000_0000	10.3.2/10-25
0x4_2400	MSGR4—Message register 4	R/W	0x0000_0000	10.3.5/10-36
0x4_2410	MSGR5—Message register 5			
0x4_2420	MSGR6—Message register 6			
0x4_1430	MSGR7—Message register 7			
0x4_2500	MER—Message enable register (for MSGR4-7)	R/W	0x0000_0000	10.3.5/10-36
0x4_2510	MSR—Message status register (for MSGG4-7)	R/W	0x0000_0000	10.3.5/10-36
Interrupt Source Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register or PCIe1-INTA vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register or PCIe1-INTA destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register or PCIe1-INTB vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register or PCIe1-INTB destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register or PCIe1-INTC vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register or PCIe1-INTC destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register or PCIe1-INTD vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register or PCIe1-INTD destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register or PCIe2-INTA vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register or PCIe2-INTA destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register or PCIe2-INTB vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register or PCIe2-INTB destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register or PCIe2-INTC vector/priority register	R/W	0x8000_0000	10.3.7/10-42

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register or PCIe2-INTC destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register or PCIe2-INTD vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register or PCIe2-INTD destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register or PCIe3-INTA vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register or PCIe3-INTA destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register or PCIe3-INTB vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register or PCIe3-INTB destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register or PCIe3-INTC vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register or PCIe3-INTC destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register or PCIe3-INTD vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register or PCIe3-INTD destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.7/10-42

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.7/10-42

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0550	IIDR26—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0600	IIVPR32—Internal interrupt 32 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0610	IIDR32—Internal interrupt 32 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0620	IIVPR33—Internal interrupt 33 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0630	IIDR33—Internal interrupt 33 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0640	IIVPR34—Internal interrupt 34 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0650	IIDR34—Internal interrupt 34 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0660	IIVPR35—Internal interrupt 35 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0670	IIDR35—Internal interrupt 35 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0680	IIVPR36—Internal interrupt 36 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0690	IIDR36—Internal interrupt 36 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_06A0	IIVPR37—Internal interrupt 37 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_06B0	IIDR37—Internal interrupt 37 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_06C0	IIVPR38—Internal interrupt 38 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_06D0	IIDR38—Internal interrupt 38 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_06E0	IIVPR39—Internal interrupt 39 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_06F0	IIDR39—Internal interrupt 39 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0700	IIVPR40—Internal interrupt 40 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0710	IIDR40—Internal interrupt 40 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0720	IIVPR41—Internal interrupt 41 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0730	IIDR41—Internal interrupt 41 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0740	IIVPR42—Internal interrupt 42 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0750	IIDR42—Internal interrupt 42 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0760	IIVPR43—Internal interrupt 43 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0770	IIDR43—Internal interrupt 43 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0780	IIVPR44—Internal interrupt 44 vector/priority register	R/W	0x8080_0000	10.3.7/10-42

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0790	IIDR44—Internal interrupt 44 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_07A0	IIVPR45—Internal interrupt 45 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_07B0	IIDR45—Internal interrupt 45 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_07C0	IIVPR46—Internal interrupt 46 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_07D0	IIDR46—Internal interrupt 46 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_07E0	IIVPR47—Internal interrupt 47 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_07F0	IIDR47—Internal interrupt 47 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0800	IIDR48—Internal interrupt 48 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0810	IIVPR48—Internal interrupt 48 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0820	IIDR49—Internal interrupt 49 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0830	IIVPR49—Internal interrupt 49 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0840	IIDR50—Internal interrupt 50 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0850	IIVPR50—Internal interrupt 50 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0860	IIDR51—Internal interrupt 51 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0870	IIVPR51—Internal interrupt 51 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0880	IIDR52—Internal interrupt 52 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0890	IIVPR52—Internal interrupt 52 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_08A0	IIDR53—Internal interrupt 53 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_08B0	IIVPR53—Internal interrupt 53 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_08C0	IIDR54—Internal interrupt 54 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_08D0	IIVPR54—Internal interrupt 54 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_08E0	IIDR55—Internal interrupt 55 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_08F0	IIVPR55—Internal interrupt 55 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0900	IIDR56—Internal interrupt 56 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0910	IIVPR56—Internal interrupt 56 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0920	IIDR57—Internal interrupt 57 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0930	IIVPR57—Internal interrupt 57 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0940	IIDR58—Internal interrupt 58 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0950	IIVPR58—Internal interrupt 58 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0960	IIDR59—Internal interrupt 59 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0970	IIVPR59—Internal interrupt 59 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_0980	IIDR60—Internal interrupt 60 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_0990	IIVPR60—Internal interrupt 60 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_09A0	IIDR61—Internal interrupt 61 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_09B0	IIVPR61—Internal interrupt 61 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_09C0	IIDR62—Internal interrupt 62 destination register	R/W	0x0000_0001	10.3.7/10-42

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_09D0	IIVPR62—Internal interrupt 62 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_09E0	IIDR63—Internal interrupt 63 destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_09F0	IIVPR63—Internal interrupt 63 vector/priority register	R/W	0x8080_0000	10.3.7/10-42
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_1680	MIVPR4—Messaging interrupt 4 (MSG 4) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_1690	MIDR4—Messaging interrupt 4 (MSG 4) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_16A0	MIVPR5—Messaging interrupt 5 (MSG 5) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_16B0	MIDR5—Messaging interrupt 5 (MSG 5) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_16C0	MIVPR6—Messaging interrupt 6 (MSG 6) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_16D0	MIDR6—Messaging interrupt 6 (MSG 6) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_16E0	MIVPR7—Messaging interrupt 7 (MSG 7) vector/priority register	R/W	0x8000_0000	10.3.7/10-42
0x5_16F0	MIDR7—Messaging interrupt 7 (MSG 7) destination register	R/W	0x0000_0001	10.3.7/10-42
0x5_1C00	MSIVPR0—Message-shared interrupt vector/priority register 0	R/W	0x8000_0000	10.3.7/10-42
0x5_1C10	MSIDR0—Message-shared interrupt destination register 0	R/W	0x0000_0001	10.3.7/10-42
0x5_1C20	MSIVPR1—Message-shared interrupt vector/priority register 1	R/W	0x8000_0000	10.3.7/10-42
0x5_1C30	MSIDR1—Message-shared interrupt destination register 1	R/W	0x0000_0001	10.3.7/10-42
0x5_1C40	MSIVPR2—Message-shared interrupt vector/priority register 2	R/W	0x8000_0000	10.3.7/10-42
0x5_1C50	MSIDR2—Message-shared interrupt destination register 2	R/W	0x0000_0001	10.3.7/10-42
0x5_1C60	MSIVPR3—Message-shared interrupt vector/priority register 3	R/W	0x8000_0000	10.3.7/10-42
0x5_1C70	MSDIR3—Message-shared interrupt destination register 3	R/W	0x0000_0001	10.3.7/10-42
0x5_1C80	MSIVPR4—Message-shared interrupt vector/priority register 4	R/W	0x8000_0000	10.3.7/10-42
0x5_1C90	MSIDR4—Message-shared interrupt destination register 4	R/W	0x0000_0001	10.3.7/10-42
0x5_1CA0	MSIVPR5—Message-shared interrupt vector/priority register 5	R/W	0x8000_0000	10.3.7/10-42
0x5_1CB0	MSIDR5—Message-shared interrupt destination register 5	R/W	0x0000_0001	10.3.7/10-42
0x5_1CC0	MSIVPR6—Message-shared interrupt vector/priority register 6	R/W	0x8000_0000	10.3.7/10-42
0x5_1CD0	MSIDR6—Message-shared interrupt destination register 6	R/W	0x0000_0001	10.3.7/10-42
0x5_1CE0	MSIVPR7—Message-shared interrupt vector/priority register 7	R/W	0x8000_0000	10.3.7/10-42
0x5_1CF0	MSDIR7—Message-shared interrupt destination register 7	R/W	0x0000_0001	10.3.7/10-42
PIC Register Address Map—Per-CPU Registers				

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x6_0040	IPIDR0—Processor core 0 interprocessor 0 dispatch register	W	all zeros	10.3.8/10-49
0x6_0050	IPIDR1—Processor core 0 interprocessor 1 dispatch register			
0x6_0060	IPIDR2—Processor core 0 interprocessor 2 dispatch register			
0x6_0070	IPIDR3—Processor core 0 interprocessor 3 dispatch register			
0x6_0080	CTPR0—Processor core 0 current task priority register	R/W	0x0000_000F	10.3.8/10-49
0x6_0090	WHOAMI0—Processor core 0 who am I register	R	all zeros	10.3.8/10-49
0x6_00A0	IACK0—Processor core 0 interrupt acknowledge register	R	all zeros	10.3.8/10-49
0x6_00B0	EOI0—Processor core 0 end of interrupt register	W	all zeros	10.3.8/10-49
0x6_1040	IPIDR0—Processor core 1 interprocessor 0 dispatch register	W	all zeros	10.3.8/10-49
0x6_1050	IPIDR1—Processor core 1 interprocessor 1 dispatch register			
0x6_1060	IPIDR2—Processor core 1 interprocessor 2 dispatch register			
0x6_1070	IPIDR3—Processor core 1 interprocessor 3 dispatch register			
0x6_1080	CTPR1—Processor core 1 current task priority register	R/W	0x0000_000F	10.3.8/10-49
0x6_1090	WHOAMI1—Processor core 1 who am I register	R	all zeros	10.3.8/10-49
0x6_10A0	IACK1—Processor core 1 interrupt acknowledge register	R	all zeros	10.3.8/10-49
0x6_10B0	EOI1—Processor core 1 end of interrupt register	W	all zeros	10.3.8/10-49
RapidIO Registers[14]				
RapidIO Architectural Registers				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0040_0002 = MPC8572E 0x0041_0002 = MPC8572	20.6.1/20-10
0xC_0004	Device information capability register (DICAR)	R	0x*nnnn_nnnn*	20.6.1/20-10
0xC_0008	Assembly identity capability register (AIDCAR)	R/W	0x*nnnn_nnnn*	20.6.1/20-10
0xC_000C	Assembly information capability register (AICAR)	R/W	0x0000_0000	20.6.1/20-10
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE0F8_00*n*9	20.6.1/20-10
0xC_0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	20.6.1/20-10
0xC_001C	Destination operations capability register (DOCAR)	R	0x0000_FCF4	20.6.1/20-10
0xC_0040	Mailbox command and status register (MCSR)	R	0x2020_0000	20.6.1/20-10
0xC_0044	Port -Write and doorbell command and status register (PWDCSR)	R	0x2000_0020	20.6.1/20-10
0xC_004C	Processing element logical layer control command and status register (PELLCCSR)	R	0x0000_0001	20.6.1/20-10
0xC_005C	Local configuration space base address 1 command and status register (LCSBA1CSR)	R/W	0x0000_0000	20.6.1/20-10
0xC_0060	Base device ID command and status register (BDIDCSR)	R/W	0x00*nn_nnnn*	20.6.1/20-10
0xC_0068	Host base device ID lock command and status register (HBDIDLCSR)	Special	0x0000_FFFF	20.6.1/20-10

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_006C	Component tag command and status register (CTCSR)	R/W	0x0000_0000	20.6.1/20-10
Extended Features Space				
1x/4x LP-Serial				
0xC_0100	Port maintenance block header 0 (PMBH0)	R	0x0600_0001	20.6.2/20-22
0xC_0120	Port link time-out control command and status register (PLTOCCSR)	R/W	0xFFFF_FF00	20.6.2/20-22
0xC_0124	Port response time-out control command and status register (PRTOCCSR)	R/W	0xFFFF_FF00	20.6.2/20-22
0xC_013C	General control command and status register (GCCSR)	R/W	0x*n*000_0000	20.6.2/20-22
0xC_0140	Link maintenance request command and status register (LMREQCSR)	R/W	0x0000_0000	20.6.2/20-22
0xC_0144	Link maintenance response command and status register (LMRESPCSR)	R	0x0000_0000	20.6.2/20-22
0xC_0148	Local ackID status command and status register (LASCSR)	Mixed	0x0000_0000	20.6.2/20-22
0xC_0158	Error and status command and status register (ESCSR)	Mixed	0x0000_0001	20.6.2/20-22
0xC_015C	Control command and status register (CCSR)	R/W	0x5060_0001	20.6.2/20-22
Error Reporting, Logical				
0xC_0600	Error reporting block header (ERBH)	R	0x0000_0007	20.6.3/20-30
0xC_0608	Logical/Transport layer error detect command and status register (LTLEDCSR)	R/W	0x0000_0000	20.6.3/20-30
0xC_060C	Logical/Transport layer error enable command and status register (LTLEECSR)	R/W	0x0000_0000	20.6.3/20-30
0xC_0614	Logical/Transport layer address capture command and status register (LTLACCSR)	R/W	0x0000_0000	20.6.3/20-30
0xC_0618	Logical/Transport layer device ID capture command and status register (LTLDIDCCSR)	R/W	0x0000_0000	20.6.3/20-30
0xC_061C	Logical/Transport layer control capture command and status register (LTLCCCSR)	R/W	0x0000_0000	20.6.3/20-30
Error Reporting, Physical				
0xC_0640	Error detect command and status register (EDCSR)	R/W	0x0000_0000	20.6.4/20-36
0xC_0644	Error rate enable command and status register (ERECSR)	R/W	0x0000_0000	20.6.4/20-36
0xC_0648	Error capture attributes command and status register (ECACSR)	R/W	0x0000_0000	20.6.4/20-36
0xC_064C	Packet/control symbol error capture command and status register 0 (PCSECCSR0)	R/W	0x0000_0000	20.6.4/20-36
0xC_0650	Packet error capture command and status register 1 (PECCSR1)	R/W	0x0000_0000	20.6.4/20-36
0xC_0654	Packet error capture command and status register 2 (PECCSR2)	R/W	0x0000_0000	20.6.4/20-36
0xC_0658	Packet error capture command and status register 3 (PECCSR3)	R/W	0x0000_0000	20.6.4/20-36
0xC_0668	Error rate command and status register (ERCSR)	R/W	0x8000_0000	20.6.4/20-36

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_066C	Error rate threshold command and status register (ERTCSR)	R/W	0xFFFF_0000	20.6.4/20-36
General				
0xD_0004	Logical layer configuration register (LLCR)	R/W	0x0000_0000	20.6.5/20-42
0xD_0010	Error / port-write interrupt status register (EPWISR)	R	0x0000_0000	20.6.5/20-42
0xD_0020	Logical retry error threshold configuration register (LRETCR)	R/W	0x0000_00FF	20.6.5/20-42
0xD_0080	Physical retry error threshold configuration register (PRETCR)	R/W	0x0000_00FF	20.6.5/20-42
0xD_0100	Alternate device ID command and status register (ADIDCSR)	R/W	0x0000_0000	20.6.5/20-42
0xD_0120	Accept-all configuration register (AACR)	R/W	0x*n*000_000*n*	20.6.5/20-42
0xD_0124	Logical Outbound Packet time-to-live configuration register (LOPTTLCR)	R/W	0x0000_0000	20.6.5/20-42
0xD_0130	Implementation error command and status register (IECSR)	w1c	0x0000_0000	20.6.5/20-42
0xD_0140	Physical configuration register (PCR)	R/W	0x0000_8010	20.6.5/20-42
0xD_0158	Serial link command and status register (SLCSR)	w1c	0x0000_0000	20.6.5/20-42
0xD_0160	Serial link error injection configuration register (SLEICR)	R/W	0x0000_0000	20.6.5/20-42
Revision Control Register				
0xD_0BF8	IP Block Revision Register 1 (IPBRR1)	R	0x01C0_0000	20.6.6/20-49
0xD_0BFC	IP Block Revision Register 2 (IPBRR2)	R	0x0000_0000	20.6.6/20-49
ATMU				
0xD_0C00	RapidIO outbound window translation address register 0 (ROWTAR0)	R/W	0xFF80_0000	20.8.5/20-95
0xD_0C04	RapidIO outbound window translation extended address register 0 (ROWTEAR0)	R/W	0x0000_003F	20.8.5/20-95
0xD_0C10	RapidIO outbound window attributes register 0 (ROWAR0)	R/W	0x8004_4023	20.8.5/20-95
0xD_0C20	RapidIO outbound window translation address register 1 (ROWTAR1)	R/W	0x0000_0000	20.8.5/20-95
0xD_0C24	RapidIO outbound window translation extended address register 1 (ROWTEAR1)	R/W	0x0000_0000	20.8.5/20-95
0xD_0C28	RapidIO outbound window base address register 1 (ROWBAR1)	R/W	0x0000_0000	20.8.5/20-95
0xD_0C30	RapidIO outbound window attributes register 1 (ROWAR1)	R/W	0x0004_4023	20.8.5/20-95
0xD_0C34	RapidIO outbound window segment 1 register 1 (ROWS1R1)	R/W	0x0044_0000	20.8.5/20-95
0xD_0C38	RapidIO outbound window segment 2 register 1 (ROWS2R1)	R/W	0x0044_0000	20.8.5/20-95
0xD_0C3C	RapidIO outbound window segment 3 register 1 (ROWS3R1)	R/W	0x0044_0000	20.8.5/20-95
0xD_0C40 - 0xD_0CFC	RapidIO outbound window 2 through outbound window 7	—	—	—
0xD_0D00	RapidIO outbound window translation address register 8 (ROWTAR8)	R/W	0x0000_0000	20.8.5/20-95
0xD_0D04	RapidIO outbound window translation extended address register 8 (ROWTEAR8)	R/W	0x0000_0000	20.8.5/20-95

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0D08	RapidIO outbound window base address register 8 (ROWBAR8)	R/W	0x0000_0000	20.8.5/20-95
0xD_0D10	RapidIO outbound window attributes register 8 (ROWAR8)	R/W	0x0004_4023	20.8.5/20-95
0xD_0D14	RapidIO outbound window segment 1 register 8 (ROWS1R8)	R/W	0x0044_0000	20.8.5/20-95
0xD_0D18	RapidIO outbound window segment 2 register 8 (ROWS2R8)	R/W	0x0044_0000	20.8.5/20-95
0xD_0D1C	RapidIO outbound window segment 3 register 8 (ROWS3R8)	R/W	0x0044_0000	20.8.5/20-95
0xD_0D60	RapidIO Inbound window translation address register 4 (RIWTAR4)	R/W	0x0000_0000	20.8.6/20-97
0xD_0D68	RapidIO Inbound window base address register 4 (RIWBAR4)	R/W	0x0000_0000	20.8.6/20-97
0xD_0D70	RapidIO inbound window attributes register 4 (RIWAR4)	R/W	0x0004_4021	20.8.6/20-97
0xD_0D80–0xD_0DBC	RapidIO inbound window 3 through inbound window 2			
0xD_0DC0	RapidIO inbound window translation address register 1 (RIWTAR1)	R/W	0x0000_0000	20.8.6/20-97
0xD_0DC8	RapidIO inbound window base address register 1 (RIWBAR1)	R/W	0x0000_0000	20.8.6/20-97
0xD_0DD0	RapidIO inbound window attributes register 1 (RIWAR1)	R/W	0x0004_4021	20.8.6/20-97
0xD_0DE0	RapidIO inbound window translation address register 0 (RIWTAR0)	R/W	0x0000_0000	20.8.6/20-97
0xD_0DF0	RapidIO inbound window attributes register 0 (RIWAR0)	Mixed	0x8004_4021	20.8.6/20-97
Message Unit				
RapidIO Outbound Message Unit 0 Registers				
0xD_3000	Outbound message 0 mode register (OM0MR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3004	Outbound message 0 status register (OM0SR)	Mixed	0x0000_0000	20.7.1/20-61
0xD_3008	Extended outbound message 0 descriptor queue dequeue pointer address register (EOM0DQDPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_300C	Outbound message 0 descriptor queue dequeue pointer address register (OM0DQDPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3010	Extended outbound message 0 source address register (EOM0SAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3014	Outbound message 0 source address register (OM0SAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3018	Outbound message 0 destination port register (OM0DPR)	R/W	0x0000_0000	20.7.1/20-61
0xD_301C	Outbound message 0 destination attributes Register (OM0DATR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3020	Outbound message 0 double-word count register (OM0DCR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3024	Extended outbound message 0 descriptor queue enqueue pointer address register (EOM0DQEPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3028	Outbound message 0 descriptor queue enqueue pointer address register (OM0DQEPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_302C	Outbound message 0 retry error threshold configuration register (OM0RETCR)	R/W	0x0000_0000	20.7.1/20-61

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_3030	Outbound message 0 multicast group register (OM0MGR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3034	Outbound message 0 multicast list register (OM0MLR)	R/W	0x0000_0000	20.7.1/20-61
RapidIO Inbound Message Unit 0 Registers				
0xD_3060	Inbound message 0 mode register (IM0MR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3064	Inbound message 0 status register (IM0SR)	Mixed	0x0000_0002	20.7.2/20-72
0xD_3068	Extended inbound message 0 frame queue dequeue pointer address register (EIM0FQDPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_306C	Inbound message 0 frame queue dequeue pointer address register (IM0FQDPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3070	Extended inbound message 0 frame queue enqueue pointer address register (EIM0FQEPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3074	Inbound message 0 frame queue enqueue pointer address register (IM0FQEPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3078	Inbound message 0 maximum interrupt report interval register (IM0MIRIR)	R/W	0xFFFF_FF00	20.7.2/20-72
RapidIO Outbound Message Unit 1 Registers				
0xD_3100	Outbound message 1 mode register (OM1MR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3104	Outbound message 1 status register (OM1SR)	Mixed	0x0000_0000	20.7.1/20-61
0xD_3108	Extended outbound message 1 descriptor queue dequeue pointer address register (EOM1DQDPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_310C	Outbound message 1 descriptor queue dequeue pointer address register (OM1DQDPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3110	Extended outbound message 1 source address register (EOM1SAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3114	Outbound message 1 source address register (OM1SAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3118	Outbound message 1 destination port register (OM1DPR)	R/W	0x0000_0000	20.7.1/20-61
0xD_311C	Outbound message 1 destination attributes register (OM1DATR)	R/W	0x0006_0000	20.7.1/20-61
0xD_3120	Outbound message 1 double-word count register (OM1DCR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3124	Extended outbound message 1 descriptor queue enqueue pointer address register (EOM1DQEPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3128	Outbound message 1 descriptor queue enqueue pointer address register (OM1DQEPAR)	R/W	0x0000_0000	20.7.1/20-61
0xD_312C	Outbound message 1 retry error threshold configuration register (OM1RETCR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3130	Outbound message 1 multicast group register (OM1MGR)	R/W	0x0000_0000	20.7.1/20-61
0xD_3134	Outbound message 1 multicast list register (OM1MLR)	R/W	0x0000_0000	20.7.1/20-61
RapidIO Inbound Message Unit 1 Registers				
0xD_3160	Inbound message 1 mode register (IM1MR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3164	Inbound message 1 status register (IM1SR)	Mixed	0x0000_0002	20.7.2/20-72

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_3168	Extended inbound message 1 frame queue dequeue pointer address register (EIM1FQDPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_316C	Inbound message 1 frame queue dequeue pointer address register (IM1FQDPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3170	Extended inbound message 1 frame queue enqueue pointer address register (EIM1FQEPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3174	Inbound message 1 frame queue enqueue pointer address register (IM1FQEPAR)	R/W	0x0000_0000	20.7.2/20-72
0xD_3178	Inbound message 1 maximum interrupt report interval register (IM1MIRIR)	R/W	0xFFFF_FF00	20.7.2/20-72
RapidIO Doorbell Registers				
0xD_3400	Outbound doorbell mode register (ODMR)	R/W	0x0000_0000	20.7.3/20-78
0xD_3404	Outbound doorbell status register (ODSR)	Mixed	0x0000_0000	20.7.3/20-78
0xD_3418	Outbound doorbell destination port register (ODDPR)	R/W	0x0000_0000	20.7.3/20-78
0xD_341C	Outbound doorbell destination attributes register (ODDATR)	R/W	0x0000_0000	20.7.3/20-78
0xD_342C	Outbound doorbell retry error threshold configuration register (ODRETCR)	R/W	0x0000_0000	20.7.3/20-78
0xD_3460	Inbound doorbell mode register (IDMR)	R/W	0x0000_0000	20.7.4/20-81
0xD_3464	Inbound doorbell status register (IDSR)	Mixed	0x0000_0002	20.7.4/20-81
0xD_3468	Extended inbound doorbell queue dequeue pointer address register (EIDQDPAR)	R/W	0x0000_0000	20.7.4/20-81
0xD_346C	Inbound doorbell queue dequeue Pointer address register (IDQDPAR)	R/W	0x0000_0000	20.7.4/20-81
0xD_3470	Extended inbound doorbell queue enqueue pointer address register (EIDQEPAR)	R/W	0x0000_0000	20.7.4/20-81
0xD_3474	Inbound doorbell Queue enqueue pointer address register (IDQEPAR)	R/W	0x0000_0000	20.7.4/20-81
0xD_3478	Inbound doorbell maximum interrupt report interval register (IDMIRIR)	R/W	0xFFFF_FF00	20.7.4/20-81
RapidIO Port-Write Registers				
0xD_34E0	Inbound port-write mode register (IPWMR)	R/W	0x0000_0000	20.7.5/20-87
0xD_34E4	Inbound port-write status register (IPWSR)	Mixed	0x0000_0000	20.7.5/20-87
0xD_34E8	Extended inbound port-write queue base address register (EIPWQBAR)	R/W	0x0000_0000	20.7.5/20-87
0xD_34EC	Inbound port-write queue base address register (IPWQBAR)	R/W	0x0000_0000	20.7.5/20-87
Global Utilities Registers				
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x*nnnn_nnnn*	23.4.1/23-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	23.4.1/23-4

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	Mixed	0x0000_007F	23.4.1/23-4
0xE_000C	PORDEVSR—POR device status register	R	see ref.	23.4.1/23-4
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	23.4.1/23-4
0xE_0014	PORDEVSR2—POR device status register 2	R	0x0000_*n*00*n*	23.4.1/23-4
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	23.4.1/23-4
Device Disables				
0xE_0070	DEVDISR—Device disable control register	R/W	0xnn0*n*_00*nn*	23.4.1/23-4
Power Management Registers				
0xE_0080	POWMGTCSR—Power management control and status register	Mixed	0x0000_0000	23.4.1/23-4
Interrupt and Reset Status and Control				
0xE_0090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	23.4.1/23-4
0xE_0094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	23.4.1/23-4
0xE_0098	ECTRSTCR—Exception reset control register	Mixed	0x0000_0000	23.4.1/23-4
0xE_009C	AUTORSTSR—Automatic reset status register	Mixed	0x0000_0000	23.4.1/23-4
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	23.4.1/23-4
0xE_00A4	SVR—System version register	R	MPC8572E system version	23.4.1/23-4
Status Registers				
0xE_00B0	RSTCR—Reset control register	R/W	0x0000_0000	23.4.1/23-4
0xE_00C0	LBCVSELCR—LBC voltage select control register	R/W	0x0000_0000	23.4.1/23-4
0xE_0B28	DDRCLKDR—DDR clock disable register	R/W	0x0000_0000	23.4.1/23-4
Debug and Control Registers				
0xE_0E00	CLKOCR—Clock out control register	R/W	0x0000_0000	23.4.1/23-4
0xE_0E20	TLUTRGCR—TLU target control register	R/W	0x0000_0000	23.4.1/23-4
0xE_0F04	SRDS1CR—SerDes1 control register	R/W	0x1100_4418	23.4.1/23-4
0xE_0F10	SRDS2CR—SerDes2 control register	R/W	0x1100_4418	23.4.1/23-4
Performance Monitor Control Registers				
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	24.3.2/24-6
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	24.3.2/24-6
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	24.3.2/24-6
0xE_1018	PMC0 (lower)—Performance monitor counter 0 upper	R/W	0x0000_0000	24.3.3/24-10
0xE_101C	PMC0 (upper)—Performance monitor counter 0 lower	R/W	0x0000_0000	24.3.3/24-10

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	24.3.2/24-6
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	24.3.2/24-6
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	24.3.3/24-10
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	24.3.2/24-6
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	24.3.2/24-6
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	24.3.3/24-10
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	24.3.2/24-6
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	24.3.2/24-6
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	24.3.3/24-10
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	24.3.2/24-6
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	24.3.2/24-6
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	24.3.3/24-10
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	24.3.2/24-6
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	24.3.2/24-6
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	24.3.3/24-10
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	24.3.2/24-6
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	24.3.2/24-6
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	24.3.3/24-10
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	24.3.2/24-6
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	24.3.2/24-6
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	24.3.3/24-10
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	24.3.2/24-6
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	24.3.2/24-6
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	24.3.3/24-10
0xE_10A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	24.3.2/24-6
0xE_10A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	24.3.2/24-6
0xE_10A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	24.3.3/24-10
0xE_10B0	PMLCA10—Performance monitor local control register A10	R/W	0x0000_0000	24.3.2/24-6
0xE_10B4	PMLCB10—Performance monitor local control register B10	R/W	0x0000_0000	24.3.2/24-6
0xE_10B8	PMC10—Performance monitor counter 10	R/W	0x0000_0000	24.3.3/24-10
0xE_10C0	PMLCA11—Performance monitor local control register A11	R/W	0x0000_0000	24.3.2/24-6
0xE_10C4	PMLCB11—Performance monitor local control register B11	R/W	0x0000_0000	24.3.2/24-6
0xE_10C8	PMC11—Performance monitor counter 11	R/W	0x0000_0000	24.3.3/24-10
Debug and Watchpoint Monitor Registers				
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	25.3.1/25-11

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	25.3.1/25-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	25.3.1/25-11
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	25.3.1/25-11
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	25.3.1/25-11
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	25.3.1/25-11
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	25.3.2/25-16
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	25.3.2/25-16
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	25.3.2/25-16
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	25.3.2/25-16
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	25.3.2/25-16
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	25.3.2/25-16
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	25.3.2/25-16
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	25.3.2/25-16
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	25.3.2/25-16
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	25.3.3/25-23
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	25.3.3/25-23

Table 2-11. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	25.3.4/25-24

1 Implementation-dependent reset values are listed in specified section/page.

2 I^2C2 has the same memory-mapped registers that are described for I^2C1 from 0x 0x0_3000 to 0x0_3014, except the offsets are from 0x0_3100 to 0x0_3114.

3 Port size for BR0 is configured from external signals during reset, hence '*nn*' is either 0x08, 0x10, or 0x18. See Section 4.4.3.4, "Boot ROM Location," for more information.

4 FMR[BOOT] is set during reset if BR0[MSEL] selects FCM as the boot controller.

5 DDR SDRAM controller 2 has the same memory-mapped registers that are described for DDR SDRAM controller 1 from 0x0_2000 to 0x0_2FFC, except the offsets are from 0x0_6000 to 0x0_6FFC.

6 PCI Express controller 3 has the same memory-mapped registers that are described for PCI Express controller 1 from 0x0_A000 to 0x0_AFFC, except the offsets are from 0x0_8000 to 0x0_8FFC.

7 PCI Express controller 2 has the same memory-mapped registers that are described for PCI Express controller 1 from 0x0_A000 to 0x0_AFFC, except the offsets are from 0x0_9000 to 0x0_9FFC.

8 DMA controller 2 has the same memory-mapped registers that are described for DMA controller 1 from 0x2_1000 to 0x2_1FFFC except the offsets are from 0x0_C000 to 0x0_CFFC.

9 TLU 2 has the same memory-mapped registers that are described for TLU 1 from 0x2_F000 to 0x2_FFFFC except the offsets are from 0x1_5000 to 0x1_5FFC.

10 Cleared on read

11 eTSEC2 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFC, except the offsets are from 0x 2_5000 to 0x2_5FFC.

12 eTSEC3 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFC, except the offsets are from 0x 2_6000 to 0x2_6FFC.

13 eTSEC4 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFC, except the offsets are from 0x 2_7000 to 0x2_7FFC.

14 Serial RapidIO registers: values indicated with *n* are read from configuration signals at reset.

Chapter 3
Signal Descriptions

This chapter describes the MPC8572E external signals. It is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions, including two lists: one by functional block and one alphabetical
- List of reset configuration signals
- List of output signal states at reset

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{\text{IRQ\_OUT}}$ (interrupt out). 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.

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.

3.1 Signals Overview

The MPC8572E signals are grouped as follows:

- I^2C interface signals
- System control, general-purpose output, power management, and debug signals
- Test, JTAG, configuration, and clock signals
- Hi-speed serial interface 1 signals (Serial RapidIO/PCI Express)
- Hi-speed serial interface 2 signals (SGMII Ethernet)
- eTSEC1–4 interface signals
- FEC interface signals
- DDR controllers 1 & 2 memory interface signals
- Enhanced local bus interface signals
- DMA controllers 1 & 2 interface signals
- PIC interface signals
- DUART interface signals
- Configuration signals

Figure 3-1, Figure 3-2, and Figure 3-3 illustrate the external signals of the MPC8572E, showing how the signals are grouped. Refer to the *MPC8572E Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications. Note that these figures show multiplexed signals multiple times, once for each multiplexed function.

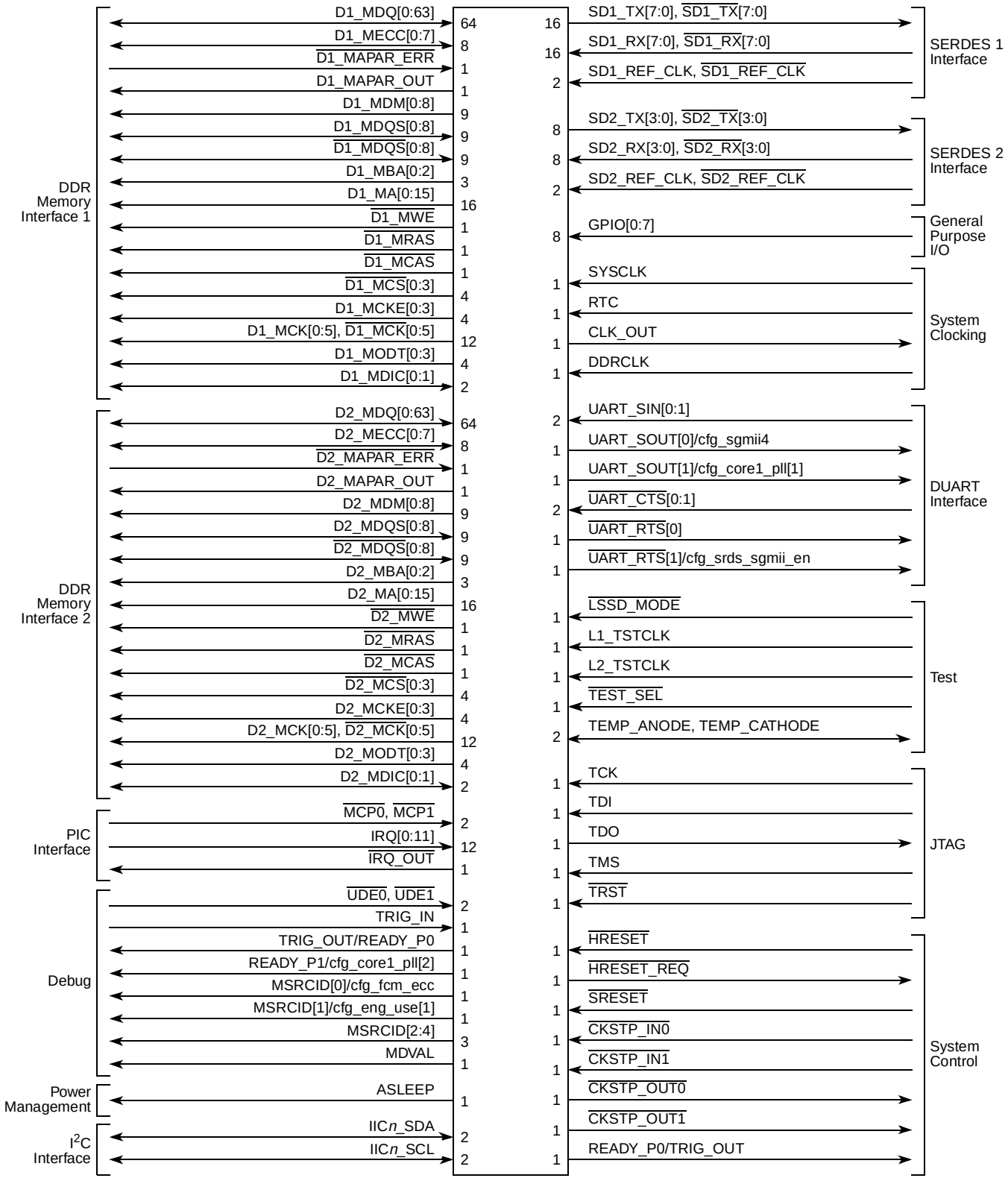


Figure 3-1. MPC8572E Signal Groupings (1/3)

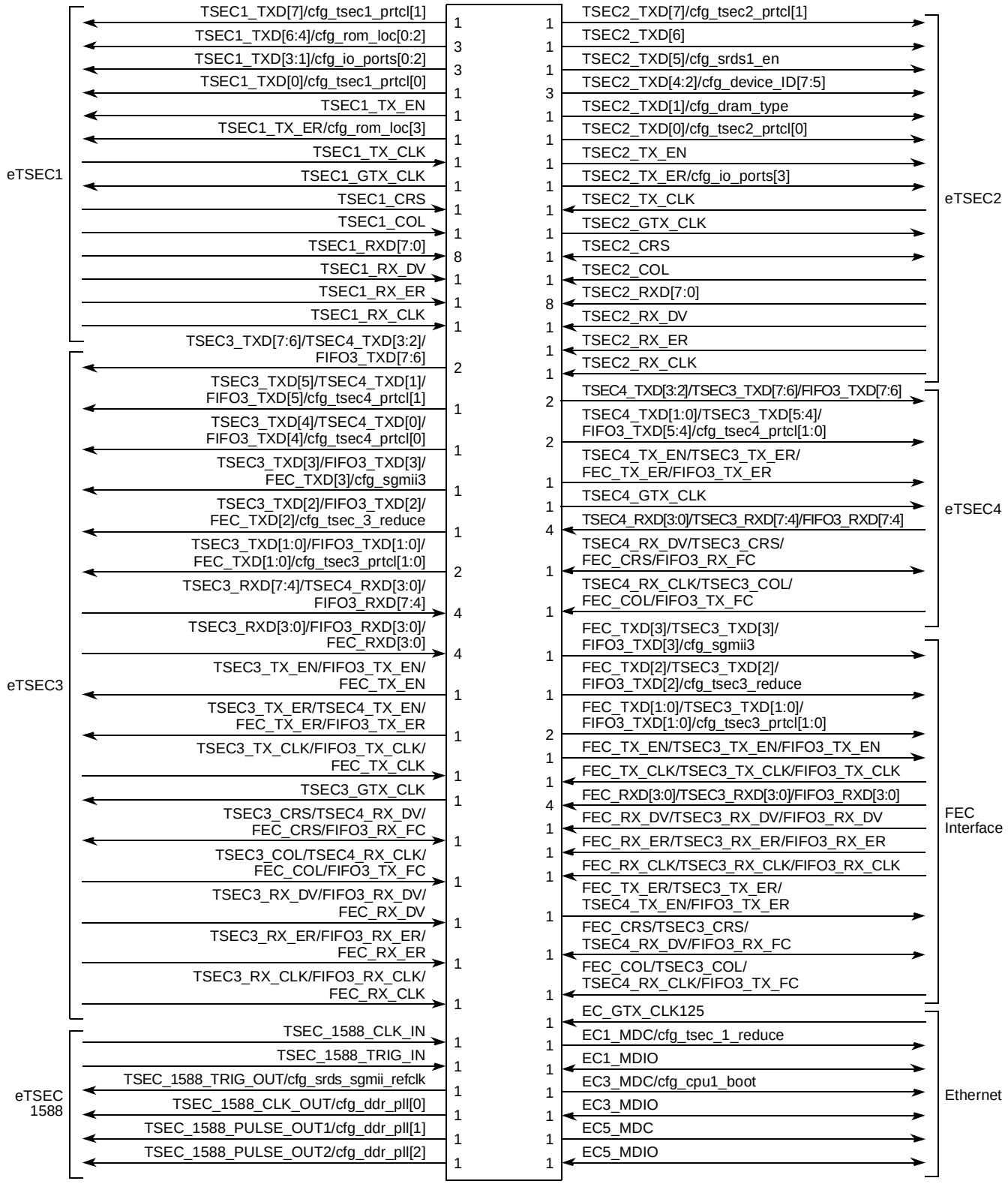


Figure 3-2. MPC8572E Signal Groupings (2/3) (Continued)

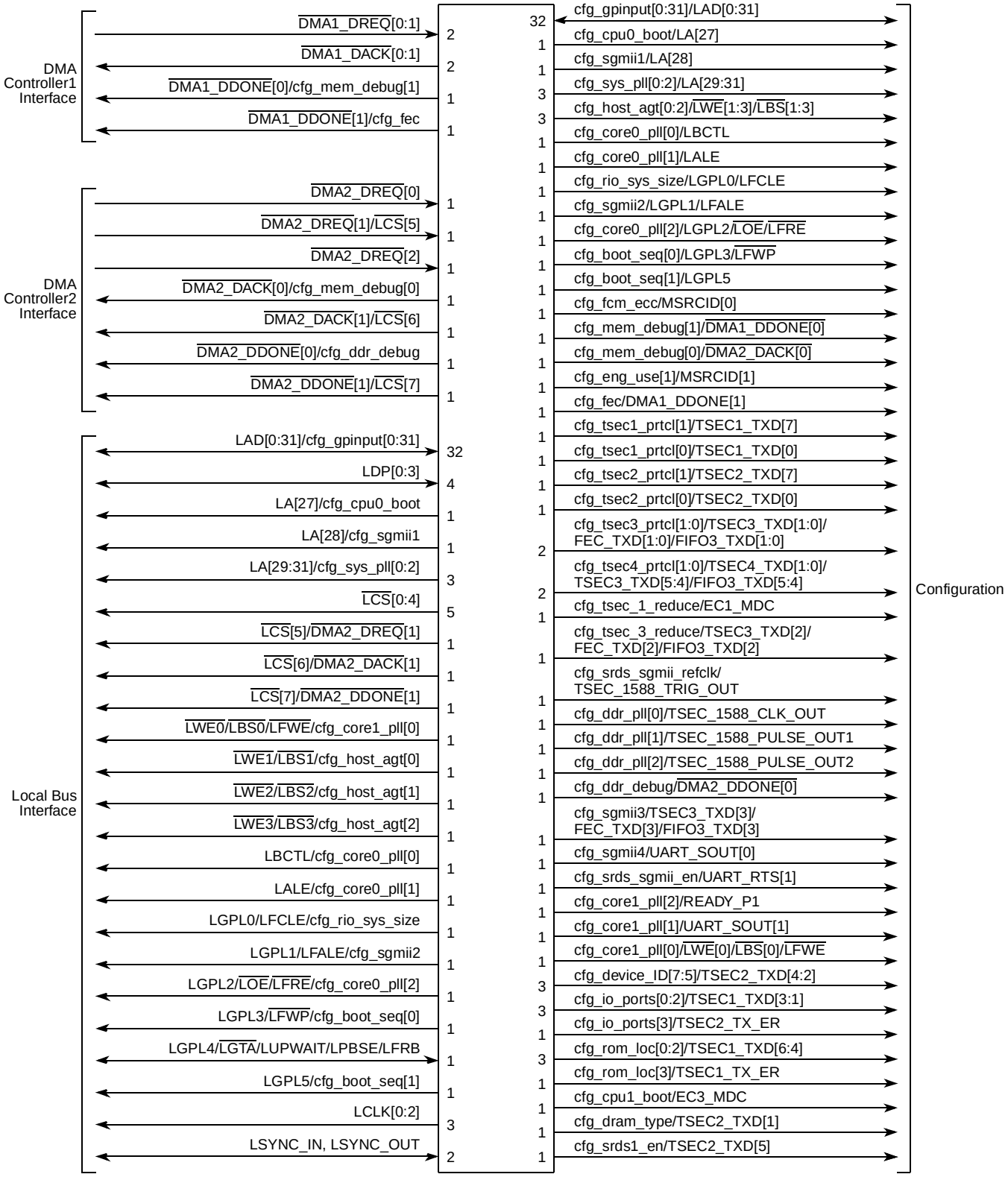


Figure 3-3. MPC8572E Signal Groupings (3/3) (Continued)

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 provide summaries of signal functions. Table 3-1 provides a summary of the signals grouped by function, and Table 3-2 provides the summary list of the signals grouped alphabetically. These tables detail 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 table provides a pointer to the table where the signal function is described.

Table 3-1. MPC8572E Signal Reference by Functional Block

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
D1_MDQ[0:63]	DDR controller data	DDR Controller 1	—	64	I/O	9-1/9-4
D1_MECC[0:7]	DDR error correcting code	DDR Controller 1	—	8	I/O	9-1/9-4
D1_MDM[0:8]	DDR data mask	DDR Controller 1	—	9	I/O	9-1/9-4
D1_MDQS[0:8]	DDR data strobe	DDR Controller 1	—	9	I/O	9-1/9-4
$\overline{\text{D1\_MDQS}}$[0:8]	DDR data strobe (complement)	DDR Controller 1	—	9	I/O	9-1/9-4
D1_MBA[2:0]	DDR bank select	DDR Controller 1	—	3	O	9-1/9-4
D1_MA[15:0]	DDR address	DDR Controller 1	—	16	O	9-1/9-4
$\overline{\text{D1\_MWE}}$	DDR write enable	DDR Controller 1	—	1	O	9-1/9-4
$\overline{\text{D1\_MRAS}}$	DDR row address strobe	DDR Controller 1	—	1	O	9-1/9-4
$\overline{\text{D1\_MCAS}}$	DDR column address strobe	DDR Controller 1	—	1	O	9-1/9-4
$\overline{\text{D1\_MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR Controller 1	—	4	O	9-1/9-4
D1_MCKE[0:3]	DDR clock enable	DDR Controller 1	—	4	O	9-1/9-4
D1_MCK[0:5], $\overline{\text{D1\_MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR Controller 1	—	12	O	9-1/9-4
D1_MODT[0:3]	DRAM On-Die Termination	DDR Controller 1	—	4	O	—
D1_MDIC[0:1]	Driver impedance calibration	DDR Controller 1	—	2	I/O	9-1/9-4
$\overline{\text{D1\_MAPAR\_ERR}}$	Address parity error	DDR Controller 1	—	1	I	9-1/9-4
D1_MAPAR_OUT	Address parity out	DDR Controller 1	—	1	O	9-1/9-4
D2_MDQ[0:63]	DDR controller data	DDR Controller 2	—	64	I/O	9-1/9-4
D2_MECC[0:7]	DDR error correcting code	DDR Controller 2	—	8	I/O	9-1/9-4
D2_MDM[0:8]	DDR data mask	DDR Controller 2	—	9	I/O	9-1/9-4
D2_MDQS[0:8]	DDR data strobe	DDR Controller 2	—	9	I/O	9-1/9-4
$\overline{\text{D2\_MDQS}}$[0:8]	DDR data strobe (complement)	DDR Controller 2	—	9	I/O	9-1/9-4
D2_MBA[2:0]	DDR bank select	DDR Controller 2	—	3	O	9-1/9-4
D2_MA[15:0]	DDR address	DDR Controller 2	—	16	O	9-1/9-4

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{D2\_MWE}}$	DDR write enable	DDR Controller 2	—	1	O	9-1/9-4
$\overline{\text{D2\_MRAS}}$	DDR row address strobe	DDR Controller 2	—	1	O	9-1/9-4
$\overline{\text{D2\_MCAS}}$	DDR column address strobe	DDR Controller 2	—	1	O	9-1/9-4
$\overline{\text{D2\_MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR Controller 2	—	4	O	9-1/9-4
D2_MCKE[0:3]	DDR clock enable	DDR Controller 2	—	4	O	9-1/9-4
D2_MCK[0:5], $\overline{\text{D2\_MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR Controller 2	—	12	O	9-1/9-4
D2_MODT[0:3]	DRAM On-Die Termination	DDR Controller 2	—	4	O	—
D2_MDIC[0:1]	Driver impedance calibration	DDR Controller 2	—	2	I/O	9-1/9-4
$\overline{\text{D2\_MAPAR\_ERR}}$	Address parity error	DDR Controller 2	—	1	I	9-1/9-4
D2_MAPAR_OUT	Address parity out	DDR Controller 2	—	1	O	9-1/9-4
SD1_RX[7:0]	Receive data	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	I	21-2/21-5
$\overline{\text{SD1\_RX}}$[7:0]	Receive data (complement)	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	I	21-2/21-5
SD1_TX[7:0]	Transmit data	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:0]	Transmit data (complement)	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	O	21-2/21-5
SD1_RX[7:4]	Receive data	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	I	21-2/21-5
$\overline{\text{SD1\_RX}}$[7:4]	Receive data (complement)	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	I	21-2/21-5
SD1_TX[7:4]	Transmit data	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:4]	Transmit data (complement)	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	O	21-2/21-5
SD1_RX[7:6]	Receive data	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	I	21-2/21-5
$\overline{\text{SD1\_RX}}$[7:6]	Receive data (complement)	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	I	21-2/21-5

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
SD1_TX[7:6]	Transmit data	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:6]	Transmit data (complement)	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	O	21-2/21-5
SD1_TX[7:4]	Transmit data	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	O	20.4.2/20-4
$\overline{\text{SD1\_TX}}$[7:4]	Transmit data (complement)	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	O	20.4.2/20-4
SD1_RX[7:4]	Receive data	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	I	20.4.2/20-4
$\overline{\text{SD1\_RX}}$[7:4]	Receive data (complement)	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	I	20.4.2/20-4
SD1_REF_CLK	PLL reference clock	SerDes 1	—	1	I	—
$\overline{\text{SD1\_REF\_CLK}}$	PLL reference clock (complement)	SerDes 1	—	1	I	—
SD2_REF_CLK	PLL reference clock	SerDes 2	—	1	I	—
$\overline{\text{SD2\_REF\_CLK}}$	PLL reference clock (complement)	SerDes 2	—	1	I	—
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	15-2/15-10
EC1_MDC	Ethernet management data clock	Ethernet management Interface 1 (eTSEC1)	cfg_tsec_1_reduce	1	O	15-2/15-10
EC1_MDIO	Ethernet management data in/out	Ethernet management Interface 1 (eTSEC1)	—	1	I/O	15-2/15-10
EC3_MDC	Ethernet management data clock	Ethernet management Interface 3 (eTSEC3)	cfg_cpu1_boot	1	O	15-2/15-10
EC3_MDIO	Ethernet management data in/out	Ethernet management Interface 3 (eTSEC3)	—	1	I/O	15-2/15-10
EC5_MDC	Ethernet management data clock	Ethernet management Interface 5 (FEC)	—	1	O	15-2/15-10

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
EC5_MDIO	Ethernet management data in/out	Ethernet management Interface 5 (FEC)	—	1	I/O	15-2/15-10
TSEC_1588_ CLK_IN	eTSEC 1588 clock in	eTSEC	—	1	I	15-2/15-10
TSEC_1588_ TRIG_IN	eTSEC 1588 trigger in	eTSEC	—	1	I	15-2/15-10
TSEC_1588_ TRIG_OUT	eTSEC 1588 trigger out	eTSEC	cfg_srds_sgmii_refclk	1	O	15-2/15-10
TSEC_1588_ CLK_OUT	eTSEC 1588 clock out	eTSEC	cfg_ddr_pll[0]	1	O	15-2/15-10
TSEC_1588_ PULSE_OUT1	eTSEC 1588 pulse out 1	eTSEC	cfg_ddr_pll[1]	1	O	15-2/15-10
TSEC_1588_ PULSE_OUT2	eTSEC 1588 pulse out 2	eTSEC	cfg_ddr_pll[2]	1	O	15-2/15-10
TSEC1_TXD[7]	TSEC1 transmit data 7	eTSEC1	cfg_tsec1_prtcl[1]	1	O	15-2/15-10
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	eTSEC1	cfg_rom_loc[0:2]	3	O	15-2/15-10
TSEC1_TXD[3:1]	TSEC1 transmit data 3–1	eTSEC1	cfg_io_ports[0:2]	3	O	15-2/15-10
TSEC1_TXD[0]	TSEC1 transmit data 0	eTSEC1	cfg_tsec1_prtcl[0]	1	O	15-2/15-10
TSEC1_TX_EN	TSEC1 transmit enable	eTSEC1	—	1	O	15-2/15-10
TSEC1_TX_ER	TSEC1 transmit error	eTSEC1	cfg_rom_loc[3]	1	O	15-2/15-10
TSEC1_TX_CLK	TSEC1 transmit clock in	eTSEC1	—	1	I	15-2/15-10
TSEC1_GTX_CLK	TSEC1 transmit clock out	eTSEC1	—	1	O	15-2/15-10
TSEC1_CRS	TSEC1 carrier sense	eTSEC1	—	1	I	15-2/15-10
TSEC1_COL	TSEC1 collision detect	eTSEC1	—	1	I	15-2/15-10
TSEC1_RXD[7:0]	TSEC1 receive data	eTSEC1	—	8	I	15-2/15-10
TSEC1_RX_DV	TSEC1 receive data valid	eTSEC1	—	1	I	15-2/15-10
TSEC1_RX_ER	TSEC1 receiver error	eTSEC1	—	1	I	15-2/15-10
TSEC1_RX_CLK	TSEC1 receive clock	eTSEC1	—	1	I	15-2/15-10
SD2_TX[0], $\overline{\text{SD2\_TX[0]}}$	SGMII transmit data (and complement)	eTSEC1	—	2	O	15-2/15-10
SD2_RX[0], $\overline{\text{SD2\_RX[0]}}$	SGMII receive data (and complement)	eTSEC1	—	2	I	15-2/15-10
TSEC2_TXD[7]	TSEC2 transmit data 7	eTSEC2	cfg_tsec2_prtcl[1]	1	O	15-2/15-10
TSEC2_TXD[6]	TSEC2 transmit data 6	eTSEC2	—	1	O	15-2/15-10
TSEC2_TXD[5]	TSEC2 transmit data 5	eTSEC2	cfg_srds1_en	1	O	15-2/15-10
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	eTSEC2	cfg_device_ID[7:5]	3	O	15-2/15-10
TSEC2_TXD[1]	TSEC2 transmit data 1	eTSEC2	cfg_dram_type	1	O	15-2/15-10

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC2_TXD[0]	TSEC2 transmit data 0	eTSEC2	cfg_tsec2_prtcl[0]	1	O	15-2/15-10
TSEC2_TX_EN	TSEC2 transmit enable	eTSEC2	—	1	O	15-2/15-10
TSEC2_TX_ER	TSEC2 transmit error	eTSEC2	cfg_io_ports[3]	1	O	15-2/15-10
TSEC2_TX_CLK	TSEC2 transmit clock in	eTSEC2	—	1	I	15-2/15-10
TSEC2_GTX_CLK	TSEC2 transmit clock out	eTSEC2	—	1	O	15-2/15-10
TSEC2_CRS	TSEC2 carrier sense	eTSEC2	—	1	I	15-2/15-10
TSEC2_COL	TSEC2 collision detect	eTSEC2	—	1	I	15-2/15-10
TSEC2_RXD[7:0]	TSEC2 receive data	eTSEC2	—	8	I	15-2/15-10
TSEC2_RX_DV	TSEC2 receive data valid	eTSEC2	—	1	I	15-2/15-10
TSEC2_RX_ER	TSEC2 receive error	eTSEC2	—	1	I	15-2/15-10
TSEC2_RX_CLK	TSEC2 receive clock	eTSEC2	—	1	I	15-2/15-10
SD2_TX[1], $\overline{\text{SD2\_TX[1]}}$	SGMII transmit data (and complement)	eTSEC2	—	2	O	15-2/15-10
SD2_RX[1], $\overline{\text{SD2\_RX[1]}}$	SGMII receive data (and complement)	eTSEC2	—	2	I	15-2/15-10
TSEC3_TXD[7]	TSEC3 transmit data 7	eTSEC3	TSEC4_TXD[3]	1	O	15-2/15-10
TSEC3_TXD[6]	TSEC3 transmit data 6	eTSEC3	TSEC4_TXD[2]	1	O	15-2/15-10
TSEC3_TXD[5]	TSEC3 transmit data 5	eTSEC3	TSEC4_TXD[1], cfg_tsec4_prtcl[1]	1	O	15-2/15-10
TSEC3_TXD[4]	TSEC3 transmit data 4	eTSEC3	TSEC4_TXD[0], cfg_tsec4_prtcl[0]	1	O	15-2/15-10
TSEC3_TXD[3]	TSEC3 transmit data 3	eTSEC3	FEC_TXD[3], cfg_sgmii3	1	O	15-2/15-10
TSEC3_TXD[2]	TSEC3 transmit data 2	eTSEC3	FEC_TXD[2], cfg_tsec_3_reduce	1	O	15-2/15-10
TSEC3_TXD[1:0]	TSEC3 transmit data 1–0	eTSEC3	FEC_TXD[1:0], cfg_tsec3_prtcl[1:0]	2	O	15-2/15-10
TSEC3_TX_ER	TSEC3 transmit error	eTSEC3	TSEC4_TX_EN, FEC_TX_ER	1	O	15-2/15-10
TSEC3_TX_EN	TSEC3 transmit enable	eTSEC3	FEC_TX_EN	1	O	15-2/15-10
TSEC3_TX_CLK	TSEC3 transmit clock in	eTSEC3	FEC_TX_CLK	1	I	15-2/15-10
TSEC3_GTX_CLK	TSEC3 transmit clock out	eTSEC3	—	1	O	15-2/15-10
TSEC3_RXD[3:0]	TSEC3 receive data 3–0	eTSEC3	FEC_RXD[3:0]	4	I	15-2/15-10
TSEC3_RX_DV	TSEC3 receive data valid	eTSEC3	FEC_RX_DV	1	I	15-2/15-10
TSEC3_RX_ER	TSEC3 receive error	eTSEC3	FEC_RX_ER	1	I	15-2/15-10
TSEC3_RX_CLK	TSEC3 receive clock	eTSEC3	FEC_RX_CLK	1	I	15-2/15-10
TSEC3_CRS	TSEC3 carrier sense	eTSEC3	TSEC4_RX_DV, FEC_CRS	1	I	15-2/15-10

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC3_COL	TSEC3 collision detect	eTSEC3	TSEC4_RX_CLK, FEC_COL	1	I	15-2/15-10
SD2_TX[2], $\overline{\text{SD2\_TX[2]}}$	SGMII transmit data (and complement)	eTSEC3	—	2	O	15-2/15-10
SD2_RX[2], $\overline{\text{SD2\_RX[2]}}$	SGMII receive data (and complement)	eTSEC3	—	2	I	15-2/15-10
TSEC4_TXD[3]	TSEC4 transmit data 3	eTSEC4	TSEC3_TXD[7]	1	O	15-2/15-10
TSEC4_TXD[2]	TSEC4 transmit data 2	eTSEC4	TSEC3_TXD[6]	1	O	15-2/15-10
TSEC4_TXD[1:0]	TSEC4 transmit data 1–0	eTSEC4	TSEC3_TXD[5:4], cfg_tsec4_prtcl[1:0]	2	O	15-2/15-10
TSEC4_TX_EN	TSEC4 transmit enable	eTSEC4	TSEC3_TX_ER, FEC_TX_ER	1	O	15-2/15-10
TSEC4_GTX_CLK	TSEC4 transmit clock out	eTSEC4	—	1	O	15-2/15-10
TSEC4_RXD[3:0]	TSEC4 receive data 3–0	eTSEC4	TSEC3_RXD[7:4]	4	I	15-2/15-10
TSEC4_RX_DV	TSEC4 receive data valid	eTSEC4	TSEC3_CRS, FEC_CRS	1	I	15-2/15-10
TSEC4_RX_CLK	TSEC4 receive clock	eTSEC4	TSEC3_COL, FEC_COL	1	I	15-2/15-10
SD2_TX[3], $\overline{\text{SD2\_TX[3]}}$	SGMII transmit data (and complement)	eTSEC4	—	2	O	15-2/15-10
SD2_RX[3], $\overline{\text{SD2\_RX[3]}}$	SGMII receive data (and complement)	eTSEC4	—	2	I	15-2/15-10
FEC_TXD[3]	FEC transmit data 3	FEC	TSEC3_TXD[3], cfg_sgmii3	1	O	16-1/16-7
FEC_TXD[2]	FEC transmit data 2	FEC	TSEC3_TXD[2], cfg_tsec_3_reduce	1	O	16-1/16-7
FEC_TXD[1:0]	FEC transmit data 1–0	FEC	TSEC3_TXD[1:0], cfg_tsec3_prtcl[1:0]	2	O	16-1/16-7
FEC_TX_EN	FEC transmit enable	FEC	TSEC3_TX_EN	1	O	16-1/16-7
FEC_TX_ER	FEC transmit error	FEC	TSEC4_TX_EN, TSEC3_TX_ER	1	O	16-1/16-7
FEC_TX_CLK	FEC transmit clock in	FEC	TSEC3_TX_CLK	1	I	16-1/16-7
FEC_RXD[3:0]	FEC receive data 3–0	FEC	TSEC3_RXD[3:0]	4	I	16-1/16-7
FEC_RX_DV	FEC receive data valid	FEC	TSEC3_RX_DV	1	I	16-1/16-7
FEC_RX_ER	FEC receive error	FEC	TSEC3_RX_ER	1	I	16-1/16-7
FEC_RX_CLK	FEC receive clock	FEC	TSEC3_RX_CLK	1	I	16-1/16-7
FEC_CRS	FEC carrier sense	FEC	TSEC4_RX_DV, TSEC3_CRS	1	I	16-1/16-7
FEC_COL	FEC collision detect	FEC	TSEC4_RX_CLK, TSEC3_COL	1	I	16-1/16-7

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
LAD[0:31]	Local bus address/data	eLBC	cfg_gpinput[0:31]	32	I/O	14-2/14-5
LDP[0:3]	Local bus data parity	eLBC	—	4	I/O	14-2/14-5
LA[27]	Local bus burst address	eLBC	cfg_cpu0_boot	1	O	14-2/14-5
LA[28]	Local bus port address	eLBC	cfg_sgmii1	1	O	14-2/14-5
LA[29:31]	Local bus port address	eLBC	cfg_sys_pll[0:2]	3	O	14-2/14-5
$\overline{\text{LCS}}$[0:4]	Local bus chip select 0–4	eLBC	—	5	O	14-2/14-5
$\overline{\text{LCS}}$[5]	Local bus chip select 5	eLBC	$\overline{\text{DMA2\_DREQ}}$[1]	1	O	14-2/14-5
$\overline{\text{LCS}}$[6]	Local bus chip select 6	eLBC	$\overline{\text{DMA2\_DACK}}$[1]	1	O	14-2/14-5
$\overline{\text{LCS}}$[7]	Local bus chip select 7	eLBC	$\overline{\text{DMA2\_DDONE}}$[1]	1	O	14-2/14-5
$\overline{\text{LWE0}}$/$\overline{\text{LBS0}}$/$\overline{\text{LFWE}}$	Local bus write enable/byte select 0/flash write enable	eLBC	cfg_core1_pll[0]	1	O	14-2/14-5
$\overline{\text{LWE}}$[1:3]/$\overline{\text{LBS}}$[1:3]	Local bus write enable/byte select 1–3	eLBC	cfg_host_agt[0:2]	3	O	14-2/14-5
LBCTL	Local bus data buffer control	eLBC	cfg_core0_pll[0]	1	O	14-2/14-5
LALE	Local bus address latch enable	eLBC	cfg_core0_pll[1]	1	O	14-2/14-5
LGPL0/LFCLE	Local bus UPM general purpose line 0/flash command latch enable	eLBC	cfg_rio_sys_size	1	O	14-2/14-5
LGPL1/$\overline{\text{LFALE}}$	Local bus GP line 1/flash address latch enable	eLBC	cfg_sgmii2	1	O	14-2/14-5
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LFRE}}$	Local bus GP line 2/output enable/flash read enable	eLBC	cfg_core0_pll[2]	1	O	14-2/14-5
LGPL3/$\overline{\text{LFWP}}$	Local bus GP line 3/flash write protect	eLBC	cfg_boot_seq[0]	1	O	14-2/14-5
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT/LPBSE/ LFRB	Local bus GP line 4/GPCM terminate access/UPM wait/parity byte select/flash ready-busy	eLBC	—	1	I/O	14-2/14-5
LGPL5	Local bus GP line 5 address	eLBC	cfg_boot_seq[1]	1	O	14-2/14-5
LCLK[0:2]	Local bus clock	eLBC	—	3	O	14-2/14-5
LSYNC_IN	Local bus PLL synchronization	eLBC	—	1	I	14-2/14-5
LSYNC_OUT	Local bus PLL synchronization	eLBC	—	1	O	14-2/14-5
$\overline{\text{DMA1\_DREQ}}$[0:1]	DMA1 request 0–1	DMA 1	—	2	I	19-3/19-5
$\overline{\text{DMA1\_DACK}}$[0:1]	DMA1 acknowledge 0–1	DMA 1	—	2	O	19-3/19-5
$\overline{\text{DMA1\_DDONE}}$[0]	DMA1 done 0	DMA 1	cfg_mem_debug[1]	1	O	19-3/19-5
$\overline{\text{DMA1\_DDONE}}$[1]	DMA1 done 1	DMA 1	cfg_fec	1	O	19-3/19-5
$\overline{\text{DMA2\_DREQ}}$[0]	DMA2 request 0	DMA 2	—	1	I	19-3/19-5
$\overline{\text{DMA2\_DREQ}}$[1]	DMA2 request 1	DMA 2	$\overline{\text{LCS}}$[5]	1	I	19-3/19-5

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{DMA2\_DREQ}}$[2]	DMA2 request 2	DMA 2	—	1	I	19-3/19-5
$\overline{\text{DMA2\_DACK}}$[0]	DMA2 acknowledge 0	DMA 2	cfg_mem_debug[0]	1	O	19-3/19-5
$\overline{\text{DMA2\_DACK}}$[1]	DMA2 acknowledge 1	DMA 2	$\overline{\text{LCS}}$[6]	1	O	19-3/19-5
$\overline{\text{DMA2\_DDONE}}$[0]	DMA2 done 0	DMA 2	cfg_ddr_debug	1	O	19-3/19-5
$\overline{\text{DMA2\_DDONE}}$[1]	DMA2 done 1	DMA 2	$\overline{\text{LCS}}$[7]	1	O	19-3/19-5
$\overline{\text{MCP0}}$	Machine check processor 0	PIC	—	1	I	10-3/10-9
$\overline{\text{MCP1}}$	Machine check processor 1	PIC	—	1	I	10-3/10-9
IRQ[0:11]	External interrupt 0–11	PIC	—	12	I	10-3/10-9
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-3/10-9
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	13-1/13-3
UART_SOUT[0]	DUART serial data out	Dual UART	cfg_sgmii4	1	O	13-1/13-3
UART_SOUT[1]	DUART serial data out	Dual UART	cfg_core1_pll[1]	1	O	13-1/13-3
$\overline{\text{UART\_CTS}}$[0:1]	DUART clear to send	Dual UART	—	2	I	13-1/13-3
$\overline{\text{UART\_RTS}}$[0]	DUART ready to send	Dual UART	—	1	O	13-1/13-3
$\overline{\text{UART\_RTS}}$[1]	DUART ready to send	Dual UART	cfg_srds_sgmii_en	1	O	13-1/13-3
IIC1_SDA	I^2C serial data	I^2C 1	—	1	I/O	12-1/12-3
IIC1_SCL	I^2C serial clock	I^2C 1	—	1	I/O	12-1/12-3
IIC2_SDA	I^2C serial data	I^2C 2	—	1	I/O	12-1/12-3
IIC2_SCL	I^2C serial clock	I^2C 2	—	1	I/O	12-1/12-3
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-1/4-1
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-1/4-1
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-1/4-1
$\overline{\text{CKSTP\_IN0}}$	Checkstop in (processor 0)	System control	—	1	I	23-1/23-2
$\overline{\text{CKSTP\_IN1}}$	Checkstop in (processor 1)	System control	—	1	I	23-1/23-2
$\overline{\text{CKSTP\_OUT0}}$	Checkstop out (processor 0)	System control	—	1	O	23-1/23-2
$\overline{\text{CKSTP\_OUT1}}$	Checkstop out (processor 1)	System control	—	1	O	23-1/23-2
GPIO[0:7]	General-purpose input/output	General-purpose I/O	—	8	I/O	22-1/22-2
READY_P0	Device ready (processor 0)	System control	TRIG_OUT	1	O	4-1/4-1
READY_P1	Device ready (processor 1)	System control	cfg_core1_pll[2]	1	O	4-1/4-1
ASLEEP	Asleep	Power mgmt	—	1	O	23-1/23-2
$\overline{\text{UDE0}}$	Unconditional debug event, core 0	Debug	—	1	I	e500 Core Ref. Man.
$\overline{\text{UDE1}}$	Unconditional debug event, core 1	Debug	—	1	I	e500 Core Ref. Man.
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	25-3/25-7

Table 3-1. MPC8572E Signal Reference by Functional Block (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TRIG_OUT	Watchpoint trigger out	Debug	READY_P0	1	O	25-3/25-7
MSRCID[0]	Memory debug source port ID 0	Debug	cfg_fcm_ecc	1	O	25-3/25-7
MSRCID[1]	Memory debug source port ID 1	Debug	cfg_eng_use[1]	1	O	25-3/25-7
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	25-3/25-7
MDVAL	Memory debug data valid	Debug	—	1	O	25-3/25-7
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	25-3/25-7
L1_TSTCLK	L1 test clock	Test	—	1	I	25-3/25-7
L2_TSTCLK	L2 test clock	Test	—	1	I	25-3/25-7
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	25-3/25-7
TEMP_ANODE	Temperature diode anode	Test	—	1	I	25-3/25-7
TEMP_CATHODE	Temperature diode cathode	Test	—	1	O	25-3/25-7
TCK	Test clock	JTAG	—	1	I	25-3/25-7
TDI	Test data in	JTAG	—	1	I	25-3/25-7
TDO	Test data out	JTAG	—	1	O	25-3/25-7
TMS	Test mode select	JTAG	—	1	I	25-3/25-7
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	25-3/25-7
SYSCLK	System clock	Clock	—	1	I	4-1/4-1
RTC	Real time clock	Clock	—	1	I	4-1/4-1
CLK_OUT	Clock out	Clock	—	1	O	4-1/4-1
DDRCLK	DDR complex clock in	Clock	—	1	I	4-1/4-1

Table 3-2. MPC8572E Signal Reference by Signal Name

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
ASLEEP	Asleep	Power mgmt	—	1	O	23-1/23-2
$\overline{\text{CKSTP\_IN0}}$	Checkstop in (processor 0)	System control	—	1	I	23-1/23-2
$\overline{\text{CKSTP\_IN1}}$	Checkstop in (processor 1)	System control	—	1	I	23-1/23-2
$\overline{\text{CKSTP\_OUT0}}$	Checkstop out (processor 0)	System control	—	1	O	23-1/23-2
$\overline{\text{CKSTP\_OUT1}}$	Checkstop out (processor 1)	System control	—	1	O	23-1/23-2
CLK_OUT	Clock out	Clock	—	1	O	4-1/4-1
D1_MA[15:0]	DDR address	DDR Controller 1	—	16	O	9-1/9-4
$\overline{\text{D1\_MAPAR\_ERR}}$	Address parity error	DDR Controller 1	—	1	I	9-1/9-4
D1_MAPAR_OUT	Address parity out	DDR Controller 1	—	1	O	9-1/9-4

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
D1_MBA[2:0]	DDR bank select	DDR Controller 1	—	3	O	9-1/9-4
$\overline{\text{D1\_MCAS}}$	DDR column address strobe	DDR Controller 1	—	1	O	9-1/9-4
D1_MCK[0:5], $\overline{\text{D1\_MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR Controller 1	—	12	O	9-1/9-4
D1_MCKE[0:3]	DDR clock enable	DDR Controller 1	—	4	O	9-1/9-4
$\overline{\text{D1\_MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR Controller 1	—	4	O	9-1/9-4
D1_MDIC[0:1]	Driver impedance calibration	DDR Controller 1	—	2	I/O	9-1/9-4
D1_MDM[0:8]	DDR data mask	DDR Controller 1	—	9	I/O	9-1/9-4
D1_MDQ[0:63]	DDR controller data	DDR Controller 1	—	64	I/O	9-1/9-4
D1_MDQS[0:8]	DDR data strobe	DDR Controller 1	—	9	I/O	9-1/9-4
$\overline{\text{D1\_MDQS}}$[0:8]	DDR data strobe (complement)	DDR Controller 1	—	9	I/O	9-1/9-4
D1_MECC[0:7]	DDR error correcting code	DDR Controller 1	—	8	I/O	9-1/9-4
D1_MODT[0:3]	DRAM On-Die Termination	DDR Controller 1	—	4	O	—
$\overline{\text{D1\_MRAS}}$	DDR row address strobe	DDR Controller 1	—	1	O	9-1/9-4
$\overline{\text{D1\_MWE}}$	DDR write enable	DDR Controller 1	—	1	O	9-1/9-4
D2_MA[15:0]	DDR address	DDR Controller 2	—	16	O	9-1/9-4
$\overline{\text{D2\_MAPAR\_ERR}}$	Address parity error	DDR Controller 2	—	1	I	9-1/9-4
D2_MAPAR_OUT	Address parity out	DDR Controller 2	—	1	O	9-1/9-4
D2_MBA[2:0]	DDR bank select	DDR Controller 2	—	3	O	9-1/9-4
$\overline{\text{D2\_MCAS}}$	DDR column address strobe	DDR Controller 2	—	1	O	9-1/9-4
D2_MCK[0:5], $\overline{\text{D2\_MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR Controller 2	—	12	O	9-1/9-4
D2_MCKE[0:3]	DDR clock enable	DDR Controller 2	—	4	O	9-1/9-4
$\overline{\text{D2\_MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR Controller 2	—	4	O	9-1/9-4
D2_MDIC[0:1]	Driver impedance calibration	DDR Controller 2	—	2	I/O	9-1/9-4
D2_MDM[0:8]	DDR data mask	DDR Controller 2	—	9	I/O	9-1/9-4
D2_MDQ[0:63]	DDR controller data	DDR Controller 2	—	64	I/O	9-1/9-4
D2_MDQS[0:8]	DDR data strobe	DDR Controller 2	—	9	I/O	9-1/9-4
$\overline{\text{D2\_MDQS}}$[0:8]	DDR data strobe (complement)	DDR Controller 2	—	9	I/O	9-1/9-4
D2_MECC[0:7]	DDR error correcting code	DDR Controller 2	—	8	I/O	9-1/9-4
D2_MODT[0:3]	DRAM On-Die Termination	DDR Controller 2	—	4	O	—
$\overline{\text{D2\_MRAS}}$	DDR row address strobe	DDR Controller 2	—	1	O	9-1/9-4
$\overline{\text{D2\_MWE}}$	DDR write enable	DDR Controller 2	—	1	O	9-1/9-4
DDRCLK	DDR complex clock in	Clock	—	1	I	4-1/4-1

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
DMA1_DACK[0:1]	DMA1 acknowledge 0–1	DMA 1	—	2	O	19-3/19-5
DMA1_DDONE[0]	DMA1 done 0	DMA 1	cfg_mem_debug[1]	1	O	19-3/19-5
DMA1_DDONE[1]	DMA1 done 1	DMA 1	cfg_fec	1	O	19-3/19-5
DMA1_DREQ[0:1]	DMA1 request 0–1	DMA 1	—	2	I	19-3/19-5
DMA2_DACK[0]	DMA2 acknowledge 0	DMA 2	cfg_mem_debug[0]	1	O	19-3/19-5
DMA2_DACK[1]	DMA2 acknowledge 1	DMA 2	LCS[6]	1	O	19-3/19-5
DMA2_DDONE[0]	DMA2 done 0	DMA 2	cfg_ddr_debug	1	O	19-3/19-5
DMA2_DDONE[1]	DMA2 done 1	DMA 2	LCS[7]	1	O	19-3/19-5
DMA2_DREQ[0]	DMA2 request 0	DMA 2	—	1	I	19-3/19-5
DMA2_DREQ[1]	DMA2 request 1	DMA 2	LCS[5]	1	I	19-3/19-5
DMA2_DREQ[2]	DMA2 request 2	DMA 2	—	1	I	19-3/19-5
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	15-2/15-10
EC1_MDC	Ethernet management data clock	Ethernet management Interface 1 (eTSEC1)	cfg_tsec_1_reduce	1	O	15-2/15-10
EC1_MDIO	Ethernet management data in/out	Ethernet management Interface 1 (eTSEC1)	—	1	I/O	15-2/15-10
EC3_MDC	Ethernet management data clock	Ethernet management Interface 3 (eTSEC3)	cfg_cpu1_boot	1	O	15-2/15-10
EC3_MDIO	Ethernet management data in/out	Ethernet management Interface 3 (eTSEC3)	—	1	I/O	15-2/15-10
EC5_MDC	Ethernet management data clock	Ethernet management Interface 5 (FEC)	—	1	O	15-2/15-10
EC5_MDIO	Ethernet management data in/out	Ethernet management Interface 5 (FEC)	—	1	I/O	15-2/15-10
FEC_COL	FEC collision detect	FEC	TSEC4_RX_CLK, TSEC3_COL	1	I	16-1/16-7
FEC_CRS	FEC carrier sense	FEC	TSEC4_RX_DV, TSEC3_CRS	1	I	16-1/16-7
FEC_RX_CLK	FEC receive clock	FEC	TSEC3_RX_CLK	1	I	16-1/16-7
FEC_RX_DV	FEC receive data valid	FEC	TSEC3_RX_DV	1	I	16-1/16-7

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
FEC_RX_ER	FEC receive error	FEC	TSEC3_RX_ER	1	I	16-1/16-7
FEC_RXD[3:0]	FEC receive data 3–0	FEC	TSEC3_RXD[3:0]	4	I	16-1/16-7
FEC_TX_CLK	FEC transmit clock in	FEC	TSEC3_TX_CLK	1	I	16-1/16-7
FEC_TX_EN	FEC transmit enable	FEC	TSEC3_TX_EN	1	O	16-1/16-7
FEC_TX_ER	FEC transmit error	FEC	TSEC4_TX_EN, TSEC3_TX_ER	1	O	16-1/16-7
FEC_TXD[1:0]	FEC transmit data 1–0	FEC	TSEC3_TXD[1:0], cfg_tsec3_prtcl[1:0]	2	O	16-1/16-7
FEC_TXD[2]	FEC transmit data 2	FEC	TSEC3_TXD[2], cfg_tsec_3_reduce	1	O	16-1/16-7
FEC_TXD[3]	FEC transmit data 3	FEC	TSEC3_TXD[3], cfg_sgmii3	1	O	16-1/16-7
GPIO[0:7]	General-purpose input/output	General-purpose I/O	—	8	I/O	22-1/22-2
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-1/4-1
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-1/4-1
IIC1_SCL	I^2C serial clock	I^2C 1	—	1	I/O	12-1/12-3
IIC1_SDA	I^2C serial data	I^2C 1	—	1	I/O	12-1/12-3
IIC2_SCL	I^2C serial clock	I^2C 2	—	1	I/O	12-1/12-3
IIC2_SDA	I^2C serial data	I^2C 2	—	1	I/O	12-1/12-3
IRQ[0:11]	External interrupt 0–11	PIC	—	12	I	10-3/10-9
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-3/10-9
L1_TSTCLK	L1 test clock	Test	—	1	I	25-3/25-7
L2_TSTCLK	L2 test clock	Test	—	1	I	25-3/25-7
LA[27]	Local bus burst address	eLBC	cfg_cpu0_boot	1	O	14-2/14-5
LA[28]	Local bus port address	eLBC	cfg_sgmii1	1	O	14-2/14-5
LA[29:31]	Local bus port address	eLBC	cfg_sys_pll[0:2]	3	O	14-2/14-5
LAD[0:31]	Local bus address/data	eLBC	cfg_gpinput[0:31]	32	I/O	14-2/14-5
LALE	Local bus address latch enable	eLBC	cfg_core0_pll[1]	1	O	14-2/14-5
LBCTL	Local bus data buffer control	eLBC	cfg_core0_pll[0]	1	O	14-2/14-5
LCLK[0:2]	Local bus clock	eLBC	—	3	O	14-2/14-5
$\overline{\text{LCS}}$[0:4]	Local bus chip select 0–4	eLBC	—	5	O	14-2/14-5
$\overline{\text{LCS}}$[5]	Local bus chip select 5	eLBC	$\overline{\text{DMA2\_DREQ}}$[1]	1	O	14-2/14-5
$\overline{\text{LCS}}$[6]	Local bus chip select 6	eLBC	$\overline{\text{DMA2\_DACK}}$[1]	1	O	14-2/14-5
$\overline{\text{LCS}}$[7]	Local bus chip select 7	eLBC	$\overline{\text{DMA2\_DDONE}}$[1]	1	O	14-2/14-5
LDP[0:3]	Local bus data parity	eLBC	—	4	I/O	14-2/14-5

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
LGPL0/LFCLE	Local bus UPM general purpose line 0/flash command latch enable	eLBC	cfg_rio_sys_size	1	O	14-2/14-5
LGPL1/$\overline{\text{LFALE}}$	Local bus GP line 1/flash address latch enable	eLBC	cfg_sgmii2	1	O	14-2/14-5
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LFRE}}$	Local bus GP line 2/output enable/flash read enable	eLBC	cfg_core0_pll[2]	1	O	14-2/14-5
LGPL3/$\overline{\text{LFWP}}$	Local bus GP line 3/flash write protect	eLBC	cfg_boot_seq[0]	1	O	14-2/14-5
LGPL4/$\overline{\text{LGTA}}$/ LUPWAIT/LPBSE/ LFRB	Local bus GP line 4/GPCM terminate access/UPM wait/parity byte select/flash ready-busy	eLBC	—	1	I/O	14-2/14-5
LGPL5	Local bus GP line 5 address	eLBC	cfg_boot_seq[1]	1	O	14-2/14-5
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	25-3/25-7
LSYNC_IN	Local bus PLL synchronization	eLBC	—	1	I	14-2/14-5
LSYNC_OUT	Local bus PLL synchronization	eLBC	—	1	O	14-2/14-5
$\overline{\text{LWE}}$[1:3]/$\overline{\text{LBS}}$[1:3]	Local bus write enable/byte select 1–3	eLBC	cfg_host_agt[0:2]	3	O	14-2/14-5
$\overline{\text{LWE0}}$/$\overline{\text{LBS0}}$/$\overline{\text{LFWE}}$	Local bus write enable/byte select 0/flash write enable	eLBC	cfg_core1_pll[0]	1	O	14-2/14-5
$\overline{\text{MCP0}}$	Machine check processor 0	PIC	—	1	I	10-3/10-9
$\overline{\text{MCP1}}$	Machine check processor 1	PIC	—	1	I	10-3/10-9
MDVAL	Memory debug data valid	Debug	—	1	O	25-3/25-7
MSRCID[0]	Memory debug source port ID 0	Debug	cfg_fcm_ecc	1	O	25-3/25-7
MSRCID[1]	Memory debug source port ID 1	Debug	cfg_eng_use[1]	1	O	25-3/25-7
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	25-3/25-7
READY_P0	Device ready (processor 0)	System control	TRIG_OUT	1	O	4-1/4-1
READY_P1	Device ready (processor 1)	System control	cfg_core1_pll[2]	1	O	4-1/4-1
RTC	Real time clock	Clock	—	1	I	4-1/4-1
SD1_REF_CLK	PLL reference clock	SerDes 1	—	1	I	
$\overline{\text{SD1\_REF\_CLK}}$	PLL reference clock (complement)	SerDes 1	—	1	I	
SD1_RX[7:0]	Receive data	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	I	21-2/21-5

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{SD1\_RX}}$[7:0]	Receive data (complement)	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	I	21-2/21-5
SD1_RX[7:4]	Receive data	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	I	21-2/21-5
$\overline{\text{SD1\_RX}}$[7:4]	Receive data (complement)	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	I	21-2/21-5
SD1_RX[7:4]	Receive data	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	I	20.4.2/20-4
$\overline{\text{SD1\_RX}}$[7:4]	Receive data (complement)	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	I	20.4.2/20-4
SD1_RX[7:6]	Receive data	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	I	21-2/21-5
$\overline{\text{SD1\_RX}}$[7:6]	Receive data (complement)	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	I	21-2/21-5
SD1_TX[7:0]	Transmit data	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:0]	Transmit data (complement)	PCI Express Controller 1	Multiplexed with PCI Express 2 & 3, Serial RapidIO	8	O	21-2/21-5
SD1_TX[7:4]	Transmit data	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:4]	Transmit data (complement)	PCI Express Controller 2	Multiplexed with PCI Express 1 & 3, Serial RapidIO	4	O	21-2/21-5
SD1_TX[7:4]	Transmit data	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	O	20.4.2/20-4
$\overline{\text{SD1\_TX}}$[7:4]	Transmit data (complement)	Serial RapidIO	Multiplexed with PCI Express 1, 2 & 3	4	O	20.4.2/20-4
SD1_TX[7:6]	Transmit data	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	O	21-2/21-5
$\overline{\text{SD1\_TX}}$[7:6]	Transmit data (complement)	PCI Express Controller 3	Multiplexed with PCI Express 1 & 2, Serial RapidIO	2	O	21-2/21-5
SD2_REF_CLK	PLL reference clock	SerDes 2	—	1	I	
$\overline{\text{SD2\_REF\_CLK}}$	PLL reference clock (complement)	SerDes 2	—	1	I	

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
SD2_RX[0], $\overline{\text{SD2\_RX[0]}}$	SGMII receive data (and complement)	eTSEC1	—	2	I	15-2/15-10
SD2_RX[1], $\overline{\text{SD2\_RX[1]}}$	SGMII receive data (and complement)	eTSEC2	—	2	I	15-2/15-10
SD2_RX[2], $\overline{\text{SD2\_RX[2]}}$	SGMII receive data (and complement)	eTSEC3	—	2	I	15-2/15-10
SD2_RX[3], $\overline{\text{SD2\_RX[3]}}$	SGMII receive data (and complement)	eTSEC4	—	2	I	15-2/15-10
SD2_TX[0], $\overline{\text{SD2\_TX[0]}}$	SGMII transmit data (and complement)	eTSEC1	—	2	O	15-2/15-10
SD2_TX[1], $\overline{\text{SD2\_TX[1]}}$	SGMII transmit data (and complement)	eTSEC2	—	2	O	15-2/15-10
SD2_TX[2], $\overline{\text{SD2\_TX[2]}}$	SGMII transmit data (and complement)	eTSEC3	—	2	O	15-2/15-10
SD2_TX[3], $\overline{\text{SD2\_TX[3]}}$	SGMII transmit data (and complement)	eTSEC4	—	2	O	15-2/15-10
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-1/4-1
SYSCLK	System clock	Clock	—	1	I	4-1/4-1
TCK	Test clock	JTAG	—	1	I	25-3/25-7
TDI	Test data in	JTAG	—	1	I	25-3/25-7
TDO	Test data out	JTAG	—	1	O	25-3/25-7
TEMP_ANODE	Temperature diode anode	Test	—	1	I	25-3/25-7
TEMP_CATHODE	Temperature diode cathode	Test	—	1	O	25-3/25-7
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	25-3/25-7
TMS	Test mode select	JTAG	—	1	I	25-3/25-7
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	25-3/25-7
TRIG_OUT	Watchpoint trigger out	Debug	READY_P0	1	O	25-3/25-7
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	25-3/25-7
TSEC_1588_ TRIG_OUT	eTSEC 1588 trigger out	eTSEC	cfg_srds_sgmii_refclk	1	O	15-2/15-10
TSEC_1588_ CLK_IN	eTSEC 1588 clock in	eTSEC	—	1	I	15-2/15-10
TSEC_1588_ CLK_OUT	eTSEC 1588 clock out	eTSEC	cfg_ddr_pll[0]	1	O	15-2/15-10
TSEC_1588_ PULSE_OUT1	eTSEC 1588 pulse out 1	eTSEC	cfg_ddr_pll[1]	1	O	15-2/15-10
TSEC_1588_ PULSE_OUT2	eTSEC 1588 pulse out 2	eTSEC	cfg_ddr_pll[2]	1	O	15-2/15-10
TSEC_1588_ TRIG_IN	eTSEC 1588 trigger in	eTSEC	—	1	I	15-2/15-10

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC1_COL	TSEC1 collision detect	eTSEC1	—	1	I	15-2/15-10
TSEC1_CRS	TSEC1 carrier sense	eTSEC1	—	1	I	15-2/15-10
TSEC1_GTX_CLK	TSEC1 transmit clock out	eTSEC1	—	1	O	15-2/15-10
TSEC1_RX_CLK	TSEC1 receive clock	eTSEC1	—	1	I	15-2/15-10
TSEC1_RX_DV	TSEC1 receive data valid	eTSEC1	—	1	I	15-2/15-10
TSEC1_RX_ER	TSEC1 receiver error	eTSEC1	—	1	I	15-2/15-10
TSEC1_RXD[7:0]	TSEC1 receive data	eTSEC1	—	8	I	15-2/15-10
TSEC1_TX_CLK	TSEC1 transmit clock in	eTSEC1	—	1	I	15-2/15-10
TSEC1_TX_EN	TSEC1 transmit enable	eTSEC1	—	1	O	15-2/15-10
TSEC1_TX_ER	TSEC1 transmit error	eTSEC1	cfg_rom_loc[3]	1	O	15-2/15-10
TSEC1_TXD[0]	TSEC1 transmit data 0	eTSEC1	cfg_tsec1_prtcl[0]	1	O	15-2/15-10
TSEC1_TXD[3:1]	TSEC1 transmit data 3–1	eTSEC1	cfg_io_ports[0:2]	3	O	15-2/15-10
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	eTSEC1	cfg_rom_loc[0:2]	3	O	15-2/15-10
TSEC1_TXD[7]	TSEC1 transmit data 7	eTSEC1	cfg_tsec1_prtcl[1]	1	O	15-2/15-10
TSEC2_COL	TSEC2 collision detect	eTSEC2	—	1	I	15-2/15-10
TSEC2_CRS	TSEC2 carrier sense	eTSEC2	—	1	I	15-2/15-10
TSEC2_GTX_CLK	TSEC2 transmit clock out	eTSEC2	—	1	O	15-2/15-10
TSEC2_RX_CLK	TSEC2 receive clock	eTSEC2	—	1	I	15-2/15-10
TSEC2_RX_DV	TSEC2 receive data valid	eTSEC2	—	1	I	15-2/15-10
TSEC2_RX_ER	TSEC2 receive error	eTSEC2	—	1	I	15-2/15-10
TSEC2_RXD[7:0]	TSEC2 receive data	eTSEC2	—	8	I	15-2/15-10
TSEC2_TX_CLK	TSEC2 transmit clock in	eTSEC2	—	1	I	15-2/15-10
TSEC2_TX_EN	TSEC2 transmit enable	eTSEC2	—	1	O	15-2/15-10
TSEC2_TX_ER	TSEC2 transmit error	eTSEC2	cfg_io_ports[3]	1	O	15-2/15-10
TSEC2_TXD[0]	TSEC2 transmit data 0	eTSEC2	cfg_tsec2_prtcl[0]	1	O	15-2/15-10
TSEC2_TXD[1]	TSEC2 transmit data 1	eTSEC2	cfg_dram_type	1	O	15-2/15-10
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	eTSEC2	cfg_device_ID[7:5]	3	O	15-2/15-10
TSEC2_TXD[5]	TSEC2 transmit data 5	eTSEC2	cfg_srds1_en	1	O	15-2/15-10
TSEC2_TXD[6]	TSEC2 transmit data 6	eTSEC2	—	1	O	15-2/15-10
TSEC2_TXD[7]	TSEC2 transmit data 7	eTSEC2	cfg_tsec2_prtcl[1]	1	O	15-2/15-10
TSEC3_COL	TSEC3 collision detect	eTSEC3	TSEC4_RX_CLK, FEC_COL	1	I	15-2/15-10
TSEC3_CRS	TSEC3 carrier sense	eTSEC3	TSEC4_RX_DV, FEC_CRS	1	I	15-2/15-10
TSEC3_GTX_CLK	TSEC3 transmit clock out	eTSEC3	—	1	O	15-2/15-10
TSEC3_RX_CLK	TSEC3 receive clock	eTSEC3	FEC_RX_CLK	1	I	15-2/15-10

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC3_RX_DV	TSEC3 receive data valid	eTSEC3	FEC_RX_DV	1	I	15-2/15-10
TSEC3_RX_ER	TSEC3 receive error	eTSEC3	FEC_RX_ER	1	I	15-2/15-10
TSEC3_RXD[3:0]	TSEC3 receive data 3–0	eTSEC3	FEC_RXD[3:0]	4	I	15-2/15-10
TSEC3_TX_CLK	TSEC3 transmit clock in	eTSEC3	FEC_TX_CLK	1	I	15-2/15-10
TSEC3_TX_EN	TSEC3 transmit enable	eTSEC3	FEC_TX_EN	1	O	15-2/15-10
TSEC3_TX_ER	TSEC3 transmit error	eTSEC3	TSEC4_TX_EN, FEC_TX_ER	1	O	15-2/15-10
TSEC3_TXD[1:0]	TSEC3 transmit data 1–0	eTSEC3	FEC_TXD[1:0], cfg_tsec3_prtcl[1:0]	2	O	15-2/15-10
TSEC3_TXD[2]	TSEC3 transmit data 2	eTSEC3	FEC_TXD[2], cfg_tsec_3_reduce	1	O	15-2/15-10
TSEC3_TXD[3]	TSEC3 transmit data 3	eTSEC3	FEC_TXD[3], cfg_sgmii3	1	O	15-2/15-10
TSEC3_TXD[4]	TSEC3 transmit data 4	eTSEC3	TSEC4_TXD[0], cfg_tsec4_prtcl[0]	1	O	15-2/15-10
TSEC3_TXD[5]	TSEC3 transmit data 5	eTSEC3	TSEC4_TXD[1], cfg_tsec4_prtcl[1]	1	O	15-2/15-10
TSEC3_TXD[6]	TSEC3 transmit data 6	eTSEC3	TSEC4_TXD[2]	1	O	15-2/15-10
TSEC3_TXD[7]	TSEC3 transmit data 7	eTSEC3	TSEC4_TXD[3]	1	O	15-2/15-10
TSEC4_GTX_CLK	TSEC4 transmit clock out	eTSEC4	—	1	O	15-2/15-10
TSEC4_RX_CLK	TSEC4 receive clock	eTSEC4	TSEC3_COL, FEC_COL	1	I	15-2/15-10
TSEC4_RX_DV	TSEC4 receive data valid	eTSEC4	TSEC3_CRS, FEC_CRS	1	I	15-2/15-10
TSEC4_RXD[3:0]	TSEC4 receive data 3–0	eTSEC4	TSEC3_RXD[7:4]	4	I	15-2/15-10
TSEC4_TX_EN	TSEC4 transmit enable	eTSEC4	TSEC3_TX_ER, FEC_TX_ER	1	O	15-2/15-10
TSEC4_TXD[1:0]	TSEC4 transmit data 1–0	eTSEC4	TSEC3_TXD[5:4], cfg_tsec4_prtcl[1:0]	2	O	15-2/15-10
TSEC4_TXD[2]	TSEC4 transmit data 2	eTSEC4	TSEC3_TXD[6]	1	O	15-2/15-10
TSEC4_TXD[3]	TSEC4 transmit data 3	eTSEC4	TSEC3_TXD[7]	1	O	15-2/15-10
$\overline{\text{UART\_CTS}}$[0:1]	DUART clear to send	Dual UART	—	2	I	13-1/13-3
$\overline{\text{UART\_RTS}}$[0]	DUART ready to send	Dual UART	—	1	O	13-1/13-3
$\overline{\text{UART\_RTS}}$[1]	DUART ready to send	Dual UART	cfg_srds_sgmii_en	1	O	13-1/13-3
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	13-1/13-3
UART_SOUT[0]	DUART serial data out	Dual UART	cfg_sgmii4	1	O	13-1/13-3
UART_SOUT[1]	DUART serial data out	Dual UART	cfg_core1_pll[1]	1	O	13-1/13-3

Table 3-2. MPC8572E Signal Reference by Signal Name (continued)

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{UDE0}}$	Unconditional debug event, core 0	Debug	—	1	I	e500 Core Ref. Man.
$\overline{\text{UDE1}}$	Unconditional debug event, core 1	Debug	—	1	I	e500 Core Ref. Man.

3.2 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 Section 4.4.3, “Power-On Reset Configuration.”

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of $\overline{\text{HRESET}}$. However, there is a setup and hold time for these signals relative to the rising edge of $\overline{\text{HRESET}}$, as described in the *MPC8572E Integrated Processor 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. Most of 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 *MPC8572E Integrated Processor Hardware Specifications.*

Table 3-3. MPC8572E Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
1588	TSEC_1588_TRIG_OUT	cfg_srds_sgmii_refclk	1
1588	TSEC_1588_CLK_OUT	cfg_ddr_pll[0]	1
1588	TSEC_1588_PULSE_OUT1	cfg_ddr_pll[1]	1
1588	TSEC_1588_PULSE_OUT2	cfg_ddr_pll[2]	1
Ethernet Management	EC1_MDC	cfg_tsec_1_reduce	1
Ethernet Management	EC3_MDC	cfg_cpu1_boot	1
eTSEC1	TSEC1_TXD[7]	cfg_tsec1_prtcl[1]	1
	TSEC1_TXD[0]	cfg_tsec1_prtcl[0]	1
	TSEC1_TXD[6:4]	cfg_rom_loc[0:2]	111
	TSEC1_TXD[3:1]	cfg_io_ports[0:2]	111
	TSEC1_TX_ER	cfg_rom_loc[3]	1

Table 3-3. MPC8572E Reset Configuration Signals (continued)

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
eTSEC2	TSEC2_TXD[7]	cfg_tsec2_prtcl[1]	1
	TSEC2_TXD[0]	cfg_tsec2_prtcl[0]	1
	TSEC2_TXD[5]	cfg_srds1_en	1
	TSEC2_TXD[4:2]	cfg_device_ID[7:5]	111
	TSEC2_TXD[1]	cfg_dram_type	1
	TSEC2_TX_ER	cfg_io_ports[3]	1
eTSEC3 / FEC	TSEC3_TXD[3]/FEC_TXD[3]	cfg_sgmii3	1
	TSEC3_TXD[2]/FEC_TXD[2]	cfg_tsec_3_reduce	1
	TSEC3_TXD[1:0] /FEC_TXD[1:0]	cfg_tsec3_prtcl[1:0]	11
eTSEC4 / eTSEC3	TSEC4_TXD[1:0] /TSEC3_TXD[5:4]	cfg_tsec4_prtcl[1:0]	11
eLBC	LAD[0:31]	cfg_gpinput[0:31]	Indeterminate if not driven (no default)
	LA27	cfg_cpu0_boot	1
	LA[28]	cfg_sgmii1	1
	LA[29:31]	cfg_sys_pll[0:2]	Must be driven
	$\overline{\text{LWE}}$[0]/$\overline{\text{LBS}}$[0]/$\overline{\text{LFWE}}$	cfg_core1_pll[0]	Must be driven
	$\overline{\text{LWE}}$[1:3]/$\overline{\text{LBS}}$[1:3]	cfg_host_agt[0:2]	111
	LBCTL	cfg_core0_pll[0]	Must be driven
	LALE	cfg_core0_pll[1]	Must be driven
	LGPL2/$\overline{\text{LOE}}$/$\overline{\text{LFRE}}$	cfg_core0_pll[2]	Must be driven
	LGPL0/LFCLE	cfg_rio_sys_size	1
	LGPL1/LFALE	cfg_sgmii2	1
	LGPL3/$\overline{\text{LFWP}}$	cfg_boot_seq[0]	1
	LGPL5	cfg_boot_seq[1]	1
DMA 1	$\overline{\text{DMA1\_DDONE}}$[0]	cfg_mem_debug[1]	1
	$\overline{\text{DMA1\_DDONE}}$[1]	cfg_fec	1
DMA 2	$\overline{\text{DMA2\_DACK}}$[0]	cfg_mem_debug[0]	1
	$\overline{\text{DMA2\_DDONE}}$[0]	cfg_ddr_debug	1
DUART	UART_SOUT[0]	cfg_sgmii4	1
	UART_SOUT[1]	cfg_core1_pll[1]	Must be driven
	$\overline{\text{UART\_RTS}}$[1]	cfg_srds_sgmii_en	1

Table 3-3. MPC8572E Reset Configuration Signals (continued)

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
Debug	READY_P1	cfg_core1_pll[2]	Must be driven
	MSRCID[0]	cfg_fcm_ecc	1
	MSRCID[1]	cfg_eng_use[1]	1

3.3 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{HRESET}}$ is asserted), the MPC8572E aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See Chapter 4, "Reset, Clocking, and Initialization," for a complete description of the reset functionality.

During reset, the MPC8572E ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state.

Chapter 4
Reset, Clocking, and Initialization

This chapter describes the reset, clocking, and some overall initialization of the MPC8572E, 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.

4.1 Overview

The reset, clocking, and control signals provide many options for the operation of the MPC8572E. Additionally, many modes are selected with reset configuration signals during a hard reset (assertion of $\overline{\text{HRESET}}$).

4.2 External Signal Descriptions

Table 4-1 summarizes the external signals described in this chapter. Table 4-2 and Table 4-3 have detailed signal descriptions, but Table 4-1 contains references to additional sections that contain more information.

Table 4-1. Signal Summary

Signal	I/O	Description	References (Section/Page)
$\overline{\text{HRESET}}$	I	Hard reset input. Causes a power-on reset (POR) sequence.	4.4.1.2/4-10
$\overline{\text{HRESET\_REQ}}$	O	Hard reset request output. An internal block requests that $\overline{\text{HRESET}}$ be asserted.	—
$\overline{\text{SRESET}}$	I	Soft reset input. Causes *mcp* assertion to the cores.	4.4.1.1/4-9
READY_P0	O	The MPC8572E has completed its reset operation and e500 core 0 is not in a power-down (nap, doze or sleep) or debug state.	4.4.2/4-10
READY_P1	O	The MPC8572E has completed its reset operation and e500 core 1 is not in a power-down (nap, doze or sleep) or debug state.	4.4.2/4-10
SYSCLK	I	Primary clock input to the MPC8572E.	4.4.4.1/4-30
DDRCLK	I	Clock input to the MPC8572E that sources the DDR controller complex PLL	4.4.4.1/4-30
RTC	I	Real time clock input.	4.4.4.5/4-32
SD_REF_CLK1/ $\overline{\text{SD\_REF\_CLK1}}$	I	SERDES high-speed interface 1 reference clock.	4.4.4.2/4-31
SD_REF_CLK2 / $\overline{\text{SD\_REF\_CLK2}}$	I	SERDES high-speed interface 2 reference clock.	4.4.4.3/4-31

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

4.2.1 System Control Signals

Table 4-2 describes some of the system control signals of the MPC8572E. Section 4.4.3, "Power-On Reset Configuration," describes the signals that also function as reset configuration signals. Note that the CKSTP_IN0/1 and CKSTP_OUT0/1 signals are described in Section 23.3, "External Signal Description."

Table 4-2. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description	
HRESET	I	Hard reset. Causes the MPC8572E to abort all current internal and external transactions and set all registers to their default values. HRESET may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated—See Chapter 3, "Signal Descriptions," and Section 4.4.3, "Power-On Reset Configuration," for more information on the interpretation of the other MPC8572E signals during reset.
		Timing	Assertion/Negation—The *MPC8572E Integrated Processor Hardware Specifications* gives specific timing information for this signal and the reset configuration signals.
HRESET_REQ	O	Hard reset request. Indicates to the board (system in which the MPC8572E is embedded) that a condition requiring the assertion of HRESET has been detected.	
		State Meaning	Asserted—A watchdog timer expiration sent to the platform (see Section 6.6.1, "Timer Control Register (TCR)"; WRC field definition), a RapidIO command, a boot sequencer failure (see Section 12.4.5, "Boot Sequencer Mode"), a local bus uncorrectable ECC error during NAND Flash boot process (see Section 14.4.3.4.2, "Boot Block Loading into the FCM Buffer RAM), or a software settable reset request (see Section 23.4.1.17, "Reset Control Register (RSTCR)") has triggered a request for hard reset. Negated—Indicates no reset request.
		Timing	Assertion/Negation—May occur any time, synchronous to the core complex bus clock. Once asserted, HRESET_REQ does not negate until HRESET is asserted.

Table 4-2. System Control Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
SRESET	I	Soft reset. Causes a machine check interrupt to the e500 cores. Note that if a given e500 core is not configured to process machine check interrupts, the assertion of SRESET causes a checkstop in that particular core. SRESET need not be asserted during a hard reset.	
		State Meaning	Asserted—Asserting SRESET causes a machine check interrupt (edge sensitive) to the e500 cores. SRESET has no effect while HRESET is asserted. However, the POR sequence is paused if SRESET is asserted during POR.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Must be asserted for at least three SYSCLK cycles.
READY_P0	O	Ready Processor 0. Multiplexed with TRIG_OUT and QUIESCE. See Chapter 25, "Debug Features and Watchpoint Facility," for more information on TOSR and TRIG_OUT.	
		State Meaning	Asserted—Indicates that the MPC8572E has completed the reset operation and e500 core 0 is not in a power-down state (nap, doze, or sleep) when TOSR[SEL] equals 0b000. See Section 4.4.2, "Power-On Reset Sequence," for more information.
		Timing	Assertion/Negation—Initial assertion of READY_P0 after reset is synchronous with SYSCLK. Subsequent assertion/negation due to power down modes occurs asynchronously.
READY_P1	O	Ready Processor 1.	
		State Meaning	Asserted—Indicates that the MPC8572E has completed the reset operation and e500 core 1 is not in a power-down state (nap, doze, or sleep). See Section 4.4.2, "Power-On Reset Sequence," for more information.
		Timing	Assertion/Negation—Initial assertion of READY_P1 after reset is synchronous with SYSCLK. Subsequent assertion/negation due to power down modes occurs asynchronously.

4.2.2 Clock Signals

Table 4-3 describes the overall clock signals of the MPC8572E. Note that some clock signals are specific to blocks within the MPC8572E, and although some of their functionality is described in Section 4.4.4, "Clocking," they are defined in detail in their respective chapters.

Note that there is also a CLK_OUT signal in the MPC8572E; the signal driven on the CLK_OUT pin is selectable and described in Section 23.4.1.20, "Clock Out Control Register (CLKOCR)."

Table 4-3. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SYSCLK	I	System clock (SYSCLK). SYSCLK is the primary clock input to the MPC8572E. It is the clock source for the e500 cores and for all devices and interfaces that operate synchronously with the core. It is multiplied up with a phased-lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock), which is used by virtually all of the synchronous system logic, including the L2 cache, DDR SDRAM memory controllers (when configured to run in synchronous mode), local bus memory controller, and other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL's in the e500 cores and the PLL that creates the local bus memory clocks.	
		Timing	Assertion/Negation—See the *MPC8572E Integrated Processor Hardware Specifications* for specific timing information for this signal.

Table 4-3. Clock Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
DDRCLK	I	DDR controller complex clock (DDRCLK). DDRCLK is the clock source for the DDR memory controller complex except in the case where synchronous mode of operation is selected (see Section 4.4.3.2, "DDR PLL Ratio," on page 4-13). This clock input is multipled up with a phased-lock loop (PLL) to create the DDR controller complex clock. The DDR memory controller complex clock is the DDR data rate on the external interface unless the given controller is configured to run in half speed (see Section 23.4.1.19, "DDR Clock Disable Register (DDRCLKDR)").
		Timing Assertion/Negation—See the *MPC8572E Integrated Processor Hardware Specifications* for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the e500 cores. The RTC timing specifications are given in the *MPC8572E Integrated Processor Hardware Specifications*, but the maximum frequency should be less than one-quarter of the CCB frequency. See Section 4.4.4.5, "Real Time Clock." This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).
		Timing Assertion/Negation—See the *MPC8572E Integrated Processor Hardware Specifications* for specific timing information for this signal.

4.3 Memory Map/Register Definition

This section describes the configuration and control registers that control access to the configuration space and to the boot code as well as guidelines for accessing these regions. It also contains a brief description of the boot sequencer which may be used to initialize configuration registers or memory before the any of CPU's are released to boot.

4.3.1 Local Configuration Control

Table 4-4 shows the memory map for local configuration control registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 4-4. Local Configuration Control Register Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-6
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-7

Table 4-4. Local Configuration Control Register Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-7
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-9

4.3.1.1 Accessing Configuration, Control, and Status Registers

The configuration, control, and status registers are memory mapped. The set of configuration, control, and status registers occupies a 1-Mbyte region of memory. Their location is programmable using the CCSR base address register (CCSRBAR). The default base address for the configuration, control, and status registers is 0x0_FF70_0000 (CCSRBAR = 0x000F_F700). CCSRBAR itself is 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. The contents of CCSRBAR are broadcast internally in the MPC8572E to all functional units that need to be able to identify or create configuration transactions.

4.3.1.1.1 Updating CCSRBAR

Updates to CCSRBAR that relocate the entire 1-Mbyte region of configuration, control, and status registers require 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 make sure this happens, these guidelines should be followed:

- CCSRBAR should be updated during initial configuration of the device when only one host or controller has access to the device.
  - If the boot sequencer is being used to initialize, it is recommended that the boot sequencer set CCSRBAR to its desired final location.
  - If an external host on PCI Express or RapidIO is configuring the device, it should set CCSRBAR to the desired final location before the e500 cores are released to boot.
  - If an e500 core is initializing the device, it should set CCSRBAR to the desired final location before enabling the other core and I/O devices to access the device.
- When an e500 core is writing to CCSRBAR, it should use the following sequence:
  - Read the current value of CCSRBAR using a load word instruction followed by an **isync**. This forces all accesses to configuration space to complete.
  - Write the new value to CCSRBAR.
  - Perform a load of an address that does not access configuration space or the on-chip SRAM, but has an address mapping already in effect (for example, boot ROM). Follow this load with an **isync**.
  - Read the contents of CCSRBAR from its new location, followed by another **isync**.

4.3.1.1.2 Configuration, Control, and Status Base Address Register (CCSRBAR)

Figure 4-1 shows the fields of CCSRBAR.

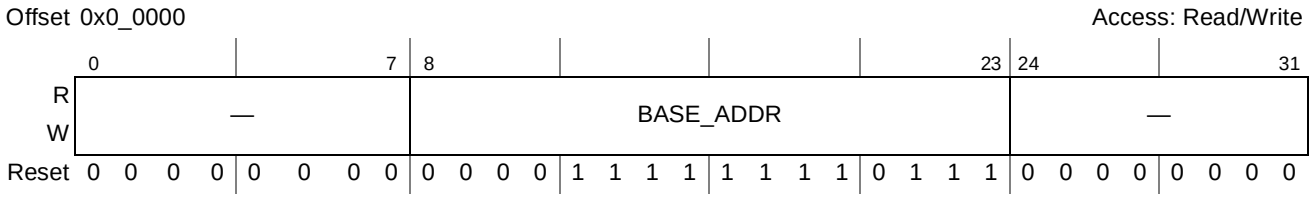


Figure 4-1. Configuration, Control, and Status Register Base Address Register (CCSRBAR)

Table 4-5 defines the bit fields of CCSRBAR.

Table 4-5. CCSRBAR Bit Settings

Bits	Name	Description
0–7	—	Write reserved, read = 0.
8–23	BASE_ADDR	Identifies the16 most-significant address bits of the window used for configuration accesses. The base address is aligned on a 1-Mbyte boundary.
24–31	—	Write reserved, read = 0

4.3.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALTCAR registers. These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. By loading the proper boot sequencer command in the serial ROM, the base address in the ALTCBAR can be combined with the 20 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 boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 12.4.5, “Boot Sequencer Mode,” for more information.

NOTE

The enable bit in the ALTCAR register should be cleared either by the boot sequencer or by the boot code that executes after the boot sequencer has completed its configuration operations. This prevents problems with incorrect mappings if subsequent configuration of the local access windows uses a different target mapping for the address specified in ALTCBAR.

4.3.1.2.1 Alternate Configuration Base Address Register (ALTCBAR)

Figure 4-2 shows the fields of ALTCBAR.

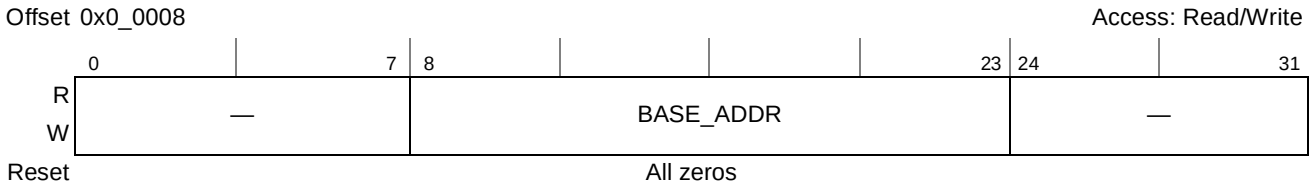


Figure 4-2. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-6 defines the bit fields of ALTCBAR.

Table 4-6. ALTCBAR Bit Settings

Bits	Name	Description
0–7	—	Write reserved, read = 0
8–23	BASE_ADDR	Identifies the16 most significant address bits of an alternate window used for configuration accesses.
24–31	—	Write reserved, read = 0

4.3.1.2.2 Alternate Configuration Attribute Register (ALTCAR)

Figure 4-3 shows the fields of ALTCAR.

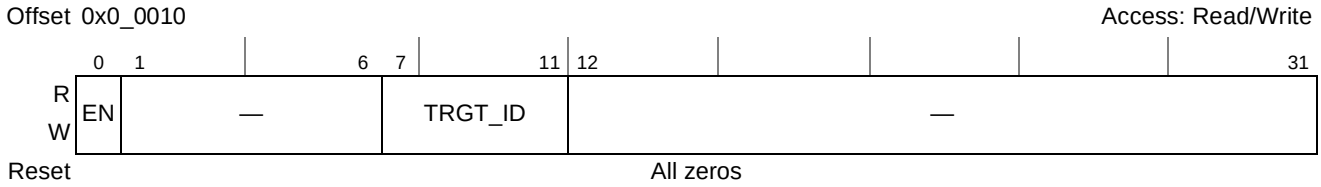


Figure 4-3. Alternate Configuration Attribute Register (ALTCAR)

Table 4-7 defines ALTCAR fields.

Table 4-7. ALTCAR Bit Settings

Bits	Name	Description
0	EN	Enable for a second configuration window. Like CCSRBAR, it has a fixed size of 1 Mbyte. 0 Second configuration window is disabled. 1 Second configuration window is enabled.
1–6	—	Write reserved, read = 0

Table 4-7. ALTCAR Bit Settings (continued)

Bits	Name	Description
7–11	TRGT_ID	Identifies the device ID to target when a transaction hits in the 1-Mbyte address range defined by the second configuration window. 00000 PCI Express interface 3 00001 PCI Express interface 2 00010 PCI Express interface 1 00011 Reserved 00100 Local bus controller 00101–01010 Reserved 01011 Interleaved DDR SDRAM 01100 Serial RapidIO 01101 Reserved 01110 Reserved 01111 DDR SDRAM interface 1 10000-10101 Reserved 10110 DDR SDRAM interface 2 10111-11111 Reserved
12–31	—	Write reserved, read = 0

4.3.1.3 Boot Page Translation

When each e500 core comes out of reset, its MMU has one 4-Kbyte page defined at 0x0_FFFF_F*nnn*. 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_FFE0. For systems in which the boot code resides at a different address, the MPC8572E provides boot page translation capability. Boot page translation is controlled by the boot page translation register (BPTR). Note that boot page translation affects transactions initiated by each of the two e500 cores in the same manner.

The boot sequencer can enable boot page translation, or the boot page translation can be set up by an external host when the MPC8572E is configured to be in boot holdoff mode. If translation is to be performed to a page outside the default boot ROM address range defined in the MPC8572E (8 Mbytes at 0x0_FF80_0000 to 0x0_FFFF_FFFF as defined in Section 4.4.3.4, "Boot ROM Location"), the external host or boot sequencer must then also set up a local access window to define the routing of the boot code fetch to the target interface that contains the boot code, because the BPTR defines only the address translation, not the target interface window. See Section 2.1, "Local Memory Map Overview and Example," and Section 12.4.5, "Boot Sequencer Mode," for more information.

4.3.1.3.1 Boot Page Translation Register (BPTR)

Figure 4-4 shows the fields of BPTR.

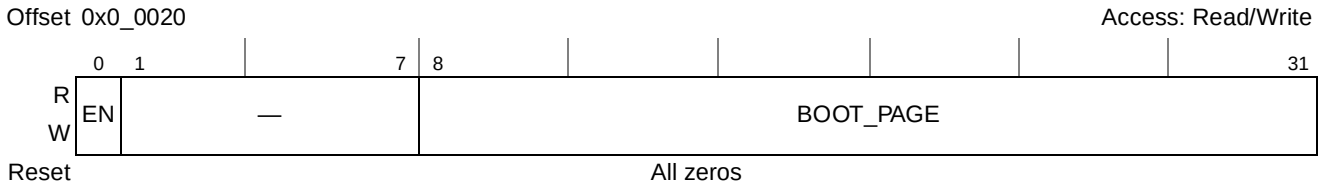


Figure 4-4. Boot Page Translation Register (BPTR)

Table 4-8 describes BPTR bit settings.

Table 4-8. BPTR Bit Settings

Bits	Name	Description
0	EN	Boot page translation enable for e500 core transactions 0 Boot page is not translated. 1 Boot page is translated as defined in the BPTR[BOOT_PAGE] parameter.
1–7	—	Write reserved, read = 0
8–31	BOOT_PAGE	Translation for boot page. If enabled, the high order 24 bits of e500 core accesses to 0x0_FFFF_F*nnn* are replaced with this value.

4.3.2 Boot Sequencer

The boot sequencer is a DMA engine that accesses a serial ROM on the I^2C1 interface and writes data to CCSR memory or the memory space pointed to by the alternate configuration base address register (ALTCBAR). See Section 4.3.1.2, "Accessing Alternate Configuration Space." The boot sequencer is enabled by reset configuration pins as described in Section 4.4.3.8, "Boot Sequencer Configuration." If the boot sequencer is enabled, the e500 cores are held in reset until the boot sequencer has completed its operation. For more details, see Section 12.4.5, "Boot Sequencer Mode," in the I^2C chapter.

4.4 Functional Description

This section describes the various ways to reset the MPC8572E device, the POR configurations, and the clocking on the device.

4.4.1 Reset Operations

The MPC8572E has reset input signals for hard and soft reset operation.

4.4.1.1 Soft Reset

Assertion of $\overline{\text{SRESET}}$ causes a machine check interrupt to both e500 cores. When this occurs, the soft reset flag is recorded in the machine check summary register in the global utilities block so that software can identify the machine check as a soft reset condition. See the *PowerPC e500 Core Complex Reference*

Manual for more information on the machine check interrupt and Section 23.4.1.11, "Machine Check Summary Register (MCPSUMR)," for more information on the setting of the soft reset flag. Note that if SRESET is asserted before a given e500 core is configured to handle a machine check interrupt, a checkstop condition occurs for that particular core, which causes CKSTP_OUT0 or CKSTP_OUT1 to assert.

4.4.1.2 Hard Reset

The MPC8572E can be completely reset by the assertion of the HRESET input. The assertion of this signal by external logic is the equivalent of a POR and causes the sequence of events described in Section 4.4.2, "Power-On Reset Sequence."

Refer to the *MPC8572E Integrated Processor Hardware Specifications* for the timing requirements for HRESET assertion and negation.

The hard reset request output signal (HRESET_REQ) indicates to external logic that a hard reset is being requested by hardware, software, or a RapidIO device. Hardware causes this signal to assert for a boot sequencer failure (see Section 12.4.5, "Boot Sequencer Mode," and Section 12.4.5.2, "EEPROM Data Format"), for a local bus uncorrectable ECC error during NAND Flash boot process (see Section 14.4.3.4.2, "Boot Block Loading into the FCM Buffer RAM), or when either e500 watchdog timer is configured to cause a hard reset request when it expires (See Section 6.6.1, "Timer Control Register (TCR)"; TCR[WRC] field). Software may request a hard reset by setting a bit in a global utilities register; see Section 23.4.1.17, "Reset Control Register (RSTCR)." A RapidIO device causes this signal to assert if it sends four consecutive RapidIO link maintenance reset commands without any other intervening packets or control symbols, except idle control symbols.

4.4.2 Power-On Reset Sequence

The POR sequence for the MPC8572E is as follows:

1. Power is applied to meet the specifications in the *MPC8572E Integrated Processor Hardware Specifications.*
2. The system asserts HRESET and TRST, causing all registers to be initialized to their default states and most I/O drivers to be three-stated (some clock, clock enables, and system control signals are active).
3. The system applies a stable SYSCLK signal and stable PLL configuration inputs, and the device PLL begins locking to SYSCLK.
4. System negates HRESET after its required hold time and after POR configuration inputs have been valid for at least 4 SYSCLK cycles.

NOTE:

If the JTAG signals are not used, then TRST may be driven negated. It is recommended that TRST not remain asserted after the negation of HRESET. TRST may be connected directly to HRESET.

There is no need to assert the $\overline{\text{SRESET}}$ signal when $\overline{\text{HRESET}}$ is asserted. If $\overline{\text{SRESET}}$ is asserted upon negation of $\overline{\text{HRESET}}$, the POR sequence is paused after the e500 core PLLs are locked and before the e500s' resets are negated. The POR sequence is resumed when $\overline{\text{SRESET}}$ is negated.

5. MPC8572E enables I/O drivers.
6. The e500 PLL configuration inputs are applied, allowing the e500 PLLs to begin locking to the device clock (the CCB clock).
7. The CCB clock is cycled for approximately 50 µs to lock the e500 PLLs.
8. The internal hard resets to the e500 cores are negated and soft resets are negated to the PLLs and other remaining I/O blocks. The PLLs begin to lock.
9. When PLL locking is completed, the Local Bus FCM is released provided NAND Flash is configured as the Boot Device, as described in Section 4.4.3.4, "Boot ROM Location." Once the FCM finishes loading pages from the NAND Flash device, the boot sequencer, if enabled, is allowed to progress, causing it to load configuration data from serial ROMs, as described in Section 4.4.3.8, "Boot Sequencer Configuration."
10. When the Local Bus FCM and boot sequencer complete, the Serial RapidIO and PCI Express interfaces begin training and are released to accept external requests, and the boot vector fetches by the e500 cores are allowed to proceed unless processor booting is further held off by POR configuration inputs as described in Section 4.4.3.7, "CPU Boot Configuration." The MPC8572E is now in its ready state.
11. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state of the system. The ready state for e500 core 0 is also indicated by the assertion of READY_P0/TRIG_OUT if TOSR[SEL] = 000. In this case, READY_P0 is asserted with the same rising edge of SYSCLK, to indicate that the e500 core 0 has reached its ready state. See Section 25.3.4.1, "Trigger Out Source Register (TOSR)," for more information on this register. READY_P1 asserts in a similar fashion to indicate that the e500 core 1 is in its ready state.

    The assertion of READY_P0 and READY_P1 allows external system monitors to know basic device status, for example, exactly when each e500 core emerges from reset, or if the particular core is in a low-power mode. For more information on the debug functions of TRIG_OUT, see Section 25.3.4, "Trigger Out Function." For more information about power management states, see Section 23.4.1, "Register Descriptions."

Figure 4-5 shows a timing diagram of the POR sequence.

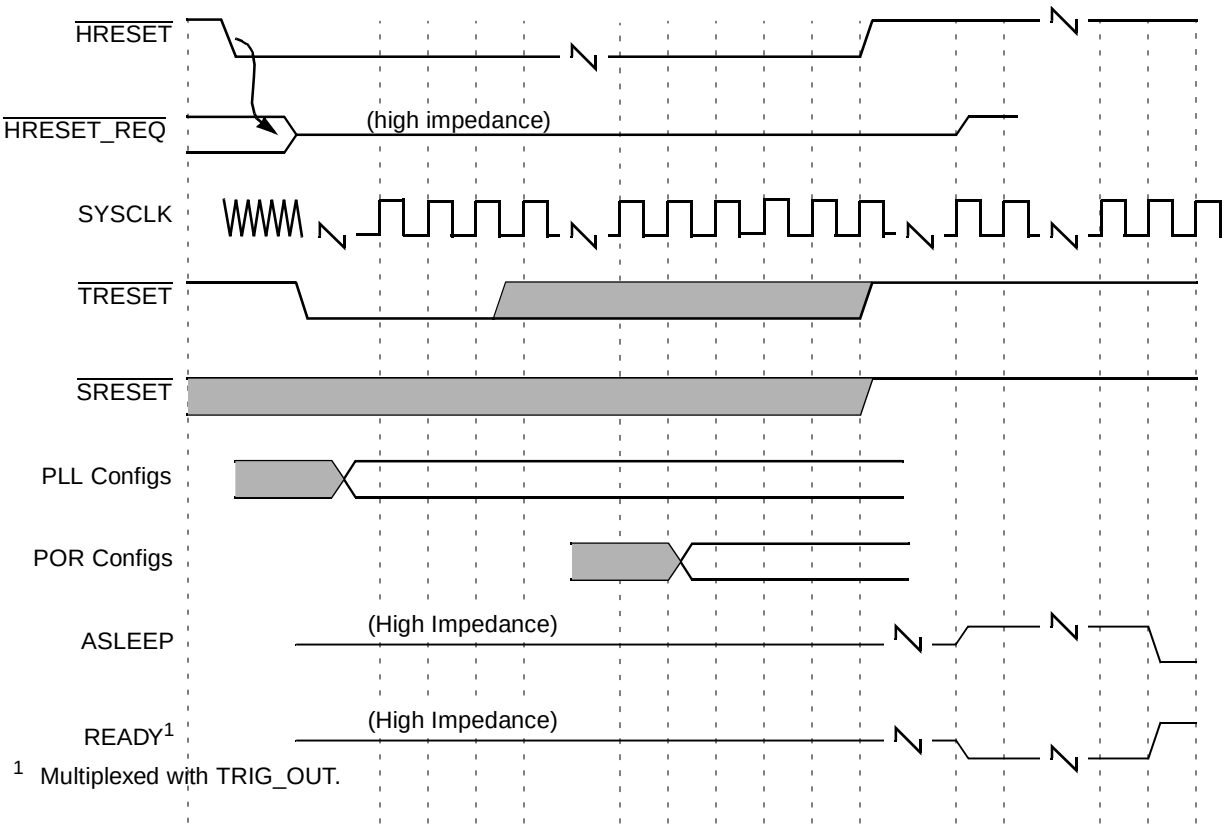


Figure 4-5. Power-On Reset Sequence

4.4.3 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during the assertion of $\overline{\text{HRESET}}$. The values of all these signals are sampled into registers while $\overline{\text{HRESET}}$ is asserted. These inputs are to be pulled high or low by external resistors. During $\overline{\text{HRESET}}$, all other signal drivers connected to these signals must be in the high-impedance state.

Most 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. Other POR configuration signals do not use pull-ups and therefore must be pulled high or low. Refer to the *MPC8572E Integrated Processor Hardware Specifications* for proper resistor values to be used for pulling POR configuration signals high or low.

This section describes the functions and modes configured by POR configuration signals. Note that many reset configuration settings are accessible to software through the following read-only memory-mapped registers described in Chapter 23, "Global Utilities:"

- POR PLL status register (PORPLLSR)
- POR boot mode status register (PORBMSR)
- POR I/O impedance status and control register (PORIMPSCR)
- POR device status registers (PORDEVSR and PORDEVSR2)

- POR debug mode status register (PORDBGMSR)
- General-purpose POR configuration register (GPPORCR)—Reports the value on LAD[0:31] during POR (can be used to external system configuration)

NOTE

In the following tables, the binary value 0b0 represents a signal pulled down to GND and a value of 0b1 represents a signal pulled up to that signal's corresponding V_{DD} voltage level, regardless of the sense of the functional signal name on the signal.

4.4.3.1 System PLL Ratio

The system PLL inputs, shown in Table 4-9, establish the clock ratio between the SYSCLK input and the platform clock used by the MPC8572E. The platform clock, also called the CCB clock, drives the L2 cache, and the e500 core complex bus (CCB). There is no default value for this PLL ratio; these signals must be pulled to the desired values. See Section 4.4.4.3.1, "Minimum Frequency Requirements," for optimal selection of this ratio with regard to available high-speed interface widths and frequencies. Note that the encoded values latched on these signals during POR--and not the actual values on the pins--are accessible in the PORPLLSR (POR PLL status register), as described in Section 23.4.1.1, "POR PLL Status Register (PORPLLSR)."

Table 4-9. CCB Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (Binary)	CCB Clock : SYSCLK Ratio
LA[29:31] No Default	cfg_sys_pll[0:2]	000	4 : 1
		001	5 : 1
		010	6 : 1
		011	8 : 1
		100	10 : 1
		101	12 : 1
		110	Reserved
		111	Reserved

4.4.3.2 DDR PLL Ratio

The DDR PLL inputs, shown in Table 4-10, establish the clock ratio between the DDRCLK input and the DDR complex clock. The DDR complex clock drives the DDR data rate, which is 2 times the rate at which commands are issued on the DDR interface. This DDR complex clock domain is asynchronous to the platform clock or CCB clock domain, and is sourced from a separate PLL than the rest of the platform, unless the DDR PLL encoding for synchronous mode of operation is selected. When synchronous mode is selected, the DDR complex is driven by the CCB clock, which becomes the DDR data rate. There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the encoded values latched on these signals during POR--and not the actual values on the pins--are accessible in the PORPLLSR (POR PLL status register), as described in Section 23.4.1.1, "POR PLL Status Register (PORPLLSR)."

Table 4-10. DDR Complex Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (Binary)	DDR Complex Clock : DDRCLK Ratio
TSEC_1588_CLK_OUT, TSEC_1588_PULSE_OUT1, TSEC_1588_PULSE_OUT2 No Default	cfg_ddr_pll[0:2]	000	3 : 1
		001	4 : 1
		010	6 : 1
		011	8 : 1
		100	10 : 1
		101	12 : 1
		110	14 : 1
		111	Synchronous mode

4.4.3.3 e500 Core PLL Ratios

Table 4-11 and Table 4-12 describe the e500 core clock PLL inputs that program the PLL's for core0 and core1 and establish the ratio between e500 core0 and core1 clock and the e500 core complex bus (CCB) clock. There are no default values for these PLL ratios; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible through the memory-mapped PORPLLSR, as described in Section 23.4.1.1, "POR PLL Status Register (PORPLLSR)," and also in the corresponding e500 core HID1 register, as described in Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

Table 4-11. e500 Core0 Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e500 Core0 : CCB ClockRatio
LBCTL, LALE, LGPL2/$\overline{\text{LOE}}$/$\overline{\text{LFRE}}$ No Default	cfg_core0_pll[0:2]	000	Reserved
		001	Reserved
		010	Reserved
		011	3 : 2 (1.5 : 1)
		100	2 : 1
		101	5 : 2 (2.5:1)
		110	3 : 1
		111	7 : 2 (3.5 : 1)

Table 4-12. e500 Core1 Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e500 Core1 : CCB ClockRatio
LWE[0]/LBS[0]/LFWE, UART_SOUT[1], READY_P1 No Default	cfg_core1_pll[0:2]	000	Reserved
		001	Reserved
		010	Reserved
		011	3 : 2 (1.5 : 1)
		100	2 : 1
		101	5 : 2 (2.5:1)
		110	3 : 1
		111	7 : 2 (3.5 : 1)

4.4.3.4 Boot ROM Location

The MPC8572E defines the default boot ROM address range to be 8 Mbytes at address 0x0_FF80_0000 to 0x0_FFFF_FFFF. However, which peripheral interface handles these boot ROM accesses can be selected at power on.

The boot ROM location inputs, shown in Table 4-13, select the physical location of boot ROM. Accesses to the boot vector and the default boot ROM region of the local address map are directed to the interface specified by these inputs.

Table 4-13. Boot ROM Location

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[6:4], TSEC1_TX_ER Default (1111)	cfg_rom_loc[0:3]	0000	PCI Express 1
		0001	PCI Express 2
		0010	Serial RapidIO
		0011	Reserved
		0100	DDR controller 1
		0101	DDR controller 2
		0110	DDR Interleaved
		0111	PCI Express 3
		1000	Local bus FCM--8-bit NAND Flash small page
		1001	Reserved
		1010	Local bus FCM--8-bit NAND Flash large page
		1011	Reserved
		1100	Reserved
		1101	Local bus GPCM—8-bit ROM
		1110	Local bus GPCM—16-bit ROM
		1111	Local Bus GPCM—32-bit ROM (default)

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 23.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

See Section 2.1, “Local Memory Map Overview and Example,” for an example memory map that relies on the default boot ROM values. Also, see Section 4.3.1.3.1, “Boot Page Translation Register (BPTR),” for information on translation of the boot page.

4.4.3.5 Host/Agent Configuration

The host/agent reset configuration inputs, shown in Table 4-14, configure the MPC8572E to act as a host or as an agent of a master on another interface. In host mode, the MPC8572E is immediately enabled to master transactions to the RapidIO and PCI Express interfaces. If the MPC8572E is an agent on the RapidIO or PCI Express interfaces, then the MPC8572E is disabled from mastering transactions on that interface until the external host enables it to do so. The external host does this by setting the control registers of the MPC8572E’s interfaces appropriately. See details in the PCI Express and RapidIO programming models described in Chapter 21, “PCI Express Interface Controller,” and Chapter 20, “Serial RapidIO Interface,” respectively.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 23.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Table 4-14. Host/Agent Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{LWE}}$[1:3]/$\overline{\text{LBS}}$[1:3] Default (111)	cfg_host_agt[0:2]	000	PCI Express 1: endpoint PCI Express 2: endpoint PCI Express 3: endpoint Serial RapidIO: agent
		001	PCI Express 1: endpoint PCI Express 2: root complex PCI Express 3: root complex Serial RapidIO: host
		010	PCI Express 1: root complex PCI Express 2: endpoint PCI Express 3: root complex Serial RapidIO: agent
		011	PCI Express 1: root complex PCI Express 2: root complex PCI Express 3: endpoint Serial RapidIO: host
		100	PCI Express 1: endpoint PCI Express 2: endpoint PCI Express 3: root complex Serial RapidIO: agent
		101	PCI Express 1: endpoint PCI Express 2: root complex PCI Express 3: endpoint Serial RapidIO: host
		110	PCI Express 1: root complex PCI Express 2: endpoint PCI Express 3: endpoint Serial RapidIO: agent
		111	PCI Express 1: root complex PCI Express 2: root complex PCI Express 3: root complex Serial RapidIO: host

4.4.3.6 I/O Port Selection

The MPC8572E can be configured with different I/O ports active. Table 4-15 shows the configuration of I/O ports and bit rates (and required reference clocks) that are possible for the Serial RapidIO and PCI Express interfaces.

Table 4-15. I/O Port Selection

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[3:1], TSEC2_TX_ER Default (1111)	cfg_IO_ports[0:3]	0000	Reserved
		0001	Reserved
		0010	PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3]
		0011	PCI Express 1 (x4) (2.5 Gbps), PCI Express 2 (x4) (2.5 Gbps) *100-MHz reference clock* PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] PCI Express 2: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7]
		0100	Reserved
		0101	Reserved
		0110	Serial RapidIO x4 (2.5 Gbps) *100 MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7]
		0111	PCI Express 1 (x4), PCI Express 2 (x2), PCI Express 3 (x2) *100-MHz reference clock* PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] PCI Express 2: RX lane[0:1] -> SD1_RX[4:5] TX lane[0:1] -> SD1_TX[4:5] PCI Express 3: RX lane[0:1] -> SD1_RX[6:7] TX lane[0:1] -> SD1_TX[6:7]

Table 4-15. I/O Port Selection (continued)

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[3:1], TSEC2_TX_ER Default (1111) *(cont.)*	cfg_IO_ports[0:3]	1000	Reserved
		1001	Reserved
		1010	Reserved
		1011	Serial RapidIO x4 (2.5 Gbps), PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* Serial RapidIO: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3]
		1100	Serial RapidIO x4 (1.25 Gbps), PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* Serial RapidIO: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3]
		1101	Serial RapidIO x4 (3.125 Gbps) *125-MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7]
		1110	Serial RapidIO x4 (1.25 Gbps) *100-MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7]
		1111	PCI Express 1 (x8) (2.5 Gbps) *100-MHz reference clock* RX lane[0:7] -> SD1_RX[0:7] TX lane[0:7] -> SD1_TX[0:7]

4.4.3.7 CPU Boot Configuration

The CPU boot configuration inputs, shown in Table 4-16, specify the boot configuration mode. If LA[27] is sampled low at reset, e500 core 0 is prevented from fetching boot code until configuration by an external master is complete. Similarly, EC3_MDC can be used to gate off e500 core 1 from fetching boot code. The external master frees e500 core 0 and/or core 1 to boot by setting EEBPCR[CPU0_EN] and/or EEBPCR[CPU1_EN], for core 0 and 1 respectively, in the ECM CCB port configuration register (EEBPCR). See Section 8.2.1.2, “ECM CCB Port Configuration Register (EEBPCR),” for more information.

Note that the value latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 23.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Note also that the value latched on these signals during POR affects all of the PCI Express interfaces' configuration ready mode. For details, see Section 21.3.10.18, "Configuration Ready Register—0x4B0."

Table 4-16. CPU Boot Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LA[27], EC3_MDC Default (11)	cfg_cpu0_boot, cfg_cpu1_boot	00	CPU boot holdoff mode for both cores. The e500 cores are prevented from booting until configured by an external master.
		01	e500 core 1 is allowed to boot without waiting for configuration by an external master, while e500 core 0 is prevented from booting until configured by an external master or the other core.
		10	e500 core 0 is allowed to boot without waiting for configuration by an external master, while e500 core 1 is prevented from booting until configured by an external master or the other core.
		11	Both e500 cores are allowed to boot without waiting for configuration by an external master (default).

4.4.3.8 Boot Sequencer Configuration

The boot sequencer configuration options, shown in Table 4-17, allow the boot sequencer to load configuration data from the serial ROM located on the I^2C1 port before the host tries to configure the MPC8572E. These options also specify normal or extended I^2C addressing modes. See Section 12.4.5, "Boot Sequencer Mode," for more information on the boot sequencer.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 23.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-17. Boot Sequencer Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LGPL3/$\overline{\text{LFWP}}$, LGPL5 Default (11)	cfg_boot_seq[0:1]	00	Reserved
		01	Normal I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C1 interface. A valid ROM must be present.
		10	Extended I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C1 interface. A valid ROM must be present.
		11	Boot sequencer is disabled. No I^2C ROM is accessed (default).

NOTE

When the boot sequencer is enabled, both processor cores are held in reset and thus prevented from fetching boot code until the boot sequencer has completed its task, regardless of the state of the CPU boot configuration signals described in Section 4.4.3.7, "CPU Boot Configuration."

4.4.3.9 DDR SDRAM Type

DDR2 requires a different voltage level from DDR3. Table 4-18 describes the configuration of the DDR SDRAM type.

Table 4-18. DDR DRAM Type

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[1] Default (1)	cfg_dram_type	0	DDR2 1.8 V, CKE low at reset
		1	DDR3 1.5 V, CKE low at reset (default)

4.4.3.10 FEC Configuration

The FEC shares parallel mode pins with both eTSEC3 and eTSEC4. So, when the FEC is enabled, eTSEC3 and eTSEC4 are powered down if not configured to be in SGMII mode. Note that the only mode of operation available when the FEC is enabled is MII mode. Table 4-19 shows how to enable the FEC.

Note that the value latched on this signal during POR is accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 23.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Table 4-19. FEC Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{DMA1\_DDONE}}$[1] Default (1)	cfg_fec	0	FEC is enabled and eTSEC3 and eTSEC4 are only available in SGMII mode.
		1	FEC is disabled and eTSEC3 and eTSEC4 are able to use their respective parallel interface pins (default).

4.4.3.11 eTSEC SGMII Mode

The eTSEC SGMII mode inputs, shown in Table 4-20, Table 4-21, Table 4-22, and Table 4-23, selects SGMII mode versus parallel mode for enhanced three-speed Ethernet controller (eTSEC) interfaces 1-4. Note that the value latched on these signals during POR are accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR).”

If the FEC is enabled (see Section 4.4.3.10, “FEC Configuration”), eTSEC3 and eTSEC4 are powered down if not configured to be in SGMII mode. Also, if a particular eTSEC is configured to run in SGMII mode, no other POR configuration inputs are pertinent for configuring that given eTSEC interface. Note that when operating in SGMII mode, the parallel interface pins normally used for a given eTSEC interface are not used by that eTSEC, but rather the corresponding SGMII SerDes lane (SD2_TX[n]/RX[n] and $\overline{\text{SD2\_TX}}$[n]/$\overline{\text{RX}}$[n]) is used.

Table 4-20. eTSEC1 SGMII Mode Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LA[28] Default (1)	cfg_sgmii1	0	eTSEC1 Ethernet interface operates in SGMII mode and uses SGMII SerDes lane 0 pins.
		1	eTSEC1 Ethernet interface operates in standard parallel interface mode and uses the TSEC1_* pins (default).

Table 4-21. eTSEC2 SGMII Mode Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
LGPL1/LFALE Default (1)	cfg_sgmii2	0	eTSEC2 Ethernet interface operates in SGMII mode and uses SGMII SerDes lane 1 pins.
		1	eTSEC2 Ethernet interface operates in standard parallel interface mode and uses the TSEC2_* pins (default).

Table 4-22. eTSEC3 SGMII Mode Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[3] Default (1)	cfg_sgmii3	0	eTSEC3 Ethernet interface operates in SGMII mode and uses SGMII SerDes lane 2 pins.
		1	eTSEC3 Ethernet interface operates in standard parallel interface mode and uses the TSEC3_* pins provided the FEC is not enabled. If the FEC is enabled, eTSEC3 is powered down (default).

Table 4-23. eTSEC4 SGMII Mode Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
UART_SOUT[0] Default (1)	cfg_sgmii4	0	eTSEC4 Ethernet interface operates in SGMII mode and uses SGMII SerDes lane 3 pins.
		1	eTSEC4 Ethernet interfaces operates in standard parallel interface mode and uses the TSEC4_* pins provided the FEC is not enabled. If the FEC is enabled, eTSEC4 is powered down (default).

4.4.3.12 SGMII SerDes Reference Clock Configuration

As shown in Table 4-24, two options are available for the frequency of the input SGMII SerDes reference clock--either a 100MHz or 125MHz LVDS differential clock. This one clock is applied to an internal PLL whose output creates the clock used by all four SGMII SerDes lanes. The result is always a 1.25Gbaud transmission/receive rate on each lane. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-24. SGMII SerDes Reference Clock Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC_1588_TRIG_OUT Default (1)	cfg_srds_sgmii_refclk	0	SGMII SerDes expects a 125MHz reference clock frequency.
		1	SGMII SerDes expects a 100MHz reference clock frequency (default).

4.4.3.13 eTSEC1 and eTSEC2 Width

The eTSEC width input, shown in Table 4-25, selects standard versus reduced width for three-speed Ethernet controller interfaces 1 and 2 operating in parallel mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, "POR Device Status Register (PORDEVSR)."

This input only affects operation of eTSEC1 or eTSEC2 if the particular interface is not configured to operate in SGMII mode. This input does not affect the width of the FIFO interface for eTSEC2, which is always an 8-bit FIFO interface. For eTSEC1, however, this bit controls whether the interface is operating as a 16-bit FIFO or an 8-bit FIFO. Note that since the full-width FIFO interface (16-bits) requires the parallel pins for both eTSEC1 and eTSEC2 controllers, eTSEC2 is unavailable and powered down if not in SGMII mode when eTSEC1 is in 16-bit FIFO mode.

Table 4-25. eTSEC1/eTSEC2 Width Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
EC1_MDC Default (1)	cfg_tsec_1_reduce	0	eTSEC1 and eTSEC2 Ethernet interfaces operate in reduced pin mode (either RTBI, RGMII, RMII, or 8-bit FIFO mode) if not configured to operate in SGMII mode.
		1	eTSEC1 and eTSEC2 Ethernet interfaces operate in their standard width TBI, GMII, MII mode if not configured to operate in SGMII mode. Or if eTSEC1 is in FIFO mode and not SGMII mode, it operates as a 16-bit FIFO. eTSEC2 FIFO width is 8 bits, if available (eTSEC1 must not be configured as a 16-bit FIFO), regardless of this configuration setting (default).

NOTE

If eTSEC1 or eTSEC2 is configured to operate in parallel mode, as opposed to SGMII mode (see Section 4.4.3.11, "eTSEC SGMII Mode"), then the width of the interface is controlled by this one configuration input, and the particular parallel protocol (TBI, GMII, MII, or FIFO) used is separately controlled with the other configuration inputs described in Section 4.4.3.15, "eTSEC1 Protocol," and/or Section 4.4.3.16, "eTSEC2 Protocol."

4.4.3.14 eTSEC3 and eTSEC4 Width

The eTSEC3 width input, shown in Table 4-26, selects standard versus reduced width for three-speed Ethernet controller interfaces 3 and 4 operating in parallel mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, "POR Device Status Register (PORDEVSR)."

This input only affects operation of eTSEC3 or eTSEC4 if the particular interface is not configured to operate in SGMII mode (see Section 4.4.3.11, "eTSEC SGMII Mode") and the FEC is disabled (see Section 4.4.3.10, "FEC Configuration"). The value of this configuration setting does not affect the width of the FIFO interface on eTSEC3, which is always 8 bits.

Because eTSEC3 and eTSEC4 share parallel mode pins, eTSEC4 in a parallel mode is only available as a reduced-pin-width interface and requires that eTSEC3 operates in reduced-pin-width parallel mode or SGMII mode. However, eTSEC4 is always available to operate in SGMII mode irrespective of the configuration of eTSEC3. Also, if eTSEC3 is in standard-pin-width parallel mode and eTSEC4 is not in SGMII mode, eTSEC4 is powered down.

Table 4-26. eTSEC3/eTSEC4 Width Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[2] Default (1)	cfg_tsec_3_reduce	0	eTSEC3 and eTSEC4 Ethernet interfaces operate in reduced pin mode (either RTBI, RGMII, RMII) if not configured to operate in SGMII mode and the FEC is disabled.
		1	eTSEC3 Ethernet interface operates in standard width TBI, GMII, MII, or 8-bit FIFO mode if not configured to operate in SGMII mode and the FEC is disabled. eTSEC 4 is unavailable unless configured to operate in SGMII mode (default).

NOTE

If eTSEC3 or eTSEC4 is configured to operate in parallel mode, as opposed to SGMII mode (see Section 4.4.3.11, "eTSEC SGMII Mode"), and the FEC is disabled (see Section 4.4.3.10, "FEC Configuration"), then the width of the interface is controlled by this one configuration input, and the particular parallel protocol (TBI, GMII, MII, or FIFO) used is separately controlled with other configuration inputs described in Section 4.4.3.17, "eTSEC3 Protocol," and Section 4.4.3.18, "eTSEC4 Protocol."

4.4.3.15 eTSEC1 Protocol

The eTSEC1 protocol inputs, shown in Table 4-27, select the protocol (FIFO, MII, GMII or TBI) used by the eTSEC1 controller operating in parallel mode. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, "POR Device Status Register (PORDEVSR)."

This input only affects operation of eTSEC1 if it is not configured to operate in SGMII mode.

Table 4-27. eTSEC1 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[0], TSEC1_TXD[7] Default (11)	cfg_tsec1_prtcl[0:1]	00	The eTSEC1 controller operates using 16-bit FIFO protocol if not configured to operate in SGMII mode (or 8-bit FIFO protocol if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”).
		01	The eTSEC1 controller operates using the MII protocol if not configured to operate in SGMII mode (or RMII if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”).
		10	The eTSEC1 controller operates using the GMII protocol if not configured to operate in SGMII mode (or RGMII if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”).
		11	The eTSEC1 controller operates using the TBI protocol if not configured to operate in SGMII mode (or RTBI if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”) (default).

4.4.3.16 eTSEC2 Protocol

The eTSEC2 protocol inputs, shown in Table 4-28, select the protocol (FIFO, MII, GMII or TBI) used by the eTSEC2 controller operating in parallel mode. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR).”

This input only affects operation of eTSEC2 if it is not configured to operate in SGMII mode.

Table 4-28. eTSEC2 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[0], TSEC2_TXD[7] Default (11)	cfg_tsec2_prtcl[0:1]	00	The eTSEC2 controller operates using 8-bit FIFO protocol if not configured to operate in SGMII mode.
		01	The eTSEC2 controller operates using the MII protocol if not configured to operate in SGMII mode (or RMII if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”)
		10	The eTSEC2 controller operates using the GMII protocol if not configured to operate in SGMII mode (or RGMII if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”).
		11	The eTSEC2 controller operates using the TBI protocol if not configured to operate in SGMII mode (or RTBI if configured in reduced mode as described in Section 4.4.3.13, “eTSEC1 and eTSEC2 Width”) (default).

4.4.3.17 eTSEC3 Protocol

The eTSEC3 protocol inputs, shown in Table 4-29, select the protocol (FIFO, MII, GMII or TBI) used by the eTSEC3 controller operating in parallel mode. Note that the value latched on these signals during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR).”

This input only affects operation of eTSEC3 if it is not configured to operate in SGMII mode (see Section 4.4.3.11, “eTSEC SGMII Mode”) and the FEC is disabled (see Section 4.4.3.10, “FEC Configuration”).

Table 4-29. eTSEC3 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC3_TXD[0], TSEC3_TXD[1] Default (11)	cfg_tsec3_prtcl[0:1]	00	The eTSEC3 controller operates using 8-bit FIFO protocol if not configured to operate in SGMII mode and the FEC is disabled.
		01	The eTSEC3 controller operates using the MII protocol if not configured to operate in SGMII mode and FEC is disabled (or RMII if configured in reduced mode as described in Section 4.4.3.14, “eTSEC3 and eTSEC4 Width”)
		10	The eTSEC3 controller operates using the GMII protocol if not configured to operate in SGMII mode and FEC is disabled (or RGMII if configured in reduced mode as described in Section 4.4.3.14, “eTSEC3 and eTSEC4 Width”).
		11	The eTSEC3 controller operates using the TBI protocol if not configured to operate in SGMII mode and FEC is disabled (or RTBI if configured in reduced mode as described in Section 4.4.3.14, “eTSEC3 and eTSEC4 Width”) (default).

4.4.3.18 eTSEC4 Protocol

The eTSEC4 protocol input, shown in Table 4-30, selects the protocol (RMII, RGMII or RTBI) used by the eTSEC4 controller operating in parallel mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR).”

This input only affects operation of eTSEC4 if it is not configured to operate in SGMII mode (see Section 4.4.3.11, “eTSEC SGMII Mode”) and the FEC is disabled (see Section 4.4.3.10, “FEC Configuration”).

Table 4-30. eTSEC4 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC4_TXD[0], TSEC4_TXD[1] Default (11)	cfg_tsec4_prtcl[0:1]	00	Reserved
		01	The eTSEC4 controller operates using the RMII protocol if not configured to operate in SGMII mode and FEC is disabled.
		10	The eTSEC4 controller operates using the RGMII protocol if not configured to operate in SGMII mode and FEC is disabled.
		11	The eTSEC4 controller operates using the RTBI protocol if not configured to operate in SGMII mode and FEC is disabled (default).

4.4.3.19 RapidIO Device ID

The RapidIO device ID inputs, shown in Table 4-31, specify the 3 lower-order bits of the device ID of the MPC8572E as used by RapidIO hosts. Note that the 5 high-order RapidIO device ID bits cannot be set via POR configuration inputs. They may be initialized via the boot sequencer or by the processor from boot ROM or by the RapidIO discovery process.

Table 4-31. RapidIO Device ID

Functional Signals	Reset Configuration Name	Meaning
TSEC2_TXD[2]	cfg_device_ID[5]	Device ID used for RapidIO hosts
TSEC2_TXD[3]	cfg_device_ID[6]	Device ID used for RapidIO hosts
TSEC2_TXD[4]	cfg_device_ID[7]	Device ID used for RapidIO hosts

If configured as a RapidIO host, the five upper-order device ID bits default to zeros. If configured as a RapidIO agent, the seven upper-order device ID bits default to ones and only cfg_device_ID[7] is used to define the device ID. Unconnected cfg_device_ID[*n*] inputs default to 1s regardless of the host/agent mode configuration.

Note that the value latched on this signal at POR is accessible through the memory-mapped PORDEVSR described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR),” as well as the memory-mapped BDIDCSR described in Section 20.6.1.12, “Base Device ID Command and Status Register (BDIDCSR).”

4.4.3.20 RapidIO System Size

The RapidIO common transport specification defines two system sizes. The large size uses 16-bit source and destination IDs allowing for 65,536 devices, while the small size uses 8-bit source and destination IDs, supporting 256 devices.

The RapidIO system size configuration shown in Table 4-32 specifies which system size is to be used by the RapidIO controller on the MPC8572E.

Note that the value latched on this signal at POR is accessible through the memory-mapped PORDEVSR described in Section 23.4.1.4, “POR Device Status Register (PORDEVSR),” as well as the memory-mapped PEFCAR described in Section 20.6.1.5, “Processing Element Features Capability Register (PEFCAR).”

Table 4-32. RapidIO System Size

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LGPL0 Default (1)	cfg_rio_sys_size	0	Large system size (up to 65,536 devices)
		1	Small system size (up to 256 devices) (default)

4.4.3.21 Memory Debug Configuration

The memory debug configuration inputs, shown in Table 4-33, select which debug outputs (DDR controller 1, DDR controller 2, or Local Bus Controller) are driven onto the MSRCID and MDVAL debug signals. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 23.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-33. Memory Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{DMA2\_DACK}}$[0], $\overline{\text{DMA1\_DDONE}}$[0] Default (11)	cfg_mem_debug[0:1]	00	Debug information from the local bus controller (LBC) is driven on the MSRCID and MDVAL signals
		01	Reserved
		10	Debug information from the DDR SDRAM controller 1 is driven on the MSRCID and MDVAL signals.
		11	Debug information from the DDR SDRAM controller 2 is driven on the MSRCID and MDVAL signals (default).

4.4.3.22 DDR Debug Configuration

The DDR debug configuration input, shown in Table 4-34, enables a DDR memory controller debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto the ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8572E in this mode.

Table 4-34. DDR Debug Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{DMA2\_}}$ $\overline{\text{DDONE}}$[0] Default (1)	cfg_ddr_debug	0	Debug information is driven on the ECC pins instead of normal ECC I/O. ECC signals from memory devices must be disconnected.
		1	Debug information is not driven on ECC pins. ECC pins function in their normal mode (default).

Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 23.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

4.4.3.23 General-Purpose POR Configuration

The LBC address/data bus inputs, shown in Table 4-35, configure the value of the general-purpose POR configuration register defined in Section 23.4.1.7, "General-Purpose POR Configuration Register (GPPORCR)." This register is intended to facilitate POR configuration of user systems. A value placed on LAD[0:31] during POR is captured and stored (read only) in the GPPORCR. Software can then 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.

Table 4-35. General-Purpose POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
LAD[0:31] No default	cfg_gpinput[0:31]	—	General-purpose POR configuration vector to be placed in GPPORCR

4.4.3.24 eLBC FCM ECC Configuration

The eLBC FCM ECC configuration input, shown in Table 4-36, determines the default configuration of the FCM ECC. Note that the value latched on this signal during POR is accessible through the PORDEVSR2, described in Section 23.4.1.6, "POR Device Status Register 2 (PORDEVSR2)."

Table 4-36. eLBC FCM ECC Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
MSRCID[0] Default (1)	cfg_fcm_ecc	0	BR*n*[DECC] = 00. Data error checking is disabled; no ECC generation for FCM.
		1	BR*n*[DECC] = 01. Data error checking is enabled, but ECC generation is disabled for FCM on full-page transfers. (default)

4.4.3.25 SerDes1 Enable

The SerDes1 high speed interface may be disabled during power-on reset by pulling low the POR configuration signal cfg_srds1_en, which exists on the external signal TSEC2_TXD[5], and is described in Table 4-37. By default, this signal is sampled high (SerDes1 enabled) unless pulled low externally (disabling SerDes1). Pulling this signal low during POR is logically equivalent to setting DEVDISR bit fields for the PCI Express interfaces and Serial RapidIO. Note that the value latched on this signal during POR is accessible through the PORDEVSR2, described in Section 23.4.1.6, "POR Device Status Register 2 (PORDEVSR2)."

Table 4-37. SerDes1 Enable POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[5] Default (1)	cfg_srds1_en	0	SerDes1 interface is disabled.
		1	SerDes1 interface is enabled. (default)

4.4.3.26 SGMII SerDes Enable

The SGMII SerDes high speed interface may be disabled during power-on reset by pulling low the POR configuration signal cfg_srds_sgmii_en, which exists on the external signal $\overline{\text{UART\_RTS}}$[1], and is described in Table 4-38. By default, this signal is sampled high (SGMII SerDes enabled) unless pulled low externally (disabling SGMII SerDes). Note that the value latched on this signal during POR is accessible through the PORDEVSR2, described in Section 23.4.1.6, "POR Device Status Register 2 (PORDEVSR2)."

Table 4-38. SGMII SerDes Enable POR Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{UART\_RTS}}$[1]	cfg_srds_sgmii_en	0	SGMII SerDes interface is disabled.
Default (1)		1	SGMII SerDes interface is enabled. (default)

4.4.3.27 Engineering Use POR Configuration

These POR configuration input may be used in the future to control functionality. It is advised that boards are built with the ability to pull up or pull down this signal. Unless this pin is redefined in the future, the default value for this signal should be logic 1. Note that the value latched on this signal during POR is accessible through the PORDEVSR2, described in Section 23.4.1.6, "POR Device Status Register 2 (PORDEVSR2)."

Table 4-39. Engineering Use POR Config Signal

External Signal	Config Signal
MSRCID[1]	cfg_eng_use[1]

4.4.4 Clocking

The following paragraphs describe the clocking within the MPC8572E device.

4.4.4.1 System Clock and DDR Controller Complex Clock

The MPC8572E takes a single input clock, SYSCLK, as its primary clock source for the e500 cores and all of the devices and interfaces that operate synchronously with the cores. As shown in Figure 4-6, the SYSCLK input (frequency) is multiplied up using a phase lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock). The CCB clock is used by virtually all of the synchronous system logic, including the L2 cache, and other internal blocks such as the DMA and interrupt controller. The CCB clock also feeds each of the PLL's in the e500 cores and the PLL that creates clocks for the local bus memory controller.

The DDR memory controller complex may use the platform clock and thus have operation of both DDR interfaces be synchronous with the platform. Alternately, an independent clock, DDRCLK, may be multiplied up using a separate PLL to create a unique DDR memory controller complex clock. In this case, the DDR complex operates asynchronous with respect to the platform clock.

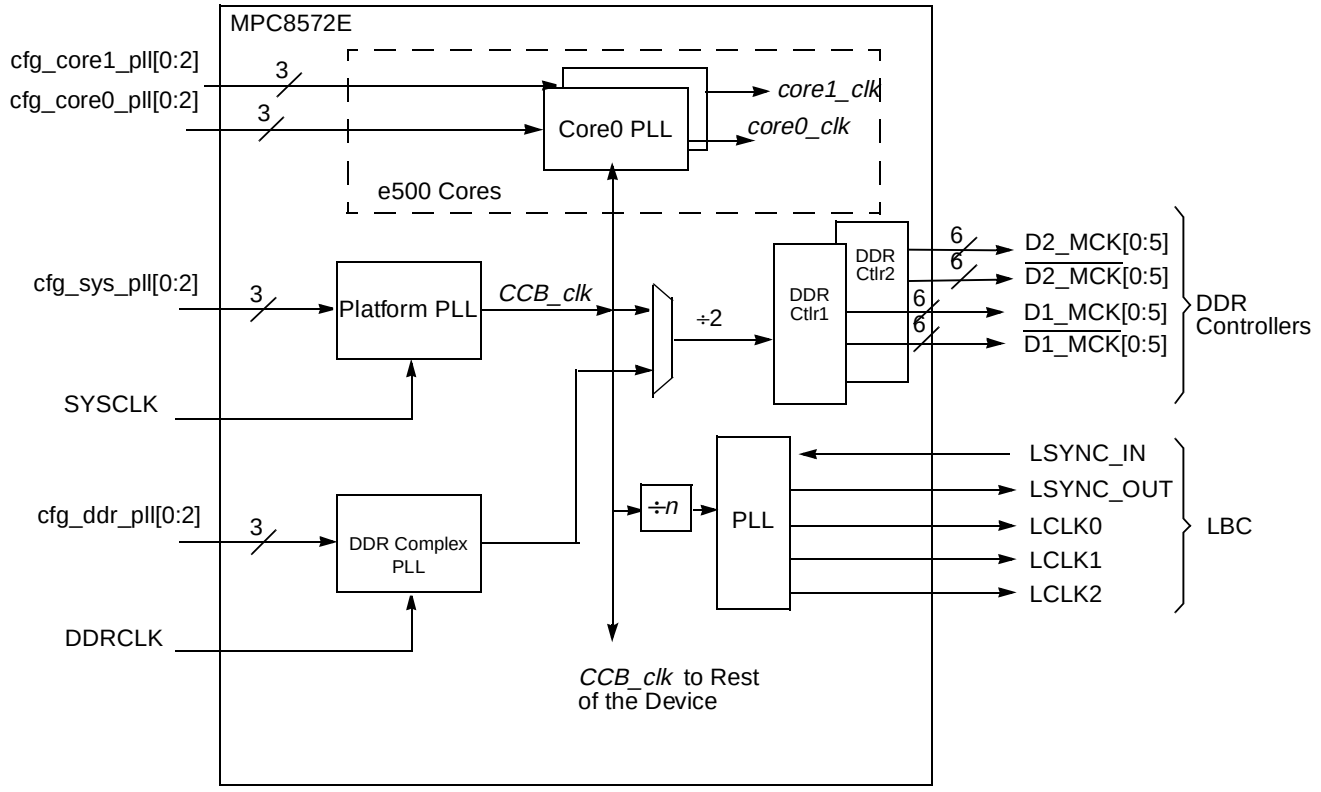


Figure 4-6. Clock Subsystem Block Diagram

4.4.4.2 RapidIO and PCI Express Clocks

Clocks for these high speed interfaces on the MPC8572E are derived from a PLL in the SerDes1 block. This PLL is driven by a reference clock(SD1_REF_CLK/$\overline{\text{SD1\_REF\_CLK}}$) whose input frequency is a function of the protocol and bit rate being used as shown in Table 4-40.

Table 4-40. High Speed Interface 1 Clocking

Interfaces	Bit Rate	Reference Clock Frequency
PCI Express	2.5 Gbps	100 MHz
Serial RapidIO PCI Express	2.5 Gbps 2.5 Gbps	100 MHz
Serial RapidIO	2.5 Gbps	100 MHz
Serial RapidIO	1.25 Gbps	100 MHz
Serial RapidIO PCI Express	1.25 Gbps 2.5 Gbps	100 MHz
Serial RapidIO	3.125 Gbps	125 MHz

4.4.4.3 SGMII Clocks

Clocks for the SGMII high speed interfaces on the MPC8572E are derived from a PLL in the SerDes2 block. This PLL is driven by a reference clock (SD2_REF_CLK/$\overline{\text{SD2\_REF\_CLK}}$) whose input frequency

may be either 100-MHz or 125-MHz to obtain the required 1.25-Gbaud rate for each lane. The configuration information of the speed of the reference clock must be set properly on the POR configuration pin cfg_srds_sgmii_refclk (see Section 4.4.3.12, "SGMII SerDes Reference Clock Configuration").

4.4.4.3.1 Minimum Frequency Requirements

Section 4.4.3.6, "I/O Port Selection," describes various high-speed interface configuration options. Note that the CCB clock frequency must be considered for proper operation of such interfaces as described below.

For proper PCI Express operation, the CCB clock frequency must be greater than:

$$\frac{500\ \text{MHz} \times (\text{PCI Express link width})}{8}$$

See Section 21.1.3.2, "Link Width," for PCI Express interface width details.

For proper serial RapidIO operation, the CCB clock frequency must be greater than:

$$\frac{2 \times (0.80) \times (\text{serial RapidIO interface frequency}) \times (\text{serial RapidIO link width})}{64}$$

See Section 20.4, "1x/4x LP-Serial Signal Descriptions," for serial RapidIO interface width and frequency details.

4.4.4.4 Ethernet Clocks

The Ethernet blocks operate asynchronously with respect to the rest of the device. These blocks use receive and transmit clocks supplied by their respective PHY chips in parallel modes, plus a 125-MHz clock input for gigabit parallel protocols. If all of the Ethernet blocks are configured to run in SGMII mode, then these input clocks are not required to be sourced. Data transfers are synchronized to the CCB clock internally.

4.4.4.5 Real Time Clock

As shown in Figure 4-7, the real time clock (RTC) input can optionally be used to clock the e500 core timer facilities. RTC can also be used (optionally) by the MPC8572E programmable interrupt controller (PIC) global timer facilities. The RTC is separate from the e500 core clocks and is intended to support relatively low frequency timing applications. The RTC frequency range is specified in the *MPC8572E Integrated Processor Hardware Specifications*, but the maximum value should not exceed one-quarter of the CCB Frequency.

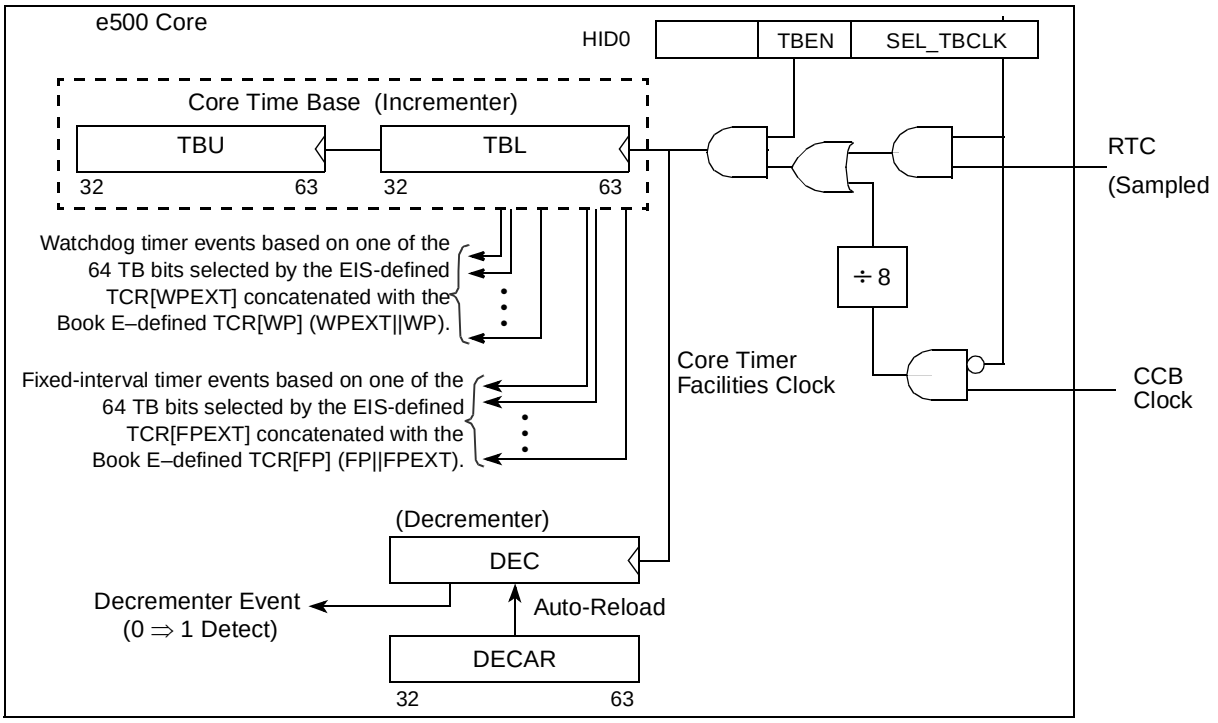
Before being distributed to the core time base, RTC is sampled and synchronized with the CCB clock.

The clock source for the core time base is specified by two fields in each core's HID0 register: time base enable (TBEN), and select time base clock (SEL_TBCLK). If the time base is enabled, (HID0[TBEN] is set), the clock source is determined as follows:

- HID0[SEL_TBCLK] = 0, the time base is updated every 8 CCB clocks
- HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of RTC

The default source of the time base is the CCB clock divided by eight. For more details, see the *PowerPC e500 Core Complex Reference Manual.*

Section 10.3.2.6, "Timer Control Registers (TCRA–TCRB)," provides additional information on the use of the RTC signal to clock the global timers in the PIC unit.



Note: The logic circuits shown depict functional relationships only; they do not represent physical implementation details.

Figure 4-7. RTC and Core Timer Facilities Clocking Options

Part II
e500 Core Complex and L2 Cache

Part II describes the many features of the MPC8572E core processors at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, “Core Complex Overview,” provides an overview of the e500v2 core processor and the L1 caches and MMU that, together with the cores, comprise the core complex.
- Chapter 6, “Core Register Summary,” provides a listing of the e500v2 registers in reference form.
- Chapter 7, “L2 Look-Aside Cache/SRAM,” describes the L2 cache of the MPC8572E. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the PowerPC architecture. This part provides additional information about the Book E architecture as it relates specifically to the e500 core complex and specific details on how its registers are accessed.

The e500 core complex interacts with the L2 cache through the core complex bus (CCB).

Chapter 5
Core Complex Overview

This chapter provides an overview of the e500 microprocessor core as it is implemented on the MPC8572E.

References to e500 are true for both the e500v1 and e500v2.

This chapter includes the following:

- An overview of architecture features as implemented in this core and a summary of the core feature set
- A summary of the instruction pipeline and flow
- An overview of the programming model
- An overview of interrupts and exception handling
- A description of the memory management architecture
- High-level details of the e500 core memory and coherency model
- A brief description of the core complex bus (CCB)
- A summary of the Power Architecture embedded category compatibility and migration from the original version of the PowerPC architecture as it is defined by Apple, IBM, and Motorola (referred to as the AIM version of the PowerPC architecture)

Specific details about the e500 are provided in the *PowerPC™ e500 Core Family Reference Manual* (Freescale Document ID No. E500CORERM). The e500 core provides features that the integrated device may not implement or may implement in a more specific way. These differences are summarized in Section 5.14, “PowerQUICC III Implementation Details.”

5.1 Overview

The e500 core is a low-power implementation of the resources for embedded processors defined by the Power ISA. The core is a 32-bit implementation using the lower words in the 64-bit general-purpose registers (GPRs).

Figure 5-1 is a block diagram of the processor core complex that shows how the functional units operate independently and in parallel. Note that this conceptual diagram does not attempt to show how these features are physically implemented.

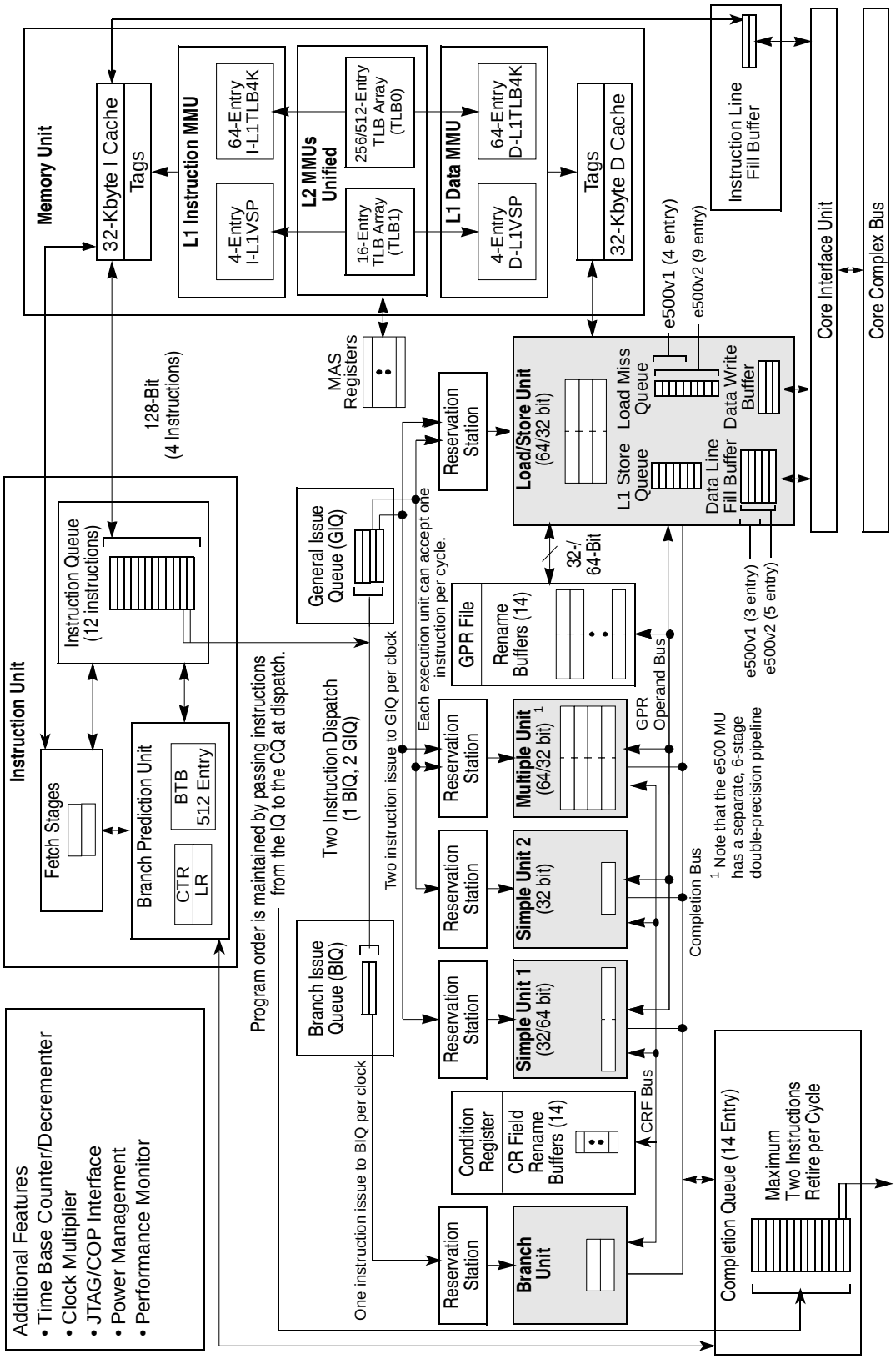


Figure 5-1. e500 Core Complex Block Diagram

The Power Architecture technology defines categories that extend the architecture that can perform computational or system management functions. One of these on the e500 is the signal processing engine (SPE), which includes a suite of vector instructions that use the upper and lower halves of the GPRs as a single two-element operand. Some extensions are defined by Freescale's embedded category implementation standards (EIS).

5.1.1 Upward Compatibility

The e500 provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits, as defined by the architecture. It also provides two-element, 64-bit data types for the SPE and embedded vector floating-point instructions, which include instructions that operate on operands comprised of two 32-bit elements. It also provides a 64-bit scalar data type for use with embedded double-precision floating-point instructions.

The embedded single-precision scalar floating-point instructions use 32-bit single-precision instructions.

NOTE

The SPE (which includes embedded floating-point functionality) is implemented in all PowerQUICC III devices. However, these instructions are not supported in devices subsequent to PowerQUICC III. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or embedded floating-point instructions at the assembly level or that uses SPE intrinsics requires rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

5.1.2 Core Complex Summary

The core complex is a superscalar processor that can issue two instructions and complete two instructions 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. All arithmetic instructions that execute in the core operate on data in the GPRs. Although the GPRs are 64 bits wide, only SPE, DPFP (e500v2 only), and embedded vector floating-point instructions operate on the upper word of the GPRs; the upper 32 bits are not affected by other 32-bit instructions.

The processor core integrates two simple instruction units (SU1, SU2), a multiple-cycle instruction unit (MU), a branch unit (BU), and a load/store unit (LSU).

The LSU and SU2 support 64- and 32-bit instructions.

The ability to execute five instructions in parallel and the use of simple instructions with short execution times yield high efficiency and throughput. Most integer instructions execute in 1 clock cycle. A series of

independent vector floating-point add instructions can be issued and completed with a throughput of one instruction per cycle.

The core complex includes independent on-chip, 32-Kbyte, eight-way set-associative, physically addressed caches for instructions and data. It also includes on-chip first-level instruction and data memory management units (MMUs) and an on-chip second-level unified MMU.

- The first-level MMUs contain two four-entry, fully-associative instruction and data translation lookaside buffer (TLB) arrays that provide support for demand-paged virtual memory address translation and variable-sized pages. They also contain two 64-entry, 4-way set-associative instruction and data TLB arrays that support 4-Kbyte pages. These arrays are maintained entirely by the hardware with a true least-recently-used (LRU) algorithm.
- The second-level MMU contains a 16-entry, fully-associative unified (instruction and data) TLB array that provides support for variable-sized pages. It also contains a unified TLB for 4-Kbyte page size support, as follows:
  - a 256-entry, 2-way set-associative unified TLB for the e500v1
  - a 512-entry, 4-way set-associative unified TLB for the e500v2

  These second-level TLBs are maintained completely by the software.

The core complex allows cache-line-based user-mode locks on the contents in either the instruction or data cache. This provides embedded applications with the capability for locking interrupt routines or other important (time-sensitive) instruction sequences into the instruction cache. It also allows data to be locked into the data cache, which supports deterministic execution time.

The core complex supports a high-speed on-chip internal bus with data tagging called the core complex bus (CCB). The CCB has two general-purpose read data buses, one write data bus, data parity bits, data tag bits, an address bus, and address attribute bits. The processor core complex supports out-of-order reads, in-order writes, and one level of pipelining for addresses with address-retry responses. It can also support single-beat and burst data transfers for memory accesses and memory-mapped I/O operations.

5.2 e500 Processor and System Version Numbers

Table 5-1 lists the revision codes in the processor version register (PVR) and the system version register (SVR). These registers can be accessed as SPRs through the e500 core (see Section 6.5.3, “Processor Version Register (PVR),” and Section 6.5.4, “System Version Register (SVR)”) or as memory-mapped registers defined by the integrated device (see “Section 23.4.1.15, “Processor Version Register (PVR),” and Section 23.4.1.16, “System Version Register (SVR)”).

Table 5-1. Device Revision Level Cross-Reference

MPC8572E Revision	Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.1	3.0	0x8021_0030	0x80E8_0011 for MPC8572E (with security) 0x80E0_0011 for MPC8572 (without security)

5.3 Features

Key features of the e500 are summarized as follows:

- 32-bit architecture
- Additional categories (formerly referred to as APUs)

  Branch target buffer (BTB) locking is specific to the e500. BTB locking gives the user the ability to lock, unlock, and invalidate BTB entries; further information is provided in Table 5-5. The EIS (see *EREF: a Reference for Freescale Book E and the e500 Core*) defines the following:

  — Integer select. This instruction is now part of the Power Architecture technology base category.

  — Performance monitor. 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 exception. 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**). See Section 5.12, “Performance Monitoring.”

  — Cache locking. Allows instructions and data to be locked into their respective caches on a cache block basis. Locking is performed by a set of touch and lock set instructions. This functionality can be enabled for user mode by setting MSR[UCLE]. The feature also provides resources for detecting and handling overlocking conditions.

  — Machine check. The machine check interrupt is treated as a separate level of interrupt. It uses its own save and restore registers (MCSRR0 and MCSRR1) and Return from Machine Check Interrupt (**rfmci**) instruction. See Section 5.8, “Interrupts and Exception Handling.”

  — Single-precision embedded scalar and vector floating-point instructions, listed in Table 5-4.

  — Signal processing engine (SPE). Note that the SPE is not a separate unit; SPE computational and logical instructions are executed in the simple and multiple-cycle units used by all other computational and logical instructions, and 64-bit loads and stores are executed in the common LSU. Figure 5-1 shows how execution logic for SU1, the MU, and the LSU is replicated to support operations on the upper halves of the GPRs.

NOTE

The SPE APU and the two single-precision floating-point APUs were combined in the original implementation of the e500 v1, as shown in Figure 5-2.

	Vector and Floating-Point APUs	e500 v1	e500 v2
Original SPE Definition	SPE vector instructions **ev**…	√	√
	Vector single-precision floating-point **evfs**…	√	√
	Scalar single-precision floating-point **efs**…	√	√
	Scalar double-precision floating-point **efd**…	—	√

Figure 5-2. Vector and Floating-Point APUs

- The e500 register set is modified as follows:
  - GPRs are widened to 64 bits to support 64-bit load, store, and merge operations. Note that the upper 32 bits are affected only by 64-bit instructions.
  - A 64-bit accumulator (ACC) has been added.
  - The signal processing and embedded floating-point status and control register (SPEFSCR) provides interrupt control and status for SPE and embedded floating-point instructions.

  These registers are shown in Figure 5-7. SPE instructions are grouped as follows:
  - Single-cycle integer add and subtract with the same latencies for SPE operations as for the 32-bit equivalent
  - Single-cycle logical operations
  - Single-cycle shift and rotates
  - Four-cycle integer pipelined multiplies
  - 4-, 11-, 19-, and 35-cycle integer divides
  - If **r**A or **r**B is zero, a floating-point divide takes 4 cycles; all other cases take 29 cycles.
  - Four-cycle SIMD pipelined multiply-accumulate (MAC)
  - 64-bit accumulator for no-stall MAC operations
  - 64-bit loads and stores
  - 64-bit merge instructions
- Cache structure—Separate 32-Kbyte, 32-byte line, 8-way set-associative level 1 instruction and data caches
  - 1.5-cycle cache array access, 3-cycle load-to-use latency
  - Pseudo-LRU (PLRU) replacement algorithm
  - Copy-back data cache that can function as a write-through cache on a page-by-page basis
  - Supports all embedded category memory coherency modes
  - Supports EIS-defined cache-locking instructions, as listed in Table 5-3
- Dual-issue superscalar control
  - Two-instructions-per-clock peak issue rate
  - Precise exception handling
- Decode unit
  - 12-entry instruction queue (IQ)
  - Full hardware detection of interlocks
  - Decodes as many as two instructions per cycle
  - Decode serialization control
  - Register dependency resolution and renaming
- Branch prediction unit (BPU)
  - Dynamic branch prediction using a 512-entry, 4-way set-associative branch target buffer (BTB) supported by the e500 BTB instructions listed in Table 5-5
  - Branch prediction is handled in the fetch stages.

- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue (CQ)
  - In-order retirement of as many as two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts, mispredicted branches, and context-synchronizing instructions
- Issue queues
  - Two-entry branch instruction issue queue (BIQ)
  - Four-entry general instruction issue queue (GIQ)
- Branch unit—The branch unit (BU) is an execution unit and is distinct from the BPU. It executes (resolves) all branch and CR logical instructions.
- Two simple units (SU1 and SU2)
  - Add and subtract
  - Shift and rotate
  - Logical operations
  - Support for 64-bit SPE instructions in SU1
- Multiple-cycle unit (MU)—The MU is shown in Figure 5-3.

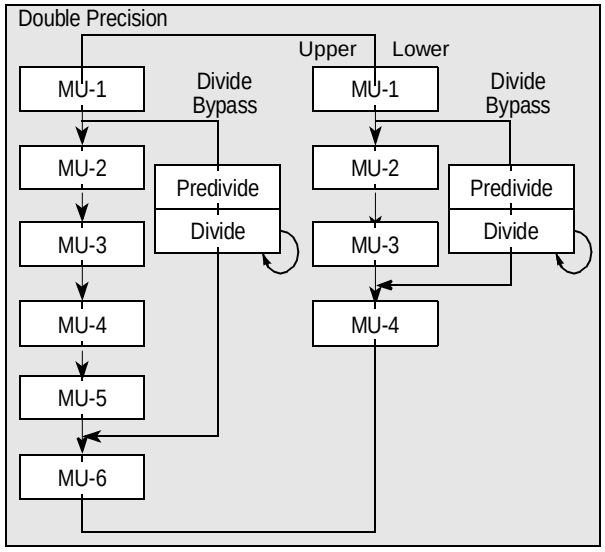


Figure 5-3. MU Pipeline, Showing Divide Bypass

The MU has the following features:

- Four-cycle latency for multiplication, including SPE integer and fractional multiply instructions and embedded scalar and vector single-precision floating-point multiply instructions. Six-cycle latency for double-precision multiplication.
- Variable-latency divide: 4, 11, 19, and 35 cycles for all integer divide instructions. If **r**A or **r**B is zero, floating-point divide instructions take 4 cycles; all others take 29. Note that although

most divide instructions take more than 4 cycles to execute, the MU allows subsequent multiply instructions to execute through all four MU stages in parallel with the divide.

— 4-cycle floating-point add and subtract

- The load/store unit (LSU) is shown in Figure 5-4.

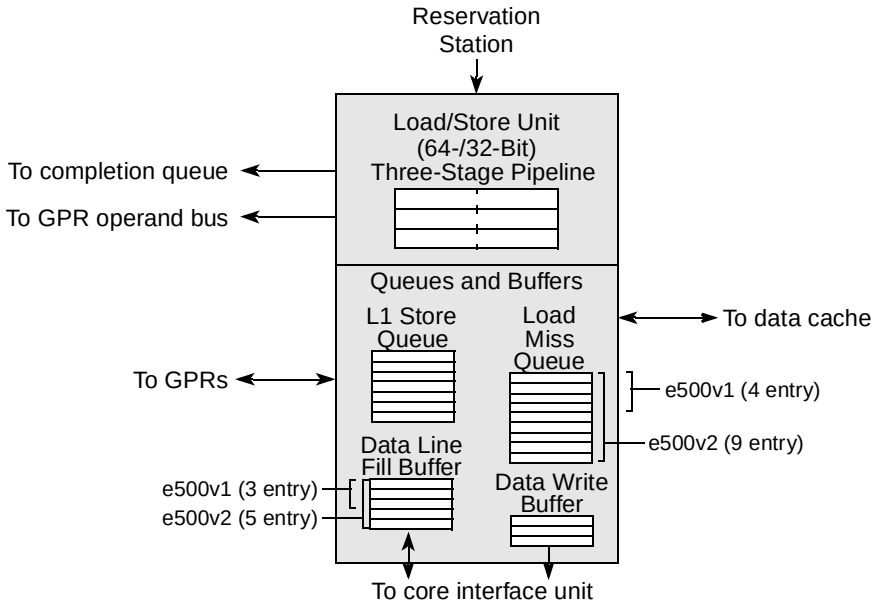


Figure 5-4. Three-Stage Load/Store Unit

The LSU has the following features:

— Three-cycle load latency

— Fully pipelined

— Load miss queue allows up to four load misses before stalling (up to nine load misses in the e500v2).

— Load hits can continue to be serviced when the load miss queue is full.

— The seven-entry L1 store queue allows full pipelining of stores.

— The three-entry data line fill buffer (five-entry on the e500v2) is used for loads and cacheable stores. Stores are allocated here so loads can access data from the store immediately.

— The data write buffer contains three entries: one dedicated for snoop pushes, one dedicated for castouts, and one that can be used for snoop pushes or cast outs.

- Cache coherency

  — Supports four-state cache coherency: modified-exclusive, exclusive, shared, and invalid (MESI). Note, however that shared state may not be accessible in some implementations.

  — Bus support for hardware-enforced coherency (bus snooping)

- Core complex bus (CCB)—internal bus

  — High-speed, on-chip local bus with data tagging

  — 32-bit address bus

  - Address protocol with address pipelining and retry/copyback derived from bus used by previous generations of processors (referred to as the 60x bus)
  - Two general-purpose read data buses and one write data bus
- Extended exception handling
  - Supports embedded category interrupt model
    - Less than 10-cycle interrupt latency
    - Interrupt vector prefix register (IVPR)
    - Interrupt vector offset registers (IVORs) 0–15 and 32–35
    - Exception syndrome register (ESR)
    - Preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfci** instruction
    - A separate set of resources for machine-check interrupts
    - SPE unavailable exception
    - Floating-point data exception
    - Floating-point round exception
    - Performance monitor
- Memory management unit (MMU)
  - 32-bit effective address translated to 32-bit real address (using a 41-bit interim virtual address) for the e500v1core and 36-bit real addressing for the e500v2 core
  - TLB entries for variable- (4-Kbyte–256-Mbyte pages for the e500v1 and 4-Kbyte–4-Gbyte pages for the e500v2) and fixed-size (4-Kbyte) pages
  - Data L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Instruction L1 MMU
    - 4-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Unified L2 MMU
    - 16-entry, fully-associative TLB array for variable-sized pages
    - e500v1—A 256-entry, 2-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
    - e500v2—A 512-entry, 4-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
  - Software reload for TLBs
  - Virtual memory support for as much as 4 Gbytes (2^{32}) of effective address space
  - Real memory support for as much as 4 Gbytes (2^{32}) of physical memory on the e500v1 and 64 Gbytes (2^{36}) on the e500v2
  - Support for big-endian and true little-endian memory on a per-page basis

- Power management
  - Low-power design
  - Power-saving modes: core-halted and core-stopped
  - Internal clock multipliers ranging from 1 to 8 times the bus clock, including integer and half-mode multipliers.The MPC8572E supports multipliers of 2, 2.5, 3, and 3.5
  - Dynamic power management of execution units, caches, and MMUs
  - NAP, DOZE, and SLEEP bits in HID0 can be used to assert *nap*, *doze*, and *sleep* output signals to initiate power-saving modes at the integrated device level.
- Testability
  - LSSD scan design
  - JTAG interface
  - ESP support
- Reliability and serviceability
  - Parity checking on caches
  - Parity checking on e500 local bus

5.3.1 e500v2 Differences

The e500v2 provides the following additional features not supported by the e500v1:

- The e500v2 uses 36-bit physical addressing, which is supported by the following:
  - MMU assist register 7 (MAS7)
  - HID0[EN_MAS7_UPDATE]
  - Programmable jumper options to specify the upper bits of the reset vector.
- The e500v2 has a 512-entry, 4-way set-associative unified TLB for TLB1.
- The maximum variable page size is extended to 4 Gbytes.
- Embedded double-precision floating-point support has been added. These instructions use the 64-bit GPRs as single, 64-bit double-precision operands. This functionality is enabled through MSR[SPE]. Latency for double-precision instructions, except divides, is 6-cycles.
- Slightly different functionality of HID1[RFXE] bit.
- The data line fill buffer in the LSU is expanded from three to five entries.
- The load miss queue in the LSU is expanded from four to nine entries.
- TBSEL and TBEE bits have been added to the performance monitor global control register 0 (PMGC0) to support monitoring of time base events.
- Minor modifications to the SPE instruction set.
- Data cache flush assist capability, supported through HID0[DCFA]. When DCFA is set, the cache miss replacement algorithm ignores invalid entries and follows the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed load or **dcbz** instructions required to flush the cache.

Detailed descriptions of these differences are provided in their respective chapters.

NOTE

Unless otherwise indicated, references to e500 apply to both e500v1 and e500v2.

5.4 Instruction Set

The e500 implements the following instructions:

- The embedded category instruction set for 32-bit implementations. This is composed primarily of the user-level instructions defined by the Power Architecture user instruction set architecture (UISA). The e500 does not include floating-point instructions that require floating-point registers (FPRs), load string, or store string instructions.
- The e500 supports the following instructions:
  - Integer select. Now part of the base category. Consists of the Integer Select instruction (**isel**), which functions as an if-then-else statement that selects between two source registers by comparison to a CR bit. This instruction eliminates conditional branches, decreases latency, and reduces the code footprint.
  - Performance monitor. Table 5-2 lists performance monitor instructions.

Table 5-2. Performance Monitor Instructions

Name	Mnemonic	Syntax
Move from Performance Monitor Register	**mfpmr**	**r**D,PMRN
Move to Performance Monitor Register	**mtpmr**	PMRN,**r**S

  - Cache locking. Consists of the instructions described in Table 5-3.

Table 5-3. Cache Locking Instructions

Name	Mnemonic	Syntax
Data Cache Block Lock Clear	**dcblc**	CT, **r**A, **r**B
Data Cache Block Touch and Lock Set	**dcbtls**	CT, **r**A, **r**B
Data Cache Block Touch for Store and Lock Set	**dcbtstls**	CT, **r**A, **r**B
Instruction Cache Block Lock Clear	**icblc**	CT, **r**A, **r**B
Instruction Cache Block Touch and Lock Set	**icbtls**	CT, **r**A, **r**B

  - Machine check. Defines the Return from Machine Check Interrupt instruction (**rfmci**).
  - SPE vector instructions. Vector instructions are defined that view the 64-bit GPRs as composed of a vector of two 32-bit elements (some instructions also read or write 16-bit elements). Some scalar instructions produce a 64-bit scalar result.
  - The embedded floating-point categories provide scalar and vector floating-point instructions. Scalar single-precision floating-point instructions use only the lower 32 bits of the GPRs; double-precision operands (e500v2 only) use all 64 bits. Table 5-4 lists embedded floating-point instructions.

Table 5-4. Scalar and Vector Embedded Floating-Point Instructions

Instruction	Mnemonic			Syntax
	Scalar SP	Scalar DP	Vector	
Convert Floating-Point Single- from Double-Precision	—	**efscfd**	—	**r**D,**r**B
Convert Floating-Point Double- from Single-Precision	—	**efdcfs**	—	**r**D,**r**B
Convert Floating-Point from Signed Fraction	**efscfsf**	**efdcfsf**	**evfscfsf**	**r**D,**r**B
Convert Floating-Point from Signed Fraction	**efscfsf**	**efdcfsf**	**evfscfsf**	**r**D,**r**B
Convert Floating-Point from Signed Integer	**efscfsi**	**efdcfsi**	**evfscfsi**	**r**D,**r**B
Convert Floating-Point from Unsigned Fraction	**efscfuf**	**efdcfuf**	**evfscfuf**	**r**D,**r**B
Convert Floating-Point from Unsigned Integer	**efscfui**	**efdcfui**	**evfscfui**	**r**D,**r**B
Convert Floating-Point to Signed Fraction	**efsctsf**	**efdctsf**	**evfsctsf**	**r**D,**r**B
Convert Floating-Point to Signed Integer	**efsctsi**	**efdctsi**	**evfsctsi**	**r**D,**r**B
Convert Floating-Point to Signed Integer with Round toward Zero	**efsctsiz**	**efdctsiz**	**evfsctsiz**	**r**D,**r**B
Convert Floating-Point to Unsigned Fraction	**efsctuf**	**efdctuf**	**evfsctuf**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer	**efsctui**	**efdctui**	**evfsctui**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer with Round toward Zero	**efsctuiz**	**efdctuiz**	**evfsctuiz**	**r**D,**r**B
Floating-Point Absolute Value	**efsabs**	**efdabs**	**evfsabs**	**r**D,**r**A
Floating-Point Add	**efsadd**	**efdadd**	**evfsadd**	**r**D,**r**A,**r**B
Floating-Point Compare Equal	**efscmpeq**	**efdcmpeq**	**evfscmpeq**	**cr**D,**r**A,**r**B
Floating-Point Compare Greater Than	**efscmpgt**	**efdcmpgt**	**evfscmpgt**	**cr**D,**r**A,**r**B
Floating-Point Compare Less Than	**efscmplt**	**efdcmplt**	**evfscmplt**	**cr**D,**r**A,**r**B
Floating-Point Divide	**efsdiv**	**efddiv**	**evfsdiv**	**r**D,**r**A,**r**B
Floating-Point Multiply	**efsmul**	**efdmul**	**evfsmul**	**r**D,**r**A,**r**B
Floating-Point Negate	**efsneg**	**efdneg**	**evfsneg**	**r**D,**r**A
Floating-Point Negative Absolute Value	**efsnabs**	**efdnabs**	**evfsnabs**	**r**D,**r**A
Floating-Point Subtract	**efssub**	**efdsub**	**evfssub**	**r**D,**r**A,**r**B
Floating-Point Test Equal	**efststeq**	**efdtsteq**	**evfststeq**	**cr**D,**r**A,**r**B
Floating-Point Test Greater Than	**efststgt**	**efdtstgt**	**evfststgt**	**cr**D,**r**A,**r**B
Floating-Point Test Less Than	**efststlt**	**efdtstlt**	**evfststlt**	**cr**D,**r**A,**r**B

— BTB locking instructions. The core complex provides a 512-entry BTB for efficient processing of branch instructions. The BTB is a branch target address cache, organized as 128 rows with 4-way set associativity, that holds the address and target instruction of the 512 most-recently taken branches. Table 5-5 lists BTB instructions.

Table 5-5. BTB Locking Instructions

Name	Mnemonic	Syntax
Branch Buffer Load Entry and Lock Set	**bblels**	—
Branch Buffer Entry Lock Reset	**bbelr**	—

5.5 Instruction Flow

The e500 core is a pipelined, superscalar processor with parallel execution units that allow instructions to execute out of order but record their results in order. Pipelining breaks instruction processing into discrete stages, so multiple instructions in an instruction sequence can occupy the successive stages: as an instruction completes one stage, it passes to the next, leaving the previous stage available to a subsequent instruction. So, even though it may take multiple cycles for an instruction to pass through all of the pipeline stages, once a pipeline is full, instruction throughput is much shorter than the latency.

A superscalar processor is one that issues multiple independent instructions into separate execution units, allowing parallel execution. The e500 core has five execution units, one each for branch (BU), load/store (LSU), and multiple-cycle operations (MU), and two for simple arithmetic operations (SU1 and SU2). The MU and SU1 arithmetic execution units also execute 64-bit SPE vector instructions, using both the lower and upper halves of the 64-bit GPRs.

The parallel execution units allow multiple instructions to execute in parallel and out of order. For example, a low-latency addition instruction that is issued to an SU after an integer divide is issued to the MU should finish executing before the higher latency divide instruction. The add instruction can make its results available to a subsequent instruction, but it cannot update the architected GPR specified as its target operand ahead of the multiple-cycle divide instruction.

5.5.1 Initial Instruction Fetch

The e500 core begins execution at fixed virtual address 0xFFFF_FFFC. The MMU has a default page translation which maps this to the identical physical address. So, the instruction at physical address 0xFFFF_FFFC must be a branch to another address within the 4-Kbyte boot page.

5.5.2 Branch Detection and Prediction

To improve branch performance, the e500 provides implementation-specific dynamic branch prediction using the BTB to resolve branch instructions and improve the accuracy of branch predictions. Each of the 512 entries in the 4-way set associative address cache of branch target addresses includes a 2-bit saturating branch history counter, whose value is incremented or decremented depending on whether the branch was taken. These bits can take on four values indicating strongly taken, weakly taken, weakly not taken, and strongly not taken. The BTB is used not only to predict branches, but to detect branches during the fetch stage, offering an efficient way to access instruction streams for branches predicted as taken.

In the e500, all branch instructions are assigned positions in the completion queue at dispatch. Speculative instructions in branch target streams are allowed to execute and proceed through the completion queue, although they can complete only after the branch prediction is resolved as correct and after the branch instruction itself completes.

If a branch resolves as correct, instructions in the target stream are marked nonspeculative and are allowed to complete. If the branch history bits in the BTB indicated weakly taken or weakly not taken, the prediction is upgraded to strongly taken or strongly not taken.

If a branch resolves as incorrect, instructions in the target stream are flushed from the execution pipeline, the branch history bits are updated in the BTB entry, and nonspeculative fetching begins from the correct path.

5.5.3 e500 Execution Pipeline

The seven stages of the e500 execution pipeline—fetch1, fetch2/predecode, decode/dispatch, issue, execute, complete, and write back—are highlighted in grey in Figure 5-5.

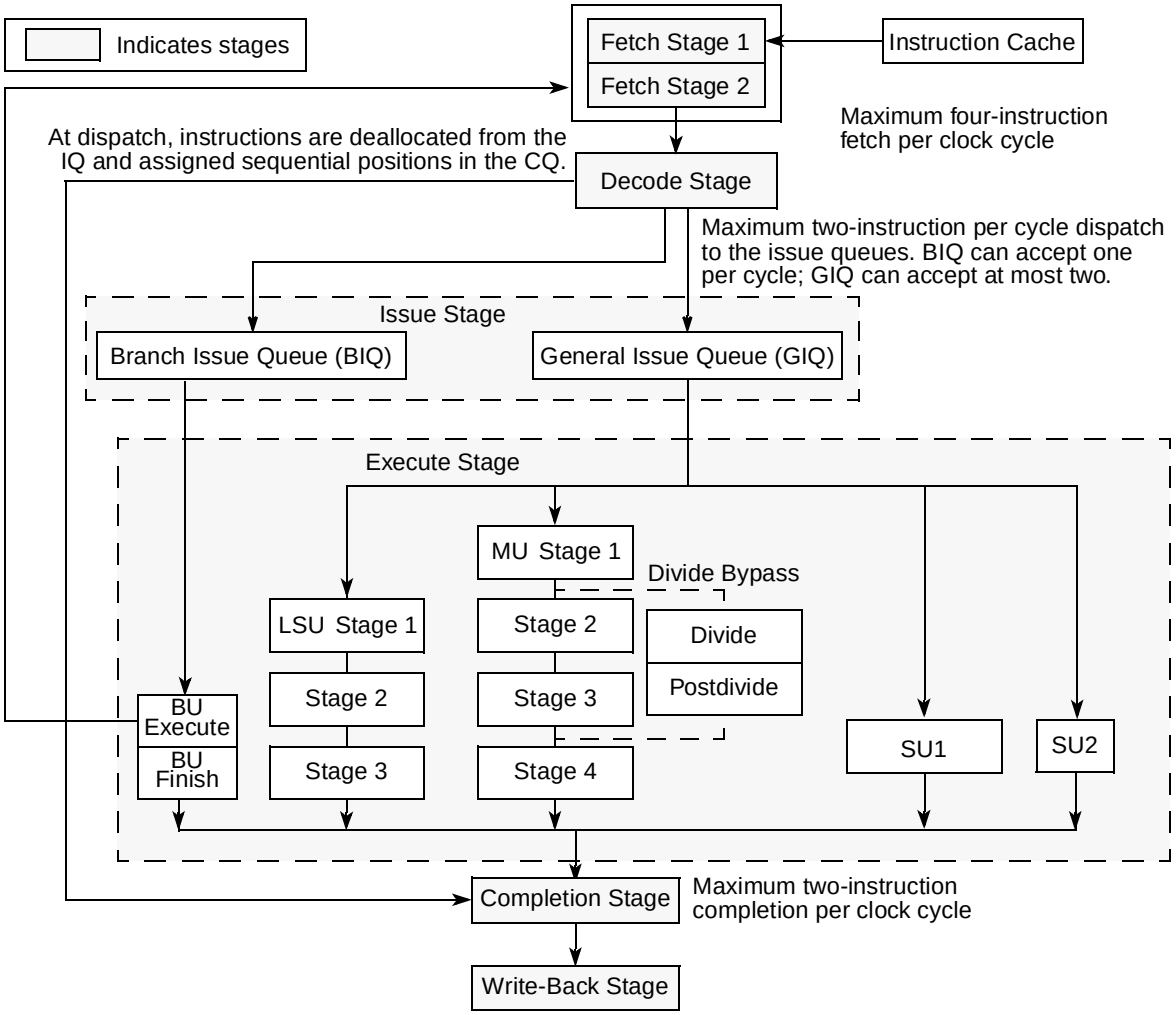


Figure 5-5. Instruction Pipeline Flow

The common pipeline stages are as follows:

- Instruction fetch—Includes the clock cycles necessary to request an instruction and the time the memory system takes to respond to the request. Instructions retrieved are latched into the instruction queue (IQ) for subsequent consideration by the dispatcher.

Instruction fetch timing depends on many variables, such as whether an instruction is in the on-chip instruction cache or an L2 cache (if implemented). Those factors increase when it is necessary to fetch instructions from system memory and include the processor-to-bus clock ratio, the amount of bus traffic, and whether any cache coherency operations are required.

Because there are so many variables, unless otherwise specified, the instruction timing examples in this chapter assume optimal performance and show the portion of the fetch stage in which the instruction is in the instruction queue. The fetch1 and fetch2 stages are primarily involved in retrieving instructions.

- The decode/dispatch stage fully decodes each instruction; most instructions are dispatched to the issue queues (however, **isync**, **rfi**, **sc**, **nop**s, and some other instructions do not go to issue queues).
- The two issue queues, BIQ and GIQ, can accept as many as one and two instructions, respectively, in a cycle. The behavior of instruction dispatch is covered in significant detail in the *e500 Software Optimization Guide*. The following simplification covers most cases:
  - Instructions dispatch only from the two lowest IQ entries—IQ0 and IQ1.
  - A total of two instructions can be dispatched to the issue queues per clock cycle.
  - Space must be available in the CQ for an instruction to decode and dispatch (this includes instructions that are assigned a space in the CQ but not in an issue queue).

Dispatch is treated as an event at the end of the decode stage. The issue stage reads source operands from rename registers and register files and determines when instructions are latched into the execution unit reservation stations. Note that the e500 has 14 rename registers, one for each completion queue entry, so instructions cannot stall because of a shortage of rename registers.

The general behavior of the two issue queues is described as follows:

- The GIQ accepts as many as two instructions from the dispatch unit per cycle. SU1, SU2, MU, and all LSU instructions (including 64-bit loads and stores) are dispatched to the GIQ, shown in Figure 5-6.

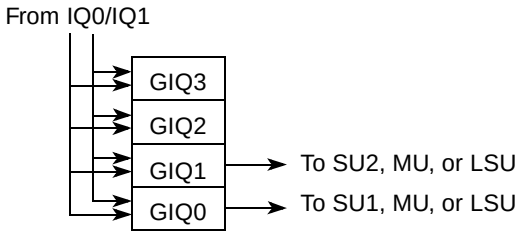


Figure 5-6. GPR Issue Queue (GIQ)

Instructions can be issued out-of-order from the bottom two GIQ entries (GIQ1–GIQ0). GIQ0 can issue to SU1, MU, and LSU. GIQ1 can issue to SU2, MU, and LSU.

Note that SU2 executes a subset of the instructions that can be executed in SU1. The ability to identify and dispatch instructions to SU2 increases the availability of SU1 to execute more computational-intensive instructions.

An instruction in GIQ1 destined for SU2 or the LSU need not wait for an MU instruction in GIQ0 that is stalled behind a long-latency divide.

- The execute stage accepts instructions from its issue queue when the appropriate reservation stations are not busy. In this stage, the operands assigned to the execution stage from the issue stage are latched.

  The execution unit executes the instruction (perhaps over multiple cycles), writes results on its result bus, and notifies the CQ when the instruction finishes. The execution unit reports any exceptions to the completion stage. Instruction-generated exceptions are not taken until the excepting instruction is next to retire.

  Most integer instructions have a 1-cycle latency, so results of these instructions are available 1 clock cycle after an instruction enters the execution unit. The MU and LSU are pipelined, as shown in Figure 5-5.

  Branches resolve in execute stage. If a branch is mispredicted, it takes 5 cycles for the next instruction to reach the execute stage.
- The complete and write-back stages maintain the correct architectural machine state and commit results to the architecture-defined registers in the proper order. If completion logic detects an instruction containing an exception status or a mispredicted branch, all following instructions are cancelled, their execution results in rename registers are discarded, and the correct instruction stream is fetched.

  The complete stage ends when the instruction is retired. Two instructions can be retired per clock cycle. If no dependencies exist, as many as two instructions are retired in program order.

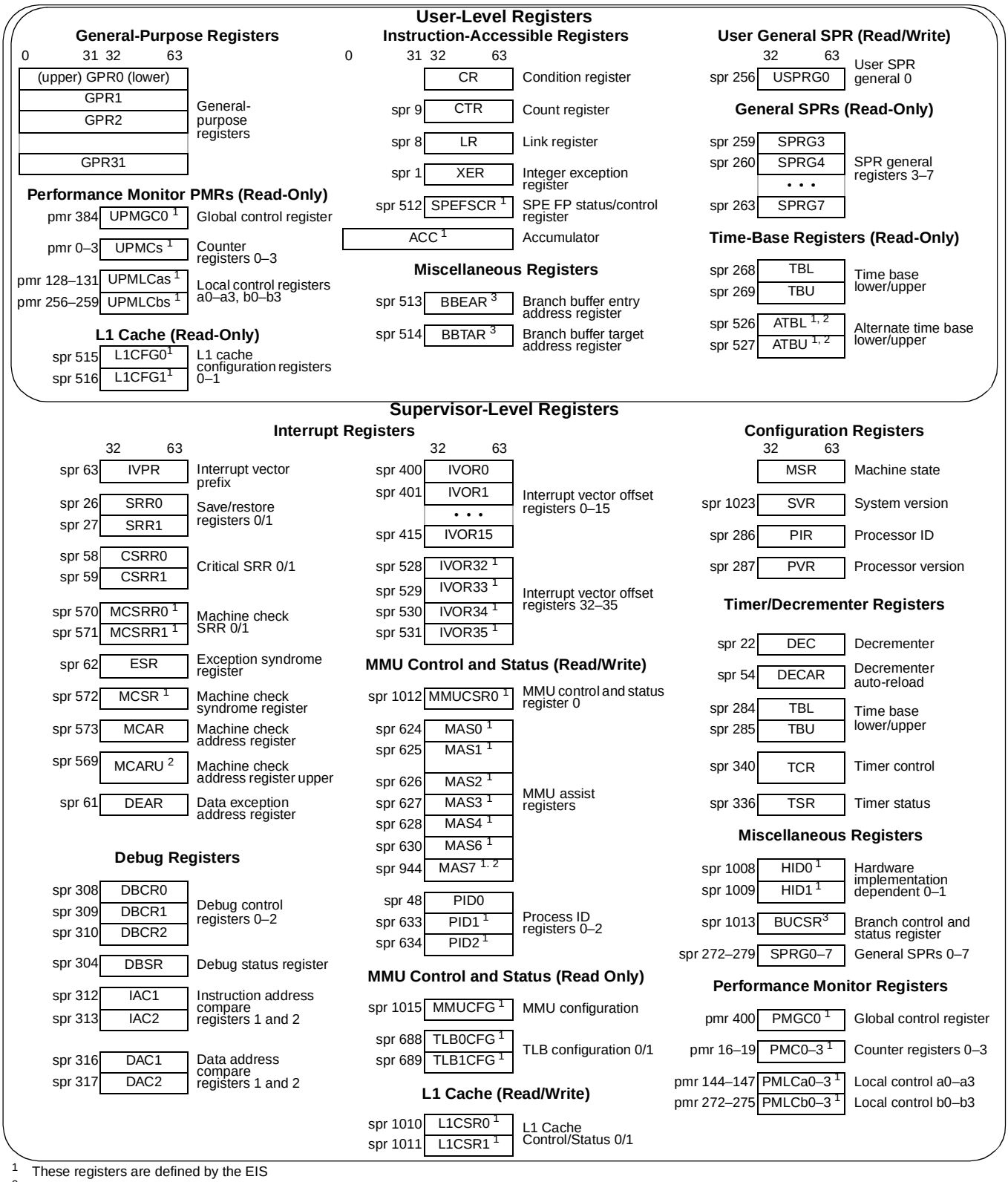
  The write-back stage occurs in the clock cycle after the instruction is retired.

The e500 core also provides new instructions that perform single-instruction, multiple-data (SIMD) operations. These signal processing instructions consist of parallel operations on both the upper and lower 32 bits of two 64-bit GPR values and produce two 32-bit results written to a 64-bit GPR.

As shown in Figure 5-5, the LSU, MU, and SU1 replicate logic to support 64-bit operations. Although a vector instruction generates separate, discrete results in the upper and lower halves of the target GPR, latency and throughput for vector instructions are the same as those for their scalar equivalents.

5.6 Programming Model

The following section describes the e500 core registers. Figure 5-7 shows the e500 register set.



1 These registers are defined by the EIS
2 e500v2 only
3 These registers are e500-specific

Figure 5-7. e500 Core Programming Model

5.7 On-Chip Cache Implementation

The core complex contains separate 32-Kbyte, eight-way set-associative, level 1 (L1) instruction and data caches to give rapid access to instructions and data.

The data cache supports four-state MESI memory coherency protocol. The core complex broadcasts all cache management functions based on the setting of the address broadcast enable bit, HID1[ABE], allowing management of other caches in the system.

On the MPC8572E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

The caches implement a pseudo-least-recently-used (PLRU) replacement algorithm.

Parity generation and checking may be enabled for both caches, and each cache can be independently invalidated through L1CSR1 and L1CSR0. Additionally, instructions are provided to perform cache locking and unlocking on both data and instruction caches on a cache-block granularity. These are listed in Section 5.10.3, “Cache Control Instructions.”

Individual instruction cache blocks and data cache blocks can be invalidated using the **icbi** and **dcbi** instructions, respectively. The entire data cache can be invalidated by setting L1CSR0[CFI]; the entire instruction cache can be invalidated by setting L1CSR1[ICFI].

5.8 Interrupts and Exception Handling

The e500 core supports an extended exception handling model, with nested interrupt capability and extensive interrupt vector programmability. The following sections define the exception model, including an overview of exception handling as implemented on the e500 core, a brief description of the exception classes, and an overview of the registers involved in the processes.

5.8.1 Exception Handling

In general, interrupt processing begins with an exception that occurs due to external conditions, errors, or program execution problems. When the exception occurs, the processor checks to verify interrupt processing is enabled for that particular exception. If enabled, the interrupt causes the state of the processor to be saved in the appropriate registers and prepares to begin execution of the handler located at the associated vector address for that particular exception.

Once the handler is executing, the implementation may need to check one or more bits in the exception syndrome register (ESR) or the SPEFSCR, depending on the exception, to verify the specific cause of the exception and take appropriate action.

The core complex provides the interrupts described in Section 5.8.5, “Interrupt Registers.”

5.8.2 Interrupt Classes

All interrupts may be categorized as asynchronous/synchronous and critical/noncritical.

- Asynchronous interrupts (such as machine check, critical input, and external interrupts) are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported in a save/restore register is the address of the instruction that would have executed next had the asynchronous interrupt not occurred.
- Synchronous interrupts are those that are caused directly by the execution or attempted execution of instructions. Synchronous inputs may be either precise or imprecise, which are described as follows:
  - Synchronous precise interrupts are those that precisely indicate the address of the instruction causing the exception that generated the interrupt or, in some cases, the address of the immediately following instruction. The interrupt type and status bits indicate which instruction is addressed in the appropriate save/restore register.
  - Synchronous imprecise interrupts are those that may indicate the address of the instruction causing the exception that generated the interrupt or some instruction after the instruction causing the interrupt. If the interrupt was caused by either the context synchronizing mechanism or the execution synchronizing mechanism, the address in the appropriate save/restore register is the address of the interrupt forcing instruction. If the interrupt was not caused by either of those mechanisms, the address in the save/restore register is the last instruction to start execution and may not have completed. No instruction following the instruction in the save/restore register has executed.

5.8.3 Interrupt Types

The e500 core processes all interrupts as either machine check, critical, or noncritical types. Separate control and status register sets are provided for each interrupt type. The core handles interrupts from these three types in the following priority order:

1. Machine check interrupt (highest priority)—The e500 defines a separate set of resources for the machine check interrupt. They use the machine check save and restore registers (MCSRR0/MCSRR1) to save state when they are taken, and they use the **rfmci** instruction to restore state. These interrupts can be masked by the machine check enable bit, MSR[ME].
2. Noncritical interrupts—First-level interrupts that allow the processor to change program flow to handle conditions generated by external signals, errors, or unusual conditions arising from program execution or from programmable timer-related events. These interrupts are largely identical to those previously defined by the OEA portion of the architecture. They use save and restore registers (SRR0/SRR1) to save state when they are taken and they use the **rfi** instruction to restore state. Asynchronous noncritical interrupts can be masked by the external interrupt enable bit, MSR[EE].
3. Critical interrupts—Critical interrupts can be taken during a noncritical interrupt or during regular program flow. They use the critical save and restore registers (CSRR0/CSRR1) to save state when they are taken and they use the **rfci** instruction to restore state. These interrupts can be masked by

the critical enable bit, MSR[CE]. The embedded category defines the critical input, watchdog timer, and machine check interrupts as critical interrupts, but the e500 implements a third set of resources for the machine check interrupt, as described in Table 5-6.

All interrupts except machine check are ordered within the two categories of noncritical and critical, such that only one interrupt of each category is reported, and when it is processed (taken), no program state is lost. Because save/restore register pairs are serially reusable, program state may be lost when an unordered interrupt is taken.

5.8.4 Upper Bound on Interrupt Latencies

Core complex interrupt latency is defined as the number of core clocks between the sampling of the interrupt signal as asserted and the initiation of the IVOR fetch (that is, the fetch of the first instruction in the handler). Core complex interrupt latency is determinate unless a guarded load or a cache-inhibited **stwcx.** is being executed, in which case the latency is indeterminate. The minimum latency is 3 core clocks and the maximum is 8, not including the 2 bus clock cycles required to synchronize the interrupt signal from the pad.

When an interrupt is taken, all instructions in the IQ are thrown away unless the oldest instruction is a load/store instruction. That is, if an asynchronous interrupt is being serviced and the oldest instruction is not a load/store instruction, the core complex goes straight from sampling the interrupt to ensuring a recoverable state and issuing an exception. If a load/store instruction is oldest, the core complex waits 4 clocks before ensuring a recoverable state. During this time, any instruction finished by the LSU is deallocated.

5.8.5 Interrupt Registers

The registers associated with interrupt and exception handling are described in Table 5-6.

Table 5-6. Interrupt Registers

Register	Description
Noncritical Interrupt Registers	
SRR0	Save/restore register 0—Holds the address of the instruction causing the exception or the address of the instruction that executes after the **rfi** instruction.
SRR1	Save/restore register 1—Holds machine state on noncritical interrupts and restores machine state after an **rfi** instruction is executed.
Critical Interrupt Registers	
CSRR0	Critical save/restore register 0—On critical interrupts, holds either the address of the instruction causing the exception or the address of the instruction that executes after the **rfci** instruction.
CSRR1	Critical save/restore register 1—Holds machine state on critical interrupts and restores machine state after an **rfci** instruction is executed.
Machine Check Interrupt Registers	
MCSRR0	Machine check save/restore register 0—Used to store the address of the instruction that executes after an **rfmci** instruction is executed.
MCSRR1	Machine check save/restore register 1—Holds machine state on machine check interrupts and restores machine state (if recoverable) after an **rfmci** instruction is executed.

Table 5-6. Interrupt Registers (continued)

Register	Description
MCAR	Machine check address register—Holds the address of the data or instruction that caused the machine check interrupt. MCAR contents are not meaningful if a signal triggered the machine check interrupt.
Syndrome Registers	
MCSR	Machine check syndrome register—Holds machine state information on machine check interrupts and restores machine state after an **rfmci** instruction is executed.
ESR	Exception syndrome register—Provides a syndrome to differentiate between the different kinds of exceptions that generate the same interrupt type. Upon generation of a specific exception type, the associated bit is set and all other bits are cleared.
SPE Interrupt Registers	
SPEFSCR	Signal processing and embedded floating-point status and control register—Provides interrupt control and status as well as various condition bits associated with the operations performed by the SPE.
Other Interrupt Registers	
DEAR	Data exception address register—Holds the address that was referenced by a load, store, or cache management instruction that caused an alignment, data TLB miss, or data storage interrupt.
IVPR IVORs	Together, IVPR[32–47] || IVOR*n* [48–59] || 0b0000 define the address of an interrupt-processing routine. See Table 5-7 and the EREF for more information.

Each interrupt has an associated interrupt vector address, obtained by concatenating the IVPR value with the address index in the associated IVOR (that is, IVPR[32–47] || IVOR*n*[48–59] || 0b0000). The resulting address is that of the instruction to be executed when that interrupt occurs. IVPR and IVOR values are indeterminate on reset, and must be initialized by the system software using **mtspr**. Table 5-7 lists IVOR registers implemented on the e500 and the associated interrupts.

Table 5-7. Interrupt Vector Registers and Exception Conditions

Register	Interrupt
Embedded Category–Defined IVORs	
IVOR0	Critical input
IVOR1	Machine check interrupt offset
IVOR2	Data storage interrupt offset
IVOR3	Instruction storage interrupt offset
IVOR4	External input interrupt offset
IVOR5	Alignment interrupt offset
IVOR6	Program interrupt offset
IVOR7	Floating-point unavailable interrupt offset
IVOR8	System call interrupt offset
IVOR9	Auxiliary processor unavailable interrupt offset
IVOR10	Decrementer interrupt offset
IVOR11	Fixed-interval timer interrupt offset
IVOR12	Watchdog timer interrupt offset
IVOR13	Data TLB error interrupt offset

Table 5-7. Interrupt Vector Registers and Exception Conditions (continued)

Register	Interrupt
IVOR14	Instruction TLB error interrupt offset
IVOR15	Debug interrupt offset
e500-Specific IVORs	
IVOR32	SPE unavailable interrupt offset
IVOR33	SPE floating-point data exception interrupt offset
IVOR34	SPE floating-point round exception interrupt offset
IVOR35	Performance monitor

5.9 Memory Management

The e500 core complex supports demand-paged virtual memory as well other memory management schemes that depend on precise control of effective-to-physical address translation and flexible memory protection as defined by the architecture. The mapping mechanism consists of software-managed TLBs that support variable-sized pages with per-page properties and permissions. The following properties can be configured for each TLB:

- User-mode page execute access
- User-mode page read access
- User-mode page write access
- Supervisor-mode page execute access
- Supervisor-mode page read access
- Supervisor-mode page write access
- Write-through required (W)
- Caching inhibited (I)
- Memory coherency required (M).
- Guarded (G)
- Endianness (E)
- User-definable (U0–U3), a 4-bit implementation-specific field

The core complex employs a two-level memory management unit (MMU) architecture. There are separate instruction and data level-1 (L1) MMUs backed up by a unified level-2 (L2) MMU.

This two-level structure is shown in Figure 5-8.

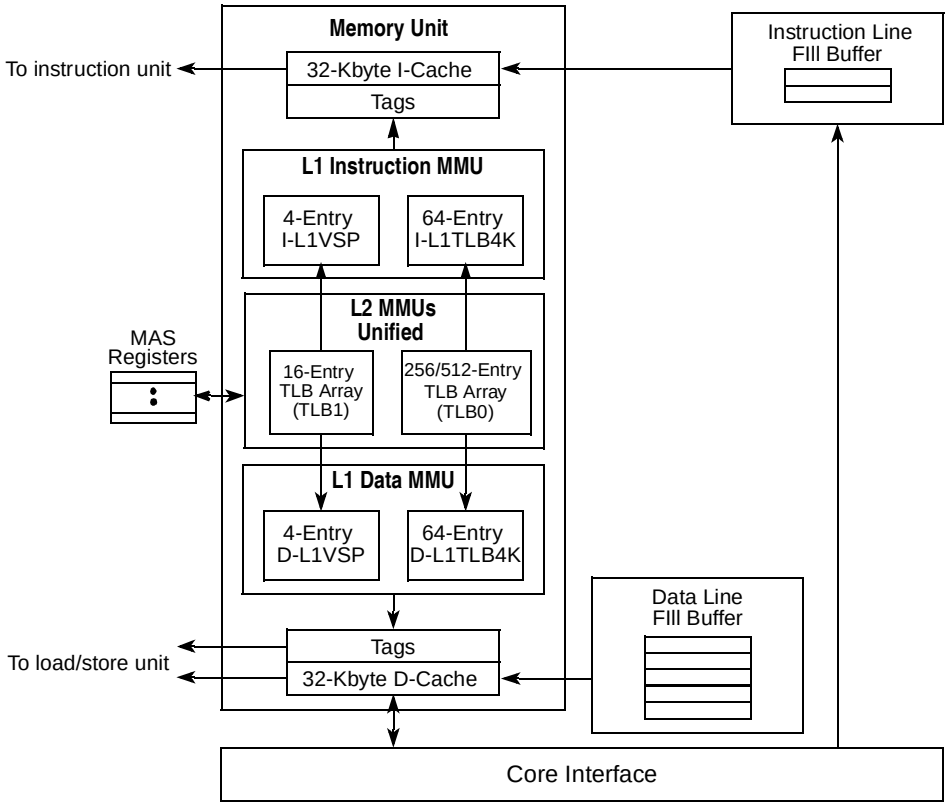


Figure 5-8. MMU Structure

Level-1 MMUs have the following features:

- Four-entry, fully associative TLB array that supports all nine page sizes
- 64-entry, 4-way set-associative TLB 4-Kbyte array that supports 4-Kbyte pages only
- Hardware partially managed by L2 MMU
- Supports snooping of TLBs by both internal and external **tlbivax** instructions

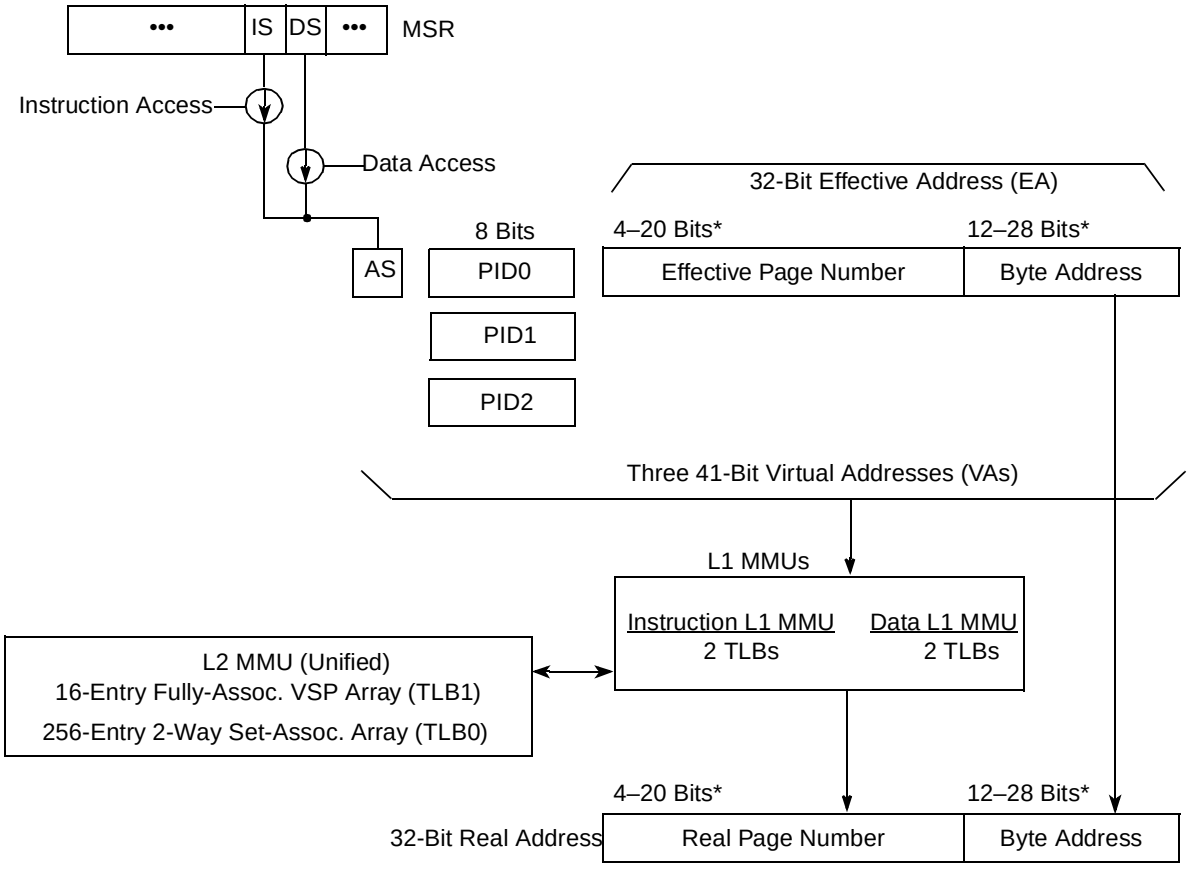
The level-2 MMU has the following features:

- A 16-entry, fully associative L2 TLB array (TLB1) that supports all nine variable page sizes
- TLB array (TLB0) that supports only 4-Kbyte pages, as follows:
  - e500v1—256-entry, 2-way set-associative TLB array
  - e500v2—512-entry, 4-way set-associative TLB array
- Hardware assist for TLB miss exceptions
- Software managed by **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, **tlbivax**, and **mtspr** instructions
- Supports snooping of TLB by both internal and external **tlbivax** instructions

5.9.1 Address Translation

The core complex fetch and load/store units generate 32-bit effective addresses. The MMU translates these addresses to real addresses (32-bit real addresses for the e500v1 core, 36-bit for the e500v2) (which are used for memory bus accesses) using an interim 41-bit virtual address.

Figure 5-9 shows the translation flow for the e500v1 core.



* Number of bits depends on page size (4 Kbytes–256 Mbytes).

Figure 5-9. Effective-to-Real Address Translation Flow

Figure 5-10 shows the same translation flow for the e500v2 core.

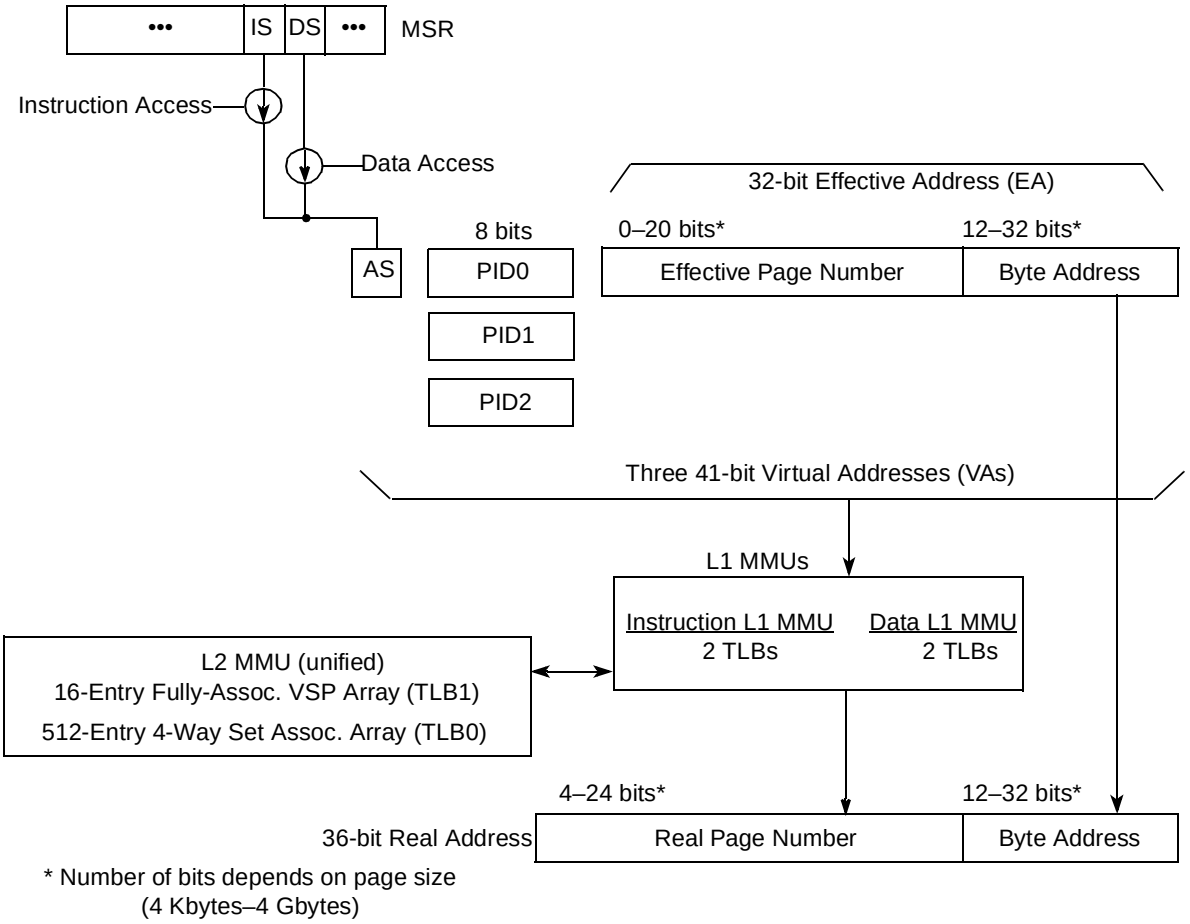


Figure 5-10. Effective-to-Real Address Translation Flow (e500v2)

The appropriate L1 MMU (instruction or data) is checked for a matching address translation. The instruction L1 MMU and data L1 MMU operate independently and can be accessed in parallel, so that hits for instruction accesses and data accesses can occur in the same clock. If an L1 MMU misses, the request for translation is forwarded to the unified (instruction and data) L2 MMU. If found, the contents of the TLB entry are concatenated with the byte address to obtain the physical address of the requested access. On misses, the L1 TLB entries are replaced from their L2 TLB counterparts using a true LRU algorithm.

5.9.2 MMU Assist Registers (MAS0–MAS4 and MAS6–MAS7)

MMU assist registers are used to hold values either read from or to be written to the TLBs and information required to identify the TLB to be accessed. MAS3 implements the real page number (RPN), the user attribute bits (U0–U3), and permission bits (UX, SX, UW, SW, UR, SR) that specify user and supervisor read, write, and execute permissions.

The e500 does not implement MAS5.

MAS registers are affected by the following instructions (see the EREF for more detailed information):

- MAS registers are accessed with the **mtspr** and **mfspr** instructions.
- The TLB Read Entry instruction (**tlbre**) causes the contents of a single TLB entry from the L2 MMU to be placed in defined locations in MAS0–MAS3 (and optionally MAS7 on the e500v2). The TLB entry to be extracted is determined by information written to MAS0 and MAS2 before the **tlbre** instruction is executed.
- The TLB Write Entry instruction (**tlbwe**) causes the information stored in certain locations of MAS0–MAS3 (and MAS7 on the e500v2) to be written to the TLB specified in MAS0.
- The TLB Search Indexed instruction (**tlbsx**) updates MAS registers conditionally, based on success or failure of a lookup in the L2 MMU. The lookup is specified by the instruction encoding and specific search fields in MAS6. The values placed in the MAS registers may differ, depending on a successful or unsuccessful search.

For TLB miss and certain MMU-related DSI/ISI exceptions, MAS4 provides default values for updating MAS0–MAS2.

5.9.3 Process ID Registers (PID0–PID2)

The e500 core complex also implements three process ID (PID) registers that hold the values used to construct the three virtual addresses for each access. These process IDs provide an extended page sharing capability. Which of these three virtual addresses is used is controlled by the TID field of a matching TLB entry, and when TID = 0x00 (identifying a page as globally shared), the PID values are ignored.

A hit to multiple TLB entries in the L1 MMU (even if they are in separate arrays) or a hit to multiple entries in the L2 MMU is considered to be a programming error.

5.9.4 TLB Coherency

The core complex provides the ability to invalidate a TLB entry, as defined by the architecture. The **tlbivax** instruction invalidates a matching local TLB entry. Execution of this instruction is also broadcast on the core complex bus (CCB) if HID1[ABE] is set. The core complex also snoops TLB invalidate transactions on the CCB from other bus masters.

On the MPC8572E the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

5.10 Memory Coherency

The core complex supports four-state memory coherency. Memory coherency is hardware-supported on the system bus through bus snooping and the retry/copyback bus protocol, and through broadcasting of cache management instructions. Translation coherency is also hardware-supported through broadcasting and bus snooping of TLB invalidate transactions. The four-state MESI protocol supports efficient large-scale real-time data sharing between multiple caching bus masters.

5.10.1 Atomic Update Memory References

The e500 core supports atomic update memory references for both aligned word forms of data using the load and reserve and store conditional instruction pair, **lwarx** and **stwcx**. Typically, a load and reserve instruction establishes a reservation and is paired with a store conditional instruction to achieve the atomic operation. However, there are restrictions and requirements for this functionality. The processor revokes reservations during a context switch, so the programmer must reacquire the reservation after a context switch occurs.

5.10.2 Memory Access Ordering

The core complex supports weakly ordered references to memory. Thus the e500 manages the order and synchronization of instructions to ensure proper execution when memory is shared between multiple processes or programs. The cache and data memory control attributes, along with **msync** and **mbar**, provide the required access control.

5.10.3 Cache Control Instructions

The core complex supports instructions for performing a full range of cache control functions, including cache locking by line. The core complex supports broadcasting and snooping of these cache control instructions on the CCB. The e500 core also supports the following e500-specific cache locking instructions:

- Data Cache Block Lock Clear (**dcblc**)
- Data Cache Block Touch and Lock Set (**dcbtls**)
- Data Cache Block Touch for Store and Lock Set (**dcbtstls**)
- Instruction Cache Block Lock Clear (**icblc**)
- Instruction Cache Block Touch and Lock Set (**icbtls**)

5.10.4 Programmable Page Characteristics

Cache and memory attributes are programmable on a per-page basis. In addition to the write-through, caching-inhibited, memory coherency enforced, and guarded characteristics defined by the WIMG bits, the endianness bit, E, allows selection of big- or little-endian byte ordering on a per-page basis.

In addition to the WIMGE bits, the MMU model defines user-definable page attribute bits U0–U3.

5.11 Core Complex Bus (CCB)

The core complex defines a versatile local bus interface that allows a wide range of system performance and system-complexity trade-offs. The interface defines the following buses:

- An address-out bus for mastering bus transactions
- An address-in bus for snooping internal resources
- Three tagged data buses

Two of the data buses are general-purpose data-in buses for reads, and the third is a data-out bus for writes. The two data-in buses feature support for out-of-order read transactions from two different sources simultaneously, and all three data buses may be operated concurrently. The address-in bus supports snooping for external management of the L1 caches and TLBs by other bus masters. The core complex broadcasts and snoops the cache and TLB management instructions accordingly. It is envisioned that a wide range of system implementations can be constructed from the defined interface.

5.12 Performance Monitoring

The e500 core provides a performance monitoring capability that allows counting of events such as processor clocks, instruction cache misses, data cache misses, mispredicted branches, and others. The count of these events may be configured to trigger a performance monitor exception following the e500 interrupt model. This interrupt is assigned to vector offset register IVOR35.

The register set associated with the performance monitoring function consists of counter registers, a global control register, and local control registers. These registers are read/write from supervisor mode, and each register is reflected to a corresponding read-only register for user mode. Two instructions, **mtpmr** and **mfpmr**, are provided for moving data to and from these registers. An overview of the performance monitoring registers is provided in the following sections.

5.12.1 Global Control Register

The PMGC0 register provides global control of the performance monitoring facility from supervisor mode. From this register all counters may be frozen, unfrozen, or configured to freeze on an enabled condition or event. Additionally, the performance monitoring facility may be disabled or enabled from this register. The contents of PMGC0 are reflected to UPMGC0, which may be read from user mode using the **mfpmr** instruction.

5.12.2 Performance Monitor Counter Registers

There are four counter registers (PCM0–PCM3) provided in the performance monitoring facility. These 32-bit registers hold the current count for software-selectable events and can be programmed to generate an exception on overflow. These registers may be written or read from supervisor mode using the **mtpmr** and **mfpmr** instructions. The contents of these registers are reflected to UPCM0–UPCM3, which can be read from user mode with **mfpmr**.

Performance monitor exceptions occur only if all of the following conditions are met:

- A counter is in the overflow state.
- The counter's overflow signaling is enabled.
- Overflow exception generation is enabled in PMGC0.
- MSR[EE] is set.

5.12.3 Local Control Registers

For each of the counter registers, there are two corresponding local control registers. These two registers specify which of the 128 available events is to be counted, what specific action is to be taken on overflow, and various options for freezing a counter value under given modes or conditions.

- PMLCa0–PMLCa3 provide fields that allow freezing of the corresponding counter in user mode, supervisor mode, or under software control. Additionally, the overflow condition may be enabled or disabled from this register. The contents of these registers are reflected to UPMLCa0–UPMLCa3, which can be read from user mode with **mfpmr**.
- PMLCb0–PMLCb3 provide count scaling for each counter register using configurable threshold and multiplier values. The threshold is a 6-bit value and the multiplier is a 3-bit encoded value, allowing eight multiplier values in the range of 1 to 128. Any counter may be configured to increment only when an event occurs more than [threshold × multiplier] times. The contents of these registers are reflected to UPMLCb0–UPMLCb3, which can be read from user mode with **mfpmr**.

5.13 Legacy Support of Power Architecture Technology

This section provides an overview of the architectural differences and compatibilities of the e500 core compared with the AIM Power Architecture technology. The two levels of the e500 programming environment are as follows:

- User level—This defines the base user-level instruction set, user-level registers, data types, memory conventions, and the memory and programming models seen by application programmers.
- Supervisor level—This defines supervisor-level resources typically required by an operating system, the memory management model, supervisor level registers, and the exception model.

Like all devices that implement the Power Architecture technology, in general, the e500 core supports the user-level architecture. The following sections are intended to highlight the main differences. For specific implementation details refer to the relevant chapter.

5.13.1 Instruction Set Compatibility

The following sections generally describe the user and supervisor instruction sets.

5.13.1.1 User Instruction Set

The e500 core executes legacy user-mode binaries and object files except for the following:

- The e500 supports vector and scalar single-precision floating-point operations as part of the SPE. The e500v2 supports scalar double-precision floating-point instructions. These instructions have different encoding than the AIM definition of the architecture. Additionally, the e500 core uses GPRs for floating-point operations, rather than the FPRs defined by the UISA. Most porting of floating-point operations can be handled by recompiling.
- String instructions are not implemented on the e500; therefore, trap emulation must be provided to ensure backward compatibility.

5.13.1.2 Supervisor Instruction Set

The supervisor mode instruction set defined by the PowerPC architecture is compatible with the e500 with the following exceptions:

- The MMU architecture is different, so some TLB manipulation instructions have different semantics.
- Instructions that support the BATs and segment registers are not implemented.

5.13.2 Memory Subsystem

The architecture provides separate instruction and data memory resources. The e500 provides additional cache control features, including cache locking.

5.13.3 Exception Handling

Exception handling is generally the same as that defined in the AIM version of the architecture for the e500, with the following differences:

- The critical interrupt provides an extra level of interrupt nesting. The critical interrupt includes external critical and watchdog timer time-out inputs.
- The machine check exception uses the Return from Machine Check Interrupt instruction, **rfmci**, and two machine check save/restore registers, MCSRR0 and MCSRR1.
- IVPR and IVORs set interrupt vectors individually, but they can be set to the address offsets defined in the OEA to provide compatibility.
- The embedded category does not define a reset vector; execution begins at a fixed virtual address, 0xFFFF_FFFC.
- Timer services are generally compatible, although the embedded category defines a new decrementer auto reload feature, the fixed-interval timer critical interrupt, and the watchdog timer interrupt, which are implemented in the e500 core.

An overview of the interrupt and exception handling capabilities of the e500 core can be found in Section 5.8, “Interrupts and Exception Handling.”

5.13.4 Memory Management

The embedded category defines resources for fixed 4-Kbyte pages and multiple, variable page sizes that can be configured in a single implementation. TLB management is provided with new instructions and SPRs.

5.13.5 Reset

Embedded category–compliant cores do not share a common reset vector with the AIM version of the architecture. Instead, at reset fetching begins at address 0xFFFF_FFFC. In addition, the Freescale MMU category defines specific aspects of the MMU page translation and protection mechanisms. Unlike the AIM version of the core, as soon as instruction fetching begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

MMU operations are described in the EREF.

5.13.6 Little-Endian Mode

Unlike the AIM version of the architecture, where little-endian mode is controlled on a system basis, the embedded category allows control of byte ordering on a memory page basis. In addition, the little-endian byte ordering used is true little-endian.

5.14 PowerQUICC III Implementation Details

Table 5-8 summarizes details of the PowerQUICC III-specific implementation of the e500 core.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation

Feature	PowerQUICC III Implementation
Cache protocol	The L2 cache is a write-through cache and does not support MESI cache protocol.
Nexus support	Nexus is not supported. The Nexus processor ID register (NPIDR) and the Nexus bus enable bit (HID1[NEXEN]) are not supported.
R1 and R2 data bus parity	R1 and R2 data bus parity are disabled on PowerQUICC III devices. HID1[R1DPE,R2DPE] are reserved.
Dynamic bus snooping	The PowerQUICC III devices do not perform dynamic bus snooping as described here. That is, when the e500 core is in core-stopped state (which is the state of the core when the PowerQUICC III device is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, L1 caches should be flushed if coherency is required during these power-down modes. For more information, see Section 23.5.1.9, “Snooping in Power-Down Modes.”
Supported TCR[WRC]	PowerQUICC III devices define values for 00, 01, 10, and 11, as described in Table 6-9 on page 6-15.
SPE and floating-point categories	The SPE (which includes the embedded vector and scalar floating-point instructions) are not implemented in the next generation of PowerQUICC devices. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses these instructions at the assembly level or that uses SPE or floating-point intrinsics requires rewriting for upward compatibility with next generation PowerQUICC devices. The e500v2 core implements SPE double-precision floating-point instructions. Freescale Semiconductor offers a libcfsl_e500 library that uses SPE instructions. Freescale Semiconductor will also provide future libraries to support next generation PowerQUICC devices.
HID0 implementation	SEL_TBCLK bit. Selects time base clock. If this bit is set and the time base is enabled, the time base is based on the TBCLK input, which on the PowerQUICC III devices is RTC.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation (continued)

Feature	PowerQUICC III Implementation
HID1 Implementation	PLL_MODE. Set to 01 PLL_CFG. This PowerQUICC III device supports the following: 0000_11 Ratio of 3:2 (1.5:1) 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) NEXEN, R1DPE, R2DPE, MPXTT, MSHARS, SSHAR, ATS, and MID are not implemented On PowerQUICC III devices, ABE must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.
	Please refer to the description of HID1[RFXE] in Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." If RFXE is 0, conditions that cause the assertion of $\overline{\textit{core\_fault\_in}}$ cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured: • ECM mapping errors: EEER[LAEE] must be set. See Section 8.2.1.6, "ECM Error Enable Register (EEER)." • L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. See Section 7.3.1.5, "L2 Error Registers." • DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See Section 9.4.1, "Register Descriptions." • PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated. Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that an parity errors can generate an interrupt. See Section 14.3.1.10, "Transfer Error Check Disable Register (LTEDR)," and Section 14.3.1.11, "Transfer Error Interrupt Enable Register (LTEIR)."
PIR value	The PIR value corresponds to the core number on multicore PowerQUICC III devices.
PVR value	The PVR reset value is 0x80*nn_nnnn.* See Table 5-1 for specific values. PVR[VERSION] = 0x80*nn* PVR[REVISION] = 0x*nnnn*
SVR value	The SVR reset value is 0x80*nn_nnnn.* See Table 5-1 for specific values.
Alternate time base	The alternate time base defines a time base counter similar to the time base defined in architecture. It is intended to be used for measuring time in implementation defined intervals. It differs from the defined Time Base in that it is not writable and always counts up, wrapping when the 64-bit count overflows. It defines two SPRs, ATBL (SPR 526) and ATBL (SPR 527).

Chapter 6
Core Register Summary

This chapter describes the e500 register model and indicates whether each register is defined by the Power Architecture technology, by the Freescale embedded category implementation standards (EIS), or by the implementation. For the programmer, drawing this distinction indicates the degree to which code is portable among Freescale processors.

This chapter provides reference material—figures for each register and complete descriptions of register fields, including how the registers are accessed, reset values, and whether they can be accessed by user- and supervisor-level software. Detailed discussions of how these registers are used are provided in *EREF: A Reference for Freescale Book E and the e500 Core* and the *PowerPC™ e500 Core Family Reference Manual*.

Note that all registers described here are implemented in the hardware as part of the e500 core.

6.1 Overview

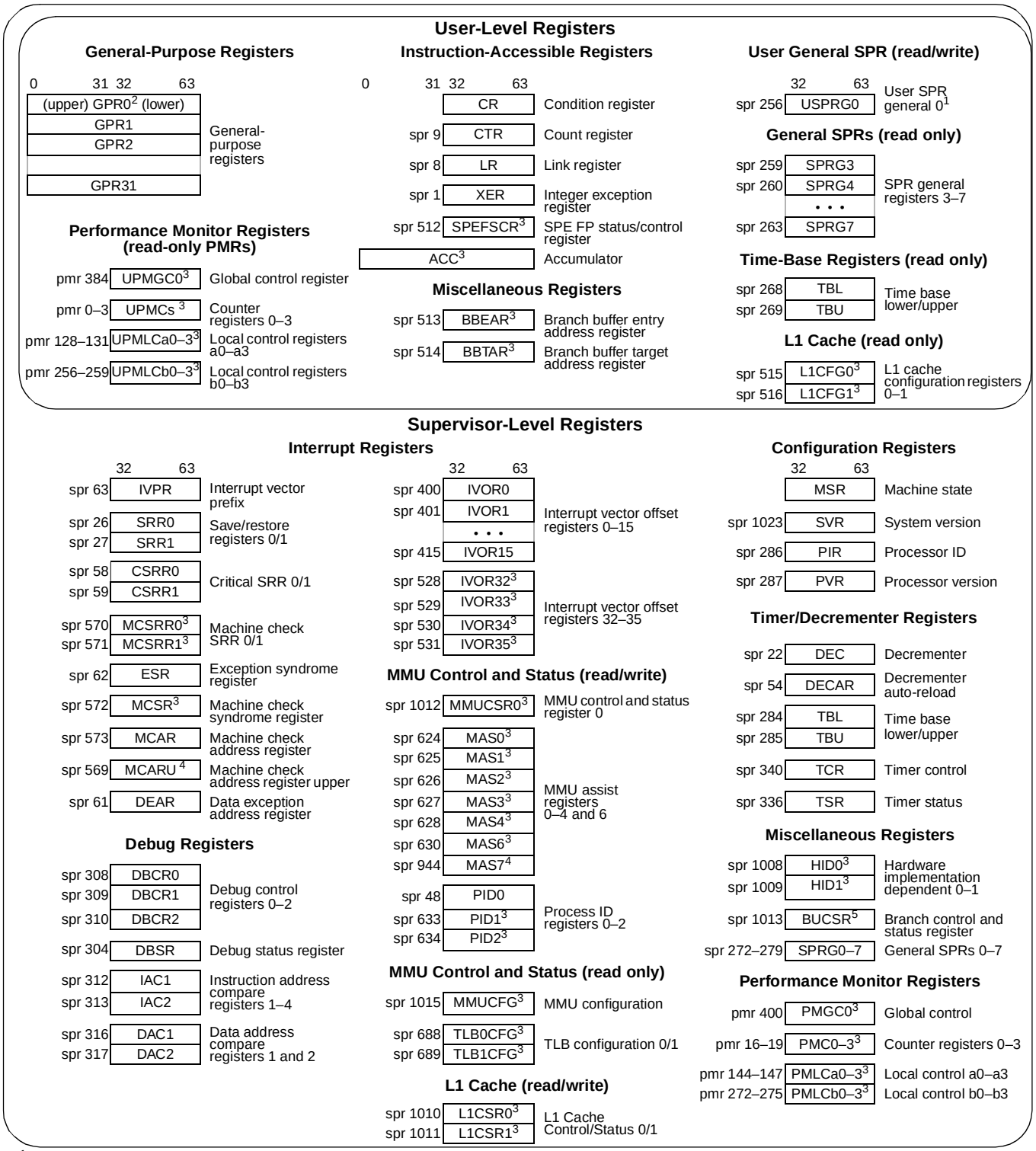
As shown in Figure 6-1, most of the registers implemented are defined by the architecture, and most of those were defined by the AIM definition of the architecture and have changed very little. Additional registers and fields within registers are defined by the EIS and by the implementation.

The Power Architecture technology defines some register fields in a very general way, leaving some details as implementation specific. In some cases, this more specific functionality is defined by the EIS; in others it is left up to the processor. This chapter identifies the level at which each features is defined.

References to e500 are true for both the e500v1 and e500v2.

6.1.1 Register Set

Table 6-1 shows the e500 register set, grouped by whether they can be accessed by user- or supervisor-level software. Unless otherwise indicated, these registers are defined by the base or embedded category.



1 USPRG0 is a separate physical register from SPRG0.
2 The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.
3 These registers are defined by the EIS and are not part of the Book E architecture.
4 e500v2 only
5 These registers are e500-specific

Figure 6-1. Core Register Model

6.2 Register Model for 32-Bit Implementations

Embedded 32-bit processors implement the following types of software-accessible registers:

- Architecture-defined registers that are accessed as part of instruction execution. These include the following:
  - Registers used for computation. These include the following:
    - General-purpose registers (GPRs)—The 32 GPRs hold source and destination operands for load, store, arithmetic, and computational instructions, and to read and write to other registers. The e500 implements these as 64-bit registers for use with 64-bit load, store, and merge instructions, as described in Section 6.3.1, “General-Purpose Registers (GPRs).”
    - Integer exception register (XER)—Bits in this register are set based on the operation of an instruction considered as a whole, not on intermediate results. (For example, the Subtract from Carrying instruction (**subfc**), the result of which is specified as the sum of three values, sets bits in the XER based on the entire operation, not on an intermediate sum.)

    These registers are described in Section 6.3, “Registers for Computational Operations.”
  - Condition register (CR)—Used to record conditions such as overflows and carries that occur as a result of executing arithmetic instructions (including those implemented by the SPE). The CR is described in Section 6.4, “Registers for Branch Operations.”
  - Machine state register (MSR)—Used by the operating system to configure parameters such as user/supervisor mode, address space, and enabling of asynchronous interrupts. This register is described in Section 6.5.1, “Machine State Register (MSR),” grouped with processor control SPRs.
- Special-purpose registers (SPRs) are accessed explicitly using **mtspr** and **mfspr** instructions. These registers are listed in Table 6-1 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale EIS– and e500-defined SPRs that are accessed explicitly using **mtspr** and **mfspr** are listed in Table 6-2 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale EIS–defined performance monitor registers (PMRs). These registers are similar to SPRs, but are accessed with Freescale EIS–defined move to and move from PMR instructions (**mtpmr** and **mfpmr**).

In this chapter, SPRs are grouped by function as follows:

- Section 6.4, “Registers for Branch Operations,” describes the count register (CTR) and the link register (LR).
- Section 6.5, “Processor Control Registers”
- Section 6.6, “Timer Registers”
- Section 6.7, “Interrupt Registers”
- Section 6.8, “Software-Use SPRs (SPRG0–SPRG7 and USPRG0),” describes SPRs defined for software use.
- Section 6.9, “Branch Target Buffer (BTB) Registers,” describes e500-specific registers defined to support the e500 tabs.
- Section 6.10, “Hardware Implementation-Dependent Registers,” describes HID0 and HID1.
- Section 6.11, “L1 Cache Configuration Registers”

- Section 6.12, “MMU Registers”
- Section 6.13, “Debug Registers”
- Section 6.14, “Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)”

The e500 core implements 64-bit GPRs, the upper 32 bits of which are used only with 64-bit load, store, and merge instructions.

6.2.1 Special-Purpose Registers (SPRs)

Table 6-1 summarizes SPRs. The SPR numbers are used in the instruction mnemonics. Bit 5 in an SPR number indicates whether an SPR is accessible from user- or supervisor-level software. An **mtspr** or **mfspr** instruction that specifies an unsupported SPR number is considered an invalid instruction.

In Table 6-1 and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

NOTE

Writing to the following registers requires synchronization, as described in the “Synchronization Requirements” section in the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

- BTB locking registers—BBEAR, BBTAR, and BUCSR
- DBCR*n*
- HID*n*
- L1CSR*n*
- MMU registers—MAS*n*, MMUCSR0, PID*n*
- SPEFSCR

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
CSRR0	Critical save/restore register 0	58	00001 11010	Read/Write	Yes	6.7.1.1/6-17
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-17
CTR	Count register	9	00000 01001	Read/Write	No	6.4.3/6-11

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
DAC1	Data address compare 1	316	01001 11100	Read/Write	Yes	6.13.4/6-45
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0 [1]	308	01001 10100	Read/Write	Yes	6.13.1/6-39
DBCR1	Debug control register 1 [1]	309	01001 10101	Read/Write	Yes	
DBCR2	Debug control register 2 [1]	310	01001 10110	Read/Write	Yes	
DBSR	Debug status register	304	01001 10000	w1c[2]	Yes	6.13.2/6-43
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.7.1.5/6-18
DEC	Decrementer	22	00000 10110	Read/Write	Yes	6.6.4/6-16
DECAR	Decrementer auto-reload	54	00001 10110	Write only		
ESR	Exception syndrome register	62	00001 11110	Read/Write	Yes	6.7.1.8/6-19
IAC1	Instruction address compare 1	312	01001 11000	Read/Write	Yes	6.13.3/6-45
IAC2	Instruction address compare 2	313	01001 11001			
IVOR0	Critical input	400	01100 10000	Read/Write	Yes	6.7.1.7/6-18
IVOR1	Machine check interrupt offset	401	01100 10001			
IVOR2	Data storage interrupt offset	402	01100 10010			
IVOR3	Instruction storage interrupt offset	403	01100 10011			
IVOR4	External input interrupt offset	404	01100 10100			
IVOR5	Alignment interrupt offset	405	01100 10101			
IVOR6	Program interrupt offset	406	01100 10110			
IVOR8	System call interrupt offset	408	01100 11000			
IVOR10	Decrementer interrupt offset	410	01100 11010			
IVOR11	Fixed-interval timer interrupt offset	411	01100 11011			
IVOR12	Watchdog timer interrupt offset	412	01100 11100			
IVOR13	Data TLB error interrupt offset	413	01100 11101			
IVOR14	Instruction TLB error interrupt offset	414	01100 11110			
IVOR15	Debug interrupt offset	415	01100 11111			
IVPR	Interrupt vector	63	00001 11111	Read/Write	Yes	6.7.1.6/6-18
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-11
PID	Process ID register [3]	48	00001 10000	Read/Write	Yes	6.12.1/6-32
PIR	Processor ID register	286	01000 11110	Read only	Yes	6.5.2/6-13
PVR	Processor version register	287	01000 11111	Read only	Yes	6.5.3/6-13

Table 6-1. Base and Embedded Category Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
SPRG0	SPR general 0	272	01000 10000	Read/Write	Yes	6.8/6-22
SPRG1	SPR general 1	273	01000 10001	Read/Write	Yes	
SPRG2	SPR general 2	274	01000 10010	Read/Write	Yes	
SPRG3	SPR general 3	259	01000 00011	Read only	No[4]	
		275	01000 10011	Read/Write	Yes	
SPRG4	SPR general 4	260	01000 00100	Read only	No	
		276	01000 10100	Read/Write	Yes	
SPRG5	SPR general 5	261	01000 00101	Read only	No	6.8/6-22
		277	01000 10101	Read/Write	Yes	
SPRG6	SPR general 6	262	01000 00110	Read only	No	
		278	01000 10110	Read/Write	Yes	
SPRG7	SPR general 7	263	01000 00111	Read only	No	
		279	01000 10111	Read/Write	Yes	
SRR0	Save/restore register 0	26	00000 11010	Read/Write	Yes	6.7.1.1/6-17
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-17
TBL	Time base lower	268	01000 01100	Read only	No	6.6.3/6-16
		284	01000 11100	Write only	Yes	
TBU	Time base upper	269	01000 01101	Read only	No	
		285	01000 11101	Write only	Yes	
TCR	Timer control register	340	01010 10100	Read/Write	Yes	6.6.1/6-14
TSR	Timer status register	336	01010 10000	w1c[5]	Yes	6.6.2/6-15
USPRG0	User SPR general 0[6]	256	01000 00000	Read/Write	No	6.8/6-22
XER	Integer exception register	1	00000 00001	Read/Write	No	6.3.2/6-8

1 Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*

2 The DBSR is read using **mfspr**. It cannot be directly written to. Instead, DBSR bits corresponding to 1 bits in the GPR can be cleared using **mtspr**.

3 Implementations may support more than one PID. For implementations with multiple PIDs, the PID defined by the embedded category is PID0.

4 User-mode read access to SPRG3 is implementation dependent.

5 The TSR is read using **mfspr**. It cannot be directly written to. Instead, TSR bits corresponding to 1 bits in the GPR can be cleared using **mtspr**.

6 USPRG0 is a separate physical register from SPRG0.

Table 6-2 describes the implementation-specific SPRs and SPRs defined by categories other than the base and embedded categories. Compilers should recognize the mnemonic names given in Table 6-2 when parsing instructions.

Table 6-2. Additional SPRs (by SPR Abbreviation)

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
BBEAR	Branch buffer entry address register [1]	513	Read/Write	No	6.9.1/6-23
BBTAR	Branch buffer target address register [1]	514	Read/Write	No	6.9.2/6-23
BUCSR	Branch unit control and status register [1]	1013	Read/Write	Yes	6.9.3/6-24
HID0	Hardware implementation dependent reg 0 [1]	1008	Read/Write	Yes	6.10.1/6-25
HID1	Hardware implementation dependent reg 1 [1]	1009	Read/Write	Yes	6.10.2/6-26
IVOR32	SPE unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-18
IVOR33	Floating-point data exception interrupt offset	529	Read/Write	Yes	
IVOR34	Floating-point round exception interrupt offset	530	Read/Write	Yes	
IVOR35	Performance monitor	531	Read/Write	Yes	
L1CFG0	L1 cache configuration register 0	515	Read only	No	6.11.3/6-30
L1CFG1	L1 cache configuration register 1	516	Read only	No	6.11.4/6-31
L1CSR0	L1 cache control and status register 0 [1]	1010	Read/Write	Yes	6.11.1/6-28
L1CSR1	L1 cache control and status register 1 [1]	1011	Read/Write	Yes	6.11.2/6-29
MAS0	MMU assist register 0 [1]	624	Read/Write	Yes	6.12.5.1/6-34
MAS1	MMU assist register 1 [1]	625	Read/Write	Yes	6.12.5.2/6-35
MAS2	MMU assist register 2 [1]	626	Read/Write	Yes	6.12.5.3/6-36
MAS3	MMU assist register 3 [1]	627	Read/Write	Yes	6.12.5.4/6-37
MAS4	MMU assist register 4 [1]	628	Read/Write	Yes	6.12.5.5/6-38
MAS6	MMU assist register 6 [1]	630	Read/Write	Yes	6.12.5.6/6-38
MAS7	MMU assist register 7 [1]	944	Read/Write	Yes	6.12.5.7/6-39
MCAR	Machine check address register	573	Read only	Yes	6.7.2.3/6-21
MCARU	Machine check address register upper	569	Read only	Yes	6.7.2.3/6-21
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.4/6-21
MCSRR0	Machine check save/restore register 0	570	Read/Write	Yes	6.7.2.1/6-20
MCSRR1	Machine check save/restore register 1	571	Read/Write	Yes	6.7.2.2/6-20
MMUCFG	MMU configuration register	1015	Read only	Yes	6.12.3/6-32
MMUCSR0	MMU control and status register 0 [1]	1012	Read/Write	Yes	6.12.2/6-32

Table 6-2. Additional SPRs (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
PID0	Process ID register 0 [1]	48	Read/Write	Yes	6.12.1/6-32
PID1	Process ID register 1 [1]	633	Read/Write	Yes	
PID2	Process ID register 2 [1]	634	Read/Write	Yes	
SPEFSCR	Signal processing and embedded floating-point status and control register [1]	512	Read/Write	No	6.14/6-45
SVR	System version register	1023	Read only	Yes	6.5.4/6-14
TLB0CFG	TLB configuration register 0	688	Read only	Yes	6.12.4/6-33
TLB1CFG	TLB configuration register 1	689	Read only	Yes	6.12.4.2/6-34

[1] Accesses to this register requires synchronization, as described in the “Synchronization Requirements” section of the “Register Model” chapter of the *PowerPC™ e500 Core Family Reference Manual*.

6.3 Registers for Computational Operations

The following sections describe general-purpose and integer exception registers.

NOTE

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

6.3.1 General-Purpose Registers (GPRs)

GPR0–GPR31 support integer operations. The instruction formats provide 5-bit fields for specifying the GPRs to be used in the execution of the instruction. Each GPR is a 64-bit register, although only 64-bit load, store, and merge instructions use GPR bits 0–31.

6.3.2 Integer Exception Register (XER)

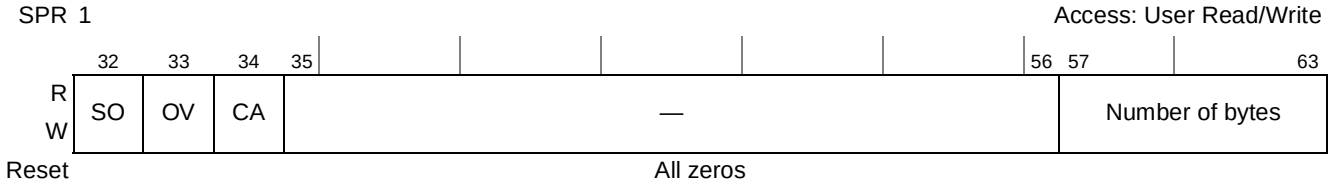


Figure 6-2. Integer Exception Register (XER)

Table 6-3. XER Field Description

Bits	Name	Description
32	SO	Summary overflow. Set when an instruction (except **mtspr**) sets the overflow bit. Once set, SO remains set until it is cleared by **mtspr[XER]** or **mcrxr**. SO is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow. Executing **mtspr[XER]**, supplying the values 0 for SO and 1 for OV, causes SO to be cleared and OV to be set.
33	OV	Overflow. X-form add, subtract from, and negate instructions having OE = 1 set OV if the carry out of bit 32 is not equal to the carry out of bit 33, and clear OV otherwise to indicate a signed overflow. X-form multiply low word and divide word instructions having OE = 1 set OV if the result cannot be represented in 32 bits (**mullwo**, **divwo**, and **divwuo**) and clear OV otherwise. OV is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow.
34	CA	Carry. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA if there is a carry out of bit 32 and clear it otherwise. CA can be used to indicate unsigned overflow for add and subtract operations that set CA. Shift right algebraic word instructions set CA if any 1 bits are shifted out of a negative operand and clear CA otherwise. Compare instructions and instructions that cannot carry (except Shift Right Algebraic Word, **mtspr[XER]**, and **mcrxr**) do not affect CA.
35–56	—	Reserved, should be cleared.
57–63	No. of bytes	Supports emulation of load and store string instructions. Specifies the number of bytes to be transferred by a load string indexed or store string indexed instruction.

6.4 Registers for Branch Operations

This section describes registers that support branch and CR operations.

6.4.1 Condition Register (CR)

Access: User Read/Write

	32–35	36–39	40–43	44–47	48–51	52–55	56–59	60–63
R W	CR0	CR1	CR2	CR3	CR4	CR5	CR6	CR7
Reset	All zeros							

Figure 6-3. Condition Register (CR)

Table 6-4. BI Operand Settings for CR Fields

CR*n* Bits	CR Bits	BI	Description
CR0[0]	32	00000	Negative (LT)—Set when the result is negative. For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR0[1]	33	00001	Positive (GT)—Set when the result is positive (and not zero). For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR0[2]	34	00010	Zero (EQ)—Set when the result is zero. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.

Table 6-4. BI Operand Settings for CR Fields (continued)

CR*n* Bits	CR Bits	BI	Description
CR0[3]	35	00011	Summary overflow (SO). Copy of XER[SO] at the instruction's completion. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CR1[0]	36	00100	Negative (LT) For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR1[1]	37	00101	Positive (GT) For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR1[2]	38	00110	Zero (EQ) For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR1[3]	39	00111	Summary overflow (SO) For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.
CR*n*[0]	40 44 48 52 56 60	01000 01100 10000 10100 11000 11100	Less than (LT) For integer compare instructions: **r**A < SIMM or rB (signed comparison) or **r**A < UIMM or **r**B (unsigned comparison). For SPE vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR*n*[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than (GT) For integer compare instructions: **r**A > SIMM or **r**B (signed comparison) or **r**A > UIMM or **r**B (unsigned comparison). For SPE vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR*n*[2]	42 46 50 54 58 62	01010 01110 10010 10110 11010 11110	Equal (EQ) For integer compare instructions: **r**A = SIMM, UIMM, or **r**B. For SPE vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR*n*[3]	43 47 51 55 59 63	01011 01111 10011 10111 11011 11111	Summary overflow (SO) For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction. For SPE vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements.

The bits of CR0 are interpreted as described in Table 6-5.

Table 6-5. CR0 Bit Descriptions

CR Bit	Name	Description
32	Negative (LT)	Bit 32 of the result is equal to 1.
33	Positive (GT)	Bit 32 of the result is equal to 0 and at least one bit from 33–63 of the result is non-zero.
34	Zero (EQ)	Bits 32–63 of the result are equal to 0.
35	Summary overflow (SO)	This is a copy of the final state of XER[SO] at the completion of the instruction.

6.4.2 Link Register (LR)

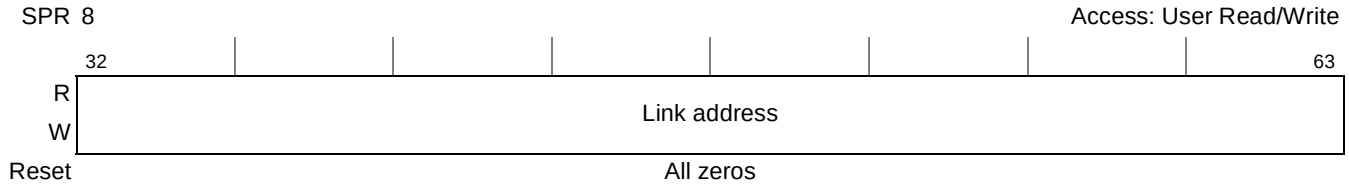


Figure 6-4. Link Register (LR)

6.4.3 Count Register (CTR)

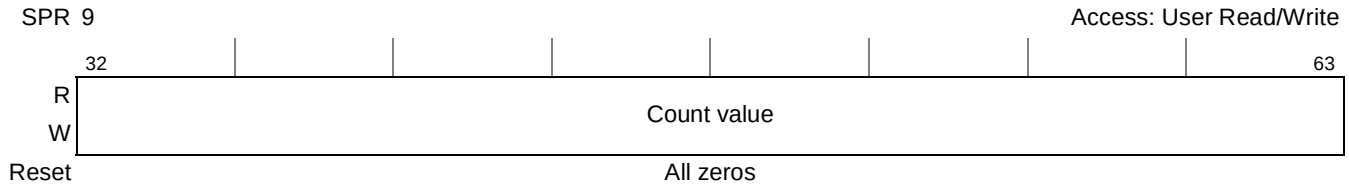


Figure 6-5. Count Register (CTR)

6.5 Processor Control Registers

This section addresses machine state, processor ID, and processor version registers.

6.5.1 Machine State Register (MSR)

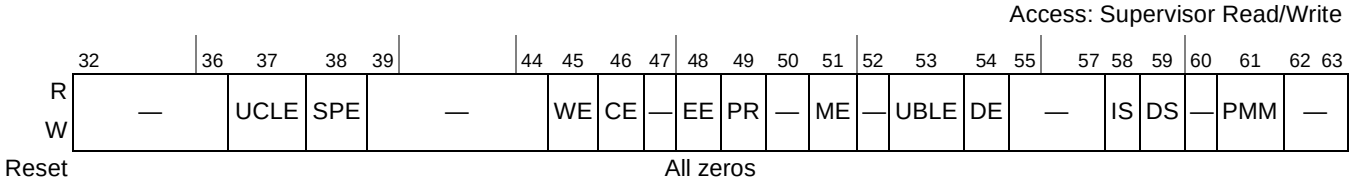


Figure 6-6. Machine State Register (MSR)

Table 6-6. MSR Field Descriptions

Bits	Name	Description
32–36	—	Reserved, should be cleared.[1]
37	UCLE	User-mode cache lock enable. Used to restrict user-mode cache-line locking by the operating system 0 Any cache lock instruction executed in user-mode takes a cache-locking DSI exception and sets either ESR[DLK] or ESR[ILK]. This allows the operating system to manage and track the locking/unlocking of cache lines by user-mode tasks. 1 Cache-locking instructions can be executed in user-mode and they do not take a DSI for cache-locking (they may still take a DSI for access violations though).
38	SPE	SPE enable. (e500-specific). 0 If software attempts to execute an instruction that accesses the upper word of a GPR, the SPE unavailable exception is taken. 1 Software can execute the following instructions: These instructions include the SPE instructions, embedded double-precision, and single-precision vector floating-point instructions. (That is, all instructions that access the upper half of the 64-bit GPRs.)
39–44	—	Reserved, should be cleared.[1]
45	WE	Wait state enable. Allows the core complex to signal a request for power management, according to the states of HID0[DOZE], HID0[NAP], and HID0[SLEEP]. 0 The processor is not in wait state and continues processing. No power management request is signaled to external logic. 1 The processor enters wait state by ceasing to execute instructions and entering low-power mode. Details of how wait state is entered and exited and how the processor behaves in the wait state are implementation-dependent. On the e500, MSR[WE] gates the DOZE, NAP, and SLEEP outputs from the core complex; as a result, these outputs negate to the external power management logic on entry to the interrupt and then return to their previous state on return from the interrupt. WE is cleared on entry to any interrupt and restored to its previous state upon return.
46	CE	Critical enable 0 Critical input and watchdog timer interrupts are disabled. 1 Critical input and watchdog timer interrupts are enabled.
47	—	Reserved, should be cleared.[1]
48	EE	External enable 0 External input, decrementer, fixed-interval timer, and performance monitor interrupts are disabled. 1 External input, decrementer, fixed-interval timer, and performance monitor interrupts are enabled.
49	PR	User mode (problem state) 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (for example, GPRs, SPRs, and the MSR). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource. PR also affects memory access control
50	—	Reserved, should be cleared.[1]
51	ME	Machine check enable 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
52	—	Reserved, should be cleared.[1]
53	UBLE	In the e500, it is the user BTB lock enable bit. 0 User-mode execution of the BTB lock instructions is disabled; privileged instruction exception taken instead. 1 User-mode execution of the BTB lock instructions for user mode is enabled.

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
54	DE	Debug interrupt enable. See the description of the DBSR[UDE] in Section 6.13.2, "Debug Status Register (DBSR)." 0 Debug interrupts are disabled. 1 Debug interrupts are enabled if DBCR0[IDM] = 1.
55–57	—	Reserved, should be cleared.[1]
58	IS	Instruction address space 0 The processor directs all instruction fetches to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs all instruction fetches to address space 1 (TS = 1 in the relevant TLB entry).
59	DS	Data address space 0 The processor directs data memory accesses to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs data memory accesses to address space 1 (TS = 1 in the relevant TLB entry).
60	—	Reserved, should be cleared.[1]
61	PMM	Performance monitor mark bit. System software can set PMM when a marked process is running to enable statistics to be gathered only during execution of the marked process. MSR[PR] and MSR[PMM] together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCax, the state for which monitoring is enabled, counting is enabled.
62–63	—	Preserved for OEA-defined RI and LE, respectively

[1] An MSR bit that is reserved may be altered by a return from interrupt instruction.

6.5.2 Processor ID Register (PIR)

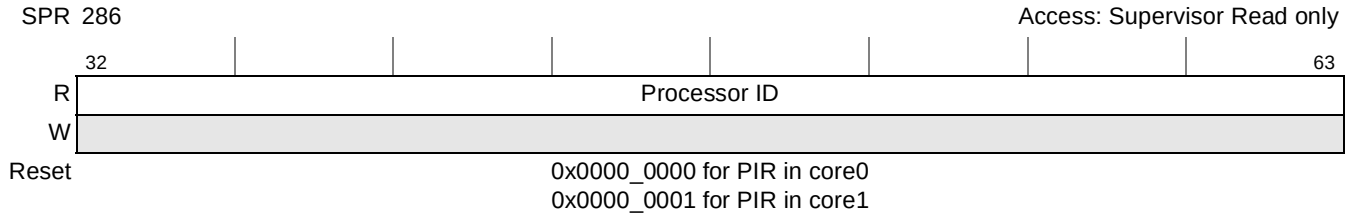


Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

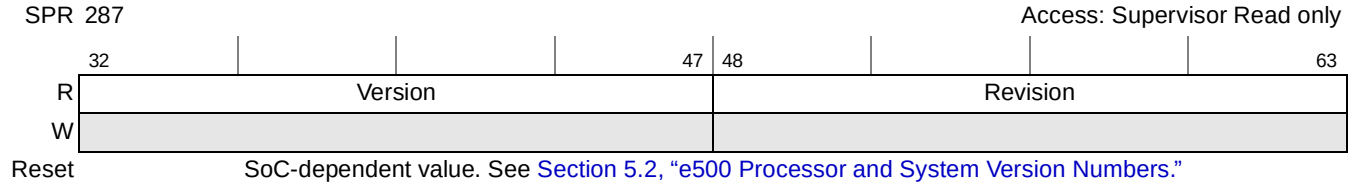


Figure 6-8. Processor Version Register (PVR)

Table 6-7. PVR Field Descriptions

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)
48–63	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)

6.5.4 System Version Register (SVR)

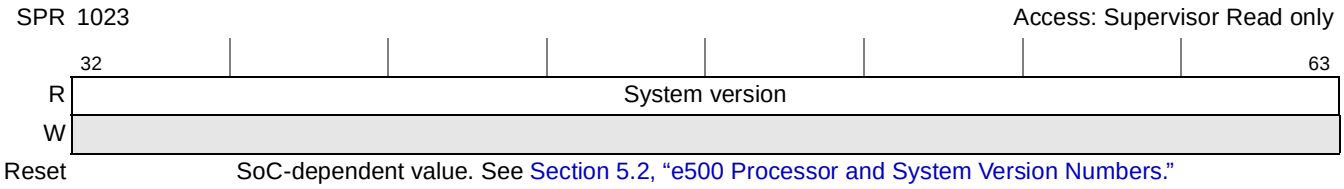


Figure 6-9. System Version Register (SVR)

Table 6-8. SVR Field Descriptions

Bits	Name	Description
32–63	System version	A 16-bit number that identifies the SoC version. See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.

6.6 Timer Registers

6.6.1 Timer Control Register (TCR)

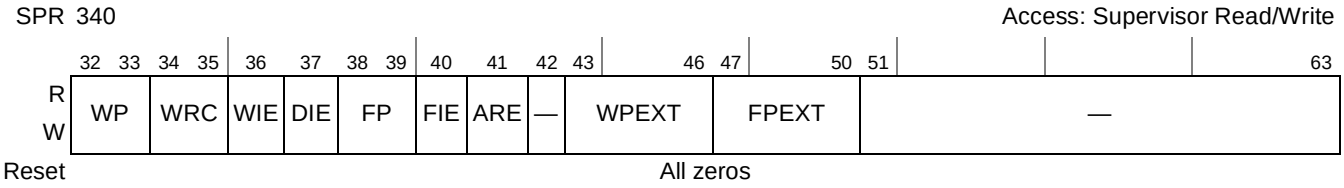


Figure 6-10. Timer Control Register (TCR)

Table 6-9. TCR Field Descriptions

Bits	Name	Description
32–33	WP	Watchdog timer period. When concatenated with WPEXT, specifies one of 64-bit locations of the time base used to signal a watchdog timer exception on a transition from 0 to 1. WPEXT[0–3] || WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) WPEXT[0–3] || WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
34–35	WRC	Watchdog timer reset control. This value is written into TSR[WRS] when a watchdog event occurs. WRC may be set by software but cannot be cleared by software, except by a software-induced reset. Once written to a non-zero value, WRC may no longer be altered by software. 00 No watchdog timer reset occurs 01 Second timeout generates machine check 10 Second timeout generates HRESET_REQ output externally 11 Second timeout automatically resets the given processor core
36	WIE	Watchdog timer interrupt enable 0 Watchdog timer interrupts disabled 1 Watchdog timer interrupts enabled
37	DIE	Decrementer interrupt enable 0 Decrementer interrupts disabled 1 Decrementer interrupts enabled
38–39	FP	Fixed interval timer period. When concatenated with FPEXT, FP specifies one of 64 bit locations of the time base used to signal a fixed-interval timer exception on a transition from 0 to 1. FPEXT[0–3] || FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) FPEXT[0–3] || FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
40	FIE	Fixed interval interrupt enable 0 Fixed interval interrupts disabled 1 Fixed interval interrupts enabled
41	ARE	Auto-reload enable. Controls whether the DECAR value is reloaded into the DEC when the DEC value reaches 0000_0001. See *EREF: A Reference for Freescale Book E and the e500 Core*. 0 Auto-reload disabled 1 Auto-reload enabled
42	—	Reserved, should be cleared.
43–46	WPEXT	Watchdog timer period extension (see the description for WP)
47–50	FPEXT	Fixed-interval timer period extension (see the description for FP)
51–63	—	Reserved, should be cleared.

6.6.2 Timer Status Register (TSR)

SPR 336 Access: Supervisor w1c[1]

	32	33	34 35	36	37	38 63
R	ENW	WIS	WRS	DIS	FIS	—
W	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros					

[1] Set by hardware. Read with **mfspr** and cleared with **mtspr** by writing ones to any TSR bit positions to be cleared and zeros in all other bit positions.

Figure 6-11. Timer Status Register (TSR)

Table 6-10. TSR Field Descriptions

Bits	Name	Description
32	ENW	Enable next watchdog time. Functions as write-one-to-clear. 0 Action on next watchdog timer time-out is to set TSR[ENW] 1 Action on next watchdog timer time-out is governed by TSR[WIS] When a watchdog timer time-out occurs while WIS = 0 and the next watchdog time-out is enabled (ENW = 1), a watchdog timer exception is generated and logged by setting WIS. This is referred to as a watchdog timer first time out. A watchdog timer interrupt occurs if enabled by TCR[WIE] and MSR[CE]. To avoid another watchdog timer interrupt once MSR[CE] is reenabled, (assuming TCR[WIE] is not cleared instead), the interrupt handler must reset TSR[WIS].
33	WIS	Watchdog timer interrupt status. Functions as write-one-to-clear. 0 A watchdog timer event has not occurred. 1 A watchdog timer event occurred. When MSR[CE] = 1 and TCR[WIE] = 1, a watchdog timer interrupt is taken. See the description of ENW for more information about how WIS is used.
34–35	WRS	Watchdog timer reset status. Functions as write-one-to-clear. Defined at reset (value = 00). Set to TCR[WRC] when a reset is caused by the watchdog timer.
36	DIS	Decrementer interrupt status. Functions as write-one-to-clear. 0 A decrementer event has not occurred. 1 A decrementer event occurred. When MSR[EE] = TCR[DIE] = 1, a decrementer interrupt is taken.
37	FIS	Fixed-interval timer interrupt status. Functions as write-one-to-clear. 0 A fixed-interval timer event has not occurred. 1 A fixed-interval timer event occurred. When MSR[EE] = 1 and TCR[FIE] = 1, a fixed-interval timer interrupt is taken.
38–63	—	Reserved, should be cleared.

6.6.3 Time Base Registers

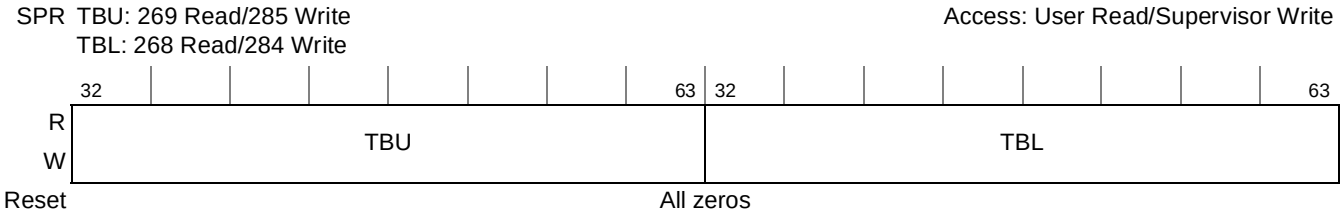


Figure 6-12. Time Base Upper/Lower Registers (TBU/TBL)

6.6.4 Decrementer Register

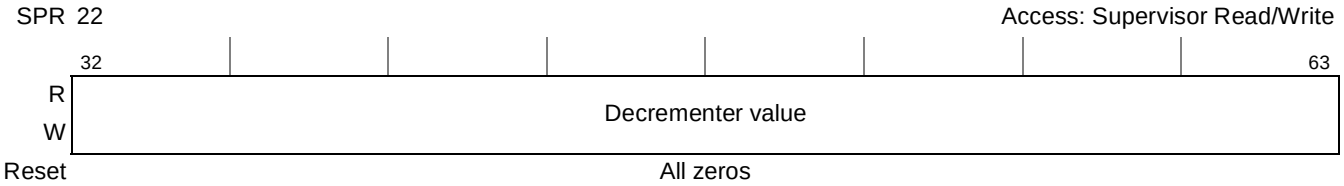


Figure 6-13. Decrementer Register (DEC)

6.6.5 Decrementer Auto-Reload Register (DECAR)

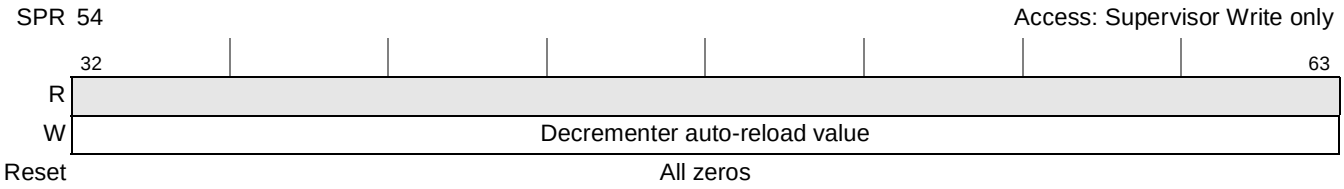


Figure 6-14. Decrementer Auto-Reload Register (DECAR)

6.7 Interrupt Registers

6.7.1 Interrupt Registers Defined by the Embedded and Base Categories

6.7.1.1 Save/Restore Register 0 (SRR0)

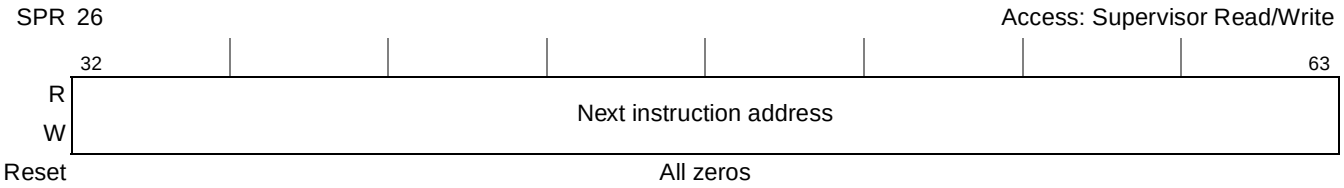


Figure 6-15. Save/Restore Register 0 (SRR0)

6.7.1.2 Save/Restore Register 1 (SRR1)

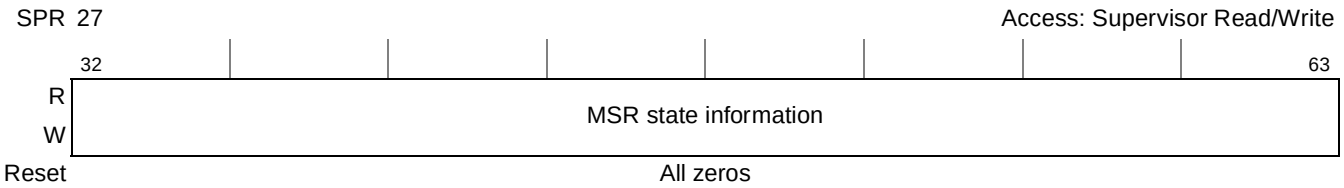


Figure 6-16. Save/Restore Register 1 (SRR1)

6.7.1.3 Critical Save/Restore Register 0 (CSRR0)

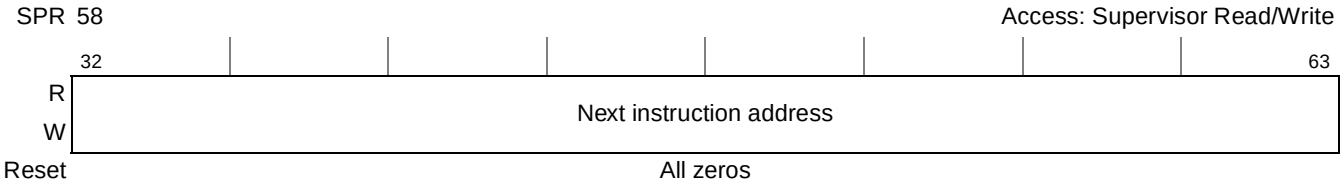


Figure 6-17. Critical Save/Restore Register 0 (CSRR0)

6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

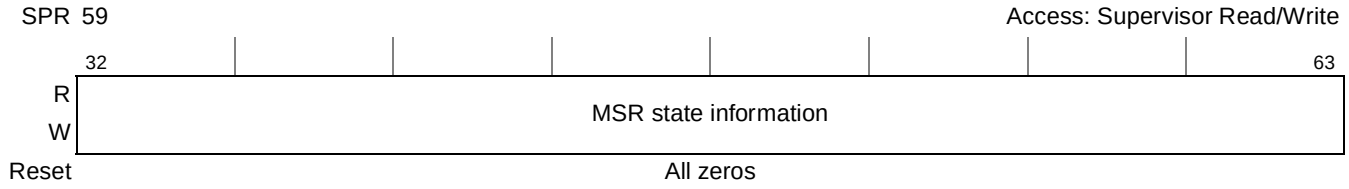


Figure 6-18. Critical Save/Restore Register 1 (CSRR1)

6.7.1.5 Data Exception Address Register (DEAR)

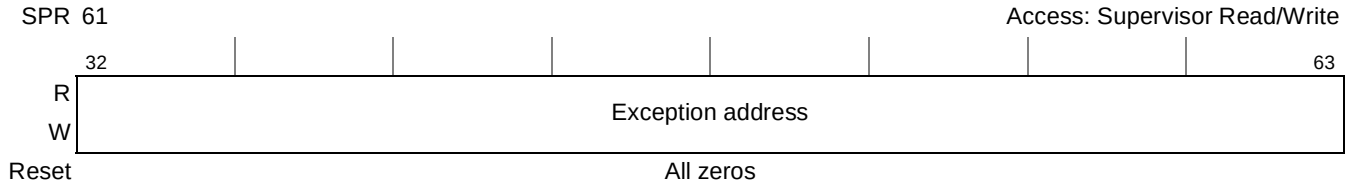


Figure 6-19. Data Exception Address Register (DEAR)

6.7.1.6 Interrupt Vector Prefix Register (IVPR)

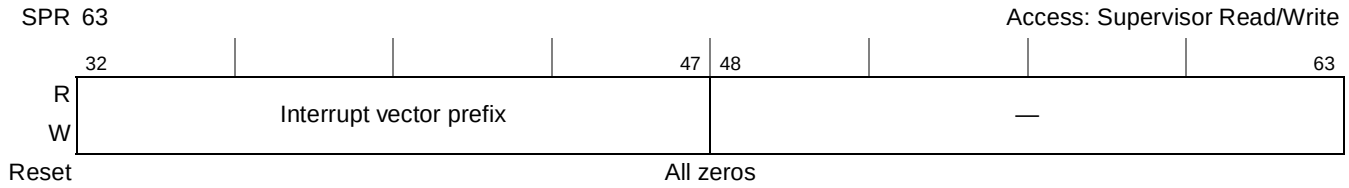


Figure 6-20. Interrupt Vector Prefix Register (IVPR)

6.7.1.7 Interrupt Vector Offset Registers (IVOR*n*)

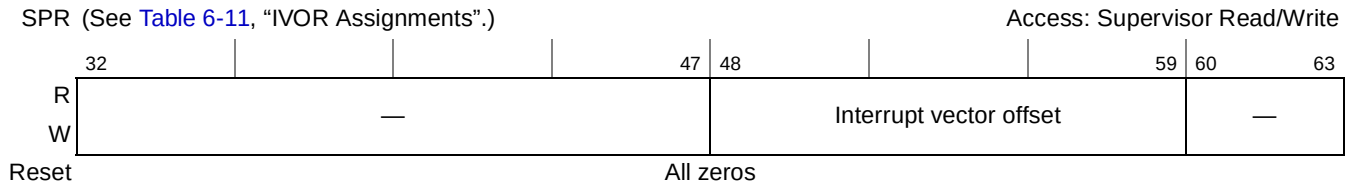


Figure 6-21. Interrupt Vector Offset Registers (IVOR*n*)

Table 6-11. IVOR Assignments

IVOR Number	SPR	Interrupt Type
IVOR0	400	Critical input
IVOR1	401	Machine check
IVOR2	402	Data storage
IVOR3	403	Instruction storage
IVOR4	404	External input

Table 6-11. IVOR Assignments (continued)

IVOR Number	SPR	Interrupt Type
IVOR5	405	Alignment
IVOR6	406	Program
IVOR8	408	System call
IVOR10	410	Decrementer
IVOR11	411	Fixed-interval timer interrupt
IVOR12	412	Watchdog timer interrupt
IVOR13	413	Data TLB error
IVOR14	414	Instruction TLB error
IVOR15	415	Debug
IVOR16–IVOR31	—	Reserved for future architectural use
IVOR32	528	SPE unavailable
IVOR33	529	Floating-point data exception
IVOR34	530	Floating-point round exception
IVOR35	531	Performance monitor
IVOR36–IVOR63	—	Allocated for implementation-dependent use

6.7.1.8 Exception Syndrome Register (ESR)

SPR 62 — Access: Supervisor Read/Write

	32–35	36	37	38	39	40	41	42	43	44–45	46	47–55	56–57	58–63
R W	—	PIL	PPR	PTR	—	ST	—	DLK	ILK	—	BO	—	SPE	—
Reset	All zeros													

Figure 6-22. Exception Syndrome Register (ESR)

Table 6-12. ESR Field Descriptions

Bits	Name	Syndrome	Interrupt Types
32–35	—	Reserved, should be cleared.	—
36	PIL	Illegal instruction exception	Program
37	PPR	Privileged instruction exception	Program
38	PTR	Trap exception	Program
39	—	Reserved and permanently cleared because the e500 does not implement this FPU. Setting it has no effect.	—
40	ST	Store operation	Alignment, data storage, data TLB error

Table 6-12. ESR Field Descriptions (continued)

Bits	Name	Syndrome	Interrupt Types
41	—	Reserved, should be cleared.	—
42	DLK	Cache locking. Settings are implementation-dependent. 0 Default 1 On the e500, DLK is set when a DSI occurs because **dcbtls**, **dcbtstls**, or **dcblc** is executed in user mode while MSR[UCLE] = 0.	Data storage
43	ILK	Set when a DSI occurs because **icbtl** or **icblc** is executed in user mode (MSR[PR] = 1) and MSR[UCLE] = 0	Data storage
44–45	—	Reserved, should be cleared.	—
46	BO	Byte-ordering exception	Data storage, instruction storage
47–55	—	Reserved, should be cleared.	—
56	SPE	SPE exception bit (e500-specific) 0 Default 1 Any exception caused by an SPE or SPFP instruction	SPE unavailable
57–63	—	Reserved, should be cleared.	—

6.7.2 Additional Interrupt Registers

6.7.2.1 Machine Check Save/Restore Register 0 (MCSRR0)

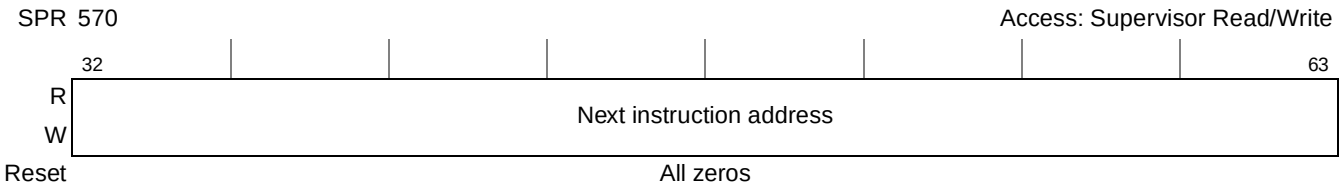


Figure 6-23. Machine Check Save/Restore Register 0 (MCSRR0)

6.7.2.2 Machine Check Save/Restore Register 1 (MCSRR1)

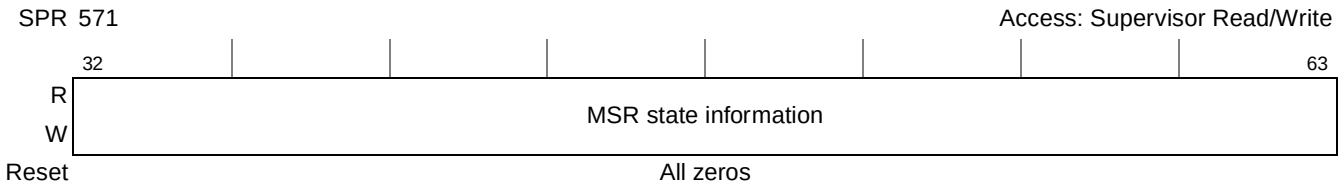


Figure 6-24. Machine Check Save/Restore Register 1 (MCSRR1)

6.7.2.3 Machine Check Address Register (MCAR/MCARU)

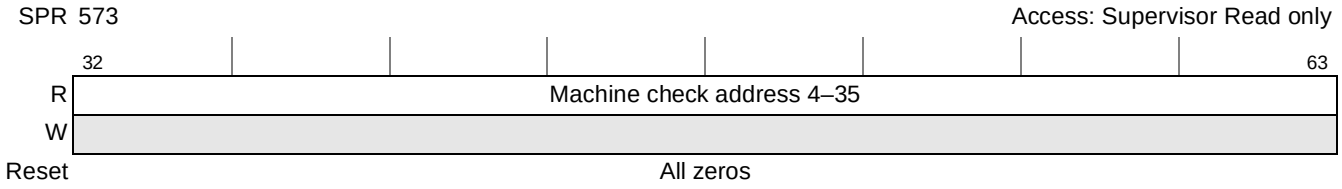


Figure 6-25. Machine Check Address Register (MCAR)

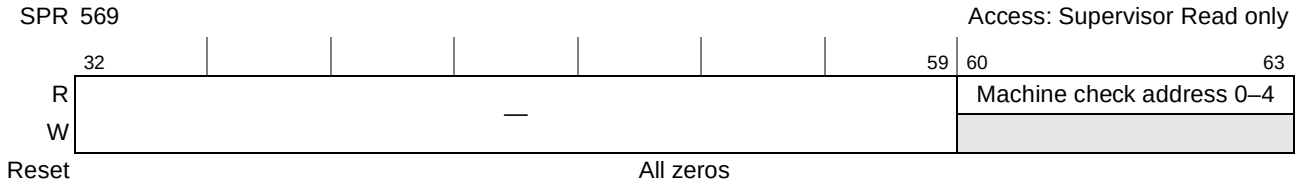


Figure 6-26. Machine Check Address Register Upper (MCARU)

6.7.2.4 Machine Check Syndrome Register (MCSR)

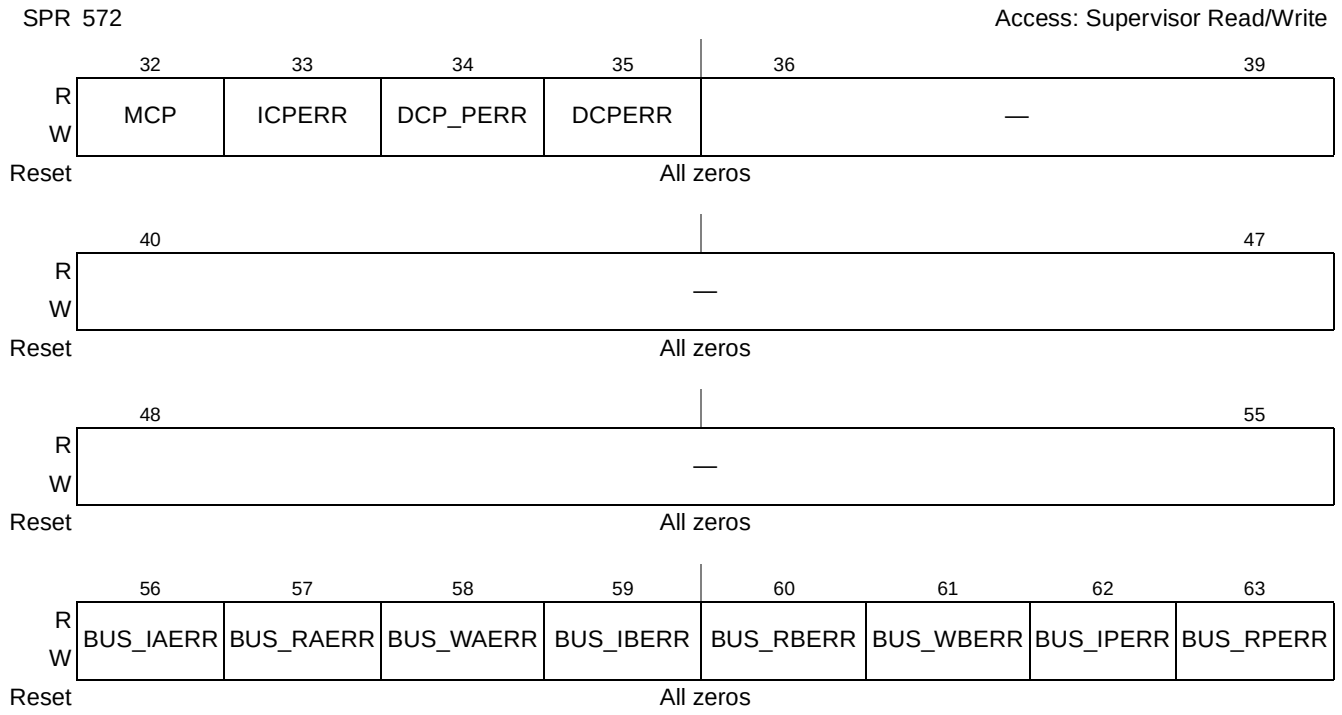


Figure 6-27. Machine Check Syndrome Register (MCSR)

Table 6-13. MCSR Field Descriptions

Bits	Name	Description
32	MCP	Machine check input pin
33	ICPERR	Instruction cache parity error

Table 6-13. MCSR Field Descriptions (continued)

Bits	Name	Description
34	DCP_PERR	Data cache push parity error
35	DCPERR	Data cache parity error
36–55	—	Reserved, should be cleared.
56	BUS_IAERR	Bus instruction address error
57	BUS_RAERR	Bus read address error
58	BUS_WAERR	Bus write address error
59	BUS_IBERR	Bus instruction data bus error
60	BUS_RBERR	Bus read data bus error
61	BUS_WBERR	Bus write bus error
62	BUS_IPERR	Bus instruction parity error
63	BUS_RPERR	Bus read parity error

6.8 Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

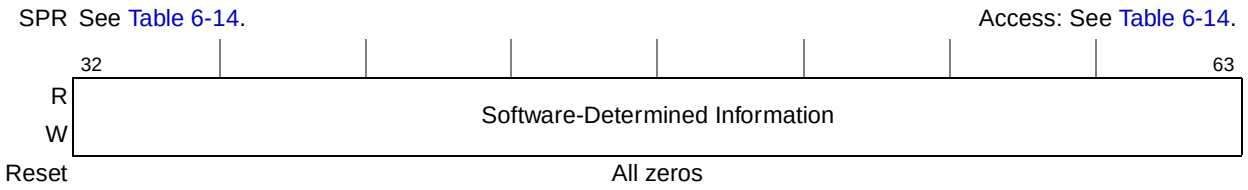


Figure 6-28. Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

Table 6-14. SPR Assignments

Name	SPR	Access	Level
SPRG0	272	Read/Write	Supervisor
SPRG1	273	Read/Write	Supervisor
SPRG2	274	Read/Write	Supervisor
SPRG3	259	Read only	User/Supervisor
	275	Read/Write	Supervisor
SPRG4	260	Read only	User/Supervisor
	276	Read/Write	Supervisor
SPRG5	261	Read only	User/Supervisor
	277	Read/Write	Supervisor
SPRG6	262	Read only	User/Supervisor
	278	Read/Write	Supervisor
SPRG7	263	Read only	User/Supervisor
	279	Read/Write	Supervisor
USPRG0	256	Read/Write	User/Supervisor

6.9 Branch Target Buffer (BTB) Registers

6.9.1 Branch Buffer Entry Address Register (BBEAR)

SPR 513 Access: User Read/Write

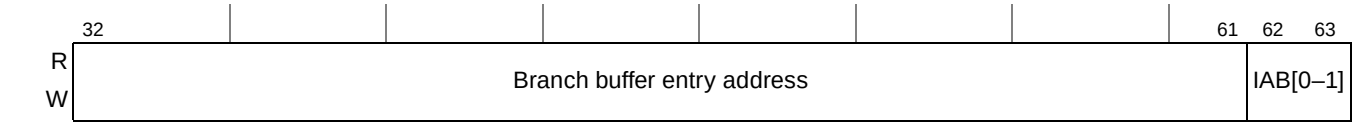


Figure 6-29. Branch Buffer Entry Address Register (BBEAR)

Table 6-15. BBEAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer entry address	Branch buffer entry effective address bits 0–29
62–63	IAB[0–1]	Instruction after branch (with BBTAR[62]). 3-bit pointer that points to the instruction in the cache line after the branch. See the description in the **bblels** instruction in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.

6.9.2 Branch Buffer Target Address Register (BBTAR)

SPR 514 Access: User Read/Write

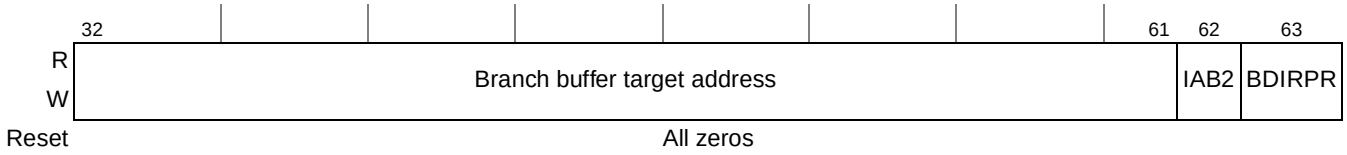


Figure 6-30. Branch Buffer Target Address Register (BBTAR)

Table 6-16. BBTAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer target address	Branch buffer target effective address bits 0–29
62	IAB2	Instruction after branch bit 2 (with BBEAR[62–63]). IAB is a 3-bit pointer that points to the instruction in the cache line after the branch. See the description for **bblels** in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.
63	BDIRPR	Branch direction prediction. The user can pick the direction of the predicted branch. 0 The locked address is always predicted as not taken. 1 The locked address is always predicted as taken.

6.9.3 Branch Unit Control and Status Register (BUCSR)

SPR 1013 Access: Supervisor Read/Write

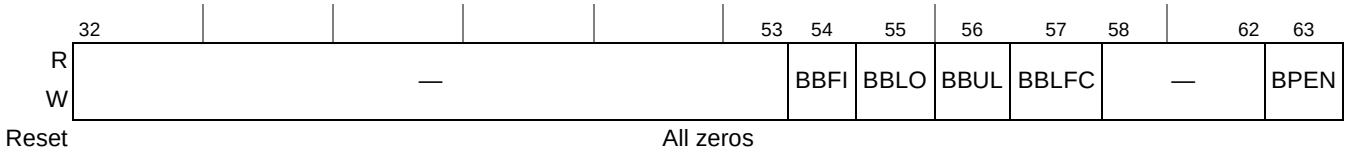


Figure 6-31. Branch Unit Control and Status Register (BUCSR)

Table 6-17. BUCSR Field Descriptions

Bits	Name	Description
32–53	—	Reserved, should be cleared.
54	BBFI	Branch buffer flash invalidate. Clearing and then setting BBFI flash clears the valid bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBFI is always read as 0.
55	BBLO	Branch buffer lock overflow status 0 Indicates a lock overflow condition was not encountered in the branch buffer 1 Indicates a lock overflow condition was encountered in the branch buffer This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
56	BBUL	Branch buffer unable to lock 0 Indicates a lock overflow condition in the branch buffer 1 Indicates a lock set instruction failed in the branch buffer, for example, if the BTB is disabled. This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
57	BBLFC	Branch buffer lock bits flash clear. Clearing and then setting BBLFC flash clears the lock bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBLFC is always read as 0.
58–62	—	Reserved, should be cleared.
63	BPEN	Branch prediction enable 0 Branch prediction disabled 1 Branch prediction enabled (enables BTB to predict branches)

6.10 Hardware Implementation-Dependent Registers

6.10.1 Hardware Implementation-Dependent Register 0 (HID0)

SPR 1008 Access: Supervisor Read/Write

	32	33–39	40	41	42–47
R W	EMCP	—	DOZE	NAP	—
Reset	All zeros				

	48	49	50	51–55	56	57	58–62	63
R W	—	TBEN	SEL_TBCLK	—	EN_MAS7_UPDATE	DCFA	—	NOPTI
Reset	All zeros							

Figure 6-32. Hardware Implementation-Dependent Register 0 (HID0)

Table 6-18. HID0 Field Descriptions

Bits	Name	Description
32	EMCP	Enable machine check pin, $\overline{\text{MCP}}$. Used to mask out further machine check exceptions caused by assertion of $\overline{\text{MCP}}$. 0 $\overline{\text{MCP}}$ is disabled. 1 $\overline{\text{MCP}}$ is enabled. If MSE[ME] = 0, asserting $\overline{\text{MCP}}$ causes checkstop. If MSR[ME] = 1, asserting $\overline{\text{MCP}}$ causes a machine check exception.
33–39	—	Reserved, should be cleared.
40	DOZE	Doze power management mode. If MSR[WE] is set, this bit controls DOZE mode. 0 Core not in doze mode 1 Core in doze mode
41	NAP	Nap power management mode. If MSR[WE] is set, this bit controls NAP mode. 0 Core not in nap mode 1 Core in nap mode
42–48	—	Reserved, should be cleared.
49	TBEN	Time base enable 0 Time base disabled (no counting) 1 Time base enabled • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 0, the time base is updated every 8 bus clocks • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of *core_tbclk* (sampled at bus rate). The maximum supported frequency can be found in the electrical specifications, but this value is approximately 25% of the bus clock frequency.
50	SEL_TBCLK	Select time base clock. If the time base is enabled, this field functions as follows: 0 Time base is based on the processor clock 1 Time base is based on the TBCLK (RTC) input
51–55	—	Reserved, should be cleared.
56	EN_MAS7_UPDATE	Enable MAS7 update (e500v2 only). Enables updating MAS7 by **tlbre** and **tlbsx**. 0 MAS7 is not updated by a **tlbre** or **tlbsx**. 1 MAS7 is updated by a **tlbre** or **tlbsx**.

Table 6-18. HID0 Field Descriptions (continued)

Bits	Name	Description
57	DCFA	Data cache flush assist (e500v2 only). Force data cache to ignore invalid sets on miss replacement selection. 0 The data cache flush assist facility is disabled 1 The miss replacement algorithm ignores invalid entries and follows the replacement sequence defined by the PLRU bits. This reduces the series of uniquely addressed load or **dcbz** instructions to eight per set. The bit should be set just before beginning a cache flush routine and should be cleared when the series of instructions is complete.
58–62	—	Reserved, should be cleared.
63	NOPTI	No-op the data and instruction cache touch instructions. 0 **dcbt**, **dcbtst**, and **icbt** are enabled. On the e500, if CT = 0, **icbt** is always a no-op, regardless of the value of NOPTI. If CT = 1, **icbt** does a touch load to an L2 cache, if one is present. 1 **dcbt**, **dcbtst**, and **icbt** are treated as no-ops; **dcblc** and **dcbtls** are not.

6.10.2 Hardware Implementation-Dependent Register 1 (HID1)

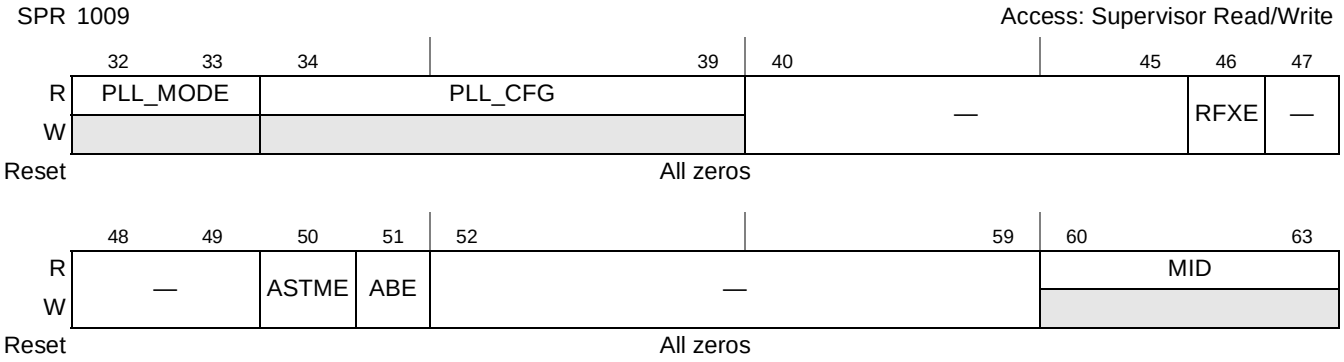


Figure 6-33. Hardware Implementation-Dependent Register 1 (HID1)

Table 6-19. HID1 Field Descriptions

Bits	Name	Description
32–33	PLL_MODE	Read-only for integrated devices. 11 Fixed value for this device
34–39	PLL_CFG	Reflected directly from configuration input pins (read-only). PLL_CFG[0–4] corresponds to the integer divide ratio and PLL_CFG5 is the half-mode bit. The following values are supported: 0001_00 Ratio of 2:1 0001_01 Ratio of 5:2 (2.5:1) 0001_10 Ratio of 3:1 0001_11 Ratio of 7:2 (3.5:1) Note that this value is also reflected to PORPLLSR[e500_Ratio]. See Section 23.4.1.1, “POR PLL Status Register (PORPLLSR).”
40–45	—	Reserved, should be cleared.

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
46	RFXE	Read fault exception enable. Enables the core to internally generate a machine check interrupt when *core_fault_in* is asserted. Depending on the value of MSR[ME], this results in either a machine check interrupt or a checkstop. 0 Assertion of *core_fault_in* cannot cause a machine check. The core does not execute any instructions from a faulty instruction fetch and does not execute any load instructions that get their data from a faulty data fetch. On the e500v2, if these instructions are eventually required by the sequential programming model (that is, they are not in a speculative execution path), the e500v2 stalls until an asynchronous interrupt is taken. The e500v1 does not stall when faulty instructions or data are received, as described in the following note. **Note:** The e500v1 does not stall when faulty instructions or data are received. Instead, it continues processing with faulty instructions or data. The only reliable way to prevent such behavior is to set RFXE, which causes a machine check before the faulty instructions or data are used. To avoid the use of faulty instructions or data and to have good error determination, software must set RFXE and program the PIC to interrupt the processor when errors occur. As a result, software must deal with multiple interrupts for the same fundamental problem. 1 Assertion of *core_fault_in* causes a machine check if MSR[ME] = 1 or a checkstop if MSR[ME] = 0. The *core_fault_in* signal is asserted to the core when logic outside of the core has a problem delivering good data to the core. For example, the front-side L2 cache asserts *core_fault_in* when an ECC error occurs and ECC is enabled. As a second example, it is asserted when there is a master abort on a PCI transaction. See “Proper Reporting of Bus Faults” in the core complex bus chapter of the *PowerPC™ e500 Core Family Reference Manual*. The RFXE bit provides flexibility in error recovery. Typically, devices outside the core have some way other than the assertion of *core_fault_in* to signal the core that an error occurred. Usually, this is done by channeling interrupt requests through a programmable interrupt controller (PIC) to the core. In these cases, the assertion of *core_fault_in* is used only to prevent the core from using bad data before receiving an interrupt from the PIC (for example, an external or critical input interrupt). Possible combinations of RFXE and PIC configuration are as follows: • RFXE = 0 and the PIC is configured to interrupt the processor. In this configuration, the assertion of *core_fault_in* does not trigger a machine check interrupt. The core does not use the faulty instructions or data and may stall. The PIC interrupts the core so that error recovery can begin. This configuration allows the core to query the PIC and the rest of the system for more information about the cause of the interrupt, and generally provides the best error recovery capabilities. • RFXE = 1 and the PIC is not configured to interrupt the processor. This configuration provides quick error detection without the overhead of configuring the PIC. When the PIC is not configured, setting RFXE avoids stalling the core when *core_fault_in* is asserted. Determination of the root cause of the problem may be somewhat more difficult than it would be if the PIC were enabled. • RFXE = 1 and the PIC is configured to interrupt the processor. In this configuration, the core may receive two interrupts for the same fundamental error. The two interrupts may occur in any order, which may complicate error handling. Therefore, this is usually not an interesting configuration for a single-core device. This may, however, be an interesting configuration for multi-core devices in which the PIC may steer interrupts to a processor other than the one that attempted to fetch the faulty data. • RFXE = 0 and the PIC is not configured to interrupt the processor. This is not a recommended configuration. The processor may stall indefinitely due to an unreported error.
47–49	—	Reserved, should be cleared.
50	ASTME	Address bus streaming mode enable. This bit, along with the ECM stream control bits in the EEBACR, enables address bus streaming on the CCB. See Section 8.2.1.1, “ECM CCB Address Configuration Register (EEBACR).” 0 Address bus streaming mode disabled 1 Address bus streaming mode enabled

Table 6-19. HID1 Field Descriptions (continued)

Bits	Name	Description
51	ABE	Address broadcast enable. The e500 broadcasts cache management instructions (**dcbst**, **dcblc** (CT = 1), **icblc** (CT = 1), **dcbf**, **dcbi**, **mbar**, **msync**, **tlbsync**, **icbi**) based on ABE. ABE must be set to allow management of external L2 caches. 0 Address broadcasting disabled 1 Address broadcasting enabled
52–59	—	Reserved, should be cleared.
60–63	MID	Unique core identifier (read only) 0000 core 0 0001 core 1 0010–1111 Reserved

6.11 L1 Cache Configuration Registers

6.11.1 L1 Cache Control and Status Register 0 (L1CSR0)

SPR 1010 Access: Supervisor Read/Write

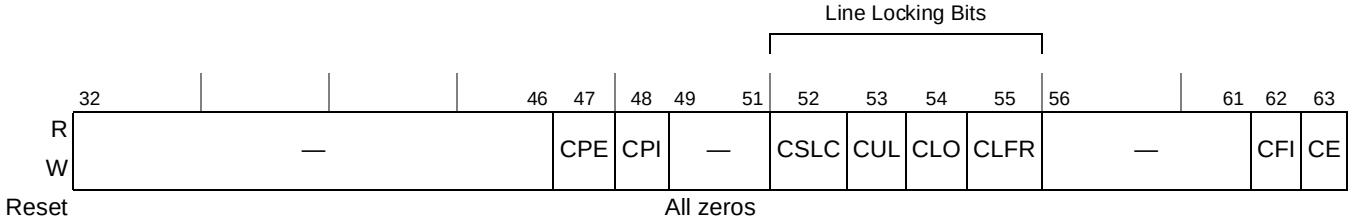


Figure 6-34. L1 Cache Control and Status Register 0 (L1CSR0)

Table 6-20. L1CSR0 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	CPE	(Data) Cache parity enable 0 Parity checking of the cache disabled 1 Parity checking of the cache enabled
48	CPI	(Data) Parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Cache parity must also be enabled (CPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	CSLC	(Data) Cache snoop lock clear. Sticky bit set by hardware if a **dcbi** snoop (either internally or externally generated) invalidated a locked cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can be cleared only by software. 0 The cache has not encountered a **dcbi** snoop that invalidated a locked line. 1 The cache has encountered a **dcbi** snoop that invalidated a locked line.
53	CUL	(Data) Cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the cache 1 Indicates a lock set instruction was not effective in the cache

Table 6-20. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
54	CLO	(Data) Cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the cache 1 Indicates a lock overflow condition was encountered in the cache
55	CLFR	(Data) Cache lock bits flash reset. Writing a 1 during a flash clear operation causes an undefined operation. Writing a 0 during a flash clear operation is ignored. Clearing occurs regardless of the enable (CE) value. 0 Default. 1 Hardware initiates a cache lock bits flash clear operation. This bit is cleared when the operation is complete.
56–61	—	Reserved, should be cleared.
62	CFI	(Data) Cache flash invalidate. 0 No cache invalidate. Writing a 0 to CFI during an invalidation operation is ignored. 1 Cache invalidation operation. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation. Invalidation occurs regardless of the enable (CE) value.
63	CE	(Data) Cache enable 0 The cache is neither accessed or updated. 1 Enables cache operation

6.11.2 L1 Cache Control and Status Register 1 (L1CSR1)

SPR 1011 Access: Supervisor Read/Write

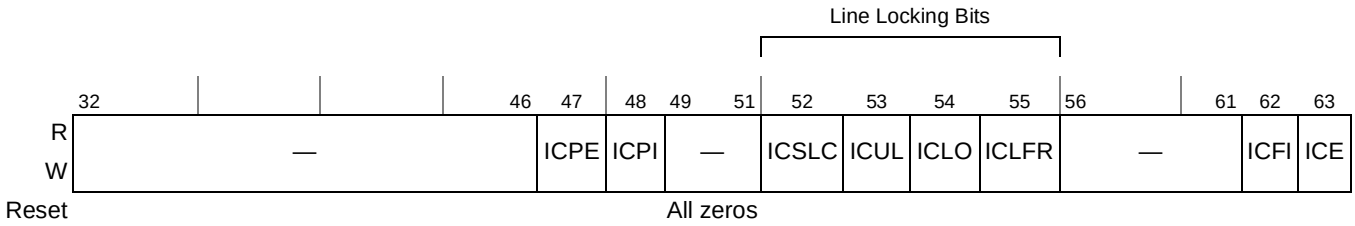


Figure 6-35. L1 Cache Control and Status Register 1 (L1CSR1)

Table 6-21. L1CSR1 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	ICPE	Instruction cache parity enable 0 Parity checking of the instruction cache disabled 1 Parity checking of the instruction cache enabled
48	ICPI	Instruction parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Note that instruction cache parity must also be enabled (ICPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.

Table 6-21. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
52	ICSLC	Instruction cache snoop lock clear. Sticky bit set by hardware if an **icbi** snoop (either internally or externally generated) invalidated a locked line in the instruction cache. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can only be cleared by software. 0 The instruction cache has not encountered an **icbi** snoop that invalidated a locked line. 1 The instruction cache has encountered an **icbi** snoop that invalidated a locked line.
53	ICUL	Instruction cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the instruction cache 1 Indicates a lock set instruction was not effective in the instruction cache
54	ICLO	Instruction cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the instruction cache 1 Indicates a lock overflow condition was encountered in the instruction cache
55	ICLFR	Instruction cache lock bits flash reset. Writing 0 and then 1 flash clears the lock bit of all entries in the instruction cache; clearing occurs independently from the value of the enable bit (ICE). ICLFR is always read as 0.
56–61	—	Reserved, should be cleared.
62	ICFI	Instruction cache flash invalidate. Written to 0 and then 1 to flash clear the valid bit of all entries in the instruction cache; operates independently from the value of the enable bit (ICE). ICFI is always read as 0.
63	ICE	Instruction cache enable 0 The instruction cache is neither accessed or updated. 1 Enables instruction cache operation

6.11.3 L1 Cache Configuration Register 0 (L1CFG0)

SPR 515 Access: User Read only

	32	33	34				38	39			42	43	44	45			49	50	52	53		55	56						63
R	CARCH		—					CBSIZE			CREPL		CLA	CPA	—				CNWAY		—		CSIZE						
W																													
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0

Figure 6-36. L1 Cache Configuration Register 0 (L1CFG0)

Table 6-22. L1CFG0 Field Descriptions

Bits	Name	Description
32–33	CARCH	Cache architecture 00 Harvard 01 Unified
34–38	—	Reserved, should be cleared.
39–40	CBSIZE	Cache line size 0 32 bytes 1 64 bytes
41–42	CREPL	Cache replacement policy 0 True LRU 1 Pseudo LRU

Table 6-22. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
43	CLA	Cache locking available 0 Unavailable 1 Available
44	CPA	Cache parity available 0 Unavailable 1 Available
45–49	—	Reserved, should be cleared.
50–52	CNWAY	Cache number of ways 111 Indicates 8 ways
53–55	—	Reserved, should be cleared.
56–63	CSIZE	Cache size 0x20 indicates 32 Kbytes

6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

SPR 516 Access: User Read only

	32						38	39	40	41	42	43	44	45											52	53											63
R	—							ICBSIZE		ICREPL		ICLA	ICPA	ICNWAY											ICSIZE													
W																																					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0					

Figure 6-37. L1 Cache Configuration Register 1 (L1CFG1)

Table 6-23. L1CFG1 Field Descriptions

Bits	Name	Description
32–38	—	Reserved, should be cleared.
39–40	ICBSIZ	Instruction cache block size 00 Indicates block size of 32 bytes
41–42	ICREPL	Instruction cache replacement policy 01 Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available 1 Indicates available
44	ICPA	Instruction cache parity available 1 Indicates available
45–52	ICNWAY	Instruction cache number of ways 111 Indicates 8 ways
53–63	ICSIZE	Instruction cache size 0x20 indicates 32 Kbytes

6.12 MMU Registers

6.12.1 Process ID Registers (PID0–PID2)

SPR 48 (PID0)
SPR 633 (PID1)
SPR 634 (PID2)

Access: Supervisor Read/Write

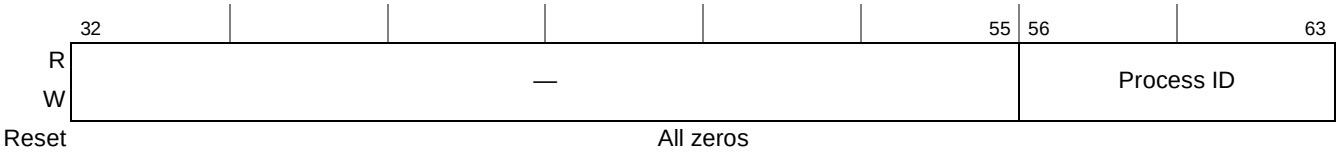


Figure 6-38. Process ID Registers (PID0–PID2)

6.12.2 MMU Control and Status Register 0 (MMUCSR0)

SPR 1012

Access: Supervisor Read/Write

Figure 6-39. MMU Control and Status Register 0 (MMUCSR0)

Table 6-24. MMUCSR0 Field Descriptions

Bits	Name	Description
32–60	—	Reserved, should be cleared.
61	L2TLB0_FI	TLB0 flash invalidate (write to 1 to invalidate)
62	L2TLB1_FI	TLB1 flash invalidate (write 1 to invalidate) 0 No flash invalidate. Writing a 0 to this bit during an invalidation operation is ignored. 1 TLB1 invalidation operation. Hardware initiates a TLB1 invalidation operation. When this operation is complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation.
63	—	Reserved, should be cleared.

6.12.3 MMU Configuration Register (MMUCFG)

SPR 1015

Access: Supervisor Read only

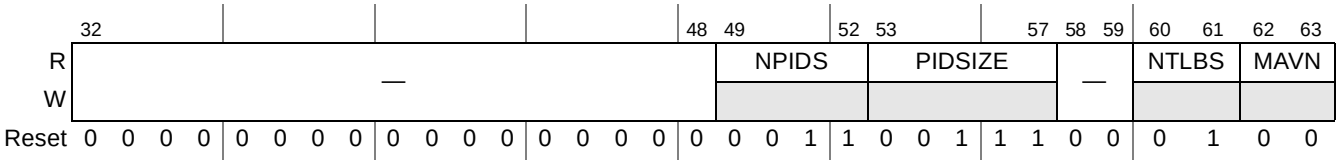


Figure 6-40. MMU Configuration Register (MMUCFG)

Table 6-25. MMUCFG Field Descriptions

Bits	Name	Description
32–48	—	Reserved, should be cleared.
49–52	NPIDS	Number of PID registers, a 4-bit field that indicates the number of PID registers provided by the processor. The e500 implements three PIDs.
53–57	PIDSIZE	PID register size. The 5-bit value of PIDSIZE is one less than the number of bits in each of the PID registers implemented by the processor. The processor implements only the least significant PIDSIZE+1 bits in the PID registers. 00111 indicates 8-bit registers. This is the value presented by the e500.
58–59	—	Reserved, should be cleared.
60–61	NTLBS	Number of TLBs. The value of NTLBS is one less than the number of software-accessible TLB structures that are implemented by the processor. NTLBS is set to one less than the number of TLB structures so that its value matches the maximum value of MAS0[TLBSEL]. 00 1 TLB 01 2 TLBs. This is the value presented by the e500. 10 3 TLBs 11 4 TLBs
62–63	MAVN	MMU architecture version number. Indicates the version number of the architecture of the MMU implemented by the processor. 0b00 indicates version 1.0.

6.12.4 TLB Configuration Registers (TLB*n*CFG)

6.12.4.1 TLB0 Configuration Register 0 (TLB0CFG)

SPR 688 Access: Supervisor Read only

	32–39	40–43	44–47	48	49	50–51	52–63
R	ASSOC	MINSIZE	MAXSIZE	IPROT	AVAIL	—	NENTRY
W							
Reset	0 0 0 0 0 1 0 0	0 0 0 1	0 0 0 1	0	0	0 0	0 0 1 0 0 0 0 0 0 0 0 0

Figure 6-41. TLB Configuration Register 0 (TLB0CFG)

Table 6-26. TLB0CFG Field Descriptions

Bits	Name	Description
32–39	ASSOC	Associativity of TLB0 0x02 indicates associativity is 2-way set associative
40–43	MINSIZE	Minimum page size of TLB0 0x1 indicates smallest page size is 4 Kbytes
44–47	MAXSIZE	Maximum page size of TLB0 0x1 indicates maximum page size is 4 Kbytes
48	IPROT	Invalidate protect capability of TLB0 0 Indicates invalidate protection capability not supported
49	AVAIL	Page size availability of TLB0 0 No variable-sized pages available (MINSIZE = MAXSIZE)

Table 6-26. TLB0CFG Field Descriptions (continued)

Bits	Name	Description
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB0 0x100 LB0 contains 256 entries 0x200TLB0 contains 512 entries (e500v2 only)

6.12.4.2 TLB1 Configuration Register 1 (TLB1CFG)

SPR 689 Access: Supervisor Read only

	32–39	40–43	44–47	48	49	50 51	52–63
R	ASSOC	MINSIZE	MAXSIZE	IPROT	AVAIL	—	NENTRY
W							
Reset	0 0 0 1 0 0 0 0	0 0 0 1	1 0 1 1	1	1	0 0	0 0 0 0 0 0 0 1 0 0 0 0

Figure 6-42. TLB Configuration Register 1 (TLB1CFG)

Table 6-27. TLB1CFG Field Descriptions

Bits	Name	Description
32–39	ASSOC	Associativity of TLB1 0x10 indicates associativity is 16
40–43	MINSIZE	Minimum page size of TLB1 0x1 indicates smallest page size is 4K
44–47	MAXSIZE	Maximum page size of TLB1 0xB Indicates maximum page size is 4 Gbytes (e500v2 only)
48	IPROT	Invalidate protect capability of TLB1 1 Indicates that TLB1 supports invalidate protection capability
49	AVAIL	Page size availability of TLB1 1 Indicates all page sizes between MINSIZE and MAXSIZE supported
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB1 0x010: TLB1 contains 16 entries

6.12.5 MMU Assist Registers

6.12.5.1 MAS Register 0 (MAS0)

SPR 624 Access: Supervisor Read/Write

	32–34	35	36–43	44–47	48–62	63
R	—	TLBSEL	—	ESEL	—	NV
W						
Reset	All zeros					

Figure 6-43. MAS Register 0 (MAS0)

Table 6-28. MAS0 Field Descriptions—MMU Read/Write and Replacement Control

Bits	Name	Descriptions
32–34	—	Reserved, should be cleared.
35	TLBSEL	Selects TLB for access 0 TLB0 1 TLB1
36–43	—	Reserved, should be cleared.
44–47	ESEL	Entry select. Number of entry in selected array to be used for **tlbwe**. This field is also updated on TLB error exceptions (misses), and **tlbsx** hit and miss cases. Only certain bits are valid, depending on the array selected in TLBSEL. Other bits should be 0. For the e500, ESEL serves as the way select for the corresponding TLB as follows: When TLBSEL = 00 (TLB0 selected), bits 46–47 are used (and bits 44–45 should be cleared). This field selects between way 0, 1, 2, or 3 of TLB0. EA bits 45–51 from MAS2[EPN] are used to index into the TLB to further select the entry for the operation. Note that for the e500v1, bit 47 selects either way 0 or way 1, and bit 46 should remain cleared. When TLBSEL = 01 (TLB1 selected), all four bits are used to select one of 16 entries in the array.
48–62	—	Reserved, should be cleared.
63	NV	Next victim. Next victim bit value to be written to TLB0[NV] on execution of **tlbwe**. This field is also updated on TLB error exceptions (misses), **tlbsx** hit and miss cases and on execution of **tlbre**. This field is updated based on the calculated next victim bit for TLB0 (based on the round-robin replacement algorithm.) Note that this field is not defined for operations that specify TLB1 (when TLBSEL = 01).

6.12.5.2 MAS Register 1 (MAS1)

SPR 625 Access: Supervisor Read/Write

	32	33	34–39	40–47	48–50	51	52–55	56–63
R W	V	IPROT	—	TID	—	TS	TSIZE	—
Reset	All zeros							

Figure 6-44. MAS Register 1 (MAS1)

Table 6-29. MAS1 Field Descriptions—Descriptor Context and Configuration Control

Bits	Name	Descriptions
32	V	TLB valid bit 0 This TLB entry is invalid. 1 This TLB entry is valid.
33	IPROT	Invalidate protect. Set to protect this TLB entry from invalidate operations due the execution of **tlbiva**[**x**] (TLB1 only). Note that not all TLB arrays are necessarily protected from invalidation with IPROT. Arrays that support invalidate protection are denoted as such in the TLB configuration registers. 0 Entry is not protected from invalidation 1 Entry is protected from invalidation.
34–39	—	Reserved, should be cleared.
40–47	TID	Translation identity. An 8-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the three virtual address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.
48–50	—	Reserved, should be cleared.
51	TS	Translation space. This bit is compared with the IS or DS fields of the MSR (depending on the type of access) to determine if this TLB entry may be used for translation.
52–55	TSIZE	Translation size. Defines the TLB entry page size. For arrays that contain fixed-size TLB entries, TSIZE is ignored. For variable page size arrays, the page size is 4^{TSIZE} Kbytes. The e500 supports the following sizes: 0001 4 Kbytes 0010 16 Kbytes 0011 64 Kbytes 0100 256 Kbytes 0101 1 Mbyte 0110 4 Mbytes 0111 16 Mbytes 1000 64 Mbytes 1001 256 Mbytes 1010 1 Gbyte (e500v2 only) 1011 4 Gbytes (e500v2 only)
56–63	—	Reserved, should be cleared.

6.12.5.3 MAS Register 2 (MAS2)

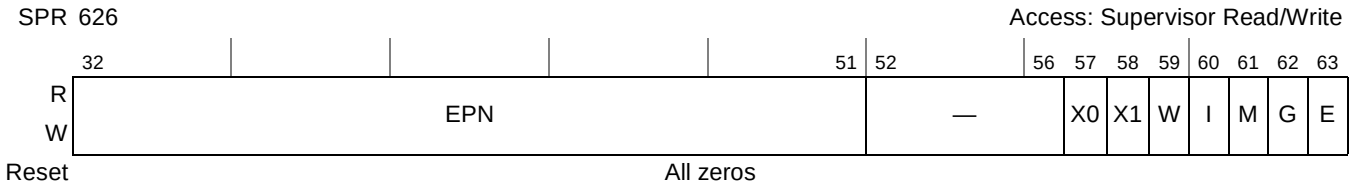


Figure 6-45. MAS Register 2 (MAS2)

Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes

Bits	Name	Description
32–51	EPN	Effective page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–56	—	Reserved for implementation-specific use
57	X0	Implementation-dependent page attribute
58	X1	Implementation-dependent page attribute

Table 6-30. MAS2 Field Descriptions—EPN and Page Attributes (continued)

Bits	Name	Description
59	W	Write-through 0 This page is considered write-back with respect to the caches in the system. 1 All stores performed to this page are written through the caches to main memory.
60	I	Caching-inhibited 0 Accesses to this page are considered cacheable. 1 The page is considered caching-inhibited. All loads and stores to the page bypass the caches and are performed directly to main memory.
61	M	Memory coherency required 0 Memory coherency is not required. 1 Memory coherency is required. This allows loads and stores to this page to be coherent with loads and stores from other processors (and devices) in the system, assuming all such devices are participating in the coherency protocol.
62	G	Guarded 0 Accesses to this page are not guarded and can be performed before it is known if they are required by the sequential execution model. 1 All loads and stores to this page that miss in the L1 cache are performed without speculation (that is, they are known to be required). Speculative loads can be performed if they hit in the L1 cache. In addition, accesses to caching-inhibited pages are performed using only the memory element that is explicitly specified.
63	E	Endianness. Determines endianness for the corresponding page. Little-endian operation is true little-endian, which differs from the modified little-endian byte-ordering model optionally available in previous devices that implement the original PowerPC architecture. See the *PowerPC™ e500 Core Family Reference Manual* for more information. 0 The page is accessed in big-endian byte order. 1 The page is accessed in true little-endian byte order.

6.12.5.4 MAS Register 3 (MAS3)

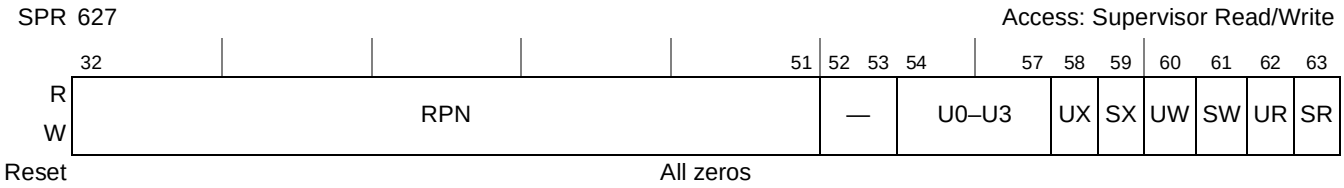


Figure 6-46. MAS Register 3 (MAS3)

Table 6-31. MAS3 Field Descriptions–RPN and Access Control

Bits	Name	Description
32–51	RPN	Real page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared. For the e500v2, the 4 high-order bits of RPN are stored in MAS7[RPN].
52–53	—	Reserved, should be cleared.
54–57	U0–U3	User attribute bits. Associated with a TLB entry and can be used by system software. For example, they can hold information useful to a page-scanning algorithm or mark more abstract page attributes.
58–63	PERMIS	Permission bits (UX, SX, UW, SW, UR, SR). User and supervisor read, write, and execute permission bits.

6.12.5.5 MAS Register 4 (MAS4)

SPR 628 Access: Supervisor Read/Write

	32–34	35	36–45	46–47	48–50	51–52	52–55	56	57	58	59	60	61	62	63
R W	—	TLBSELD	—	TIDSELD	—		TSIZED	—	X0D	X1D	WD	ID	MD	GD	ED
Reset	All zeros														

Figure 6-47. MAS Register 4 (MAS4)

Table 6-32. MAS4 Field Descriptions—Hardware Replacement Assist Configuration

Bits	Name	Description
32–34	—	Reserved, should be cleared.
35	TLBSELD	TLBSEL default value. The default value to be loaded in MAS0[TLBSEL] on a TLB miss exception. 0 TLB0 1 TLB1
36–45	—	Reserved, should be cleared.
46–47	TIDSELD	TID default selection value. A 2-bit field that specifies which of the current PID registers should be used to load the MAS1[TID] field on a TLB miss exception.
		The e500 implementation defines this field as follows: 00 PID0 01 PID1 10 PID2 11 TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
52–55	TSIZED	Default TSIZE value. Specifies the default value to be loaded into MAS1[TSIZE] on a TLB miss exception.
56	—	Reserved, should be cleared.
57	X0D	Default X0 value. Specifies the default value to be loaded into MAS2[X0] on a TLB miss exception.
58	X1D	Default X1 value. Specifies the default value to be loaded into MAS2[X1] on a TLB miss exception.
59	WD	Default W value. Specifies the default value to be loaded into MAS2[W] on a TLB miss exception.
60	ID	Default I value. Specifies the default value to be loaded into MAS2[I] on a TLB miss exception.
61	MD	Default M value. Specifies the default value to be loaded into MAS2[M] on a TLB miss exception.
62	GD	Default G value. Specifies the default value to be loaded into MAS2[G] on a TLB miss exception.
63	ED	Default E value. Specifies the default value to be loaded into MAS2[E] on a TLB miss exception.

6.12.5.6 MAS Register 6 (MAS6)

SPR 630 Access: Supervisor Read/Write

	32–39	40–47	48–62	63
R W	—	SPID0	—	SAS
Reset	All zeros			

Figure 6-48. MAS Register 6 (MAS6)

Table 6-33. MAS6—TLB Search Context Register 0

Bits	Name	Comments, or Function when Set
32–39	—	Reserved, should be cleared.
40–47	SPID0	Specifies the PID value (recent value of PID0) used when searching the TLB during execution of **tlbsx**.
48–62	—	Reserved, should be cleared.
63	SAS	Address space (AS) value for searches. Specifies the value of AS used when searching the TLB (during execution of **tlbsx**).

6.12.5.7 MAS Register 7 (MAS7)

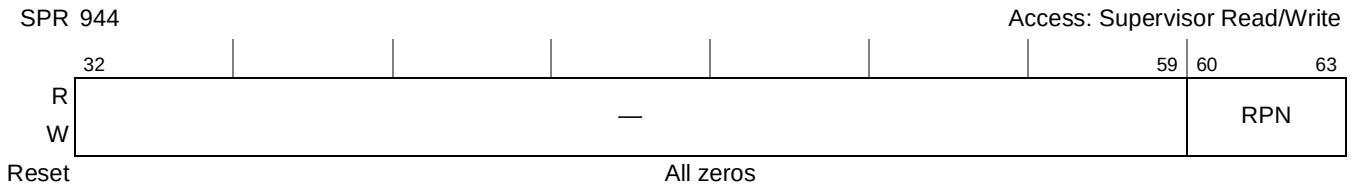


Figure 6-49. MAS Register 7 (MAS7)

Table 6-34. MAS 7 Field Descriptions—High Order RPN

Bits	Name	Description
32–63	RPN	Real page number. Four high-order bits of the RPN. The 20 low-order bits of RPN are in MAS3.

6.13 Debug Registers

6.13.1 Debug Control Registers (DBCR0–DBCR2)

6.13.1.1 Debug Control Register 0 (DBCR0)

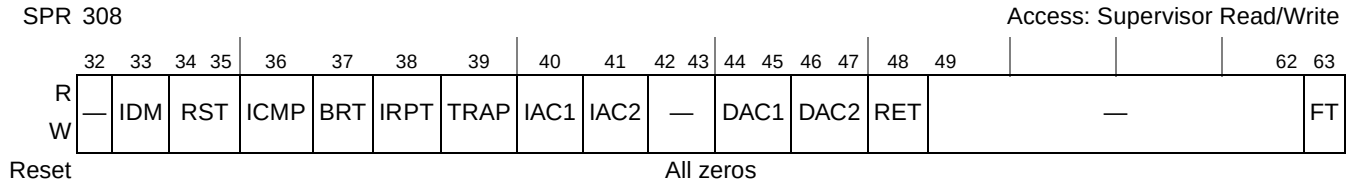


Figure 6-50. Debug Control Register 0 (DBCR0)

Table 6-35. DBCR0 Field Descriptions

Bits	Name	Description
32	—	Reserved, should be cleared.
33	IDM	Internal debug mode 0 Debug interrupts are disabled. No debug interrupts are taken and debug events are not logged. 1 If MSR[DE] = 1, the occurrence of a debug event or the recording of an earlier debug event in the DBSR when MSR[DE] = 0 or DBCR0[IDM] = 0 causes a debug interrupt. **Programming note:** Software must clear debug event status in the DBSR in the debug interrupt handler when a debug interrupt is taken before re-enabling interrupts through MSR[DE]. Otherwise, redundant debug interrupts are taken for the same debug event.
34–35	RST	Reset. The e500 implements these bits as follows: 0*x* Default (No action) 1*x* Causes a hard reset if MSR[DE] and DBCR0[IDM] are set. Always cleared on subsequent cycle. This causes a hard reset to the core only.
36	ICMP	Instruction completion debug event enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled Note: Instruction completion does not cause an ICMP debug event if MSR[DE] = 0.
37	BRT	Branch taken debug event enable 0 BRT debug events are disabled 1 BRT debug events are enabled Note: Taken branches do not cause a BRT debug event if MSR[DE] = 0.
38	IRPT	Interrupt taken debug event enable. This bit affects only noncritical interrupts. 0 IRPT debug events are disabled 1 IRPT debug events are enabled
39	TRAP	Trap debug event enable 0 TRAP debug events cannot occur 1 TRAP debug events can occur
40	IAC1	Instruction address compare 1 debug event enable 0 IAC1 debug events cannot occur 1 IAC1 debug events can occur
41	IAC2	Instruction address compare 2 debug event enable 0 IAC2 debug events cannot occur 1 IAC2 debug events can occur
42–43	—	Reserved, should be cleared.
44–45	DAC1	Data address compare 1 debug event enable 00 DAC1 debug events cannot occur 01 DAC1 debug events can occur only if a store-type data storage access 10 DAC1 debug events can occur only if a load-type data storage access 11 DAC1 debug events can occur on any data storage access
46–47	DAC2	Data address compare 2 debug event enable 00 DAC2 debug events cannot occur 01 DAC2 debug events can occur only if a store-type data storage access 10 DAC2 debug events can occur only if a load-type data storage access 11 DAC2 debug events can occur on any data storage access

Table 6-35. DBCR0 Field Descriptions (continued)

Bits	Name	Description
48	RET	Return debug event enable 0 RET debug events cannot occur 1 RET debug events can occur Note: An **rfci** does not cause an RET debug event if MSR[DE] = 0 at the time that **rfci** executes.
49–62	—	Reserved, should be cleared.
63	FT	Freeze timers on debug event 0 Enable clocking of timers 1 Disable clocking of timers if any DBSR bit is set (except MRR)

6.13.1.2 Debug Control Register 1 (DBCR1)

SPR 309 — Access: Supervisor Read/Write

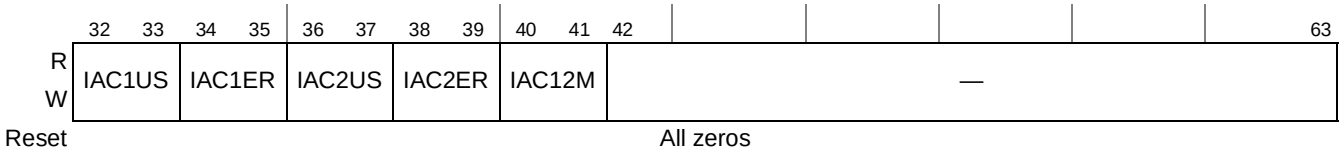


Figure 6-51. Debug Control Register 1 (DBCR1)

Table 6-36. DBCR1 Field Descriptions

Bits	Name	Description
32–33	IAC1US	Instruction address compare 1 user/supervisor mode 00 IAC1 debug events can occur 01 Reserved 10 IAC1 debug events can occur only if MSR[PR] = 0 11 IAC1 debug events can occur only if MSR[PR] = 1
34–35	IAC1ER	Instruction address compare 1 effective/real mode 00 IAC1 debug events are based on effective addresses 01 Reserved on the e500 10 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 1
36–37	IAC2US	Instruction address compare 2 user/supervisor mode 00 IAC2 debug events can occur 01 Reserved 10 IAC2 debug events can occur only if MSR[PR] = 0 11 IAC2 debug events can occur only if MSR[PR] = 1
38–39	IAC2ER	Instruction address compare 2 effective/real mode 00 IAC2 debug events are based on effective addresses 01 Reserved on the e500 10 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 1

Table 6-36. DBCR1 Field Descriptions (continued)

Bits	Name	Description
40–41	IAC12M	Instruction address compare 1/2 mode 00 Exact address compare. IAC1 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 and IAC2 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, plus ANDed with the contents of IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined. 10 Inclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined. 11 Exclusive address range compare. IAC1 and IAC2 debug events occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. If IAC1US ≠ IAC2US or IAC1ER ≠ IAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

6.13.1.3 Debug Control Register 2 (DBCR2)

SPR 310

Access: Supervisor Read/Write

	32	33	34	35	36	37	38	39	40	41	42					63
R W	DAC1US		DAC1ER		DAC2US		DAC2ER		DAC12M		—					
Reset	All zeros															

Figure 6-52. Debug Control Register 2 (DBCR2)

Table 6-37. DBCR2 Field Descriptions

Bits	Name	Description
32–33	DAC1US	Data address compare 1 user/supervisor mode 00 DAC1 debug events can occur 01 Reserved 10 DAC1 debug events can occur only if MSR[PR] = 0. 11 DAC1 debug events can occur only if MSR[PR] = 1.
34–35	DAC1ER	Data address compare 1 effective/real mode 00 DAC1 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
36–37	DAC2US	Data address compare 2 user/supervisor mode 00 DAC2 debug events can occur. 01 Reserved 10 DAC2 debug events can occur only if MSR[PR] = 0. 11 DAC2 debug events can occur only if MSR[PR] = 1.

Table 6-37. DBCR2 Field Descriptions (continued)

Bits	Name	Description
38–39	DAC2ER	Data address compare 2 effective/real mode 00 DAC2 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
40–41	DAC12M	Data address compare 1/2 mode 00 Exact address compare. DAC1 debug events can occur only if the address of the data storage access is equal to the value specified in DAC1. DAC2 debug events can occur only if the address of the data storage access is equal to the value specified in DAC2. 01 Address bit match. DAC1 and DAC2 debug events can occur only if the address of the data storage access, ANDed with the contents of DAC2 are equal to the contents of DAC1, also ANDed with the contents of DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 10 Inclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 11 Exclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

6.13.2 Debug Status Register (DBSR)

SPR: 304 Access: Supervisor w1c

	32	33	34 35	36	37	38	39	40	41	42 43	44	45	46	47
R	IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—	DAC1R	DAC1W	DAC2R	DAC2W
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c
Reset	0	0	undefined	0	0	0	0	0	0	0 0	0	0	0	0

	48	49	63
R	RET	—	
W	w1c		
Reset	All zeros		

Figure 6-53. Debug Status Register (DBSR)

Table 6-38. DBSR Field Descriptions

Bits	Name	Description
32	IDE	Imprecise debug event. Set if MSR[DE] = 0 and a debug event causes its respective DBSR bit to be set. Functions as write-one-to-clear.
33	UDE	Unconditional debug event. Set if an unconditional debug event occurred. Functions as write-one-to-clear. If $\overline{\text{UDE}}$ (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE]DBCR0[IDM]Action X 0 No action. 0 1 UDE is set. 1 1 UDE is set and a debug interrupt is taken.
34–35	MRR	Most recent reset. Functions as write-one-to-clear. Undefined at power-on. The e500 implements HRESET as follows: 0x No hard reset occurred since this bit was last cleared by software. 1x The previous reset was a hard reset.
36	ICMP	Instruction complete debug event. Set if an instruction completion debug event occurred and DBCR0[ICMP] = 1. Functions as write-one-to-clear.
37	BRT	Branch taken debug event. Set if a branch taken debug event occurred (DBCR0[BRT] = 1). Functions as write-one-to-clear.
38	IRPT	Interrupt taken debug event. Set if an interrupt taken debug event occurred (DBCR0[IRPT] = 1). Functions as write-one-to-clear.
39	TRAP	Trap instruction debug event. Set if a trap Instruction debug event occurred (DBCR0[TRAP] = 1). Functions as write-one-to-clear.
40	IAC1	Instruction address compare 1 debug event. Set if an IAC1 debug event occurred (DBCR0[IAC1] = 1). Functions as write-one-to-clear.
41	IAC2	Instruction address compare 2 debug event. Set if an IAC2 debug event occurred (DBCR0[IAC2] = 1). Functions as write-one-to-clear.
42–43	—	Reserved, should be cleared.
44	DAC1R	Data address compare 1 read debug event. Set if a read-type DAC1 debug event occurred (DBCR0[DAC1] = 10 or 11). Functions as write-one-to-clear.
45	DAC1W	Data address compare 1 write debug event. Set if a write-type DAC1 debug event occurred (DBCR0[DAC1] = 01 or 11). Functions as write-one-to-clear.
46	DAC2R	Data address compare 2 read debug event.Set if a read-type DAC2 debug event occurred (DBCR0[DAC2] = 10 or 11). Functions as write-one-to-clear.
47	DAC2W	Data address compare 2 write debug event. Set if a write-type DAC2 debug event occurred (DBCR0[DAC2] = 01 or 11). Functions as write-one-to-clear.
48	RET	Return debug event. Set if a return debug event occurred (DBCR0[RET] = 1). Functions as write-one-to-clear.
49–63	—	Reserved, should be cleared.

6.13.3 Instruction Address Compare Registers (IAC1–IAC2)

SPR 312 (IAC1); SPR 313 (IAC2) Access: Supervisor Read/Write

	32							61	62	63
R W	Instruction address								—	
Reset	All zeros									

Figure 6-54. Instruction Address Compare Registers (IAC1–IAC2)

6.13.4 Data Address Compare Registers (DAC1–DAC2)

SPR 316 (DAC1); SPR 317 (DAC2) Access: Supervisor Read/Write

	32							63
R W	Data address							
Reset	All zeros							

Figure 6-55. Data Address Compare Registers (DAC1–DAC2)

6.14 Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

SPR: 512 Access: User mixed

	High-Word Error Bits										Status Bits					
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
R W	SOVH	OVH	FGH	FXH	FINVH	FDBZH	FUNFH	FOVFH	—		FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE
Reset	All zeros															

	Enable Bits															
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R W	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	—	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
Reset	All zeros															

Figure 6-56. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Table 6-39. SPEFSCR Field Descriptions

Bits	Name	Function
32	SOVH	Summary integer overflow high. Set whenever an instruction (except **mtspr**) sets OVH. SOVH remains set until it is cleared by an **mtspr[SPEFSCR]**.
33	OVH	Integer overflow high. An overflow occurred in the upper half of the register while executing a SPE integer instruction
34	FGH	Embedded floating-point guard bit high. Floating-point guard bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.

Table 6-39. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
35	FXH	Embedded floating-point sticky bit high. Floating bit from the upper half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
36	FINVH	Embedded floating-point invalid operation error high. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
37	FDBZH	Embedded floating-point divide by zero error high. Set if the dividend is non-zero and the divisor is zero.
38	FUNFH	Embedded floating-point underflow error high
39	FOVFH	Embedded floating-point overflow error high
40–41	—	Reserved, should be cleared.
42	FINXS	Embedded floating-point inexact sticky. FINXS = FINXS | FGH | FXH | FG | FX
43	FINVS	Embedded floating-point invalid operation sticky. Location for software to use when implementing true IEEE floating point.
44	FDBZS	Embedded floating-point divide by zero sticky. FDBZS = FDBZS | FDBZH | FDBZ
45	FUNFS	Embedded floating-point underflow sticky. Storage location for software to use when implementing true IEEE floating point.
46	FOVFS	Embedded floating-point overflow sticky. Storage location for software to use when implementing true IEEE floating point.
47	MODE	Embedded floating-point mode (read only on e500)
48	SOV	Integer summary overflow. Set whenever an SPE instruction (except **mtspr**) sets OV. SOV remains set until it is cleared by **mtspr[SPEFSCR]**.
49	OV	Integer overflow. An overflow occurred in the lower half of the register while a SPE integer instruction is being executed.
50	FG	Embedded floating-point guard bit. Floating-point guard bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
51	FX	Embedded floating-point sticky bit. Floating bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
52	FINV	Embedded floating-point invalid operation error. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
53	FDBZ	Embedded floating-point divide by zero error. Set of the dividend is non-zero and the divisor is zero.
54	FUNF	Embedded floating-point underflow error
55	FOVF	Embedded floating-point overflow error
56	—	Reserved, should be cleared.
57	FINXE	Embedded floating-point inexact enable
58	FINVE	Embedded floating-point invalid operation/input error exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FINV or FINVH is set by a floating-point instruction.

Table 6-39. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
59	FDBZE	Embedded floating-point divide-by-zero exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FDBZ or FDBZH is set by a floating-point instruction.
60	FUNFE	Embedded floating-point underflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FUNF or FUNFH is set by a floating-point instruction.
61	FOVFE	Embedded floating-point overflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FOVF or FOVFH is set by a floating-point instruction.
62–63	FRMC	Embedded floating-point rounding mode control 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward –infinity

6.14.1 Accumulator (ACC)

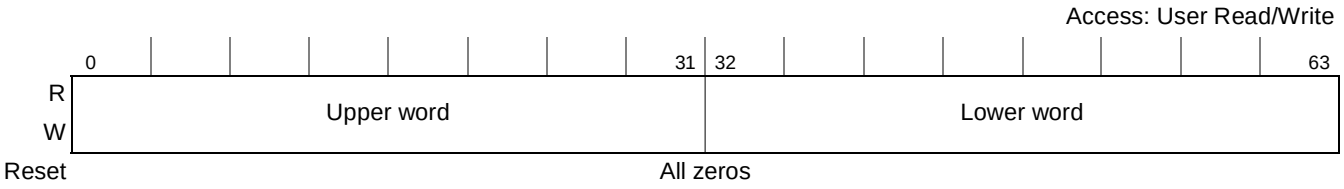


Figure 6-57. Accumulator (ACC)

Table 6-40. ACC FIeld Descriptions

Bits	Name	Function
0–31	Upper word	Holds the upper-word accumulate value for SPE multiply with accumulate instructions
32–63	Lower word	Holds the lower-word accumulate value for SPE multiply with accumulate instructions

6.15 Performance Monitor Registers (PMRs)

Table 6-41. Supervisor-Level PMRs (PMR[5] = 1)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
PMC0	Performance monitor counter 0	16	00000	10000	6.15.4/6-52
PMC1	Performance monitor counter 1	17	00000	10001	
PMC2	Performance monitor counter 2	18	00000	10010	
PMC3	Performance monitor counter 3	19	00000	10011	
PMGC0	Performance monitor global control register 0	400	01100	10000	6.15.1/6-49
PMLCa0	Performance monitor local control a0	144	00100	10000	6.15.2/6-50
PMLCa1	Performance monitor local control a1	145	00100	10001	
PMLCa2	Performance monitor local control a2	146	00100	10010	
PMLCa3	Performance monitor local control a3	147	00100	10011	
PMLCb0	Performance monitor local control b0	272	01000	10000	6.15.3/6-51
PMLCb1	Performance monitor local control b1	273	01000	10001	
PMLCb2	Performance monitor local control b2	274	01000	10010	
PMLCb3	Performance monitor local control b3	275	01000	10011	

Table 6-42. User-Level PMRs (PMR[5] = 0) (Read Only)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
UPMC0	User performance monitor counter 0	0	00000	00000	6.15.4/6-52
UPMC1	User performance monitor counter 1	1	00000	00001	
UPMC2	User performance monitor counter 2	2	00000	00010	
UPMC3	User performance monitor counter 3	3	00000	00011	
UPMLCa0	User performance monitor local control a0	128	00100	00000	6.15.3/6-51
UPMLCa1	User performance monitor local control a1	129	00100	00001	
UPMLCa2	User performance monitor local control a2	130	00100	00010	
UPMLCa3	User performance monitor local control a3	131	00100	00011	
UPMLCb0	User performance monitor local control b0	256	01000	00000	6.15.3/6-51
UPMLCb1	User performance monitor local control b1	257	01000	00001	
UPMLCb2	User performance monitor local control b2	258	01000	00010	
UPMLCb3	User performance monitor local control b3	259	01000	00011	
UPMGC0	User performance monitor global control register 0	384	01100	00000	6.15.2/6-50

6.15.1 Global Control Register 0 (PMGC0, UPMGC0)

PMGC0 (PMR400)
UPMGC0 (PMR384)

Access: PMGC0: Supervisor- Read/Write
UPMGC0: Supervisor/User Read only

1 e500v2 only

Figure 6-58. Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0)

Table 6-43. PMGC0 Field Descriptions

Bits	Name	Description
32	FAC	Freeze all counters. When FAC is set by hardware or software, PMLCx[FC] maintains its current value until it is changed by software. 0 The PMCs are incremented (if permitted by other PM control bits). 1 The PMCs are not incremented.
33	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs.
34	FCECE	Freeze counters on enabled condition or event 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0[FAC] is set. It is up to software to clear FAC.
35–50	—	Reserved, should be cleared.
51–52	TBSEL	Time base selector. Selects the time base bit that can cause a time base transition event (the event occurs when the selected bit changes from 0 to 1). 00 TB[63] (TBL[31]) 01 TB[55] (TBL[23]) 10 TB[51] (TBL[19]) 11 TB[47] (TBL[15]) Time base transition events can be used to periodically collect information about processor activity. In multiprocessor systems in which TB registers are synchronized among processors, time base transition events can be used to correlate the performance monitor data obtained by the several processors. For this use, software must specify the same TBSEL value for all processors in the system. Because the time-base frequency is implementation-dependent, software should invoke a system service program to obtain the frequency before choosing a value for TBSEL.
53–54	—	Reserved, should be cleared.

Table 6-43. PMGC0 Field Descriptions (continued)

Bits	Name	Description
55	TBEE	Time base transition event exception enable. 0 Exceptions from time base transition events are disabled. 1 Exceptions from time base transition events are enabled. A timebase transition is signaled to the performance monitor if the TB bit specified in PMGC0[TBSEL] changes from 0 to 1. Timebase transition events can be used to freeze the counters (PMGC0[FCECE]) or signal an exception (PMGC0[PMIE]). Changing PMGC0[TBSEL] while PMGC0[TBEE] is enabled may cause a false 0 to 1 transition that signals the specified action (freeze, exception) to occur immediately. Although the interrupt signal condition may occur with MSR[EE] = 0, the interrupt cannot be taken until MSR[EE] = 1.
56–63	—	Reserved, should be cleared.

6.15.2 Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3)

PMLCa0 (PMR144)
PMLCa1 (PMR145)
PMLCa2 (PMR146)
PMLCa3 (PMR147)

UPMLCa0 (PMR128)
UPMLCa1 (PMR129)
UPMLCa2 (PMR130)
UPMLCa3 (PMR131)

Access: PMLCa0–PMLCa3: Supervisor Read/Write
UPMLCa0–UPMLCa3: Supervisor/User Read only

	32	33	34	35	36	37	38	40	41	47	48	63
R W	FC	FCS	FCU	FCM1	FCM0	CE	—	EVENT			—	
Reset	All zeros											

Figure 6-59. Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)

Table 6-44. PMLCa0–PMLCa3 Field Descriptions

Bits	Name	Description
32	FC	Freeze counter 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented.
33	FCS	Freeze counter in supervisor state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 0.
34	FCU	Freeze counter in user state 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PR] = 1.
35	FCM1	Freeze counter while mark = 1 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 1.
36	FCM0	Freeze counter while mark = 0 0 The PMC is incremented (if permitted by other PM control bits). 1 The PMC is not incremented if MSR[PMM] = 0.

Table 6-44. PMLCa0–PMLCa3 Field Descriptions (continued)

Bits	Name	Description
37	CE	Condition enable 0 PMC*x* overflow conditions cannot occur (PMC*x* cannot cause interrupts, cannot freeze counters) 1 Overflow conditions occur when the most-significant bit of PMC*x* is equal to 1. It is recommended that CE be cleared when counter PMC*x* is selected for chaining.
38–40	—	Reserved, should be cleared.
41–47	EVENT	Event selector. Up to 128 events selectable. These events are described in the *PowerPC™ e500 Core Family Reference Manual.*
48–63	—	Reserved, should be cleared.

6.15.3 Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3)

PMLCb0 (PMR272) UPMLCb0 (PMR256)
PMLCb1 (PMR273) UPMLCb1 (PMR257)
PMLCb2 (PMR274) UPMLCb2 (PMR258)
PMLCb3 (PMR275) UPMLCb3 (PMR259)

Access: PMLCb0–PMLCb3: Supervisor Read/Write
UPMLCb0–UPMLCb3: Supervisor/User Read only

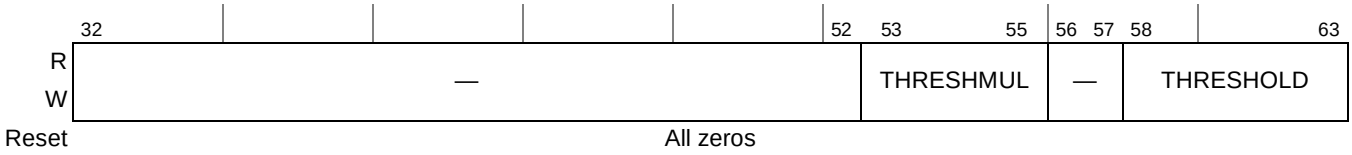


Figure 6-60. Local Control B Registers (PMLCb0–PMLCb3)/User Local Control B Registers (UPMLCb0–UPMLCb3)

Table 6-45. PMLCb0–PMLCb3 Field Descriptions

Bits	Name	Description
32–52	—	Reserved, should be cleared.
53–55	THRESHMUL	Threshold multiple 000 Threshold field is multiplied by 1 (PMLCb*n*[THRESHOLD] × 1) 001 Threshold field is multiplied by 2 (PMLCb*n*[THRESHOLD] × 2) 010 Threshold field is multiplied by 4 (PMLCb*n*[THRESHOLD] × 4) 011 Threshold field is multiplied by 8 (PMLCb*n*[THRESHOLD] × 8) 100 Threshold field is multiplied by 16 (PMLCb*n*[THRESHOLD] × 16) 101 Threshold field is multiplied by 32 (PMLCb*n*[THRESHOLD] × 32) 110 Threshold field is multiplied by 64 (PMLCb*n*[THRESHOLD] × 64) 111 Threshold field is multiplied by 128 (PMLCb*n*[THRESHOLD] × 128)
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the threshold value are counted. Such events are implementation-dependent as are the dimension (for example duration in cycles) and granularity with which the value is interpreted. By varying the value, software can obtain a profile of the event characteristics subject to thresholding. For example, if PMC1 is configured to count cache misses that last longer than the threshold value, software can obtain the distribution of cache miss durations for a given program by monitoring the program repeatedly using a different threshold each time.

6.15.4 Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3)

PMC0 (PMR16) UPMC0 (PMR0)
PMC1 (PMR17) UPMC1 (PMR1)
PMC2 (PMR18) UPMC2 (PMR2)
PMC3 (PMR19) UPMC3 (PMR3)

Access: PMC0–PMC3: Supervisor Read/Write
UPMC0–UPMC3: Supervisor/User Read only

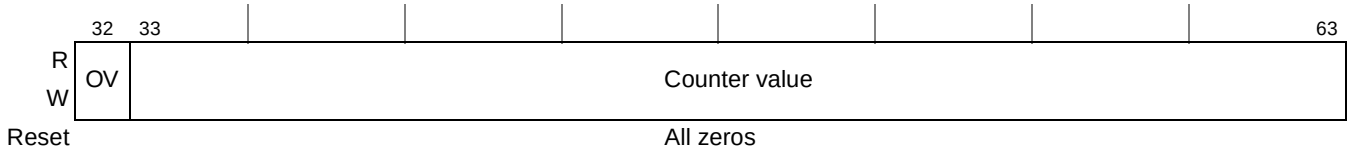


Figure 6-61. Performance Monitor Counter Registers (PMC0–PMC3)/User Performance Monitor Counter Registers (UPMC0–UPMC3)

Table 6-46. PMC0–PMC3 Field Descriptions

Bits	Name	Description
32	OV	Overflow. When this bit is set, it indicates this counter reaches its maximum value.
33–63	Counter value	Indicates the number of occurrences of the specified event

Chapter 7
L2 Look-Aside Cache/SRAM

This chapter describes the organization of the on-chip L2/SRAM, cache coherency rules, cache line replacement algorithm, cache control instructions, and various cache operations. It also describes the interaction between the L2/SRAM and the e500 core complex.

7.1 L2 Cache Overview

The integrated 1024-Kbyte L2 cache is organized as 4096 eight-way sets of 32-byte cache lines based on 36-bit physical addresses, as shown in Figure 7-1.

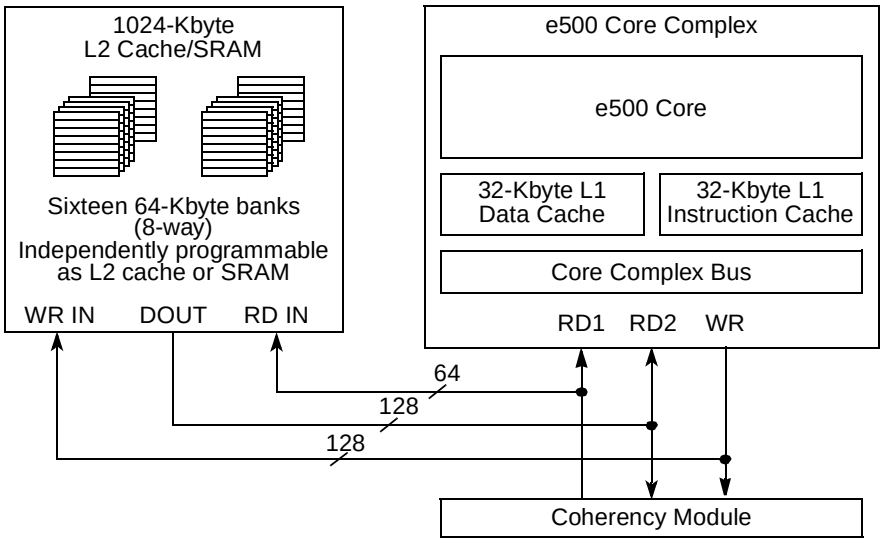


Figure 7-1. L2 Cache/SRAM Configuration

The SRAM can be configured with memory-mapped registers as externally accessible memory-mapped SRAM in addition to or instead of cache. The L2 cache can operate in the following modes, described in Section 7.2, “L2 Cache and SRAM Organization”:

- Full cache mode (1024-Kbyte cache)
- Full memory-mapped SRAM mode
- Partial SRAM and partial cache mode, in which one eighth, one quarter, or one half the total on-chip memory can be allocated to 1 or 2 SRAM regions

7.1.1 L2 Cache and SRAM Features

The L2 cache has the following characteristics:

- Supports 36-bit address space

- Write-through, front-side cache
  - Front-side design provides easier cache access for the I/O masters, such as Ethernet, PCI Express, and RapidIO
  - Write-through design is more efficient on the processor bus for front-side caches
- Valid, locked, and stale states (no modified state)
- Two input data buses (64 and 128 bits wide) and one output data bus (128 bits wide)
- All accesses are fully pipelined and non-blocking (allows hits under misses)
- 1024-Kbyte array organized as 4096 eight-way sets of 32-byte cache lines
- Eight-way set associativity with a pseudo-LRU (7-bit) replacement algorithm.
  - High level of associativity yields good performance even with many lines locked or used as SRAM regions
- I/O devices can store data into the cache in a process called ‘stashing.’
  - Stashing is indicated for global I/O writes either by a transaction attribute or by a programmable memory range
  - Regions of the cache can be reserved exclusively for stashing to prevent pollution of processor cache regions.
- Processor L2 cache regions are configurable to allocate instructions, data, or both.
  - External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types (stashing).
    - 1, 2, or 4 ways can be configured for stashing only
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag arrays use 19 tag bits and 1 tag parity bit per line.
- Multiple cache locking methods supported
  - Individual line locks are set and cleared by using e500 cache locking APU instructions—Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**).
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR*n*).
  - The entire cache can be locked by setting a configuration register appropriately.
- Lock clearing methods
  - Individual locks can be cleared by cache-locking APU instructions—Data Cache Block Lock Clear (**dcblc**) and Instruction Cache Block Lock Clear (**icblc**)—or by a snooped flush unless entire cache is locked.
  - Flash clearing of all instruction and/or data locks is done by writes to configuration registers.
  - An unlock attribute can be attached to a read instruction.
- Error injection modes supported for testing error handling

SRAM features include the following:

- SRAM regions are created by configuring 1, 2, 4 or 8 ways of each set to be reserved for memory-mapped SRAM.
- Regions can reside at any location in the memory map aligned to the SRAM size.
- SRAM memory is byte addressable; for accesses of less than a cache line, ECC is updated using read-modify-write transactions.
- I/O devices access SRAM regions by marking transactions as snoopable (global).

Table 7-1 lists the possible L2 cache/SRAM configurations.

Table 7-1. Available L2 Cache/SRAM Configurations

Cache	Stash-only Region	SRAM Region 1	SRAM Region 2
1024 Kbytes	—	—	—
896 Kbytes	—	128 Kbytes	—
	128 Kbytes	—	—
768 Kbytes	—	256 Kbytes	—
		128 Kbytes	128 Kbytes
	128 Kbytes	128 Kbytes	—
	256 Kbytes	—	—
640 Kbytes	128 Kbytes	256 Kbytes	—
		128 Kbytes	128 Kbytes
	256 Kbytes	128 Kbytes	—
512 Kbytes	—	512 Kbytes	—
		256 Kbytes	256 Kbytes
	256 Kbytes	256 Kbytes	—
		128 Kbytes	128 Kbytes
	256 Kbytes	—	—
384 Kbytes	128 Kbytes	512 Kbytes	—
		256 Kbytes	256 Kbytes
	512 Kbytes	128 Kbytes	—
256 Kbytes	256 Kbytes	512 Kbytes	—
		256 Kbytes	256 Kbytes
	512 Kbytes	256 Kbytes	—
		128 Kbytes	128 Kbytes
—	—	1024 Kbytes	—
		512 Kbytes	512 Kbytes
	512 Kbytes	512 Kbytes	—
		256 Kbytes	256 Kbytes

7.2 L2 Cache and SRAM Organization

The on-chip memory array has sixteen banks, each containing 256 sets of eight cache blocks (or ‘ways’), as shown in Figure 7-2. Each block consists of 32 bytes of data and a tag.

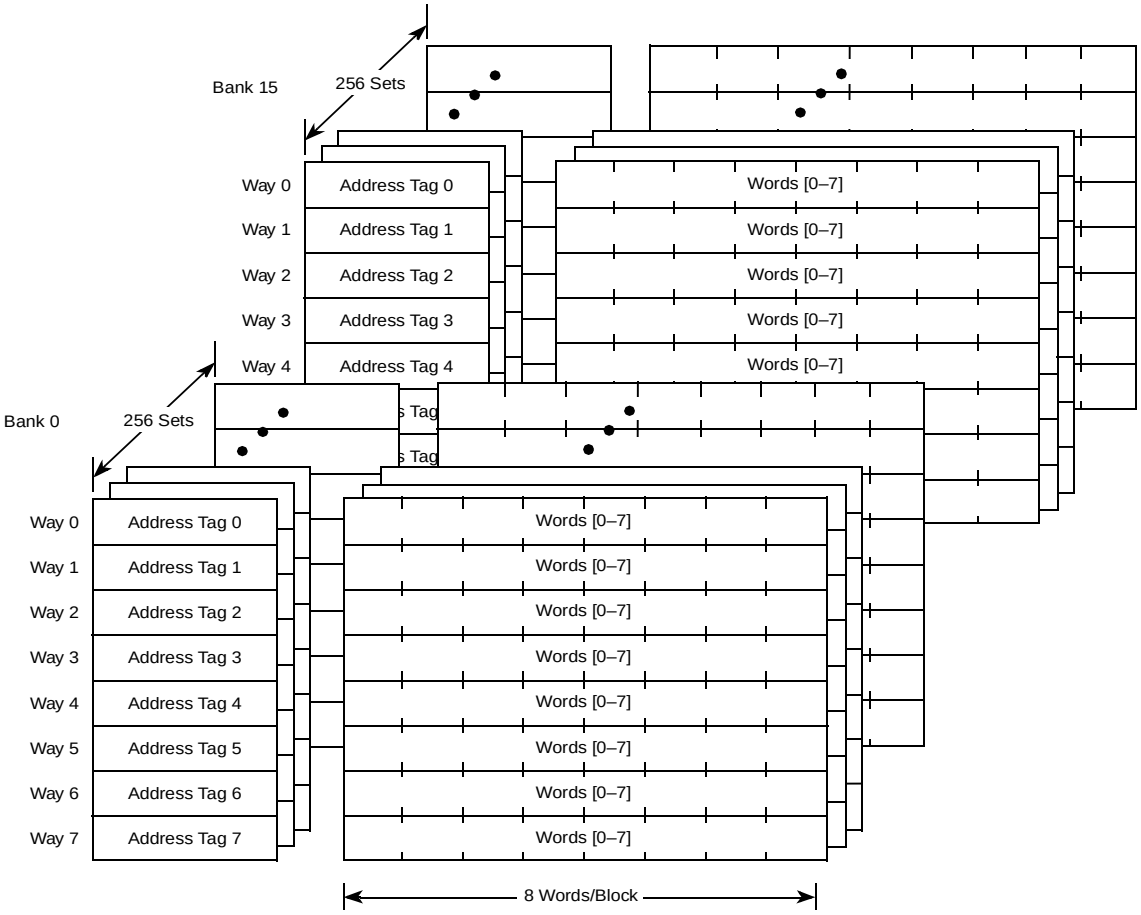


Figure 7-2. Cache Organization

7.2.1 Accessing the On-Chip Array as an L2 Cache

Figure 7-3 shows how physical address bits are used to access the L2 cache.

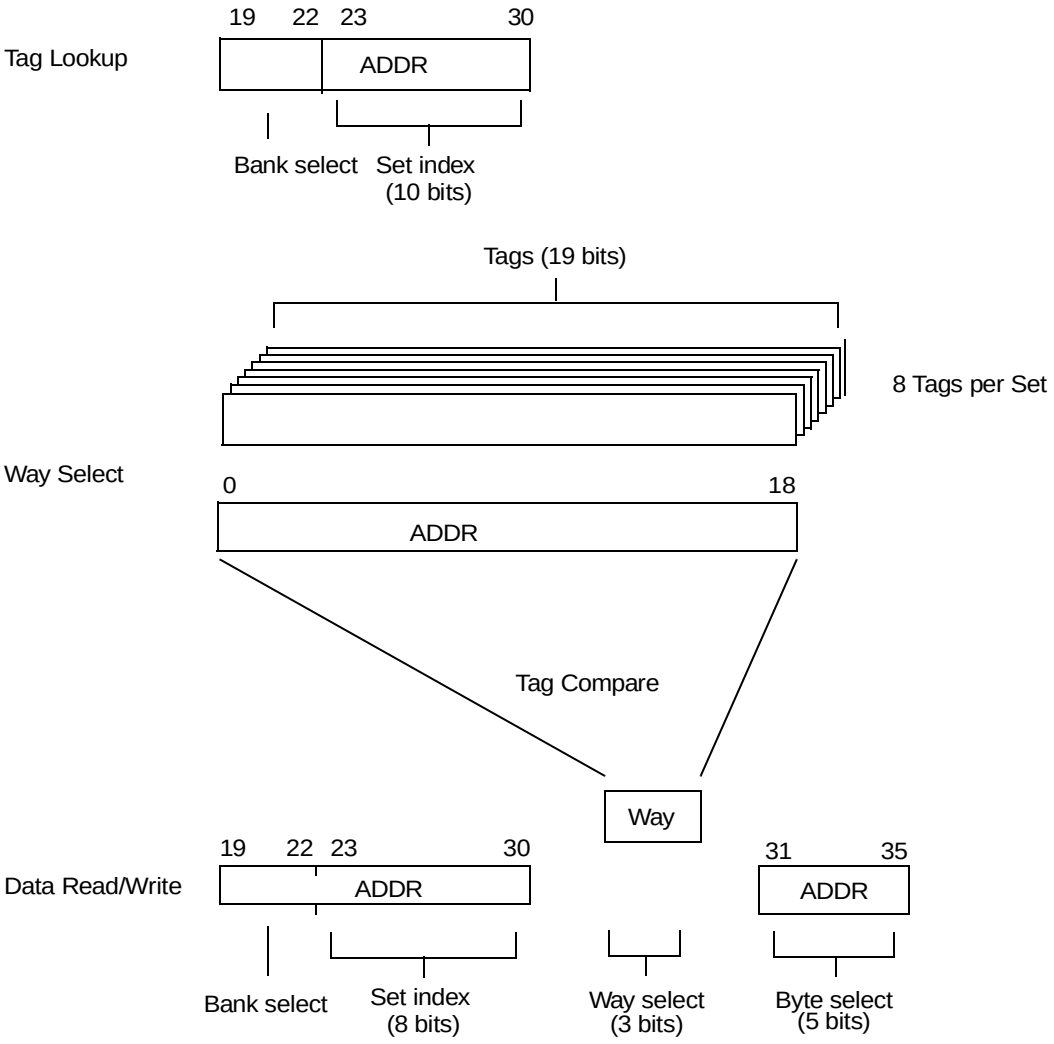


Figure 7-3. Physical Address Usage for L2 Cache Accesses

Physical address bits 19–30 identify the bank and set of the tag and data. Physical address bits 0–18 are compared against the tags of all eight ways. A match of a valid tag selects a 32-byte block of data (or way) within the set. Physical address bits 31–35 identify the byte or bytes of data within the block.

7.2.2 Accessing the On-Chip Array as an SRAM

When all or part of the array is dedicated to memory mapped SRAM, individual ways of each set are reserved for that purpose. SRAM accesses use physical address bits 16–18 in conjunction with the SRAM mode to select a way of the indexed set.

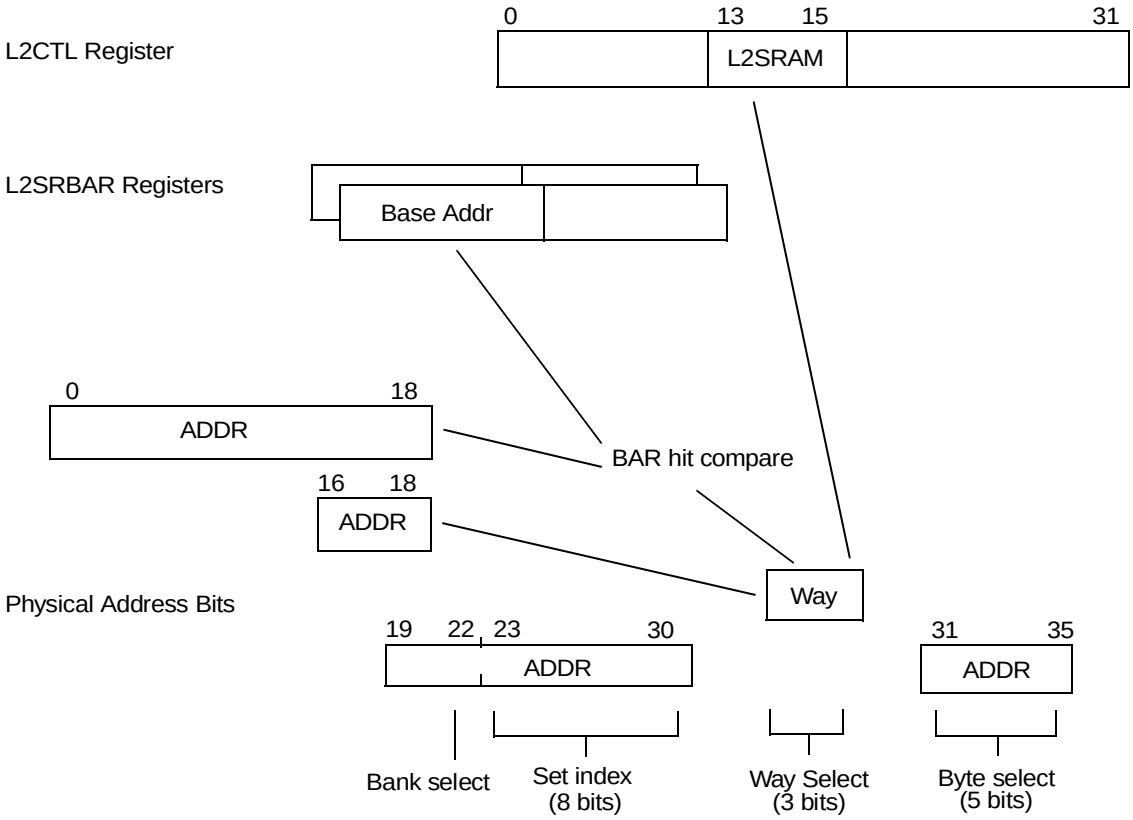


Figure 7-4. Physical Address Usage for SRAM Accesses

The mapping of address bits and SRAM mode to a way select is shown below in Table 7-2. SRAM size is dependent on the value of L2CTL[L2SIZ], since L2SIZ can be 256 Kbytes, 512 Kbytes, or 1024 Kbytes.

Table 7-2. Way Selection for SRAM Accesses

Description	L2SRAM	BAR 0 Hit	BAR 1 Hit	Addr[16:18]	Way Select
No SRAM	000	—	—	—	—
Entire Array is SRAM (single 1024KB SRAM if L2SIZ=1024KB)	001	1	0	000	0
				001	1
				010	2
				011	3
				100	4
				101	5
				110	6
				111	7

Table 7-2. Way Selection for SRAM Accesses (continued)

Description	L2SRAM	BAR 0 Hit	BAR 1 Hit	Addr[16:18]	Way Select
One half of array is an SRAM (single 512KB SRAM if L2SIZ=1024KB)	010	1	0	x00	0
				x01	1
				x10	2
				x11	3
Both halves of array are SRAM (two 512KB SRAMs if L2SIZ=1024KB)	011	1	0	x00	0
				x01	1
				x10	2
				x11	3
		0	1	x00	4
				x01	5
				x10	6
				x11	7
One quarter of the array is SRAM (single 256KB SRAM if L2SIZ=1024KB)	100	1	0	xx0	0
				xx1	1
Two quarters of the array are SRAMs (two 256KB SRAMs if L2SIZ=1024KB)	101	1	0	xx0	0
				xx1	1
		0	1	xx0	2
				xx1	3
One eighth of the array is an SRAM (single 128KB SRAM if L2SIZ=1024KB)	110	1	0	—	0
Two eighths of the array are SRAM (two 128KB SRAMs if L2SIZ=1024KB)	111	1	0	—	0
		0	1	—	1

7.2.3 Connection of the On-Chip Memory to the System

The e500 core connects to the L2 cache and the system interface through the high-speed core complex bus (CCB). The e500 core and the L2 cache connect to the rest of the integrated device through the e500 coherency module (ECM). Figure 7-5 shows the data connections of the e500 core and L2/SRAM. The e500 core can simultaneously read 128 bits of data from the L2/SRAM, read 64 bits of data from the system interface, and write 128 bits of data to the L2/SRAM and/or system interface.

The L2/SRAM can be accessed by the e500 core or the system interface through the ECM. The L2 cache does not initiate transactions. Figure 7-5 shows the data bus connections of the e500 core and L2/SRAM.

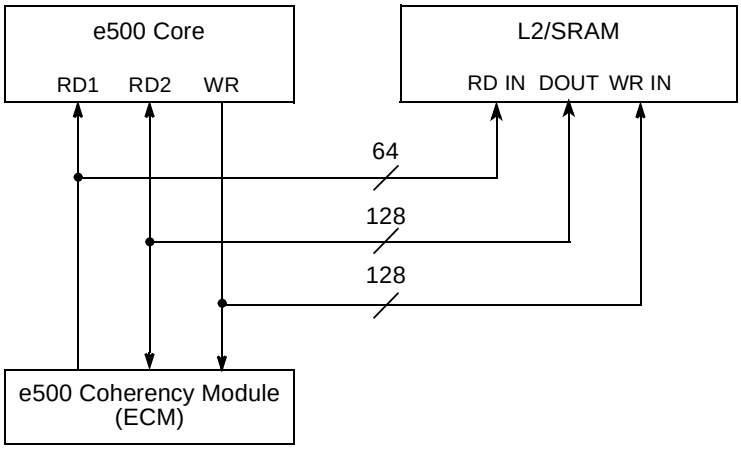


Figure 7-5. Data Bus Connection of CCB

Figure 7-6 shows address connections of the e500 core and L2/SRAM.

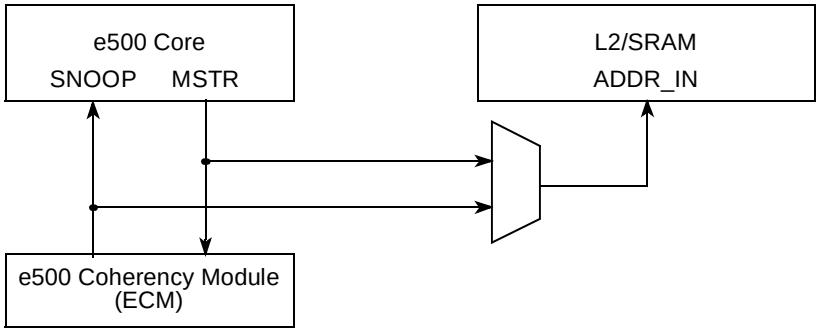


Figure 7-6. Address Bus Connection of CCB

In SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3 Memory Map/Register Definition

Table 7-3 shows the memory map for the L2/SRAM registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 7-3. L2/SRAM Memory-Mapped Registers

Offset	Register	Access	Reset	Section/Page
0x2_0000	L2CTL—L2 control register	R/W	0x3000_0000	7.3.1.1/7-9
0x2_0004	L2CWAP—L2 cache way allocation for processors	R/W	0x0000_0000	7.3.1.2/7-13
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.3.1/7-15
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3.3/7-16
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.3.1/7-15
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3.3/7-16
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.3.1/7-15
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3.3/7-16
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.3.1/7-15
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3.3/7-16
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4.1/7-18
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4.1/7-18
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5.1/7-20
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5.1/7-20
0x2_0E08	L2ERRINJCTL—L2 error injection mask control register	R/W	0x0000_0000	7.3.1.5.1/7-20
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5.2/7-22
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5.2/7-22
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5.2/7-22
0x2_0E40	L2ERRDET—L2 error detect register	w1c	0x0000_0000	7.3.1.5.2/7-22
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5.2/7-22
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5.2/7-22
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5.2/7-22
0x2_0E50	L2ERRADDRL—L2 error address capture register low	R	0x0000_0000	7.3.1.5.2/7-22
0x2_0E54	L2ERRADDRH—L2 error address capture register high	R	0x0000_0000	7.3.1.5.2/7-22
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5.2/7-22

7.3.1 L2/SRAM Register Descriptions

The following sections describe registers that control and configure the L2/SRAM array.

7.3.1.1 L2 Control Register (L2CTL)

The L2 control register (L2CTL), shown in Figure 7-7, controls configuration and operation of the L2/SRAM array. The sequence for modifying L2CTL is as follows:

1. **mbar**
2. **isync**
3. **stw** (WIMG = 01xx) CCSRBAR+0x2_0000

4. **lwz** (WIMG = 01xx) CCSRBAR+0x2_0000
5. **mbar**

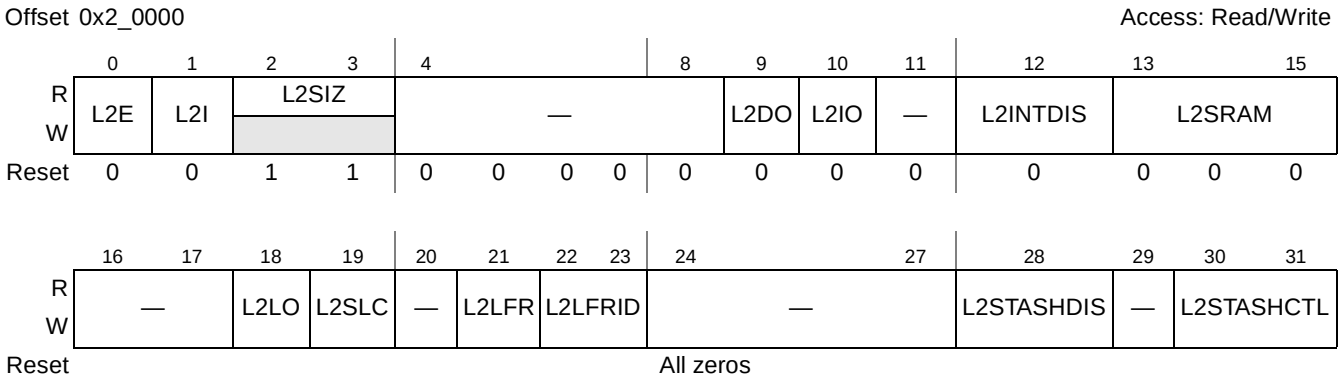


Figure 7-7. L2 Control Register (L2CTL)

Table 7-4 describes L2CTL fields.

Table 7-4. L2CTL Field Descriptions

Bits	Name	Description
0	L2E	L2 enable. Used to enable the L2 array (cache or memory-mapped SRAM). 0 The L2 SRAM (cache and memory-mapped SRAM) is disabled and is not accessed for reads, snoops, or writes. Setting the L2 flash invalidate bit (L2I) is allowed. 1 The L2 SRAM (cache or memory-mapped SRAM) is enabled. Note that L2I can be set regardless of the value of L2E.
1	L2I	L2 flash invalidate. 0 The L2 status and LRU bits are not being cleared. 1 Setting L2I invalidates the L2 cache globally by clearing the all the L2 status bits, as well as the LRU algorithm. Memory-mapped SRAM is unaffected. Data to memory-mapped SRAM are unaffected by the flash invalidate. The hardware automatically clears L2I when the invalidate is complete.
2–3	L2SIZ	L2 SRAM size (read only). Indicates the total available size of on-chip memory array (to be configured as cache or memory-mapped SRAM). 00 Reserved 01 256 Kbyte 10 512 Kbyte 11 1024 Kbyte
4–8	—	Reserved
9	L2DO	L2 data-only. Reserved in full memory-mapped SRAM mode. L2DO may be changed while the L2 is enabled or disabled. 0 The L2 cache allocates entries for instruction fetches that miss in the L2. 1 The L2 cache allocates entries for processor data loads that miss in the L2 and for processor L1 castouts but does not allocate entries for instruction fetches that miss in the L2. Instruction accesses that hit in the L2, data accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.

Table 7-4. L2CTL Field Descriptions (continued)

Bits	Name	Description
10	L2IO	L2 instruction-only. Reserved in full memory-mapped SRAM mode. Causes the L2 cache to allocate lines for instruction cache transactions only. L2IO may be changed while the L2 is enabled or disabled. 0 The L2 cache entries are allocated for data loads that miss in the L2 and for processor L1 castouts. 1 The L2 cache allocates entries for instruction fetch misses, but does not allocate entries for processor data transactions. Data accesses that hit in the L2, instruction accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.
11	—	Reserved
12	L2INTDIS	Cache read intervention disable. Reserved for full memory-mapped SRAM mode. Used to disable cache read intervention. May be changed while the L2 is enabled or disabled. 0 Cache intervention is enabled. The ECM ensures that if a data read from another device hits in the L2 cache, it is serviced from the L2 cache. 1 Cache intervention is disabled
13–15	L2SRAM	L2 SRAM configuration. Determines the L2 cache/memory-mapped SRAM allocation of the on-chip memory array. SRAM size depends on the value of L2SIZ. Since L2SIZ can be 256KB, 512KB or 1024KB, SRAM can have sizes from 32KB to 1024KB. 000 No SRAM. Entire array is cache. 001 Entire array is a single SRAM (1024-Kbyte SRAM for L2SIZ = 1024Kbytes) 010 One half of the array is an SRAM (512-Kbyte SRAM for L2SIZ = 1024Kbytes) 011 Both halves of the array are SRAMs (two 512-Kbyte SRAMs for L2SIZ = 1024Kbytes) 100 One quarter of the array is an SRAM (one 256-Kbyte SRAM for L2SIZ = 1024Kbytes) 101 Two quarters of the array are SRAMs (two 256-Kbyte SRAMs for L2SIZ = 1024Kbytes) 110 One eighth of the array is an SRAM (one 128-Kbyte SRAM for L2SIZ = 1024Kbytes) 111 Two eighths of the array are SRAMs (two 128-Kbyte SRAMs for L2SIZ = 1024Kbytes) For one SRAM region L2SRBAR0 is used and for two SRAM regions L2SRBAR0 and L2SRABAR1 are used. Regions of the array that are not allocated to SRAMs are used as cache memory. To change these bits, the L2 must be disabled (L2CTL[L2E] = 0).
16–17	—	Reserved
18	L2LO	L2 cache lock overflow. Reserved in full memory-mapped SRAM mode. This sticky bit is set if an overlock condition is detected in the L2 cache. A lock overflow is triggered either by executing instruction or data cache block touch and lock set instructions or by performing L2 cache external writes with lock set. If all ways are locked and an attempt to stash is made, the stash is not allocated. 0 The L2 cache did not encounter a lock overflow. L2LO is cleared only by software. 1 The L2 cache encountered a lock overflow condition.
19	L2SLC	L2 snoop lock clear. This sticky bit is set if a snoop invalidated a locked data cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. L2SLC is reserved in full memory-mapped SRAM mode. 0 A snoop did not invalidate a locked L2 cache line. L2SLC is cleared only by software. 1 The L2 cache encountered a snoop that invalidated a locked line.
20	—	Reserved

Table 7-4. L2CTL Field Descriptions (continued)

Bits	Name	Description
21	L2LFR	L2 cache lock bits flash reset. The L2 cache must be enabled (L2CTL[L2E] = 1) for reset to occur. This field is reserved in full memory-mapped SRAM mode. 0 The L2 cache lock bits are not cleared or the clear operation completed. 1 A reset operation is issued that clears each L2 cache line's lock bits. Depending on the L2LFRID value, data or instruction locks, or both, can be reset. Cache access is blocked during this time. After L2LFR is set, the L2 cache unit automatically clears L2LFR when the reset operation is complete (if L2CTL[L2E] is set).
22–23	L2LFRID	L2 cache lock bits flash reset select instruction or data. Indicates whether data or instruction lock bits or both are reset. 00 Not used 01 Reset data locks if L2LFR = 1. 10 Reset instruction locks if L2LFR = 1. 11 Reset both data and instruction locks if L2LFR = 1.
24–27	—	Reserved
28	L2STASHDIS	L2 stash allocate disable. Disables allocation of lines for stashing. 0 The L2 cache allocates lines for global writes that hit in a stash range or that have the stashing attribute set. 1 The L2 does not allocate lines for stashed writes. **Note:** This bit does NOT affect the updating of lines that are already resident in the cache and have the stash attribute set or hit a stash range. Such lines are updated even if this bit is set.To change this bit, the L2 must be disabled (L2CTL[L2E] = 0).
29	—	Reserved
30–31	L2STASH	L2 stash configuration. This field reserves regions of the cache for stash-only operation. That is, blocks of each cache set are reserved so that they can only be allocated for stash data. If such a region is created, processor reads and writes are not allocated into this region; it can only be populated by stash writes. Similarly, stash writes are only allocated into this region. This prevents processor and stashed I/O data from polluting one another. 00 No stash-only region. Stashed writes are allocated across the entire cache and can evict processor data and can be evicted by processor data. 01 One half of the array is a stash-only cache (way4, way5, way6 & way7 of each set) 10 One quarter of the array is a stash-only cache (way6 & way7 of each set) 11 One eighth of the array is a stash-only cache (way7 of each set) Like L2SRAM configuration, stash-only regions subtract from the amount of the on-chip memory that is available to the processor as cache. If the L2SRAM configuration uses the entire on-chip memory array as SRAM, then no stash-only region can be created. To change these bits, the L2 must be disabled (L2CTL[L2E] = 0). This field has no effect if the L2STASHDIS bit is set.

7.3.1.2 L2 Cache Way Allocation for Processors Register (L2CWAP)

The L2 cache way allocation for processors (L2CWAP) register, shown in Figure 7-8, allows the L2 cache to be programmed to distribute 8 ways between two processor cores. A pointer is used to set the ways allocated between processor 0 and processor 1.

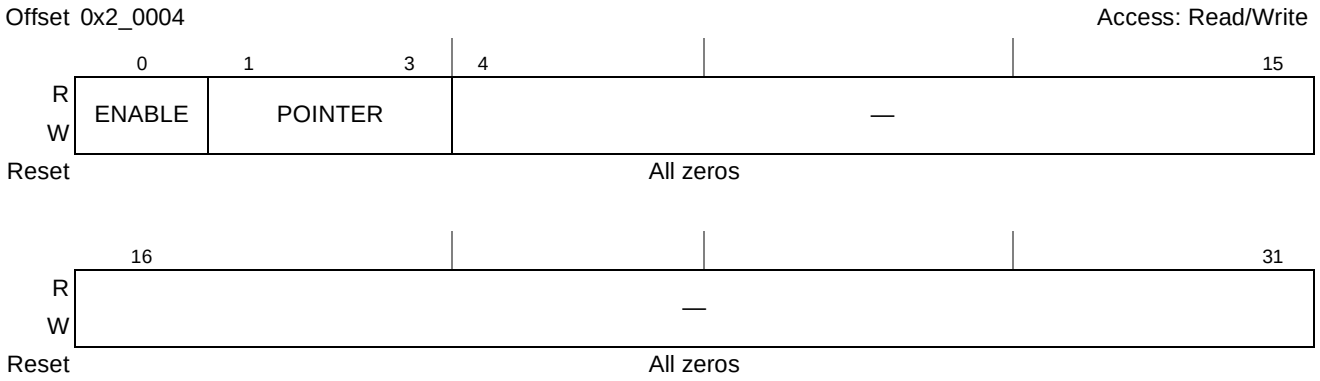


Figure 7-8. L2 Cache Way Allocation for Processors Register (L2CWAP)

Table 7-5 describes L2CWAP fields.

Table 7-5. L2CWAP Field Descriptions

Bits	Name	Description
0	ENABLE	L2 cache way pointer enable. The L2 cache uses the pointer value only when ENABLE is set. 0 The L2 way pointer is disabled. No ways are allocated only for processor 0 or for processor 1. All ways (except ways allocated for SRAM and stashing) are shared between processors. 1 The L2 way pointer is enabled. Ways are allocated for processors based on way pointer.
1–3	POINTER	L2 cache way pointer. This pointer is used to allocate ways between processor 0 and processor 1. To change this field the L2 must be disabled (L2CTL[L2E] = 0). The following description for way pointer values assumes that no ways are allocated for L2SRAM (L2CTL[13:15]) and L2STASH (L2CTL[30:31]). 000 Way0 – way7 are allocated for processor 1 and no ways are allocated for processor 0. 001 Way0 is allocated for processor 0 and way1 – way7 are allocated for processor 1. 010 Way0 – way1 are allocated for processor 0 and way2 – way7 are allocated for processor 1. 011 Way0 – way2 are allocated for processor 0 and way3 – way7 are allocated for processor 1. 100 Way0 – way3 are allocated for processor 0 and way4 – way7 are allocated for processor 1. 101 Way0 – way4 are allocated for processor 0 and way5 – way7 are allocated for processor 1. 110 Way0 – way5 are allocated for processor 0 and way6 – way7 are allocated for processor 1. 111 Way0 – way6 are allocated for processor 0 and way7 is allocated for processor 1. If ways have been allocated for SRAM (L2CTL[13:15]) or stashing (L2CTL[30:31]) way allocation is affected as noted in the following examples. In general, when allocating ways among SRAM, stashing, and processors, the controller gives SRAM first preference, then stashing, then processors. Suppose 2 ways are allocated for SRAM and 2 ways are allocated for stashing and POINTER = 100... Way0 -> SRAM Way1 -> SRAM Way2 -> Processor 0 Way3 -> Processor 0 Way4 -> Processor 1 Way5 -> Processor 1 Way6 -> Stash Way7 -> Stash Suppose 4 ways are allocated for SRAM and 2 ways are allocated for stashing and POINTER = 100... Way0 -> SRAM Way1 -> SRAM Way2 -> SRAM Way3 -> SRAM Way4 -> Processor 1 Way5 -> Processor 1 Way6 -> Stash Way7 -> Stash Suppose 2 ways are allocated for SRAM and no ways are allocated for stashing and POINTER = 100... Way0 -> SRAM Way1 -> SRAM Way2 -> Processor 0 / Stash Way3 -> Processor 0 / Stash Way4 -> Processor 1 / Stash Way5 -> Processor 1 / Stash Way6 -> Processor 1 / Stash Way7 -> Processor 1 / Stash
4–31	—	Reserved

7.3.1.3 L2 Cache External Write Registers

The MPC8572E supports allocating and locking L2 cache lines from external agents such as PCI. This functionality is called stashing. Four sets of registers are provided to support this feature; each set has three registers that specify a programmed memory range that can be locked with a snoop write transaction. All three registers in a set must be configured in order to use an external write address.

These registers are the L2 cache external write address registers 0–3 (L2CEWAR*n*), the L2 cache external write address registers extended address 0–3 (L2CEWAREA*n*), and the L2 cache external write control registers 0–3 (L2CEWCR*n*). L2CEWAR*n* contain the lower 24 bits of the external write base address and L2CEWAREA*n* contain the upper 4 bits. The base address specified in the address registers must be naturally aligned to the window size in the corresponding control register.

Further details on the locations and fields of these registers are given in the following sections.

7.3.1.3.1 L2 Cache External Write Address Registers 0–3 (L2CEWAR*n*)

The L2CEWAR*n* registers contain the lower 24 bits of the 28-bit L2 cache external write base address. Each of these registers has identical fields, as shown in Figure 7-9.

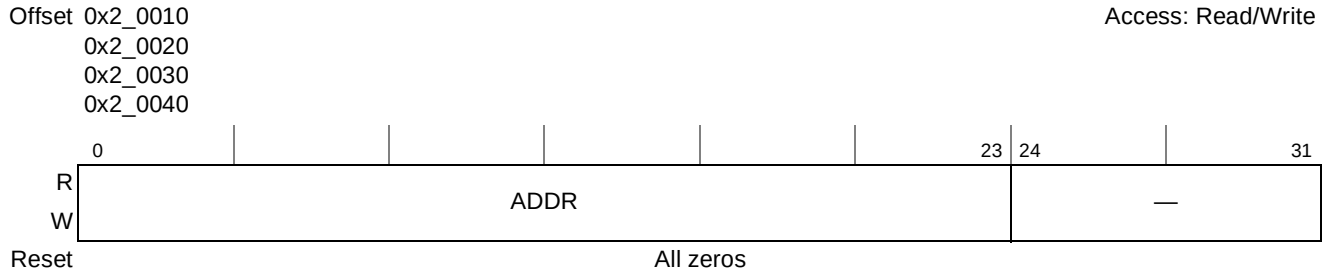


Figure 7-9. Cache External Write Address Registers (L2CEWAR*n*)

Table 7-6 describes L2CEWAR*n* fields.

Table 7-6. L2CEWAR*n* Field Descriptions

Bits	Name	Description
0–23	ADDR	Contains the lower 24 bits of the 28-bit L2 cache external write base address. Note that the upper 4 bits of the base address are in L2CEWAREA*n*[ADDR].
24–31	—	Reserved

7.3.1.3.2 L2 Cache External Write Address Registers Extended Address (L2CEWAREA*n*)

The L2 cache external write address registers extended address (L2CEWAREA*n*), shown in Figure 7-10, contain the upper 4 bits of the 28-bit L2 cache external write base address.

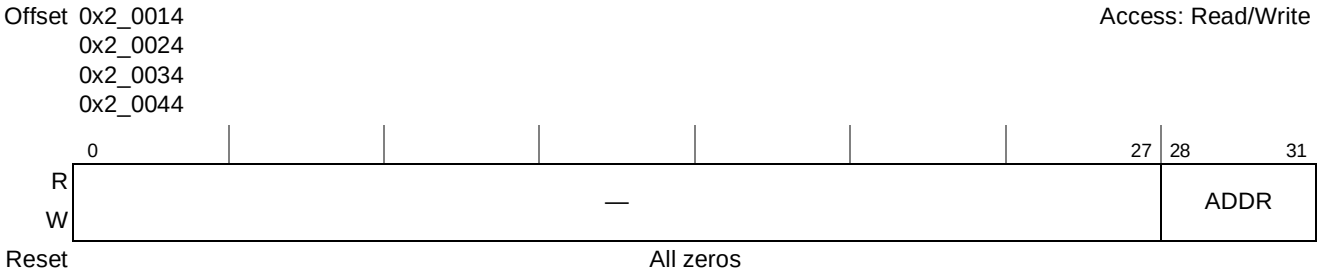


Figure 7-10. Cache External Write Address Registers Extended Address (L2CEWAREA*n*)

Table 7-7 describes the fields of L2CEWAREA*n*.

Table 7-7. L2CEWAREA*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Contains the upper 4 bits of the L2 cache external write base address. Note that the rest of the base address is in L2CEWAR*n*[ADDR].

7.3.1.3.3 L2 Cache External Write Control Registers 0–3 (L2CEWCR*n*)

The L2CEWAR*n*/L2CEWAREA*n* address registers work with the L2 cache external write control registers 0–3 (L2CEWCR*n*), shown in Figure 7-11, to control cache external write functionality.

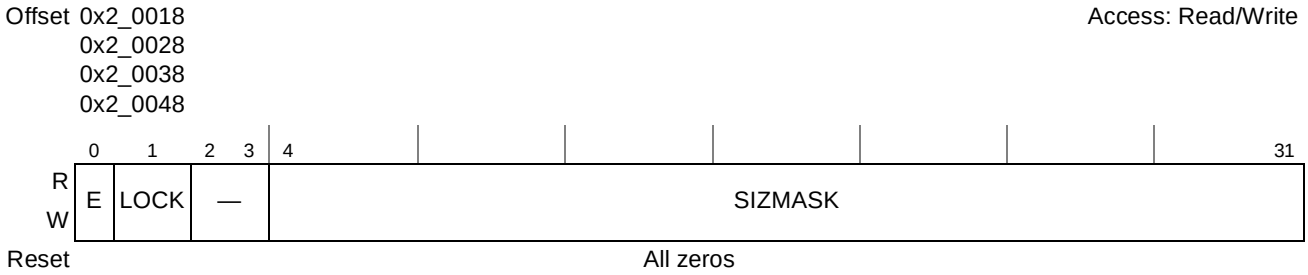


Figure 7-11. Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

The L2CEWCR*n* registers contain identical fields, which are described in Table 7-8.

Table 7-8. L2CEWCR*n* Field Descriptions

Bits	Name	Description
0	E	External write enable. An external write matching the address window defined by L2CEWAR*n*/L2CEWAREA*n*/L2CEWCR*n* is allocated or updated in the L2 cache. 0 External writes for the L2CEWAR*n*/L2CEWAREA*n*/L2CEWCR*n* set are disabled. 1 External writes are enabled for the L2CEWAR*n*/L2CEWAREA*n*/L2CEWCR*n* set.
1	LOCK	Lock lines in the targeted cache. An external write matching the address window defined by L2CEWAR*n*/L2CEWAREA*n*/L2CEWCR*n* is locked in the L2 cache when it is allocated or updated. 0 The locked bit is not set when a line is allocated unless explicitly specified by transaction attributes. 1 Cache lines are allocated as locked. A hit to a valid, unlocked lines sets the lock.
2–3	—	Reserved
4–31	SIZMASK	Mask size. Defines the size of the naturally aligned address region for cache external writes. The address region must be aligned to a boundary that is a multiple of the mask size. Any value not listed below is illegal and produces boundedly undefined results. 1111 1111 1111 1111 1111 1111 1111 256 bytes 1111 1111 1111 1111 1111 1111 1110 512 bytes 1111 1111 1111 1111 1111 1111 1100 1 Kbyte 1111 1111 1111 1111 1111 1111 1000 2 Kbytes 1111 1111 1111 1111 1111 1111 0000 4 Kbytes 1111 1111 1111 1111 1111 1110 0000 8 Kbytes 1111 1111 1111 1111 1111 1100 0000 16 Kbytes 1111 1111 1111 1111 1111 1000 0000 32 Kbytes 1111 1111 1111 1111 1111 0000 0000 64 Kbytes 1111 1111 1111 1111 1110 0000 0000 128 Kbytes 1111 1111 1111 1111 1100 0000 0000 256 Kbytes 1111 1111 1111 1111 1000 0000 0000 512 Kbytes 1111 1111 1111 1111 0000 0000 0000 1 Mbyte 1111 1111 1111 1110 0000 0000 0000 2 Mbytes 1111 1111 1111 1100 0000 0000 0000 4 Mbytes 1111 1111 1111 1000 0000 0000 0000 8 Mbytes 1111 1111 1111 0000 0000 0000 0000 16 Mbytes 1111 1111 1110 0000 0000 0000 0000 32 Mbytes 1111 1111 1100 0000 0000 0000 0000 64 Mbytes 1111 1111 1000 0000 0000 0000 0000 128 Mbytes 1111 1111 0000 0000 0000 0000 0000 256 Mbytes 1111 1110 0000 0000 0000 0000 0000 512 Mbytes 1111 1100 0000 0000 0000 0000 0000 1 Gbyte 1111 1000 0000 0000 0000 0000 0000 2 Gbytes 1111 0000 0000 0000 0000 0000 0000 4 Gbytes 1110 0000 0000 0000 0000 0000 0000 8 Gbytes 1100 0000 0000 0000 0000 0000 0000 16 Gbytes 1000 0000 0000 0000 0000 0000 0000 32 Gbytes 0000 0000 0000 0000 0000 0000 0000 64 Gbytes

7.3.1.4 L2 Memory-Mapped SRAM Registers

The registers described in this section, the L2 memory-mapped SRAM base address registers 0–1 (L2SRBAR*n*) and the L2 memory-mapped SRAM base address registers extended address 0–1 (L2SRBAREA*n*), control the memory-mapped SRAM mode functionality. Together, these two pairs of registers define memory blocks that can be mapped into the L2 cache.

Specified SRAM base addresses must be aligned to the size of the SRAM region. If L2CTL[L2SRAM] specifies one memory-mapped SRAM block, its base address must be written to the pair L2SRBAR0 and L2SRBAREA0; if it specifies two memory-mapped SRAM blocks, L2SRBAR1 and L2SRBAREA1 are used as well.

7.3.1.4.1 L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR*n*)

The L2 memory-mapped SRAM base address registers (L2SRBAR*n*), shown in Figure 7-12, contain the lower 18 bits of the 22-bit SRAM base address.

Figure 7-12. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)

L2SRBAR bits are described in Table 7-9.

Table 7-9. L2SRBAR*n* Field Descriptions

Bits	Name	Description
0–17	ADDR	Contains the lower 18 bits of the 22-bit L2 memory-mapped SRAM base address; the upper 4 bits are contained in L2SRBAREA*n*[ADDR]. (Note that some of these bits may not be needed, depending on how the L2 cache is partitioned.) The combined base address from L2SRBAREA*n*[ADDR] || L2SRBAR*n*[ADDR] is used as follows: SRAM Partition — Bits Required for SRAM Offset — Bits Used for Actual Base Address 64 Kbytes — 16 — 20 (0–19) 128 Kbytes — 17 — 19 (0–18) 256 Kbytes — 18 — 18 (0–17) 512 Kbytes — 19 — 17 (0–16) 1024 Kbytes — 20 — 16 (0–15) Unused bits of the base address are masked off by the hardware.
18–31	—	Reserved

When enabled, the windows defined in L2SRBAR*n* and L2SRBAREA*n* supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR (see Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register (CCSRBAR).”) Overlapping SRAM and local access windows is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if SRAM and local access windows overlap.

7.3.1.4.2 L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA*n*)

The L2 memory-mapped SRAM base address registers extended address (L2SRBAREA*n*), shown in Figure 7-13, contain the upper 4 bits of the L2 cache SRAM base address.

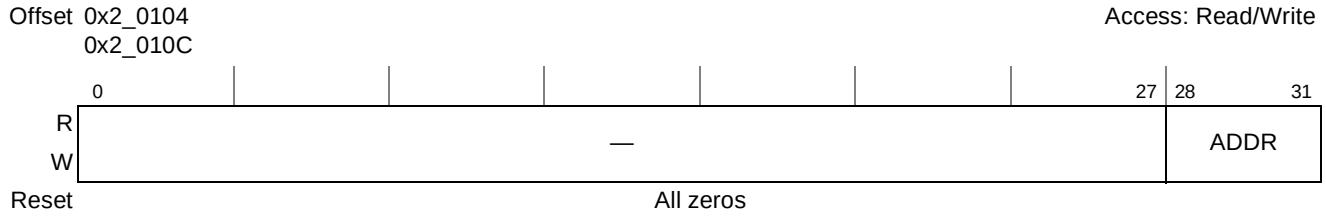


Figure 7-13. L2 Memory-Mapped SRAM Base Address Registers Extended Address 0–1 (L2SRBAREA*n*)

Table 7-10 describes the fields of L2SRBAREA*n*.

Table 7-10. L2SRBAREA*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Contains the upper 4 bits of the L2 cache SRAM base address. Note that the 18 low-order bits of the base address are contained in L2SRBAR*n*[ADDR].

7.3.1.5 L2 Error Registers

L2 error detection, reporting, and injection allow flexible handling of ECC and parity errors in the L2 data and tag arrays. When the MPC8572E detects an L2 error, the appropriate bit in the error detect register (L2ERRDET) is set. Error detection is disabled by setting the corresponding bit in the error disable register (L2ERRDIS).

The address and attributes of the first detected error are also saved in the error capture registers (L2ERRADDR, L2ERRATTR, L2CAPTDATAHI, L2CAPTDATALO, and L2CAPTACC). Subsequent errors set error bits in the error detection registers, but information is saved only for the first one. Error reporting (by generating an interrupt) is enabled by setting the corresponding bit in the error interrupt enable register (L2ERRINTEN). Note that the error detect bit is set regardless of the state of the interrupt enable bit. When an error is detected, if error detection is enabled the L2 cache/SRAM always asserts an internal error signal with read data to prevent the L1 caches and architectural registers from being loaded with corrupt data. If error detection is disabled, the detected error bit is not set and no internal signal is asserted.

The L2 error detect register (L2ERRDET) is implemented as a bit-reset type register. Reading from this register 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 0x0200_0000 is written to the register.

Note that in SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3.1.5.1 Error Injection Registers

The L2 cache includes support for injecting errors into the L2 data, data ECC, or tag. This may be used to test error recovery software by deterministically creating error scenarios.

The preferred method for error injection is to set all data pages to cache-inhibited (MMU TLB entry I = 1) except a scratch page, set L2CTL[L2DO] to prevent allocation of instruction accesses, and invalidate the L2 by setting L2CTL[L2I] = 1. The following code sequence triggers an error, then detects it (A is an address in the scratch page):

```
dcbz A          | allocates the line in the L1 in the modified state
dcbtls_L2 A     | forces the line from the L1 and allocates the line in the L2
lwz A
```

Data or tag errors are injected into the line, according to the error injection settings in L2ERRINJHI, L2ERRINJLO, and L2ERRINJCTL, at allocation. The final load detects and reports the error (if enabled) and allows software to examine the offending data, address, and attributes.

Note that error injection enable bits in L2ERRINJCTL must be cleared by software and the L2 must be invalidated (by setting L2CTL[L2I]) before resuming L2 normal operation. Figure 7-14 shows the L2 error injection mask high register (L2ERRINJHI).

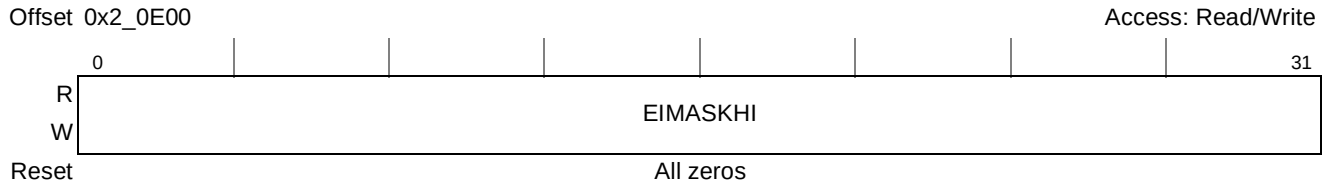


Figure 7-14. L2 Error Injection Mask High Register (L2ERRINJHI)

Table 7-11 describes L2ERRINJHI[EIMASKHI].

Table 7-11. L2ERRINJHI Field Description

Bits	Name	Description
0–31	EIMASKHI	Error injection mask/high word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache/SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-15 shows the L2 error injection mask low register (L2ERRINJLO).

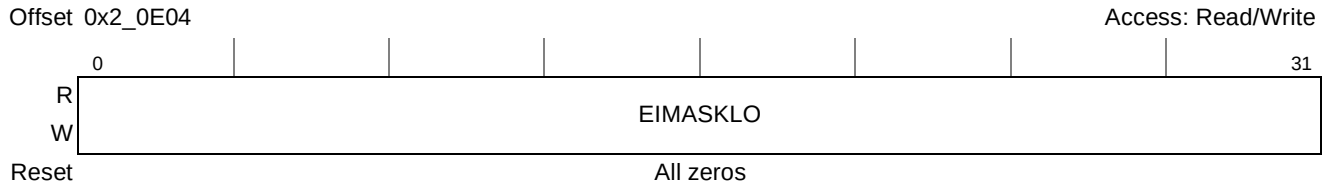


Figure 7-15. L2 Error Injection Mask Low Register (L2ERRINJLO)

Table 7-12 describes L2ERRINJLO[EIMASKLO].

Table 7-12. L2ERRINJLO Field Description

Bits	Name	Description
0–31	EIMASKLO	Error injection mask/low word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-16 shows the L2 error injection mask control register (L2ERRINJCTL).

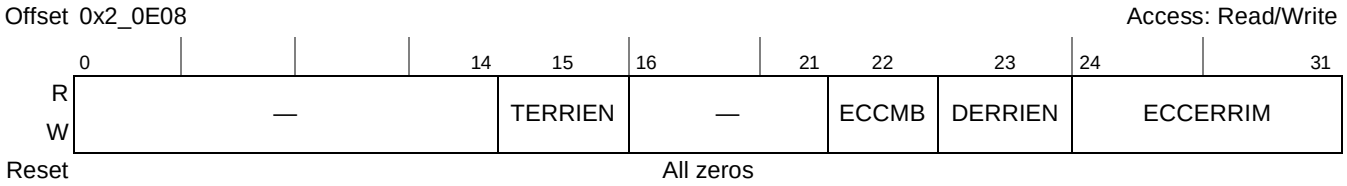


Figure 7-16. L2 Error Injection Mask Control Register (L2ERRINJCTL)

Table 7-13 describes L2ERRINJCTL fields.

Table 7-13. L2ERRINJCTL Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	TERRIEN	L2 tag array error injection enable 0 No tag errors are injected. 1 All subsequent entries written to the L2 tag array have the parity bit inverted.
16–21	—	Reserved
22	ECCMB	ECC mirror byte enable. 0 ECC byte mirroring is disabled 1 The most significant data path byte is mirrored onto the ECC byte if DERRIEN = 1.
23	DERRIEN	L2 data array error injection enable: 0 No data errors are injected. 1 Subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data and ECC error injection masks and/or data path byte mirrored onto ECC as specified by ECC mirror byte enable. **Note:** Note: if both ECC mirror byte and data error injection are enabled, ECC mask error injection is performed on the mirrored ECC.
24–31	ECCERRIM	Error injection mask for the ECC bits. A set bit corresponding to an ECC bit causes that bit to be inverted on SRAM writes if DERRIEN = 1.

7.3.1.5.2 Error Control and Capture Registers

The error control and capture registers control detection and reporting of tag parity, ECC and L2 configuration errors. L2 configuration errors are illegal combinations of L2 size and block size and are detected when the L2 is enabled (L2CTL[L2E] = 1). Figure 7-17 shows the L2 error capture data high register (L2CAPTDATAHI).

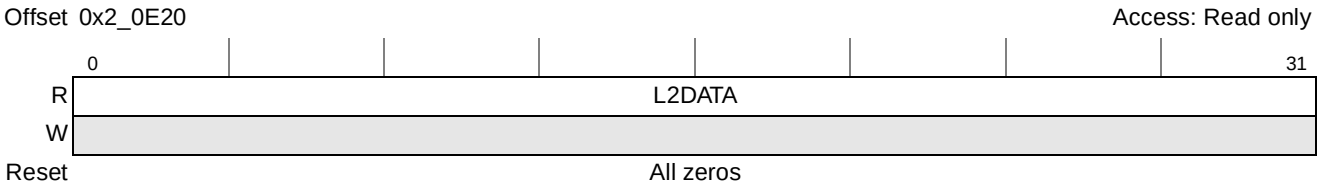


Figure 7-17. L2 Error Capture Data High Register (L2CAPTDATAHI)

Table 7-14 describes L2CAPTDATAHI[L2DATA].

Table 7-14. L2CAPTDATAHI Field Description

Bits	Name	Description
0–31	L2DATA	L2 data high word

Figure 7-18 shows the L2 error capture data low register (L2CAPTDATALO).

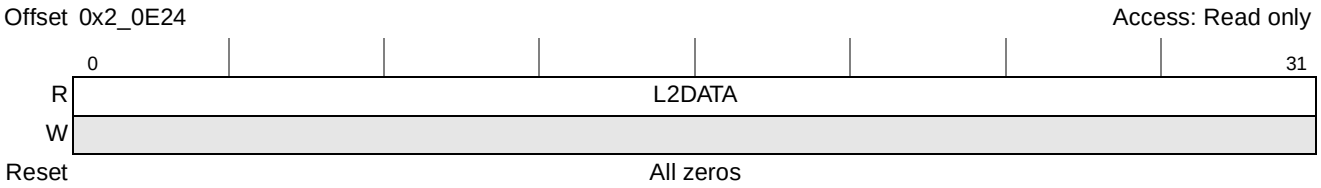


Figure 7-18. L2 Error Capture Data Low Register (L2CAPTDATALO)

Table 7-15 describes L2CAPTDATALO[L2DATA].

Table 7-15. L2CAPTDATALO Field Description

Bits	Name	Description
0–31	L2DATA	L2 data low word

Figure 7-19 shows the L2 error syndrome register (L2CAPTECC).

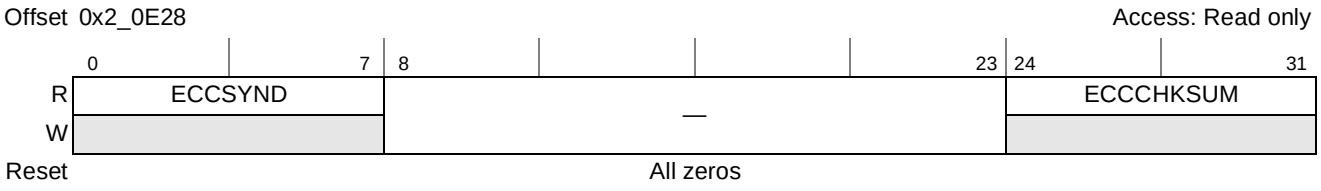


Figure 7-19. L2 Error Syndrome Register (L2CAPTECC)

Table 7-16 describes L2CAPTECC fields.

Table 7-16. L2CAPTECC Field Descriptions

Bits	Name	Description
0–7	ECCSYND	The calculated ECC syndrome of the failing double word
8–23	—	Reserved
24–31	ECCCHKSUM	The data path ECC of the failing double word

Figure 7-20 shows the L2 error detect register (L2ERRDET).

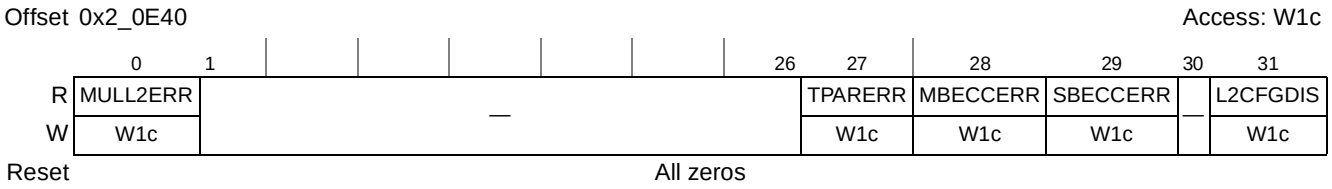


Figure 7-20. L2 Error Detect Register (L2ERRDET)

Table 7-17 describes L2ERRDET fields.

Table 7-17. L2ERRDET Field Descriptions

Bits	Name	Description
0	MULL2ERR	Multiple L2 errors (bit reset, write 1 to clear) 0 Multiple L2 errors of the same type were not detected 1 Multiple L2 errors of the same type were detected
1–26	—	Reserved
27	TPARERR	Tag parity error (bit reset, write 1 to clear) 0 Tag parity error was not detected 1 Tag parity error was detected
28	MBECCERR	Multiple-bit ECC error (bit reset, write 1 to clear) 0 Multiple-bit ECC errors were not detected 1 Multiple-bit ECC errors were detected
29	SBECCERR	Single-bit ECC error (bit reset, write 1 to clear) 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected.
30	—	Reserved
31	L2CFGERR	L2 configuration error (bit reset, write 1 to clear) 0 L2 configuration errors were not detected 1 L2 illegal configuration error detected. Reports inconsistencies between the L2SRAM, L2STASHDIS and L2STASH fields of the L2 Control Register (L2CTL)

Figure 7-21 shows the L2 error disable register (L2ERRDIS).

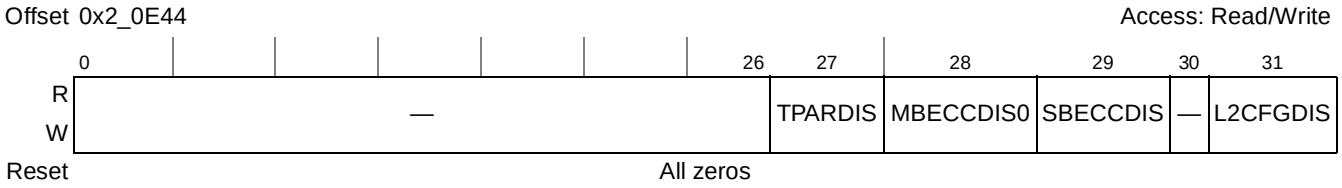


Figure 7-21. L2 Error Disable Register (L2ERRDIS)

Table 7-18 describes L2ERRDIS fields.

Table 7-18. L2ERRDIS Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARDIS	Tag parity error disable 0 Tag parity error detection enabled 1 Tag parity error detection disabled
28	MBECCDIS	Multiple-bit ECC error disable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBECCDIS must be cleared and L2ERRINTEN[MBECCINTEN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 Multiple-bit ECC error detection enabled 1 Multiple-bit ECC error detection disabled
29	SBECCDIS	Single-bit ECC error disable 0 Single-bit ECC error detection enabled 1 Single-bit ECC error detection disabled
30	—	Reserved
31	L2CFGDIS	L2 configuration error disable 0 L2 configuration error detection enabled 1 L2 configuration error detection disabled

Figure 7-22 shows the L2 error interrupt enable register (L2ERRINTEN). When an enabled error condition exists, the L2 signals an interrupt to the core through the internal $\overline{int}$ signal.

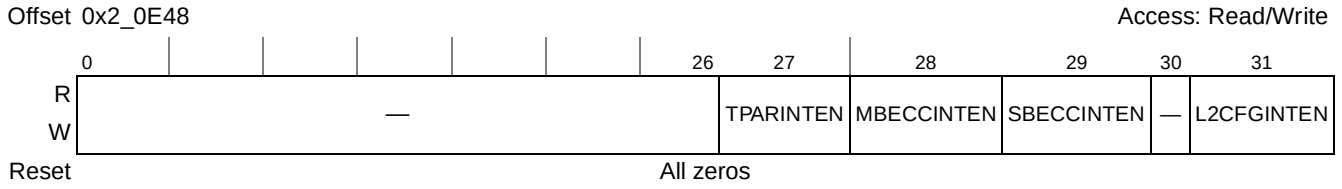


Figure 7-22. L2 Error Interrupt Enable Register (L2ERRINTEN)

Table 7-19 describes L2ERRINTEN fields.

Table 7-19. L2ERRINTEN Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	TPARINTEN	Tag parity error reporting enable 0 Tag parity error reporting disabled 1 Tag parity error reporting enabled
28	MBECCINTEN	Multiple-bit ECC error reporting enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, L2ERRDIS[MBECCDIS] must be cleared and MBECCINTEN must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 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	—	Reserved
31	L2CFGINTEN	L2 configuration error reporting enable 0 L2 configuration error reporting disabled 1 L2 configuration error reporting enabled

Figure 7-23 shows the L2 error attributes capture register (L2ERRATTR).

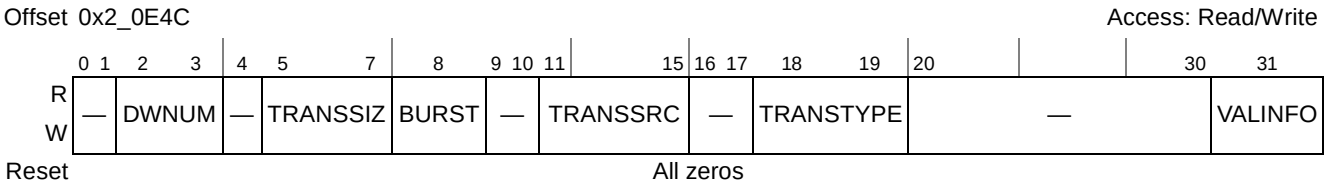


Figure 7-23. L2 Error Attributes Capture Register (L2ERRATTR)

Table 7-20 describes L2ERRATTR fields.

Table 7-20. L2ERRATTR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–3	DWNUM	Double-word number of the detected error (data ECC errors only)
4	—	Reserved
5–7	TRANSSIZ	Transaction size for detected error Single-beat / Burst 000 8 bytes / Reserved 001 1 byte / 16 bytes 010 2 bytes / 32 bytes 011 3 bytes / Reserved 100 4 bytes / Reserved 101 5 bytes / Reserved 110 6 bytes / Reserved 111 7 bytes / Reserved

Table 7-20. L2ERRATTR Field Descriptions (continued)

Bits	Name	Description
8	BURST	Burst transaction for detected error 0 Single-beat (≤ 64 bits) transaction 1 Burst transaction
9–10	—	Reserved
11–15	TRANSSRC	Transaction source for detected error 00000 External (system logic) 10000 Processor (instruction) 10001 Processor (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error 00 Snoop (tag/status read) 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VALINFO	L2 capture registers valid 0 L2 capture registers contain no valid information or no enabled errors were detected. 1 L2 capture registers contain information of the first detected error which 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.

Figure 7-24 shows the L2 error address capture register low (L2ERRADDRL).

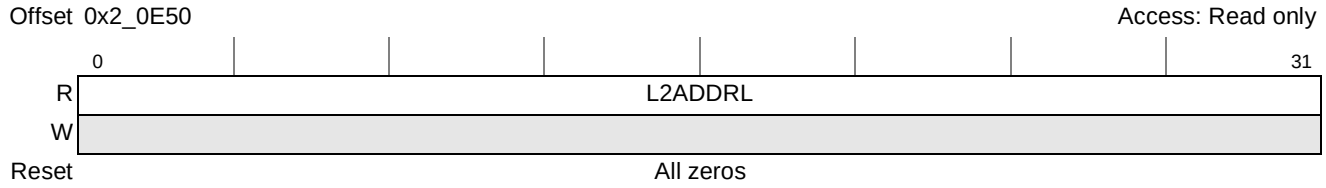


Figure 7-24. L2 Error Address Capture Register (L2ERRADDRL)

Table 7-21 describes L2ERRADDRL[L2ADDRL].

Table 7-21. L2ERRADDRL Field Description

Bits	Name	Description
0–31	L2ADDRL	L2 address bit 4-35 corresponding to detected error

Figure 7-25 shows the L2 error address capture register high (L2ERRADDRH).

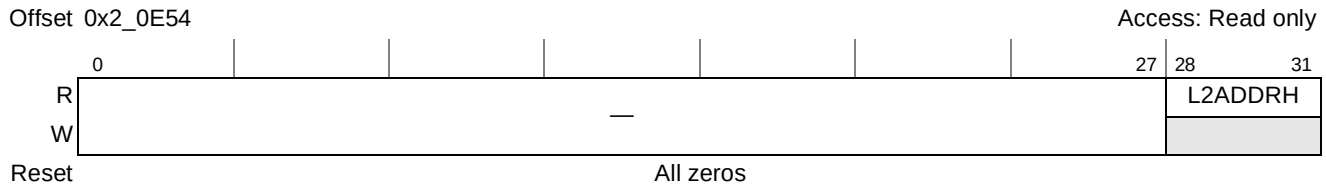


Figure 7-25. L2 Error Address Capture Register (L2ERRADDRH)

Table 7-22 describes L2ERRADDRH[L2ADDRH].

Table 7-22. L2ERRADDRH Field Description

Bits	Name	Description
0–27	—	Reserved
28–31	L2ADDRH	L2 address bits 0–3 corresponding to detected error

Figure 7-26 shows the L2 error control register (L2ERRCTL).

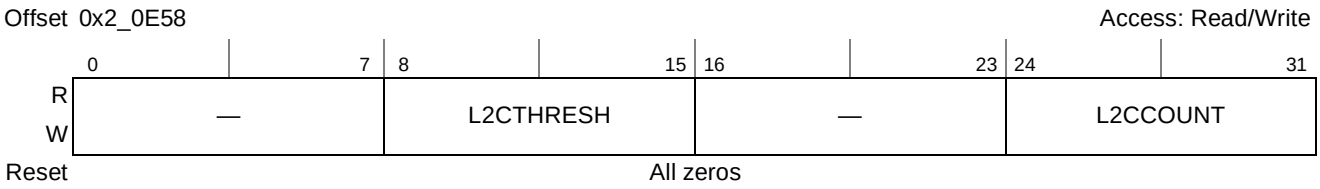


Figure 7-26. L2 Error Control Register (L2ERRCTL)

Table 7-23 describes L2ERRCTL fields.

Table 7-23. L2ERRCTL Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	L2CTHRESH	L2 cache threshold. Threshold value for the number of ECC single-bit errors that are detected before reporting an error condition.
16–23	—	Reserved
24–31	L2CCOUNT	L2 count. Counts ECC single-bit errors detected. If L2CCOUNT equals the ECC single-bit error trigger threshold, an error is reported if single-bit error reporting is enabled.

7.4 External Writes to the L2 Cache (Cache Stashing)

Data from an I/O master can be allocated into the L2 cache while simultaneously being written to memory. External (stashed) writes can be performed from any I/O master. For example:

- Ethernet
- RapidIO
- PCI-Express
- DMA

Stashing is controlled either by an attribute from the initiator of a write or by address range registers in the L2 cache. New cache lines are allocated for full-cache-line writes (unless the line is already resident in the cache). Sub-cache-line write data is stashed only if the line is already valid in the cache. For these sub-cache-line writes, a read-modify-write process is used to merge the write data with the valid data already in the cache.

For information on how to initiate cache stashing from an I/O master, see the respective chapters for the I/O masters that support stashing.

For address range based control of stashing, the L2 cache external write address registers 0–3 (L2CEWAR*n*) are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*) to control the cache stashing functionality. Each register pair (for example, L2CEWAR0 and L2CEWCR0) specifies a programmed memory range that can be allocated and optionally locked with a global write transaction. The address register must be naturally aligned to the window size in the corresponding control register. For more information, see Section 7.3.1.3.1, "L2 Cache External Write Address Registers 0–3 (L2CEWARn)," and Section 7.3.1.3.3, "L2 Cache External Write Control Registers 0–3 (L2CEWCRn)."

Note that stashing can occur regardless of whether the L1 cache is enabled or whether the cache-inhibited bit in the MMU is set for the page.

7.4.1 Stash-Only Cache Regions

In order to prevent stashed I/O data from polluting processor data in the L2 cache (and vice versa), it is possible to create stash-only regions. This is controlled by the L2STASH field of L2CTL. See Section 7.3.1.1, "L2 Control Register (L2CTL)."

If a stash-only region is created, then that region of the cache is only used for stashed I/O data, and stashed I/O data does not cause the eviction of processor data; they are kept in separate ways of each set. The processor may allocate data into the ways of the cache that are not allocated to SRAM or stash-only memory. Replacement within the stash-only region and the processor region is governed by a pseudo-LRU algorithm modified by masks that allow only applicable ways of a cache set to be considered for replacement.

7.5 L2 Cache Timing

Table 7-24 shows the timing of back-to-back loads that miss in the L1 data cache and hit in the L2 cache, assuming the core is running at 2 1/2 times the L2 cache frequency. The L2 returns the 128 bits containing the requested data (critical quad word) first. This data is forwarded to the result register before the full cache line reloads the L1.

Table 7-24. Fastest Read Timing—Hit in L2

Core Clocks	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
e500 core load 1	to D-cache	D-cache miss	to CIU	CIU Q												to CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus						
e500 core load 2		to D-cache	D-cache miss	to CIU	CIU Q																to CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus	
CCB clocks	<1	2			3			4		5			6		7			8		9			10			11>
CCB addr bus load 1		BG			TS					AACK			HIT DATA-COMING													
CCB addr bus load 2								BG		TS					AACK			HIT DATA-COMING								
CCB data bus load 1															DATA			DATA								
CCB data bus load 2																				DATA			DATA			

7.6 L2 Cache and SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term ‘snoop transaction’ refers to transactions initiated by the MPC8572E system logic or by I/O traffic, as opposed to e500 core-initiated transactions.

7.6.1 L2 Cache Coherency Rules

L2 cache coherency rules are as follows:

- The L2 is non-inclusive of the L1—valid L1 lines may be valid or invalid in the L2.
- The L2 cache holds no modified data. Data is in one of four states—invalid, exclusive, exclusive locked, and stale.

- The L2 allocates entries for data cast out or pushed (non-global, non-write-through write with kill) from the L1 caches.
- Lines for e500 core-initiated burst read transactions are allocated as exclusive in the L2.
- The L2 supports I/O devices reading data from valid lines in the L2 cache (data intervention) if L2CTL[L2INTDIS] = 0. An optional unlock attribute causes I/O reads to clear a lock when the read is performed.
- The L2 cache does not respond to cache-inhibited read transactions.
- e500 core-initiated, cache-inhibited store transactions invalidate the line when they hit on a valid L2 line. If the line is locked, it goes to the stale state. For other write transactions the cache-inhibited bit is ignored.
- Non-burst cacheable write transactions from the e500 core (generated by write-through cacheable stores) update a valid L2 cache line through a read-modify-write operation.
- e500 core cast out transactions that hit on a stale line in the L2 cache cause a data update of the line and a change to the valid locked state for that line.
- An e500 core-initiated, cacheable, non-write-through store that misses in L1 and hits on a line in the L2 invalidates that line in the L2. If the line is marked exclusive locked, the L2 marks the line as stale.
- Transactions that hit a stale L2 cache line that would cause an allocate if they miss cause a data update of the line (when data arrives from memory) and a change to the line's valid locked state. Data is not supplied by the L2 cache for the read in this case.
- The following transactions kill the data and the respective locks when they hit a valid L2 line:
  - **dcbf**
  - **dcbi**
- The L2 cache supports mixed cache external writes and core-initiated writes to the same addresses if the core-initiated writes are marked coherency-required, caching allowed, not write-through (WIMG = 001x) and the external writes are marked coherency-required, caching-allowed.
- The L2 cache supports writes to the L2 cache from peripheral devices or from I/O controllers through snoop write transactions with addresses that hit in a programmed memory range. Full cache line (32-byte) write transactions update the data for a valid line in the L2 and if the line is not valid in the L2, a line is allocated. Sub-cache line write transactions update the data only for valid L2 cache lines through read-modify-write operations.
- The L2 cache supports burst writes that lock an L2 cache line from peripheral devices or from I/O controllers through write transactions with addresses that hit in a programmed memory range that has the lock attribute set.
- The L2 cache supports burst writes that allocate and/or lock an L2 cache line from peripheral devices or I/O controllers through a write allocate transaction. See the system logic programming model (for example, that of the DMA controller) for details on how to set the transaction type for cache external writes to the L2.

7.6.2 Memory-Mapped SRAM Coherency Rules

Memory-mapped SRAM coherency rules are as follows:

- External (non–core-initiated) accesses to memory-mapped SRAM must be marked coherency-required. External accesses marked coherency-not-required to memory-mapped SRAM may cause an address unavailable error.
- Accesses to memory-mapped SRAM are cacheable only in the corresponding e500 L1 caches. External accesses must be marked cache-inhibited or be performed with non-caching transactions.

7.7 L2 Cache Locking

The MPC8572E caches can be locked and cleared using the following methods:

- Cache locking methods
  - Individual line locks are set and cleared by using instructions defined by the e500 cache locking APU, which is part of the Freescale Book E Implementation Standards (EIS). These instructions include Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**). For detailed information about these instructions, see the *PowerPC e500 Core Reference Manual*.
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR*n*).
  - The entire cache can be locked by setting a configuration registers appropriately
- Methods for clearing locks
  - Individual locks can be cleared by cache locking APU instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless the entire cache is locked.
  - Flash clearing of all instruction and/or data locks can be done by writes to configuration registers.
  - An unlock attribute can be attached to I/O read operations.

7.7.1 Locking the Entire L2 Cache

The entire L2 cache can be locked by setting L2CTL[L2DO] = 1 and L2CTL[L2IO] = 1. This has the effect of preventing any further allocation of new lines in the cache by core requests. If there are lines in the cache that are not valid, they cannot be used by core requests until the cache is unlocked. While the cache is locked, read requests are serviced as normal, and snooping continues as normal to maintain coherency. Lines invalidated to satisfy coherency requirements cannot be reallocated by core requests while the cache remains locked. The L2 cache can be unlocked by clearing L2CTL[L2IO] and/or L2CTL[L2DO]. Note that L2CTL[L2DO] and L2CTL[L2IO] have no effect on cache external write allocations or memory-mapped SRAM.

Note that this form of cache locking does not use the lock bits of the cache and cannot be cleared by resetting the cache or lock bits.

7.7.2 Locking Programmed Memory Ranges

A programmed memory range can be locked with a snoop write transaction that matches a cache external write address range (specified by L2CEWAR*n* and L2CEWCR*n*). There are no clearing of locks through the programmed address ranges. Locks can be cleared using clear lock instructions, flushes or read-and-clear-lock snoop (RWNITC with clear lock attribute), or flash clear locks.

7.7.3 Locking Selected Lines

Individual lines are locked when the L2 receives one of the following burst transactions:

- **icbtls** (CT = 1)—Instruction Cache Block Touch and Lock Set instruction
- **dcbtls** (CT = 1)—Data Cache Block Touch and Lock Set instruction
- **dcbtstls** (CT = 1)—Data Cache Block Touch for Store and Lock Set instruction
- Snoop burst write—If the address hits on a programmed cache external write space with the lock attribute set, or if the write allocate transaction type is used
- Snoop non-burst write—If the address hits on a programmed cache external write space with the lock attribute set

Note that the core complex broadcasts these instructions to the L2 if the CT field in the instruction specifies the L2 cache (CT = 1). When the L2 cache is specified, data is not placed in the L1, only the L2. If the L1 cache is specified (CT = 0), the L2 does not lock the line, and the data is placed in the L1 (and locked).

When the touch lock set L2 instruction (**dcbtls** or **dcbtstls**) hits are modified in the L1 cache, the modified data is allocated into the L2 cache (and written back to main memory) and a data lock is set. The L1 line state transitions to invalid.

Note that if the L2 receives a request to allocate and lock a line, but all lines in the selected way are locked, the requested L2 line is not allocated and the L2 cache lock overflow bit (L2CTL[L2LO]) is set.

Lines invalidated to satisfy coherency requirements cannot be reallocated while the cache remains locked.

7.7.4 Clearing Locks on Selected Lines

Individual locks in the L2 are cleared by a lock clear (**icblc** or **dcblc,** CT = 1) instruction. This directs the L2 cache to clear a lock on that line if it hits in the L2 cache. Both data and instruction locks are cleared by the **icblc** and **dcblc** instructions.

Note that the lock on a line is cleared if the line is invalidated by a snooped Flush transaction, and the line in the cache is available for allocation of a new line of instruction or data unless the entire cache is locked.

7.7.5 Flash Clearing of Instruction and Data Locks

Locks for instructions and data are recorded separately in the L2 cache, and they can be flash cleared separately by writing the appropriate value to the L2 cache control register (L2CTL[L2LFR] and

L2CTL[L2LFRID]). Flash invalidating of the L2 (setting L2CTL[L2I]) clears all locks on both instructions and data.

Note that flash clearing is the only way to clear data locks without clearing instruction locks, or to clear instruction locks without clearing data locks. All instructions and snoop transactions that clear locks clear both data and instruction locks.

7.7.6 Locks with Stale Data

If data is locked in the L2 and either the e500 core performs a cacheable copyback store or a **dcbtst** misses in the L1, the L2 invalidates the line; however, the L2 clears the valid bit for the data, the lock remains, and the line cannot be victimized. If the e500 core casts out modified data or pushes it in response to a non-flush snoop, the L2 updates the data and sets the valid bit again, maintaining the lock and keeping the data in the cache hierarchy.

7.8 PLRU L2 Replacement Policy

Line replacement is determined using a pseudo least-recently-used (PLRU) algorithm. There is a valid bit (V0–V7) for each line. To determine the replacement victim (the line to be cast out), there are seven PLRU bits (P0–P6) for each set. PLRU bits are updated every time a new line is allocated, and every time an existing line is read by the processor, or updated by a write or invalidated.

Figure 7-27 shows the binary decision tree used to generate the victim line. The eight ways of the L2 cache are labeled W0–W7; the 7 PLRU bits are labeled P0–P6.

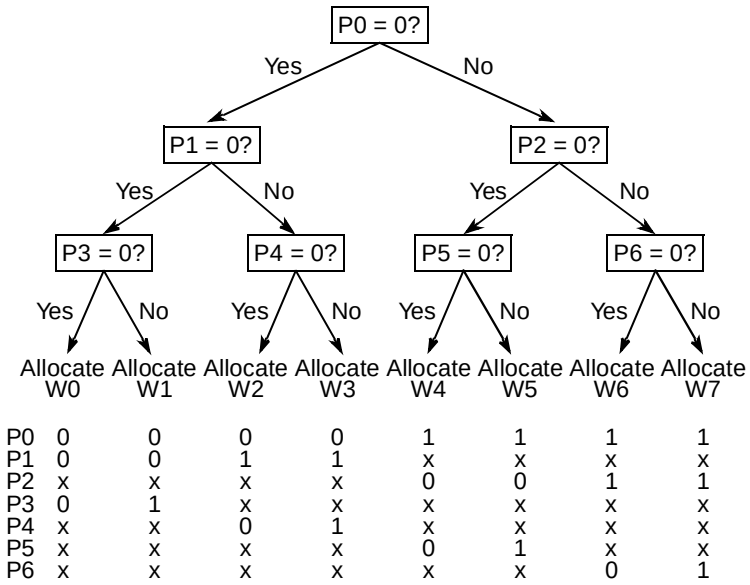


Figure 7-27. L2 Cache Line Replacement Algorithm

7.8.1 PLRU Bit Update Considerations

PLRU bit updates depend on which cache way was last accessed, as summarized in Table 7-25.

Table 7-25. PLRU Bit Update Algorithm

Last Way Accessed	PLRU Bits						
	P0	P1	P2	P3	P4	P5	P6
0	1	1	—	1	—	—	—
1	1	1	—	0	—	—	—
2	1	0	—	—	1	—	—
3	1	0	—	—	0	—	—
4	0	—	1	—	—	1	—
5	0	—	1	—	—	0	—
6	0	—	0	—	—	—	1
7	0	—	0	—	—	—	0

When an L2 line is invalidated, the PLRU bits are updated, marking the corresponding way as least-recently used. This causes the invalidated way to be selected as the next victim.

7.8.2 Allocation of Lines

The general PLRU algorithm described above must be modified to take into account special features of the L2 cache; namely SRAM regions, line locking, and stash-only regions. Each of these features reserves ways within each cache set such that some ways are not eligible for allocation/victimization by the general LRU algorithm.

To preserve the state of the ways that are set aside for other special functions, the PLRU pointers are modified by a mask that is a function of the L2 configuration registers, the lock bits in the cache status array, and initiator of the transaction. The mask effectively points the PLRU algorithm away from ways that are not to be considered for replacement.

L2 cache lines are locked through the status array lock bits. There are two lock bits for each way of each set (1024 sets by eight ways). These bits are set or cleared through special L2 controller commands.There are two sets of lock bits, one for instructions (I0–I7) and one for data (D0–D7) for every line. The lock bits act as a mask over the PLRU bits to determine victim selection. The PLRU bits are updated regardless of line locking

Lock bits are used at allocate time to steer the PLRU algorithm away from selecting locked victims. In the following discussion, the eight lock bits for a particular set are called L0–L7.

```
Where Lock Way i: Li = Di | Ii, i=0…7 (Di = data lock, Ii = instruction lock)
```

An effective value of each PLRU bit is calculated as follows:

```
P0_eff = f(P0,L0,L1,L2,L3,L4,L5,L6,L7) = (L0 & L1 & L2 & L3) | (P0 & ~(L4 & L5 & L6 & L7))
P1_eff = f(P1,L0,L1,L2,L3) = (L0 & L1) | (P1 & ~(L2 & L3))
P2_eff = f(P2,L4,L5,L6,L7) = (L4 & L5) | (P2 & ~(L6 & L7))
```

```
P3_eff = f(P3,L0,L1) = L0 | (P3 & ~L1)
P4_eff = f(P4,L2,L3) = L2 | (P4 & ~L3)
P5_eff = f(P5,L4,L5) = L4 | (P5 & ~L5)
P6_eff = f(P6,L6,L7) = L6 | (P6 & ~L7)
```

These effective PLRU bits are used to select a victim, as indicated in Table 7-26.

Table 7-26. PLRU-Based Victim Selection Mechanism

Way Selected	Effective PLRU State (Binary)	Reduced Logic Equation (Using Effective PLRU Bits)
W0	00x0xxx	~P0 & ~P1 & ~P3
W1	00x1xxx	~P0 & ~P1 & P3
W2	01xx0xx	~P0 & P1 & ~P4
W3	01xx1xx	~P0 & P1 & P4
W4	1x0xx0x	P0 & ~P2 & ~P5
W5	1x0xx1x	P0 & ~P2 & P5
W6	1x1xxx0	P0 & P2 & ~P6
W7	1x1xxx1	P0 & P2 & P6

7.9 L2 Cache Operation

This section describes the behavior of the L1 and L2 cache in response to various operations and in various configurations.

7.9.1 Initialization

7.9.1.1 L2 Cache Initialization

After power-on reset the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a flash invalidate before using the array as an L2 cache. This is done by writing a one to the L2I field of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

7.9.1.2 Memory-Mapped SRAM Initialization

After power-on reset the contents of the data and ECC arrays are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the processor or any other device that uses sub-cache-line transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cache-line writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See Section 7.3.1.5.2, “Error Control and Capture Registers.” If the array is initialized by a DMA engine using cache-line writes, then ECC checking can remain enabled during the initialization process.

7.9.2 Flash Invalidation of the L2 Cache

The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core connect bus while the flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a flash invalidation of the L2 cache regions of the array.

7.9.3 Managing Errors

7.9.3.1 ECC Errors

An individual soft error that causes a single- or multi-bit ECC error can be cleared from the L2 array simply by performing a **dcbf** instruction on the address captured in the L2ERRADDR register. This invalidates the line in the L2 cache. When the load that caused the ECC error is performed again, the data is reallocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be flash invalidated to clear out all single-bit errors.

Note that no data is lost by **dcbf**s or flash invalidates, since the L2 cache is write-through and contains no modified data.

7.9.3.2 Tag Parity Errors

A tag parity error must be fixed by flash invalidating the L2 cache. Note that a **dcbf** operation to the address that caused the error to be reported is not sufficient since a tag parity error is seen as an L2 miss and does not cause invalidation of the bad tag. Proper L2 operation cannot be guaranteed if an L2 tag parity error is not repaired by a flash invalidation of the entire array.

7.9.4 L2 Cache States

The L2 status array uses four bits for each line to determine the status of the line. Different combinations of these bits result in different L2 states. The status bits are as follows:

- Valid (V)
- Instruction locked (IL)
- Data locked (DL)
- Stale (T)

Table 7-27 shows L2 cache states. Note that these conventions are also used in Table 7-28.

Table 7-27. L2 Cache States

V	T	IL	DL	L2 states
0	x	x	x	Invalid (I)
1	0	0	0	Exclusive (E)
1	0	0	1	Exclusive data locked (EDL)
1	0	1	0	Exclusive instruction locked (EIL)
1	0	1	1	Exclusive instruction and data locked (EL)
1	1	0	0	Stale (data invalid, locks invalid) (T)
1	1	0	1	Stale (data invalid, dlock valid) (TDL)
1	1	1	0	Stale (data invalid, ilock valid) (TIL)
1	1	1	1	Stale (data invalid, locks valid) (TL)

7.9.5 L2 State Transitions

Table 7-28 lists state transitions for all e500 core-initiated transactions that change the L2 cache state. Core-initiated transactions caused when the core executes **msync**, **mbar**, **tlbivax**, or **tlbsync** do not change the L2 cache state. The table does not list initial L1 states for transactions that hit in the L1 (iL1 or dL1) and are not sent to the L2.

In the table, the heading ‘L2 hit’ indicates that the L2 provides (on a read) or captures (on a write) data for an existing line. Some entries list two final L1 states. L2 touch instructions never allocate into iL1 or dL1.

Note that if the L2 SRAM is disabled, the L2 initial and final states are always I and the L2 never hits. Similarly, if the L2 SRAM is in full memory-mapped SRAM mode, the L2 initial and final states are always I and the L2 never hits for addresses not in the memory-mapped SRAM address range. The L2 always hits for addresses in the enabled memory-mapped SRAM address ranges.

Table 7-28. State Transitions Due to Core-Initiated Transactions

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable instruction fetch icbtls_L1	iL1 I	I//T	No	I/V	same	L2CTL[L2DO] = 1. L2 touch instructions not allocated in L1
		I	No	I/V	E	L2CTL[L2DO] = 0
icbt_L2	dL1 I,E	E/EL	Yes	I/V	same	—
		T	No	I/V	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus

Table 7-28. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
icbtls_L2	dL1 I,E	I/T	No	I	same	L2CTL[L2DO] = 1
		E	Yes	I	I	L2CTL[L2DO] = 1
		EL	Yes	I	T	L2CTL[L2DO] = 1
		I	No	I	EL	L2CTL[L2DO] = 0
		E	Yes	I	EL	L2CTL[L2DO] = 0
		EL	Yes	I	same	L2CTL[L2DO] = 0
		T	No	I	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited instruction fetch	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cacheable load (4-state) Cacheable **lwarx** (4-state) dcbt_L1 (4-state) dcbtls_L1 (4-state)	dL1 I	I/T	No	E	same	L2CTL[L2IO] = 1
		E	Yes	E	I	L2CTL[L2IO] = 1
		EL	Yes	E	T	L2CTL[L2IO] = 1
		I	No	E	E	L2CTL[L2IO] = 0
		E/EL	Yes	E	same	L2CTL[L2IO] = 0
		T	No	EL	EL	L2CTL[L2IO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited load	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cache-inhibited **lwarx**	N/A	N/A	No	N/A	N/A	No L2 effect
Writeback Store	dL1 I	I/T	No	M	same	L2 allocates when a line is cast out of L1.
		E	Yes	M	I	—
		EL	Yes	M	T	—
Writeback **stwcx**	dL1 I	I/T	No	M	same	—
		E	Yes	M	I	—
		EL	Yes	M	T	—
Cacheable load (3-state) Cacheable **lwarx** (3-state) dcbt_L1 (3-state) dcbtls_L1 (3-state)	dL1 I	I	No	E/I	I	L2CTL[L2IO] = 1
		T	No	E/I	T	L2CTL[L2IO] = 1
		E	Yes	E/I	I	L2CTL[L2IO] = 1
		EL	Yes	E/I	T	L2CTL[L2IO] = 1
		I	No	E/I	E	L2CTL[L2IO] = 0
dcbt_L2 dcbtst_L2	dL1 I,E	E/EL	Yes	E/I	same	L2CTL[L2IO] = 0
		T	No	E/I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
dcbtst_L1 dcbtstls_L1	dL1 I	I/T	No	E	same	—
		E	Yes	E	I	—
		EL	Yes	E	T	—

Table 7-28. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbtls_L2 dcbtstls_L2	dL1 I,E	I	No	I	I	L2CTL[L2IO] = 1
		T	No	I	T	L2CTL[L2IO] = 1
		E	Yes	I	I	L2CTL[L2IO] = 1
		EL	Yes	I	T	L2CTL[L2IO] = 1
		I	No	I	EL	L2CTL[L2IO] = 0
		E/EL	Yes	I	EL	L2CTL[L2IO] = 0
		T	No	I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
Write-through store	dL1 I,E,M	I/T	No	same	I	—
		E/EL	Yes	same	same	Read-modify-write
Cache-inhibited store	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
Cache-inhibited **stwcx**	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
dcblc_L2 icblc_L2	dL1 I,E,M	I/E	No	same	same	—
		EL	No	same	E	—
		T	No	same	I	—
Victim castout dcbt_L2 icbt_L2 dcbtst_L2	dL1 M	I/T	No	I	same	L2CTL[L2IO] = 1. If software sharing cache lines between instructions and data wishes to capture instruction lines in L2 with L2CTL[L2IO] = 1, it must perform **dcbst** to flush the line out of the dL1 before fetching it into L2.
		I	No	I	E	L2CTL[L2IO] = 0
		E/EL	No	I	I/T	L2CTL[L2IO] = 1.
			Yes	I	Same	L2CTL[L2IO] = 0.
		T	Yes	I	EL	L2CTL[L2IO = 0.
dcbtls_L2 icbtls_L2 dcbtstls_L2	dL1 M	I	No	I	EL	An icbtls_L2 that hits modified in L1 cannot be distinguished from dcbtls_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set hillocks in L2, it must perform **dcbst** to flush the line out of the dL1 before locking it in L2.
		E/EL/T	Yes	I	EL	—
Snoop push	dL1 M	I/E	No	I/E	I	—
		EL/T	No	I/E	T	Invalidate data, keep lock
dcbf **dcbst**	dL1 M	I/E/EL	No	I	I	—
dcbz **dcba**	dL1 I	I/E	No	M	I	—
		EL	No	M	T	—

Table 7-28. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbi	dL1 I,E,M	I/ E/EL/T	No	I	I	—
dcbf **dcbst**	dL1 I,E	I/ E/EL/T	No	I	I	—
icbi	iL1 I,V	I/ E/EL/T	No	I	I	—

Table 7-29 lists L2 cache state transitions for all system-initiated (non-core) transactions that change the L2. Table 7-29 accounts for changes caused by L1 snoop pushes triggered by snoops, listed in Table 7-28.

Table 7-29. State Transitions Due to System-Initiated Transactions

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Clean IKill	x	x	0	I/E/EL/T	Same	—
Flush	x	x	0	I/E/EL/T	I	—
Write allocate	x	1	0	I/E/EL/T	EL	Allocate and lock regardless of cache external write (CEW) window
	x	1	0	I/E	E	Allocate regardless of CEW window
				EL/T	EL	—
	x	0	0	I/E	I	No allocate if cache-inhibited
				EL/T	T	Invalidate data, keep lock
WWK 32-byte WWF 32-byte WWF atomic	x	1	0	I/E/EL/T	I	Miss in cache external write windows
				I	E/EL	Hit in cache external write window
				EL	Same	Hit in cache external write window
				E	E/EL	Hit in cache external write window
				T	EL	Hit in cache external write window
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock
< 32-byte WWF < 32-byte WWF atomic	x	1	0	I/E	I	Miss in cache external write windows
				EL/T	T	Miss in cache external write windows.
				I/T	Same	Hit in CEW window but need burst data
				EL	Same	Hit in cache external write window
				E	E/EL	Hit in cache external write window. Set lock if CEW lock attribute set.
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock

Table 7-29. State Transitions Due to System-Initiated Transactions (continued)

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
Read Read atomic	1	1	0	I/T	Same	—
				E	E	—
				EL	EL	—
	x	0	0	N/A	N/A	No L1/L2 effect
RWNITC	1	1	0	I//T	Same	—
				E	E	—
				EL	EL	—
	0	1	0	I	Same	Read-and-clear-lock
				EL	E	—
				T	I	—
	x	0	0	N/A	N/A	No L1/L2 effect
Kill RWITM RWITM atomic RClaim	x	1	0	I/E	I	—
				EL/T	T	Invalidate data, keep lock
	x	0	0	I/E/EL/T	Same	—

7.9.6 Error Checking and Correcting (ECC)

The L2 cache supports error checking and correcting (ECC) for the data path between the core master and system memory. It 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. Multiple-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 double- or multi-bit errors generate an interrupt.

The syndrome encodings for the ECC code are shown in Table 7-30 and Table 7-31.

Table 7-30. L2 Cache 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				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 7-31. L2 Cache 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								•

Part III
Memory, Security, and I/O Interfaces

Part III defines the memory, security and I/O interfaces of the MPC8572E and it describes how these blocks interact with other blocks on the device. The following chapters are included:

- Chapter 8, "e500 Coherency Module," defines the e500 coherency module and how it facilitates communication between the e500 cores, the L2 cache, and the other blocks that comprise the coherent memory domain of the MPC8572E.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500v2 cores in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500v2- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8572E.
- Chapter 9, "DDR Memory Controllers," describes the two DDR2/DDR3 SDRAM memory controllers of the MPC8572E. These fully programmable controllers support most DDR memories available today, including both registered and unbuffered devices. The 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 like ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller (PIC)," describes the embedded multiprocessor programmable interrupt controller (PIC) of the MPC8572E. The PIC is an OpenPIC-compliant interrupt controller that provides multiprocessor interrupt management and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to a CPU for servicing.
- Chapter 11, "Security Engine (SEC) 3.0," describes the security controller of the MPC8572E. The SEC 3.0 off-loads computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor cores of the MPC8572E. It is optimized to process all cryptographic algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, 802.11i, 3G, A5/3 for GSM and EDGE, and GEA3 for GPRS.
- Chapter 12, "I^2C Interfaces," describes the two inter-IC (IIC or I^2C) bus controllers of the MPC8572E. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8572E can be configured to power up in boot sequencer mode which allows the I^2C1 controller to initialize configuration registers.
- Chapter 13, "DUART," describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- Chapter 14, "Enhanced Local Bus Controller," describes the enhanced local bus controller (eLBC) of the MPC8572E. The main component of the enhanced local bus controller is its memory

controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a general-purpose chip-select machine (GPCM), a NAND flash control machine (FCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.

- Chapter 15, "Enhanced Three-Speed Ethernet Controllers," describes the four enhanced three-speed Ethernet controllers (eTSECs) of the MPC8572E. These controllers provide 10/100/1Gb Ethernet support with a complete set of media-independent interface options including MII, RMII, GMII, RGMII, SGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8572E memory coherency module. The controllers provide full-duplex FIFO interface modes, hardware offload acceleration capabilities, and quality of service support. They are backward compatible with PowerQUICC III TSEC controllers.
- Chapter 16, "10/100 Fast Ethernet Controller," describes the fast Ethernet controller (FEC) port on the MPC8572E. This controller provides 10/100 fast Ethernet support suitable for diagnostic and maintenance interface with a multicore system.
- Chapter 17, "Pattern Matcher (PM) 1.1," describes the pattern matching engine (PME) of the MPC8572E. The PME assists in data organization based on regular expression pattern matching or state rule pattern matching. The PME also includes a DEFLATE engine to decompress DEFLATE compress format data, include zlib and gzip.
- Chapter 18, "Table Lookup Unit," describes the two table lookup units (TLUs) of the MPC8572E. The TLUs give access to application-defined routing topology, control, and statistics tables in external memory. The TLUs support several types of table lookup algorithms and provide resources for efficient generation of table entry addresses in memory, hash generation of addresses, and binary table searching algorithms for both exact-match and longest-prefix match strategies.
- Chapter 19, "DMA Controllers," describes both of the four-channel general-purpose DMA controllers of the MPC8572E. The DMA controllers transfer blocks of data independent of the e500v2 cores or external hosts. Data movement occurs among the local address space. Each DMA controller has four high-speed channels. Both the e500 cores and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 20, "Serial RapidIO Interface," describes the Serial RapidIO interface of the MPC8572E. The Serial RapidIO interface conforms to the *RapidIO Interconnect Specification, Revision 1.2*. Both 1x and 4x LP-serial link interfaces are possible, with transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 2.5 Gbps) per lane. Also included is a RapidIO messaging unit (RMU) which manages two inbox/outbox mailboxes for data and one doorbell message structure.The message unit can also multicast a single-segment 156-byte message to up to 32 different destination DevIDs.
- Chapter 21, "PCI Express Interface Controller," describes the three PCI-Express controllers of the MPC8572E. Each controller is compliant with the *PCI Express Base Specification Revision 1.0a*. The physical layer of these controllers operate at 2.5 Gbaud per lane. Configuration options allow multiple width configurations among the three controllers.

- Chapter 22, “General Purpose I/O (GPIO),” describes the general-purpose input and output signals of the MPC8572E.

Chapter 8
e500 Coherency Module

8.1 Introduction

The e500 coherency module (ECM) provides a flexible, easily expandable switching structure for routing e500- and I/O-initiated transactions to target modules on the device. Figure 8-1 shows a high-level block diagram of the ECM.

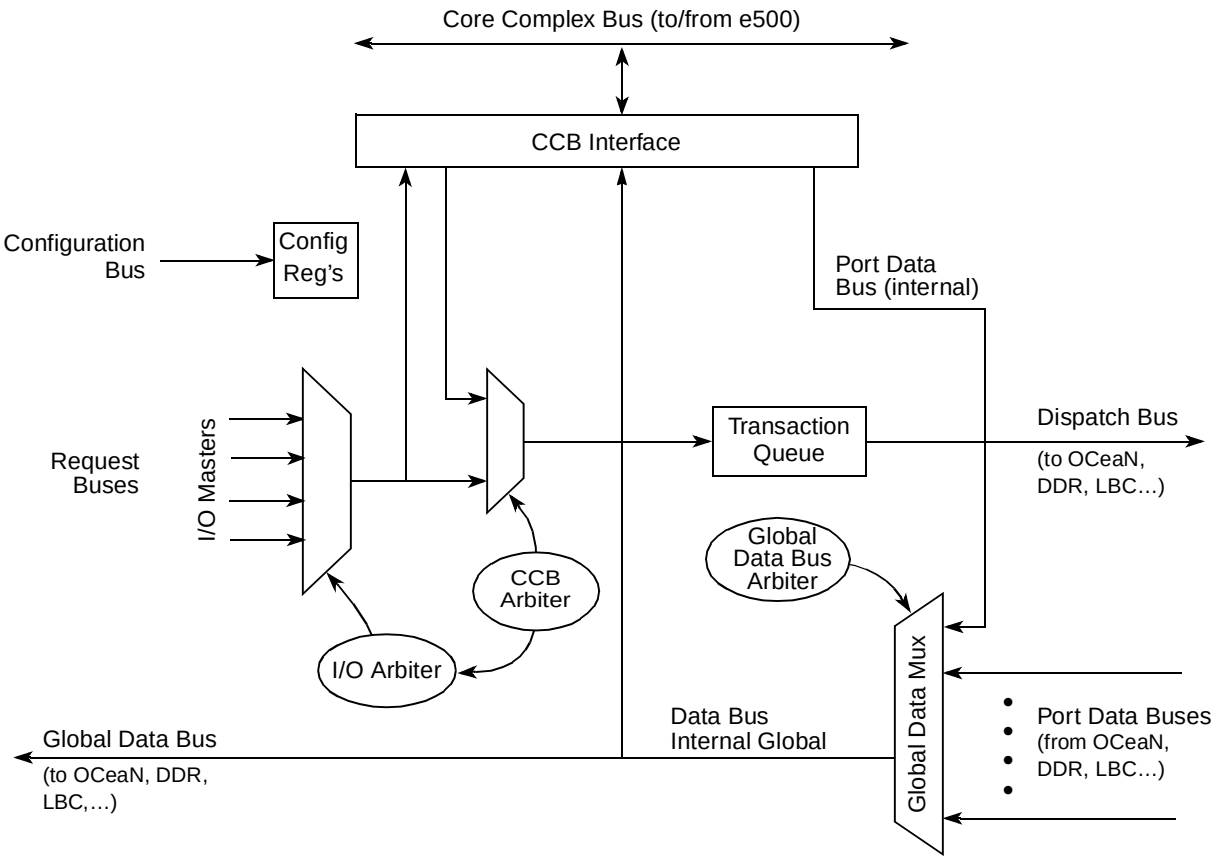


Figure 8-1. e500 Coherency Module Block Diagram

8.1.1 Overview

The ECM routes transactions initiated by the e500 cores to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the ECM forwards I/O-initiated transactions that are tagged with the global attribute onto the core complex bus (CCB). This allows on-chip caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory.

8.1.2 Features

The ECM includes these distinctive features:

- Support for two e500 cores and an L2/SRAM on the CCB, including a CCB arbiter.
- It sources a 64-bit data bus for returning read data from the ECM to the e500 cores and routing write data from the ECM to the L2/SRAM. It sinks a 128-bit data bus for receiving data from the L2/SRAM and two 128-bit write data buses—one from each of the e500 cores.
- Four connection points for I/O initiating (mastering into the device) interfaces. The ECM supports five connection points for I/O targets. The DDR memory controllers, enhanced local bus, OCeaN targets, and configuration register access block all have a target port connection to the ECM.
- Split transaction support—separate address and data tenures allow for pipelining of transactions and out-of-order data tenures between initiators and targets.
- Proper ordering of I/O-initiated transactions.
- Speculative read bus for low-latency dispatch of reads to the DDR controllers.
- Low-latency paths for returning read data from DDR to the e500 cores.
- Error registers trap transactions with invalid addresses. Errors can be programmed to generate interrupts to the e500 core, as described in the following sections:
  - Section 8.2.1.5, “ECM Error Detect Register (EEDR)”
  - Section 8.2.1.6, “ECM Error Enable Register (EEER)”
  - Section 8.2.1.7, “ECM Error Attributes Capture Register (EEATR)”
  - Section 8.2.1.8, “ECM Error Low Address Capture Register (EELADR)”
  - Section 8.2.1.9, “ECM Error High Address Capture Register (EEHADR)”
- Errors from reading I/O devices terminate with data sent to the master with a corrupt attribute. If the master is the e500 core, the ECM asserts *core_fault_in* to the core, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and one of these errors occurs, appropriate interrupts must be enabled to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

8.2 Memory Map/Register Definition

Table 8-1 shows the ECM's memory map. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 8-1. ECM Memory Map

Local Memory Offset	Register	Access	Reset	Section/page
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*n*00_0000	8.2.1.2/8-4
0x0_1BF8	ECM IP Block Revision Register 1	R	0x0001_0000	8.2.1.3/8-5
0x0_1BFC	ECM IP Block Revision Register 2	R	0x0000_0000	8.2.1.4/8-6
0x0_1E00	EEDR—ECM error detect register	w1c	0x0000_0000	8.2.1.5/8-6
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.6/8-7
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.7/8-8
0x0_1E10	EELADR—ECM error low address capture register	R	0x0000_0000	8.2.1.8/8-9
0x0_1E14	EEHADR—ECM error high address capture register	R	0x0000_0000	8.2.1.9/8-9

8.2.1 Register Descriptions

This section consists of detailed descriptions of those registers summarized in Table 8-1. Note that these registers are shown in big-endian format.

8.2.1.1 ECM CCB Address Configuration Register (EEBACR)

The ECM CCB address configuration register, shown in Figure 8-2, controls arbitration and streaming policies for the CCB.

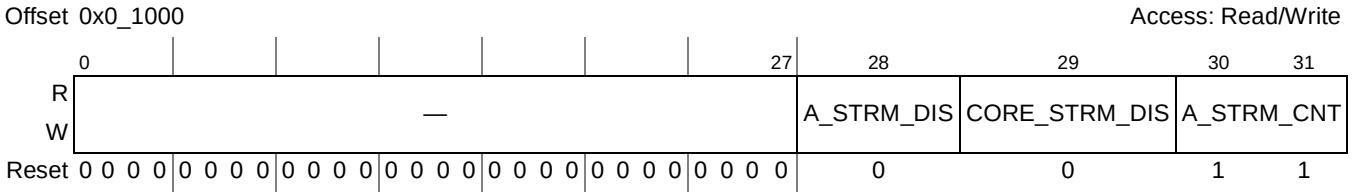


Figure 8-2. ECM CCB Address Configuration Register (EEBACR)

Table 8-2 describes the EEBACR fields.

Table 8-2. EEBACR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	A_STRM_DIS	Controls whether the ECM allows any streaming to occur. 0 Streaming is enabled. 1 Streaming is disabled.
29	CORE_STRM_DIS	With A_STRM_DIS, controls whether the e500 cores can stream commands onto the CCB. A_STRM_DIS and CORE_STRM_DIS must both be cleared for the e500 cores to be enabled to stream their address tenures that they master. 0 Stream address tenures initiated by the e500 cores, provided A_STRM_DIS is cleared. 1 Streaming of address tenures initiated by the e500 cores not allowed.
30–31	A_STRM_CNT	Stream count. Specifies the maximum number of transactions that any master can stream (issue sequentially without preemption) on the CCB following an initial transaction. 00 Reserved 01 One transaction can be streamed with the initial transaction. 10 Two transactions can be streamed with the initial transaction. 11 Three transactions can be streamed with the initial transaction. Default.

8.2.1.2 ECM CCB Port Configuration Register (EEBPCR)

The ECM CCB port configuration register (EEBPCR) is shown in Figure 8-3.

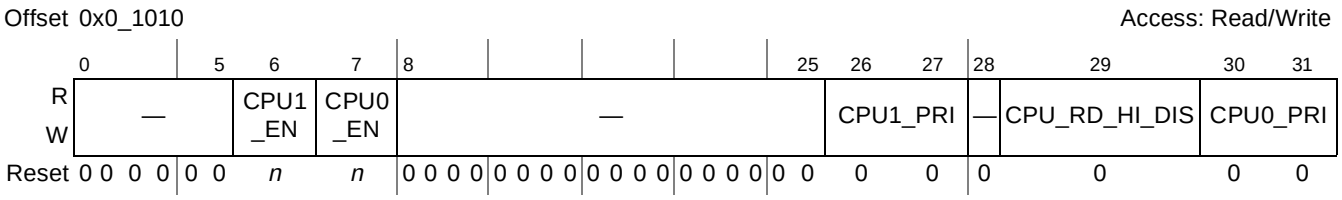


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

Table 8-3 describes EEBPCR fields.

Table 8-3. EEBPCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	CPU1_EN	CPU1 port enable. Controls boot holdoff mode when the device is an agent of an external host or when the other core is used to perform initialization prior to allowing this core to boot. Specifies whether the e500 core (CPU) port is enabled to run transactions on the CCB. The CPU boot configuration power-on reset signal (cfg_cpu1_boot) determines the initial value of this bit. If the signal is sampled as a logic 1 at the negation of reset, CPU1 is enabled to boot at the end of the POR sequence. Otherwise, CPU1 cannot fetch its boot vector until an external host or another core sets the CPU1_EN bit. 0 Boot holdoff mode. CPU1 arbitration is disabled on the CCB and no bus grants are issued. 1 CPU1 is enabled and receives bus grants in response to bus requests for the boot vector. After this bit is set, it should not be cleared by software. It is not intended to dynamically enable and disable CPU operation. It is only intended to end boot holdoff mode. See Section 4.4.3.7, “CPU Boot Configuration,” for more information.

Table 8-3. EEBPCR Field Descriptions (continued)

Bits	Name	Description
7	CPU0_EN	CPU0 port enable. Controls boot holdoff mode when the device is an agent of an external host or when the other core is used to perform initialization prior to allowing this core to boot. Specifies whether the e500 core (CPU) port0 is enabled to run transactions on the CCB. The CPU boot configuration power-on reset pin (cfg_cpu0_boot) determines the initial value of this bit. If the pin is sampled as a logic 1 at the negation of reset, CPU0 is enabled to boot at the end of the POR sequence. Otherwise, CPU0 cannot fetch its boot vector until an external host or another core sets the CPU0_EN bit. 0 Boot holdoff mode. CPU0 arbitration is disabled on the CCB and no bus grants are issued. 1 CPU0 is enabled and receives bus grants in response to bus requests for the boot vector. After this bit is set, it should not be cleared by software. It is not intended to dynamically enable and disable CPU operation. It is only intended to end boot holdoff mode. See Section 4.4.3.7, "CPU Boot Configuration," for more information.
8–25	—	Reserved
26–27	CPU1_PRI	Identifies the priority level of the e500 core 1 (CPU) port. This priority level is used to determine whether a particular port's bus request can cause the CCB Arbiter to terminate another port's streaming of address tenures. 00 Lowest priority level 01 Second lowest priority level 10 Highest priority level 11 Reserved
28	—	Reserved
29	CPU_RD_HI_DIS	Identifies which read queue of DDR targets is assigned to the e500 core (CPU) ports' read transactions (in understressed system). 0 Read high queue (higher bandwidth DDR queue) is assigned for the e500 cores' read transactions 1 Read low queue (lower bandwidth DDR queue) is assigned for the e500 cores' read transactions
30–31	CPU0_PRI	Specifies the priority level of the e500 core 0 (CPU) port. This priority level is used to determine whether a particular port's bus request can cause the CCB arbiter to terminate another port's streaming of address tenures. 00 Lowest priority level 01 Second lowest priority level 10 Highest priority level 11 Reserved

8.2.1.3 ECM IP Block Revision Register 1 (EIPBRR1)

The ECM IP block revision register 1 is shown in Figure 8-4.

Offset 0x0_1BF8 Access: Read only

	0															15	16							23	24							31
R	IP_ID																IP_MJ								IP_MN								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-4. ECM IP Block Revision Register 1 (EIPBRR1)

Table 8-4 describes EIPBRR1 fields.

Table 8-4. EIPBRR1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP block ID
16–23	IP_MJ	Major revision
24–31	IP_MN	Minor revision

8.2.1.4 ECM IP Block Revision Register 2 (EIPBRR2)

The ECM IP block revision register 2 is shown in Figure 8-5.

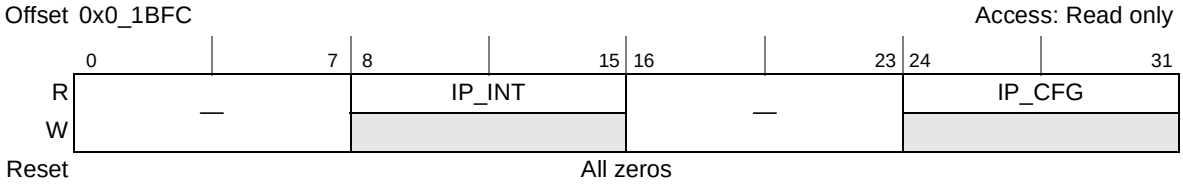


Figure 8-5. ECM IP Block Revision Register 2 (EIPBRR2)

Table 8-5 describes EIPBRR2 fields.

Table 8-5. EIPBRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

8.2.1.5 ECM Error Detect Register (EEDR)

The ECM error detect register (EEDR) is shown in Figure 8-6.

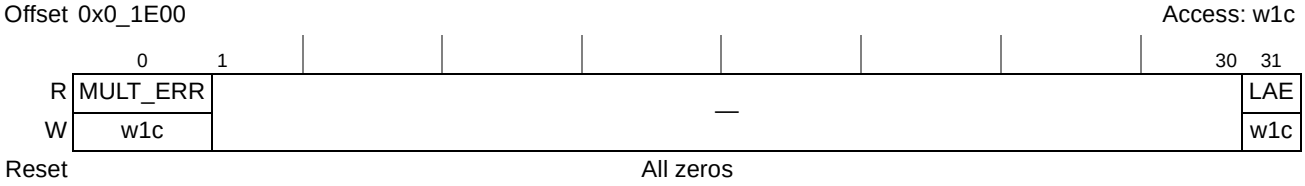


Figure 8-6. ECM Error Detect Register (EEDR)

Table 8-6 describes EEDR fields.

Table 8-6. EEDR Field Descriptions

Bits	Name	Description
0	MULT_ERR	Multiple error. Indicates the occurrence of multiple errors of the same type. Write 1 to clear. 0 Multiple errors of the same type were not detected. 1 Multiple errors of the same type were detected.
1–30	—	Reserved
31	LAE	Local access error. Write 1 to clear. Two cases can generate LAEs, as follows: • Transaction does not map to any target. In this case the ECM injects read responses (with the corrupt attribute set) and write data is dropped. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, EEER[LAEE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." • Source and target IDs indicate that an OCN port initiated a transaction that targets an OCN port. This loopback behavior can result from programming errors where inbound ATMU window targets are inconsistent with targets configured in the local access windows for a given address range. For this type of LAE, the dispatch (to OCN target in this case) is not screened off; the LAE error is reported, but the transaction is still sent to its OCN target. 0 Local access error has not occurred. 1 Local access error occurred.

8.2.1.6 ECM Error Enable Register (EEER)

The ECM error enable register (EEER) shown in Figure 8-7 enables the reporting of error conditions to the e500 core through the internal $\overline{int}$ interrupt signal.

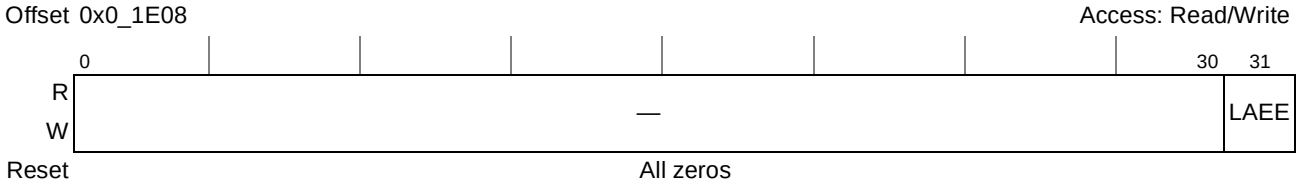


Figure 8-7. ECM Error Enable Register (EEER)

Table 8-7 describes EEER fields.

Table 8-7. EEER Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	LAEE	Local access error enable. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If HID1[RFXE] is zero and this error occurs, LAEE must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Disable reporting local access errors as interrupts. 1 Enable reporting local access errors as interrupts.

8.2.1.7 ECM Error Attributes Capture Register (EEATR)

The ECM error attributes capture register (EEATR) is shown in Figure 8-8.

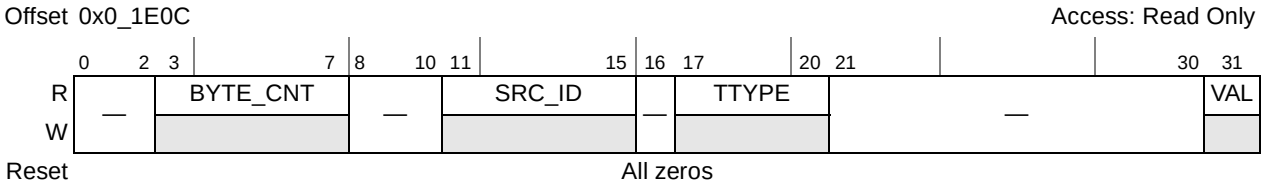


Figure 8-8. ECM Error Attributes Capture Register (EEATR)

Table 8-8 describes EEATR fields.

Table 8-8. EEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	BYTE_CNT	Byte count. Specifies the transaction byte count. 00000 32 bytes 00001 1 byte 00010 2 bytes 00100 4 bytes 01000 8 bytes 10000 16 bytes
8–10	—	Reserved
11–15	SRC_ID	Source ID. Specifies the source device mastering the transaction. 00000 PCI Express 3 00001 PCI Express 2 00010 PCI Express 1 00011 Reserved 00100 Pattern Match Engine (PME) 00101–00110Reserved 00111 Security 01000 FEC 01001 Reserved 01010 Boot sequencer 01011 Reserved 01100 Serial RapidIO 01101 Reserved 01110 TLU 1 01111 TLU 2 10000 Processor 0 (instruction) 10001 Processor 0 (data) 10010 Processor 1 (instruction) 10011 Processor 1 (data) 10100 Reserved 10101 DMA 1 10110 DMA 2 10111 Reserved 11000 eTSEC 1 11001 eTSEC 2 11010 eTSEC 3 11011 eTSEC 4 11100 RapidIO message unit 11101 RapidIO doorbell unit 11110 RapidIO port-write unit 11111 Reserved
16	—	Reserved
17–20	TTYPE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction 0101–0111Reserved 1000 Read 1001 Read with unlock 101x Reserved 1100 Read with clear atomic 1101 Read with set atomic 1110 Read with decrement atomic 1111 Read with increment atomic

Table 8-8. EEATR Field Descriptions (continued)

Bits	Name	Description
21–30	—	Reserved
31	VAL	Register data valid. 0 ECM error attribute capture register does not contain valid information. 1 ECM error attribute capture register contains valid information.

8.2.1.8 ECM Error Low Address Capture Register (EELADR)

The ECM error low address capture register (EELADR) is shown in Figure 8-9.

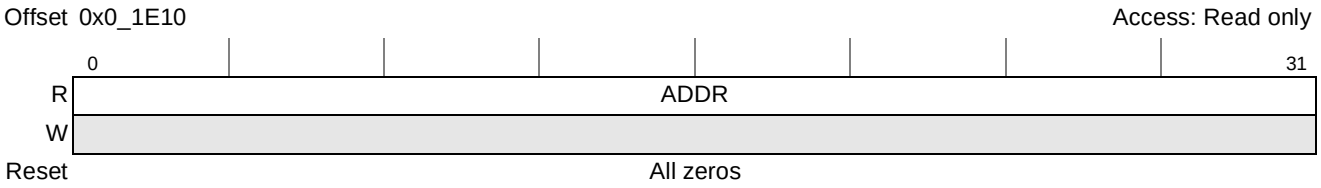


Figure 8-9. ECM Error Low Address Capture Register (EELADR)

Table 8-9 describes EELADR fields.

Table 8-9. EELADR Field Descriptions

Bits	Name	Description
0–31	ADDR	Address. Specifies the lower-order 32 bits of the 36-bit address of the transaction. Qualified by EEATR[VAL].

8.2.1.9 ECM Error High Address Capture Register (EEHADR)

The ECM error high address capture register (EEHADR) is shown in Figure 8-10.

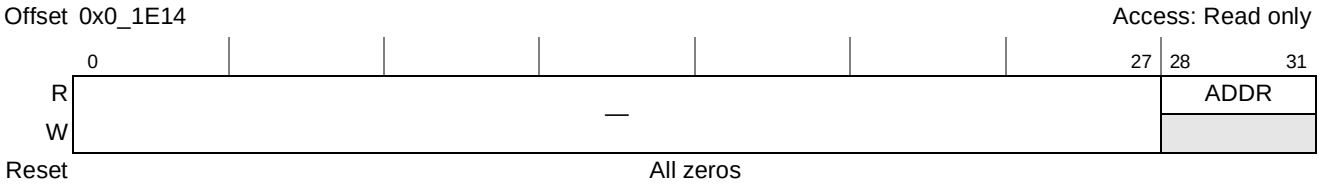


Figure 8-10. ECM Error High Address Capture Register (EEHADR)

Table 8-10 describes EEHADR fields.

Table 8-10. EEHADR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ADDR	Address. Specifies the high-order 4 bits of the 36-bit address of the transaction. Qualified by EEATR[VAL].

8.3 Functional Description

The following is a very general discussion of ECM operation.

8.3.1 I/O Arbiter

Figure 8-1 shows the I/O arbiter block that manages I/O-initiated address tenure requests arriving on the request buses. Four request buses compete for access to the ECM, which can only process one request at a time. The ECM uses two factors to select the winning request bus: the primary factor is requested bandwidth and the secondary factor is longest waiting/least recently granted status. By default all requesters start requesting low levels of bandwidth. A starvation avoidance algorithm ensures that low bandwidth requesters make forward progress in the presence of high bandwidth requesters.The transaction from the winning request bus competes with e500 core requests for the CCB and entry into the transaction queue.

8.3.2 CCB Arbiter

Figure 8-1 shows the CCB arbiter block coordinating the entry of new transactions into the ECM's transaction queue. It handles arbitration for requests to use the CCB from the e500 cores and the winning request bus and consequently controls when these new transactions can enter the transaction queue.

Because the CCB bus operates most efficiently when it streams commands from one initiator, the CCB arbiter alternates grants between streams of transactions from the e500 cores and from the winner of the I/O arbiter. The length of a stream (number of back-to-back transactions) is limited by the A_STRM_CNT field in the EEBACR register. However, the arbiter also uses the priority of the requests to limit streaming. If the priority of a new request is higher than that of a stream in progress, then the higher priority transaction interrupts the other stream. The priority of e500 transactions is set by the CPU[0/1]_PRI field in EEBPCR register. Depending how the CPU_RD_HI_DIS field in EEBPCR register is set, read transactions from the e500 cores are initially assigned to either the higher- or lower-bandwidth queue of the DDR target.

8.3.3 Transaction Queue

The ECM's transaction queue performs four basic functions: arbitration across the e500 cores and I/O masters, target mapping and dispatching, enforcement of ordering, and enforcement of coherency. The address of each transaction is compared against each local access window, and the transaction is then routed to the appropriate target interface associated with the local access window that the address hits within. Even though the CCB and ECM allow the pipelining of transactions, the address tenures of all transactions issued from I/O masters (masters other than the e500 cores) may still be ordered. For those transactions accessing address space marked as snoopable, or space that may be cached by the e500 cores, the ECM enforces coherency, snooping those transactions on the CCB, and taking castouts from the e500 cores as is necessary.

8.3.4 Global Data Multiplexor

Figure 8-1 shows how the global data multiplexor takes data bus connections and multiplexes them onto one 128-bit global data bus. The global data mux allows initiators of write transactions to route data to their targets and read targets to return data to the initiators.

8.3.5 CCB Interface

Figure 8-1 shows the CCB interface for both CCB address and data tenures. This interface formats CCB address tenures for the ECM transaction queue. It also contains the queueing and buffering needed to manage outstanding CCB data tenures. The buffers receive e500 core-initiated write and I/O-initiated read data (that hit in the L2/SRAM module) from the e500 write (128-bit wide) and read (128-bit wide) data buses and route them through the global data mux to the global data bus. The buffers also receive e500 core-initiated read and I/O-initiated write data (that hit in the L2/SRAM module) from the global data bus and forward them onto the CCB data bus (64 bits).

8.4 Initialization/Application Information

If either e500 core is used to initialize the device, the applicable CPU boot configuration power-on reset pin should be pulled high to initially set EEBPCR[CPU0_EN] or EEBPCR[CPU1_EN]. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on power-up reset initialization.

If any device other than the e500 cores (such as PCI Express) is used to initialize the device, the CPU boot configuration power-on reset pins should be pulled low to initially clear EEBPCR[CPU0_EN] and EEBPCR[CPU1_EN]. This prevents the e500 cores from accessing any configuration registers or local memory space during initialization. However, in any such system, one step near the end of the initialization routine must set EEBPCR[CPU0_EN] and EEBPCR[CPU1_EN] to re-enable the e500 cores. Note that for basic functionality, EEBPCR[CPU0/1_EN] is the only field that must be written (provided a device other than the e500 cores is used to initialize the device) in the ECM.

EEBPCR[CPU0_PRI] and EEBPCR[CPU1_PRI] specify the priority level associated with all e500 core initiated transactions. These values allow users running time-critical applications to adjust the average response latency of transactions initiated by the cores compared to those initiated by I/O masters. These priority levels affect whether e500 core requests can interrupt the streaming of address tenures initiated by (the ECM on behalf of) I/O masters or the other e500 core. Only transactions with a priority greater than the current CCB transaction can interrupt streaming. The higher the core’s priority, the lower the average latency needed for it to obtain bus grants from the ECM, because it can interrupt lower priority streaming. The default value of zero gives all core-initiated transactions the lowest priority, which prevents the cores from interrupting I/O master transaction streams.

EEBACR[A_STRM_CNT] allows users to balance response latency with throughput and should prove useful in tuning systems with multiple time-critical tasks. The default value of 0b11 causes the ECM to attempt to stream as many as four transactions initiated from the same CCB master. Increasing this value increases the maximum number of transactions that may be streamed together from any one CCB master. Raising this value can increase throughput for high priority transactions, but may increase latency for lower priority transactions from another CCB master. Note that the e500 cores must also have streaming enabled (through HID1[ASTME] in each core) for the CCB to stream.

Chapter 9
DDR Memory Controllers

9.1 Introduction

The two fully programmable DDR SDRAM controllers support most JEDEC standard x8, x16, or x32 DDR2 and DDR3 memories available. 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.

Figure 9-1 is a high-level block diagram of the DDR memory controller with its associated interfaces. Section 9.5, “Functional Description,” contains detailed figures of the controller.

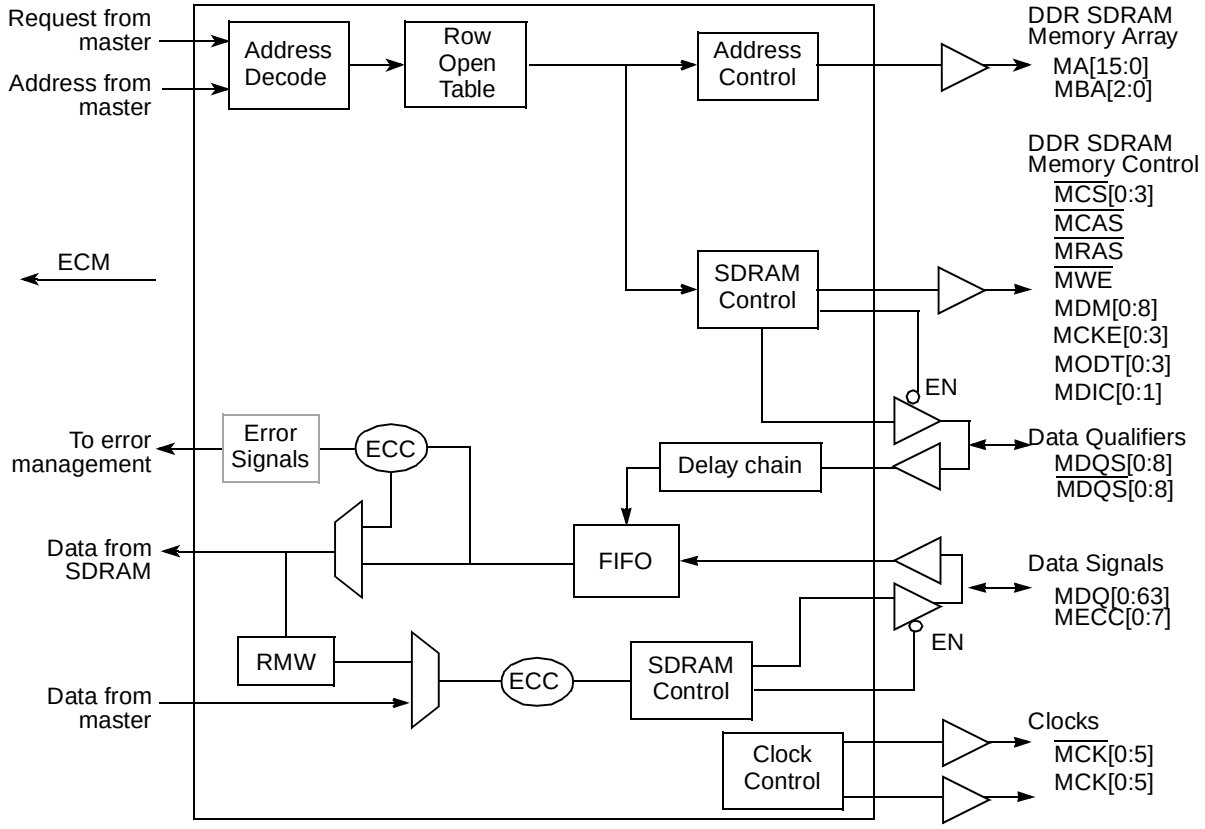


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR2 and DDR3 SDRAM
- Dual independent controllers
- 64-/72-bit SDRAM data bus. 32-/40-bit SDRAM 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 4-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
- Write leveling supported for DDR3 memories
- Support for up to eight posted refreshes
- Memory controller clock frequency of two or four times the SDRAM clock with support for sleep power management
- Support for error injection

9.2.1 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].

9.3 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.

9.3.1 Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 9-1 shows how DDR memory controller external signals are grouped. The device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 9-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
DDRC1 Memory Controller				
$\overline{\text{D1\_MAPAR\_ERR}}$	DDRC1 Address Parity Error	One	1	I
D1_MAPAR_OUT	DDRC1 Address Parity Out	Zero	1	O
D1_MDQ[0:63]	DDRC1 Data bus	All zeros	64	I/O
D1_MDQS[0:8]	DDRC1 Data strobes	All zeros	9	I/O
$\overline{\text{D1\_MDQS}}$[0:8]	DDRC1 Complement data strobes	All ones	9	I/O
D1_MECC[0:7]	DDRC1 Error checking and correcting	All zeros	8	I/O
$\overline{\text{D1\_MCAS}}$	DDRC1 Column address strobe	One	1	O
D1_MA[15:0]	DDRC1 Address bus	All zeros	16	O
D1_MBA[2:0]	DDRC1 Logical bank address	All zeros	3	O
$\overline{\text{D1\_MCS}}$[0:3]	DDRC1 Chip selects	All ones	4	O
$\overline{\text{D1\_MWE}}$	DDRC1 Write enable	One	1	O
$\overline{\text{D1\_MRAS}}$	DDRC1 Row address strobe	One	1	O
D1_MDM[0:8]	DDRC1 Data mask	All zeros	9	O
D1_MCK[0:5]	DDRC1 DRAM clock outputs	All zeros	6	O
$\overline{\text{D1\_MCK}}$[0:5]	DDRC1 DRAM clock outputs (complement)	All zeros	6	O
D1_MCKE[0:3]	DDRC1 DRAM clock enable	All zeros	4	O
D1_MODT[0:3]	DDRC1 DRAM on-die termination	All zeros	4	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	O
D1_MDIC[0:1]	DDRC1 Driver impedance calibration		2	I/O
DDRC2 Memory Controller				
D2_MDQ[0:63]	DDRC2 Data bus	All zeros	64	I/O
D2_MDQS[0:8]	DDRC2 Data strobes	All zeros	9	I/O
$\overline{\text{D2\_MDQS}}$[0:8]	DDRC2 Complement data strobes	All ones	9	I/O
D2_MECC[0:7]	DDRC2 Error checking and correcting	All zeros	8	I/O
$\overline{\text{D2\_MCAS}}$	DDRC2 Column address strobe	One	1	O

Table 9-1. DDR Memory Interface Signal Summary (continued)

Name	Function/Description	Reset	Pins	I/O
D2_MA[15:0]	DDRC2 Address bus	All zeros	16	O
D2_MBA[2:0]	DDRC2 Logical bank address	All zeros	3	O
$\overline{\text{D2\_MCS}}$[0:3]	DDRC2 Chip select	All ones	4	O
$\overline{\text{D2\_MWE}}$	DDRC2 Write enable	One	1	O
$\overline{\text{D2\_MRAS}}$	DDRC2 Row address strobe	One	1	O
D2_MDM[0:8]	DDRC2 Data mask	All zeros	9	O
D2_MCK[0:5]	DDRC2 DRAM clock outputs	All zeros	6	O
$\overline{\text{D2\_MCK}}$[0:5]	DDRC2 DRAM clock outputs (complement)	All ones	6	O
D2_MCKE[0:3]	DDRC2 DRAM clock enable	All zeros	4	O
D2_MODT[0:3]	DDRC2 DRAM on-die termination	All zeros	4	O
D2_MDIC[0:1]	DDRC2 Driver impedance calibration	b10	2	I/O
$\overline{\text{D2\_MAPAR\_ERR}}$	DDRC2 Address Parity Error	One	1	I
D2_MAPAR_OUT	DDRC2 Address Parity Out	Zero	1	O

Note that some devices implementing two DDR controllers may share one set of MDVAL and MSRCID[0:4] signals between them. Please refer to the signals and debug chapters for clarification on implementation.

Table 9-2 shows the memory address signal mappings.

Table 9-2. 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)[1]
	MA9	A9
	MA8	A8 (alternate AP for DDR)[2]
	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

1 Auto-precharge for DDR signaled on A10 when DDR_SDRAM_CFG[PCHB8] = 0

2 Auto-precharge for DDR signaled on A8 when DDR_SDRAM_CFG[PCHB8] = 1

9.3.2 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 D*n*_ in front of the signal names. When a DDR signal is

discussed specifically elsewhere in this book it is referred to without the D*n*_ prefix. For example, the D1_MDQ[0] and D2_MDQ[0] signals are both referred to as just MDQ[0].

9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
D*n*_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.
D*n*_MDQS[0:8]/ D*n*_$\overline{\text{MDQS}}$[0:8]	I/O	Data strobes. Inputs with read data, outputs with write data.	
	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 9-47 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 9-47 for byte lane assignments.
		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.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D*n*_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. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See Section 9.5.11, "Error Checking and Correcting (ECC)," and Section 25.4.2.2, "Debug Information on ECC Pins," 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
D*n*_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 9-51 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 $\overline{\text{MCS}}n$ is active). High impedance—When the memory controller is disabled
D*n*_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 9-51 describes the mapping of these signals in all cases.
		Timing	Assertion/Negation—Same timing as MA*n* High impedance—Same timing as MA*n*

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
D*n*_$\overline{\text{MCAS}}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{\text{MCAS}}$ 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 9-57 for more information on the states required on $\overline{\text{MCAS}}$ 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 Section 9.4.1.5, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 9.4.1.6, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 9.4.1.7, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 9.4.1.4, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is disabled.
D*n*_$\overline{\text{MRAS}}$	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 9-57 for more information on the states required on $\overline{\text{MRAS}}$ 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 Section 9.4.1.5, "DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)," Section 9.4.1.6, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)," Section 9.4.1.7, "DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)," and Section 9.4.1.4, "DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)." High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is disabled.
D*n*_$\overline{\text{MCS}}$[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 Section 9.4.1.1, "Chip Select Memory Bounds (CSn_BNDS)," and Section 9.4.1.2, "Chip Select Configuration (CSn_CONFIG)." The DDR controller asserts one of the $\overline{\text{MCS}}$[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.
D*n*_$\overline{\text{MWE}}$	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 9-57 for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands. Negated—Indicates a memory read operation.
		Timing	Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands. High impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is disabled.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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 9-47 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 18.2-Ω precision 1% resistors; MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV_{DD}. SeeSection 9.4.1.25, “DDR Control Driver Register 2 (DDRCDR_2),” for more information on these signals.	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These are driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.
D*n*_$\overline{\text{MAPAR_ERR}}$	I	Address parity error. Reflects whether an address parity error has been detected by the DRAM. This signal is active low.	
		State Meaning	Asserted—An error has been detected. Negated—An error has not been detected.
		Timing	Assertion/Negation—are driven by the registered DIMMs one DRAM cycle after the parity bit has been driven by the memory controller. This error signal should be held valid for two DRAM cycles.
D*n*_MAPAR_OUT	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[0:3], MODT[0:3], or $\overline{\text{MCS}}$[0:3] signals.	
		State Meaning	Asserted—The parity bit is high. Negated—The parity bit is low.
		Timing	Assertion/Negation—are issued one DRAM cycle after the chip select for each command.

9.3.2.2 Clock Interface Signals

Table 9-4 contains the detailed descriptions of the clock signals of the DDR controller.

Table 9-4. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:5], $\overline{\text{MCK}}$[0:5]	O	DRAM clock outputs and their complements. See Section 9.5.4.1, "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_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 $\overline{\text{MCS}}$ and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as $\overline{\text{MCS}}$[0] 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 $\overline{\text{MCK}}$. MCK/$\overline{\text{MCK}}$ 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. Are asserted again when exiting dynamic power management or self refresh. High impedance—Always driven.

9.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in Section 25.4.2, "DDR SDRAM Interface Debug."

9.4 Memory Map/Register Definition

Table 9-5 shows the register memory map for the DDR memory controllers. Note that while the table lists only the DDRC1 registers, the offsets are defined for both DDRC1 and DDRC2. Memory-mapped registers for DDRC1 begin at offset 0x0_2000 and the memory-mapped registers for DDRC2 begin at offset 0x0_6000.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 9-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
DDRC1 Registers				
0x000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-13
0x008	CS1_BNDS—Chip select 1 memory bounds	R/W	0x0000_0000	9.4.1.1/9-13
0x010	CS2_BNDS—Chip select 2 memory bounds	R/W	0x0000_0000	9.4.1.1/9-13
0x018	CS3_BNDS—Chip select 3 memory bounds	R/W	0x0000_0000	9.4.1.1/9-13
0x080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-14
0x084	CS1_CONFIG—Chip select 1 configuration	R/W	0x0000_0000	9.4.1.2/9-14
0x088	CS2_CONFIG—Chip select 2 configuration	R/W	0x0000_0000	9.4.1.2/9-14
0x08C	CS3_CONFIG—Chip select 3 configuration	R/W	0x0000_0000	9.4.1.2/9-14
0x0C0	CS0_CONFIG_2—Chip select 0 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0C4	CS1_CONFIG_2—Chip select 1 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0C8	CS2_CONFIG_2—Chip select 2 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x0CC	CS3_CONFIG_2—Chip select 3 configuration 2	R/W	0x0000_0000	9.4.1.3/9-17
0x100	TIMING_CFG_3—DDR SDRAM timing configuration 3	R/W	0x0000_0000	9.4.1.4/9-17
0x104	TIMING_CFG_0—DDR SDRAM timing configuration 0	R/W	0x0011_0105	9.4.1.5/9-19
0x108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.6/9-21
0x10C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.7/9-23
0x110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.8/9-25
0x114	DDR_SDRAM_CFG_2—DDR SDRAM control configuration 2	R/W	0x0000_0000	9.4.1.9/9-28
0x118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.10/9-30
0x11C	DDR_SDRAM_MODE_2—DDR SDRAM mode configuration 2	R/W	0x0000_0000	9.4.1.11/9-31
0x120	DDR_SDRAM_MD_CNTL—DDR SDRAM mode control	R/W	0x0000_0000	9.4.1.12/9-32
0x124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.13/9-34
0x128	DDR_DATA_INIT—DDR SDRAM data initialization	R/W	0x0000_0000	9.4.1.14/9-35
0x130	DDR_SDRAM_CLK_CNTL—DDR SDRAM clock control	R/W	0x0200_0000	9.4.1.15/9-35
0x140–0x144	Reserved	—	—	—
0x148	DDR_INIT_ADDR—DDR training initialization address	R/W	0x0000_0000	9.4.1.16/9-36
0x14C	DDR_INIT_EXT_ADDRESS—DDR training initialization extended address	R/W	0x0000_0000	9.4.1.17/9-36
0x150–0x15F	Reserved	—	—	—
0x160	TIMING_CFG_4— DDR SDRAM timing configuration 4	R/W	0x0000_0000	9.4.1.18/9-37
0x164	TIMING_CFG_5— DDR SDRAM timing configuration 5	R/W	0x0000_0000	9.4.1.19/9-39
0x168–0x16F	Reserved	—	—	—
0x170	DDR_ZQ_CNTL— DDR ZQ calibration control	R/W	0x0000_0000	9.4.1.20/9-40
0x174	DDR_WRLVL_CNTL— DDR write leveling control	R/W	0x0000_0000	9.4.1.21/9-42

Table 9-5. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x178–0xB1F	Reserved	—	—	—
0xB20	DDRDSR_1—DDR Debug Status Register 1	R	0x0000_0000	9.4.1.22/9-45
0xB24	DDRDSR_2—DDR Debug Status Register 2	R	0x0000_0000	9.4.1.23/9-46
0xB28	DDRCDR_1—DDR Control Driver Register 1	R/W	0x0000_0000	9.4.1.24/9-46
0xB2C	DDRCDR_2—DDR Control Driver Register 2	R/W	0x0000_0000	9.4.1.25/9-49
0xB30–0xBF7	Reserved	—	—	—
0xBF8	DDR_IP_REV1—DDR IP block revision 1	R	0x*nnnn_nnnn*[1]	9.4.1.26/9-50
0xBFC	DDR_IP_REV2—DDR IP block revision 2	R	0x00*nn*_00*nn*[1]	9.4.1.27/9-50
0xE00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.28/9-51
0xE04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.29/9-51
0xE08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.30/9-52
0xE20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.31/9-52
0xE24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.32/9-53
0xE28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.33/9-53
0xE40	ERR_DETECT—Memory error detect	w1c	0x0000_0000	9.4.1.34/9-54
0xE44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.35/9-55
0xE48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.36/9-56
0xE4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.37/9-57
0xE50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.38/9-58
0xE54	CAPTURE_EXT_ADDRESS—Memory error extended address capture	R/W	0x0000_0000	9.4.1.39/9-59
0xE58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.40/9-59
DDRC2 Registers				
Block Base Address: 0x0_6000				
DDRC2 registers **Note:** All registers defined for DDRC1 are also defined for DDRC2; the base address of DDRC2 registers is 0x0_6*nnn*.				

1 Implementation-dependent reset values are listed in specified section/page.

9.4.1 Register Descriptions

This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

9.4.1.1 Chip Select Memory Bounds (CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip selects. 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 are used, and the other chip selects' bounds registers are unused. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS are used, and all fields in CS1_BNDS are unused.

CS*n*_BNDS are shown in Figure 9-2.

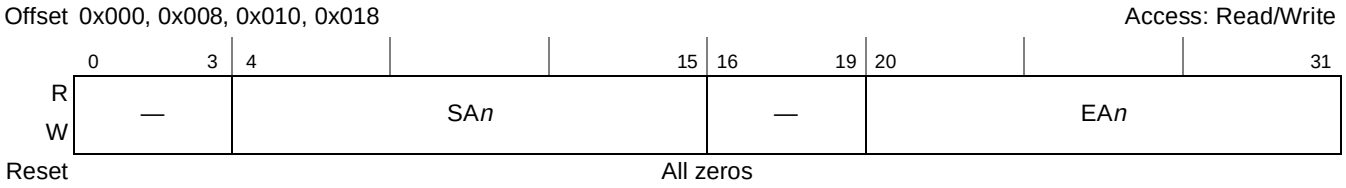


Figure 9-2. Chip Select Bounds Registers (CS*n*_BNDS)

Table 9-6 describes the CS*n*_BNDS register fields.

Table 9-6. CS*n*_BNDS Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–15	SA*n*	Starting address for chip select (bank) *n*. This value is compared against the 12 msbs of the 36-bit address.
16–19	—	Reserved
20–31	EA*n*	Ending address for chip select (bank) *n*. This value is compared against the 12 msbs of the 36-bit address.

9.4.1.2 Chip Select Configuration (CS*n*_CONFIG)

The chip select configuration (CS*n*_CONFIG) registers shown in Figure 9-3 enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers 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 are used, and the other registers' fields are 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 are used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG are used.

Offset 0x080, 0x084, 0x088, 0x08C Access: Read/Write

	0	1	2	3	4 7	8	9 11	12	13 15
R W	CS_*n*_EN	—	INTLV_EN	—	INTLV_CTL	AP_*n*_EN	ODT_RD_CFG	—	ODT_WR_CFG
Reset	All zeros								

	16 17	18 20	21 23	24 28	29 31
R W	BA_BITS_CS_*n*	—	ROW_BITS_CS_*n*	—	COL_BITS_CS_*n*
Reset	All zeros				

Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

Table 9-7 describes the CS*n*_CONFIG register fields.

Table 9-7. CS*n*_CONFIG Field Descriptions

Bits	Name	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.
1	—	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. 0 No interleaving 1 Interleaving between 2 memory controllers If this bit is set, INTLV_CTL must be set to select the bit(s) used to control interleaving.
3	—	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 30 of the 36-bit physical address is used to select the memory controller. Note that cache line interleaving should not be enabled when using DDR3 with a 64-bit bus. 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. 0100–1111Reserved
8	AP_*n*_EN	Chip select *n* auto-precharge enable 0 Chip select *n* is only auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n* always issues an auto-precharge for read and write transactions.

Table 9-7. CS*n*_CONFIG Field Descriptions (continued)

Bits	Name	Description
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 DDR2 or DDR3 memories. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS*n* 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 101–111 Reserved
12	—	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 DDR2 or DDR3 memories. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS*n* 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 101–111 Reserved
16–17	BA_BITS_CS_*n*	Number of bank bits for SDRAM on chip select *n*. These bits correspond to the sub-bank bits driven on MBA*n* in Table 9-51and Table 9-51. 00 2 logical bank bits 01 3 logical bank bits 10–11 Reserved
18–20	—	Reserved
21–23	ROW_BITS_CS_*n*	Number of row bits for SDRAM on chip select *n*. See Table 9-51and Table 9-51 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	—	Reserved
29–31	COL_BITS_CS_*n*	Number of column bits for SDRAM on chip select *n*. 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

9.4.1.3 Chip Select Configuration 2 (CS*n*_CONFIG_2)

The chip select configuration (CS*n*_CONFIG_2) registers shown in Figure 9-4 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 are used, and the other registers' fields are unused.

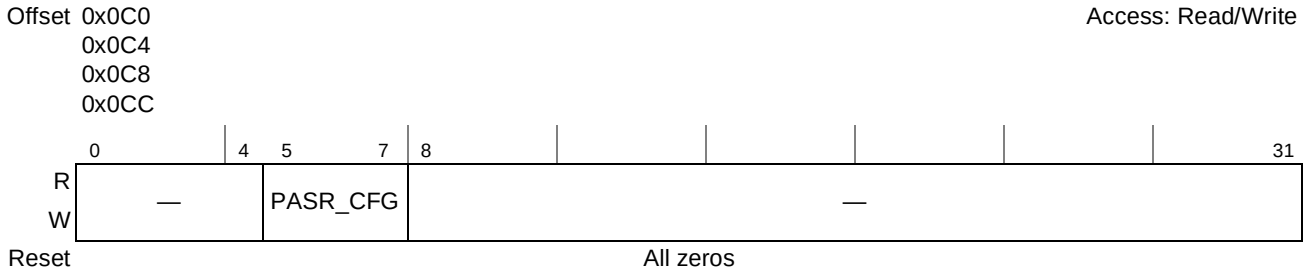


Figure 9-4. Chip Select Configuration Register 2 (CS*n*_CONFIG_2)

Table 9-8 describes the CS*n*_CONFIG_2 register fields.

Table 9-8. CS*n*_CONFIG_2 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	PASR_CFG	Partial array self refresh config. Controls the bits that are placed on MA[2:0] during the write to the EMRS(2) register when the automatic hardware DRAM initialization is used (DDR_SDRAM_CFG[BI] is cleared when DDR_SDRAM_CFG[MEM_EN] is set). If this field is a non-zero value, then it overrides 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* is optimized for partial array self refresh, as shown in Section 9.5.2, "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	—	Reserved

9.4.1.4 DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

DDR SDRAM timing configuration register 3, shown in Figure 9-5, sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

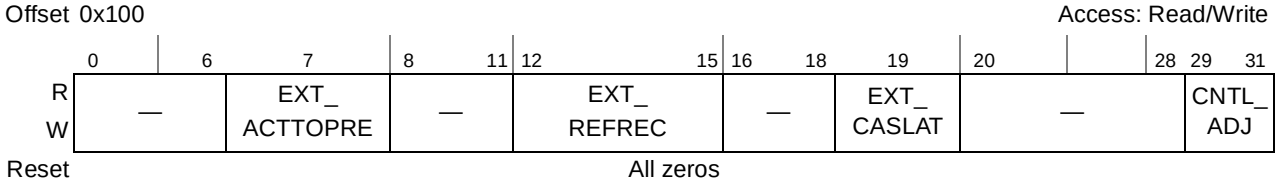


Figure 9-5. DDR SDRAM Timing Configuration 3 (TIMING_CFG_3)

Table 9-9 describes TIMING_CFG_3 fields.

Table 9-9. TIMING_CFG_3 Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
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
8–11	—	Reserved, should be cleared.
12–15	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 8bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 8bit value of the refresh recovery. t_{RFC} = {EXT_REFREC || REFREC} + 8, such that t_{RFC} is calculated as follows: 0000 0 clocks 1000 128 clocks 0001 16 clocks 1001 144 clocks 0010 32 clocks 1010 160 clocks 0011 48 clocks 1011 176 clocks 0100 64 clocks 1100 192 clocks 0101 80 clocks 1101 208 clocks 0110 96 clocks 1110 224 clocks 0111 112 clocks 1111 240 clocks
16–18	—	Reserved, should be cleared.
19	EXT_CASLAT	Extended $\overline{\text{MCAS}}$ 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	—	Reserved, should be cleared.
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], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] 000 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched aligned with the other DRAM address and control signals. 001 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched 1/2 platform cycle later than the other DRAM address and control signals. 010 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched 1 platform cycle later than the other DRAM address and control signals. 011 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched 3/2 platform cycles later than the other DRAM address and control signals. 100 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched 2 platform cycles later than the other DRAM address and control signals. 101 MODT[0:3], $\overline{\text{MCS}}$[0:3], and MCKE[0:3] are launched 5/2 platform cycles later than the other DRAM address and control signals. 110-111Reserved

9.4.1.5 DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

DDR SDRAM timing configuration register 0, shown in Figure 9-6, sets the number of clock cycles between various SDRAM control commands.

Offset 0x104 Access: Read/Write

	0 1	2 3	4 5	6 7	8	9 11	12 15	16 19	20 23	24 27	28 31
R W	RWT	WRT	RRT	WWT	—	ACT_PD_EXIT	PRE_PD_EXIT	—	ODT_PD_EXIT	—	MRS_CYC
Reset	0 0	0 0	0 0	0 0	0	0 0 1	0 0 0 1	0 0 0 0	0 0 0 1	0 0 0 0	0 1 0 1

Figure 9-6. DDR SDRAM Timing Configuration 0 (TIMING_CFG_0)

Table 9-10 describes TIMING_CFG_0 fields.

Table 9-10. TIMING_CFG_0 Field Descriptions

Bits	Name	Description
0–1	RWT	Read-to-write turnaround (t_{RTW}). Specifies how many extra cycles are added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller uses 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 determines 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 are added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller uses 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 determines 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. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
4–5	RRT	Read-to-read turnaround. Specifies how many extra cycles are added between reads to different chip selects. As a default, 3 cycles are 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 are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
6–7	WWT	Write-to-write turnaround. Specifies how many extra cycles are added between writes to different chip selects. As a default, 2 cycles are 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 are the default. Note that DDR2 does not support 8-beat bursts. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
8	—	Reserved, should be cleared.

Table 9-10. TIMING_CFG_0 Field Descriptions (continued)

Bits	Name	Description
9–11	ACT_PD_EXIT	Active powerdown exit timing (t_{XARD} and t_{XARDS}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12–15	PRE_PD_EXIT	Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any 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
16–19	—	Reserved, should be cleared.
20–23	ODT_PD_EXIT	ODT powerdown exit timing (t_{AXPD}). Specifies how many clocks must pass after exiting powerdown before ODT may be asserted. 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 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27	—	Reserved, should be cleared.
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

9.4.1.6 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 9-7, sets the number of clock cycles between various SDRAM control commands.

Offset 0x108 Access: Read/Write

	0–3	4–7	8–11	12–15	16–19	20–23	24	25–27	28	29–31
R W	PRETOACT	ACTTOPRE	ACTTORW	CASLAT	REFREC	WRREC	—	ACTTOACT	—	WRTORD
Reset	All zeros									

Figure 9-7. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

Table 9-11 describes TIMING_CFG_1 fields.

Table 9-11. TIMING_CFG_1 Field Descriptions

Bits	Name	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 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 1110 14 clocks 1111 15 clocks

Table 9-11. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
12–15	CASLAT	$\overline{\text{MCAS}}$ 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 Section 9.4.1.9, "DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)." 0000 Reserved — 1000 4.5 clocks 0001 1 clock — 1001 5 clocks 0010 1.5 clocks — 1010 5.5 clocks 0011 2 clocks — 1011 6 clocks 0100 2.5 clocks — 1100 6.5 clocks 0101 3 clocks — 1101 7 clocks 0110 3.5 clocks — 1110 7.5 clocks 0111 4 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 7-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 7-bit value of the refresh recovery, such that t_{RFC} is calculated as follows: t_{RFC} = {EXT_REFREC || REFREC} + 8. 0000 8 clocks — 0011 11 clocks 0001 9 clocks — ... 0010 10 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. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to *(t_{WR} + 2* cycles). 0000 Reserved — 1000 8 clocks 0001 1 clock — 1001 9 clocks 0010 2 clocks — 1010 10 clocks 0011 3 clocks — 1011 11 clocks 0100 4 clocks — 1100 12 clocks 0101 5 clocks — 1101 13 clocks 0110 6 clocks — 1110 14 clocks 0111 7 clocks — 1111 15 clocks
24	—	Reserved, should be cleared.
25–27	ACTTOACT	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). 000 Reserved — 100 4 clocks 001 1 clock — 101 5 clocks 010 2 clocks — 110 6 clocks 011 3 clocks — 111 7 clocks

Table 9-11. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
28	—	Reserved, should be cleared.
29–31	WRTORD	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. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to *(t_{WTR} + 2* cycles). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

9.4.1.7 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 9-8, sets the clock delay to data for writes.

Offset 0x10C Access: Read/Write

	0–3	4–8	9–12	13–15	16–18	19–21	22–23	25–26	31
R W	ADD_LAT	CPO	WR_LAT	—	RD_TO_PRE	WR_DATA_DELAY	—	CKE_PLS	FOUR_ACT
Reset	All zeros								

Figure 9-8. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

Table 9-12 describes the TIMING_CFG_2 fields.

Table 9-12. TIMING_CFG_2 Field Descriptions

Bits	Name	Description
0–3	ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. (DDR2-specific) 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 15 clocks

Table 9-12. TIMING_CFG_2 Field Descriptions (continued)

Bits	Name	Description
4–8	CPO[1]	$\overline{\text{MCAS}}$-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 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 Reserved
9–12	WR_LAT	Write latency. Note that the total write latency for DDR2 is equal to WR_LAT + ADD_LAT; the write latency for DDR1 is 1. DDR1 is not supported for the MPC8572E. If a write latency of 1 is desired, then the additive latency must also be set to at least 1 cycle. 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
13–15	—	Reserved
16–18	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. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to *(t_{RTP} + 2* cycles) 000 Reserved 001 1 cycle 010 2 cycles 011 3 cycles 100 4 cycles 101 5 cycles 110 6 cycles 111 7 cycles
19–21	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 Section 9.5.7, "DDR SDRAM Write Timing Adjustments," for details. The write preamble typically is driven high for 1/2 DRAM cycle, and then it is driven low for 1/2 DRAM cycle. However, for WR_DATA_DELAY settings of 0 clocks and 1/4 clocks, the write preamble is 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. 000 0 clock delay 001 1/4 clock delay 010 1/2 clock delay 011 3/4 clock delay 100 1 clock delay 101 5/4 clock delay 110 3/2 clock delay 111 Reserved
22	—	Reserved

Table 9-12. TIMING_CFG_2 Field Descriptions (continued)

Bits	Name	Description
23–25	CKE_PLS	Minimum CKE pulse width (t_{CKE}). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 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 ...01111030 cycles 011111 31 cycles 100000 32 cycles 100001–111111 Reserved

1 For CPO decodings other than 00000 and 11111, 'READ_LAT' is rounded up to the next integer value.

9.4.1.8 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 9-9, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Offset 0x110 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R/W	MEM_EN	SREN	ECC_EN	RD_EN	—	SDRAM_TYPE			—		DYN_PWR	DBW		8_BE	NCAP	—
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R/W	2T_EN	BA_INTLV_CTL							—		x32_EN	PCHB8	HSE	—	MEM_HALT	BI
Reset	All zeros															

Figure 9-9. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 9-13 describes the DDR_SDRAM_CFG fields.

Table 9-13. DDR_SDRAM_CFG Field Descriptions

Bits	Name	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.

Table 9-13. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
2	ECC_EN	ECC enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and ECC_EN must be one to ensure an interrupt is generated. See Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs. Note that RD_EN and 2T_EN must not both be set at the same time.
4	—	Reserved
5–7	SDRAM_TYPE	Type of SDRAM device to be used. This field is used when issuing the automatic hardware initialization sequence to DRAM via Mode Register Set and Extended Mode Register Set commands. 000–001Reserved 010 Reserved 011 DDR2 SDRAM 100 Reserved 101 Reserved 110 Reserved 111 DDR3 SDRAM
8–9	—	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	—	Reserved
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. 0 4-beat bursts are used on the DRAM interface. 1 8-beat bursts are used on the DRAM interface. **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 bus mode
14	NCAP	Non-concurrent auto-precharge. Some older DDR DRAMs do not support concurrent auto precharge. If one of these devices is used, then this bit needs to be set if auto precharge is used. 0 DRAMs in system support concurrent auto-precharge. 1 DRAMs in system do not support concurrent auto-precharge.
15	—	Reserved

Table 9-13. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
16	2T_EN	Enable 2T timing. 0 1T timing is enabled. The DRAM command/address are held for only 1 cycle on the DRAM bus. 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. Note that RD_EN and 2T_EN must not both be set at the same time.
17–23	BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('*x*' denotes a don't care bit value. All unlisted field values are reserved.) 0000000 No external memory banks are interleaved 1000000 External memory banks 0 and 1 are interleaved 0100000 External memory banks 2 and 3 are interleaved 1100000 External memory banks 0 and 1 are interleaved together **and** banks 2 and 3 are interleaved together *xx*00100 External memory banks 0 through 3 are all interleaved together
24–25	—	Reserved
26	x32_EN	x32 enable. 0 Either x8 or x16 discrete DRAM chips are used. In this mode, each data byte has a dedicated corresponding data strobe. 1 x32 discrete DRAM chips are used. In this mode, DQS0 is used to capture DQ[0:31], DQS4 is used to capture DQ[32:63] and DQS8 is used to capture ECC[0:7].
27	PCHB8	Precharge bit 8 enable. 0 MA[10] is used to indicate the auto-precharge and precharge all commands. 1 MA[8] is used to indicate the auto-precharge and precharge all commands. If x32_EN is cleared, then PCHB8 should be cleared as well.
28	HSE	Global half-strength override Sets I/O driver impedance to half strength. This impedance is 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 inSection 9.4.1.24, "DDR Control Driver Register 1 (DDRCDR_1) ."This bit should be cleared if using automatic hardware calibration. 0 I/O driver impedance is configured to full strength. 1 I/O driver impedance is configured to half strength.
29	—	Reserved

Table 9-13. DDR_SDRAM_CFG Field Descriptions (continued)

Bits	Name	Description
30	MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller does 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. 0 DDR controller accepts new transactions. 1 DDR controller finishes any remaining transactions, and then it remains halted until this bit is cleared by software.
31	BI	Bypass initialization 0 DDR controller cycles through initialization routine based on SDRAM_TYPE 1 Initialization routine is bypassed. Software is responsible for initializing memory through DDR_SDRAM_MODE2 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 does 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. See Section 9.4.1.16, “DDR Initialization Address (DDR_INIT_ADDR),” for details on avoiding ECC errors in this mode.

9.4.1.9 DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2, shown in Figure 9-10, provides more control configuration for the DDR controller.

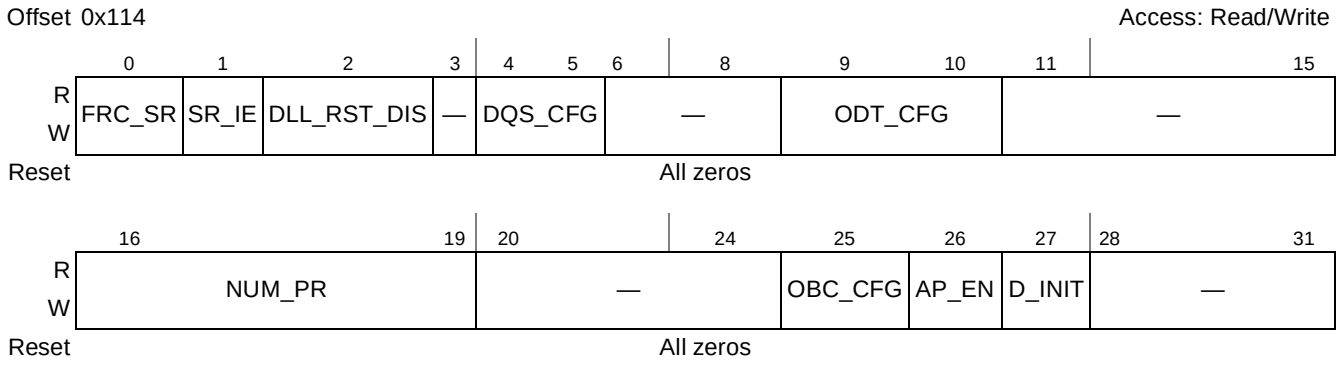


Figure 9-10. DDR SDRAM Control Configuration Register 2 (DDR_SDRAM_CFG_2)

Table 9-14 describes the DDR_SDRAM_CFG_2 fields.

Table 9-14. DDR_SDRAM_CFG_2 Field Descriptions

Bits	Name	Description
0	FRC_SR	Force self refresh 0 DDR controller operates in normal mode. 1 DDR controller enters 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 IRQ_OUT. This is considered a 'panic interrupt' by the DDR controller, and it enters self refresh as soon as possible. DDR_SDRAM_CFG[SREN] must also be set if the panic interrupt is used. 0 DDR controller does not enter self-refresh mode if panic interrupt is asserted. 1 DDR controller enters self-refresh mode if panic interrupt is asserted.
2	DLL_RST_DIS	DLL reset disable. The DDR controller typically issues a DLL reset to the DRAMs when exiting self refresh. However, this function may be disabled by setting this bit during initialization. 0 DDR controller issues a DLL reset to the DRAMs when exiting self refresh. 1 DDR controller does not issue a DLL reset to the DRAMs when exiting self refresh.
3	—	Reserved
4–5	DQS_CFG	DQS configuration 00 Only true DQS signals are used. 01 Differential DQS signals are used for DDR2 support. 10 Reserved 11 Reserved
6–8	—	Reserved
9–10	ODT_CFG	ODT configuration. This field defines how ODT is driven to the on-chip IOs. See Section 9.4.1.24, "DDR Control Driver Register 1 (DDRCDR_1)," which defines the termination value that is 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
16–19	NUM_PR	Number of posted refreshes. This determines 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 is issued at a time 0010 2 refreshes is issued at a time 0011 3 refreshes is issued at a time ... 1000 8 refreshes is issued at a time 1001–1111 Reserved

Table 9-14. DDR_SDRAM_CFG_2 Field Descriptions (continued)

Bits	Name	Description
25	OBC_CFG	On-The-Fly Burst Chop Configuration. Determines if on-the-fly Burst Chop is used. This bit should only be set if DDR3 memories are used. If on-the-fly Burst Chop mode is not used with DDR3 memories, then fixed Burst Chop mode may be used if the proper turnaround times are programmed into TIMING_CFG_0 and TIMING_CFG_4. DDR_SDRAM_CFG[8_BE] should be cleared for both on-the-fly Burst Chop mode or fixed Burst Chop mode when using a 64-bit data bus with DDR3 memories. 0 On-the-fly Burst Chop mode is disabled. Fixed burst lengths as defined in DDR_SDRAM_CFG[8_BE] are used. If fixed Burst Chop is used (with DDR3 memories), then DDR_SDRAM_CFG[8_BE] should be cleared. 1 On-the-fly Burst Chop mode is used. DDR_SDRAM_CFG[8_BE] should be cleared for on-the-fly Burst Chop mode. DDR_SDRAM_CFG[DBW] should also be cleared for on-the-fly Burst Chop mode
26	AP_EN	Address Parity Enable. Determines if address parity is generated and checked for the address and control signals when using registered DIMMs. If address parity is used, the MAPAR_OUT and MAPAR_ERR pins are used to drive the parity bit and to receive errors from the open-drain parity error signal. Even parity is used, and parity is generated for the MA[15:0], MBA[2:0], MRAS, MCAS, MWE signals. Parity does not generate for the MCKE[0:3], MODT[0:3], or MCS[0:3] signals. Note that address parity should not be used for non-zero values of TIMING_CFG_3[CNTL_ADJ]. 0 Address parity is not used 1 Address parity is used
27	D_INIT	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 automatically initializes DRAM after it is enabled. This bit is automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller initializes memory once it is enabled. This bit remains asserted until the initialization is complete. The value in DDR_DATA_INIT register is used to initialize memory.
28–31	—	Reserved

9.4.1.10 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in Figure 9-11, sets the values loaded into the DDR's mode registers.

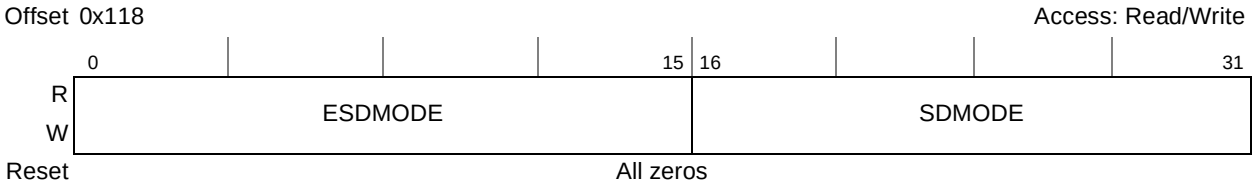


Figure 9-11. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 9-15 describes the DDR_SDRAM_MODE fields.

Table 9-15. DDR_SDRAM_MODE Field Descriptions

Bits	Name	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 lsb of ESDMODE, which, in the big-endian convention shown in Figure 9-11, 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[BI] is set, this field is still used during write leveling.
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 lsb of SDMODE, which, in the big-endian convention shown in Figure 9-11, 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].

9.4.1.11 DDR SDRAM Mode 2 Configuration (DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register, shown in Figure 9-12, sets the values loaded into the DDR's extended mode 2 and 3 registers (for DDR2).

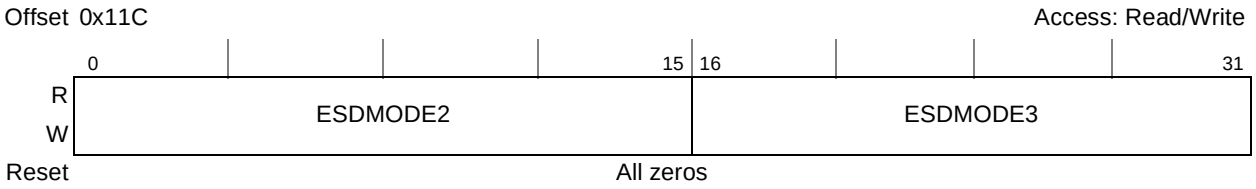


Figure 9-12. DDR SDRAM Mode 2 Configuration Register (DDR_SDRAM_MODE_2)

Table 9-16 describes the DDR_SDRAM_MODE_2 fields.

Table 9-16. DDR_SDRAM_MODE_2 Field Descriptions

Bits	Name	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 Figure 9-12, 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 Figure 9-12, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

9.4.1.12 DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register, shown in Figure 9-13, 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 signals to a specific value

Table 9-17 describes the fields of this register. Table 9-18 shows the user how to set the fields of this register to accomplish the above tasks.

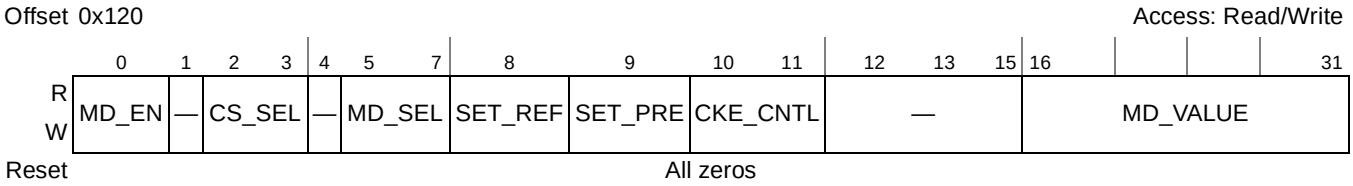


Figure 9-13. DDR SDRAM Mode Control Register (DDR_SDRAM_MD_CNTL)

Table 9-17 describes the DDR_SDRAM_MD_CNTL fields.

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 9-17. DDR_SDRAM_MD_CNTL Field Descriptions

Bits	Name	Description
0	MD_EN	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: • MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 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. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1	—	Reserved
2–3	CS_SEL	Select chip select. Specifies the chip select that is driven active due to any command forced by software in DDR_SDRAM_MD_CNTL. 00 Chip select 0 is active 01 Chip select 1 is active 10 Chip select 2 is active 11 Chip select 3 is active
4	—	Reserved

Table 9-17. DDR_SDRAM_MD_CNTL Field Descriptions (continued)

Bits	Name	Description
5–7	MD_SEL	Mode register select. MD_SEL specifies one of the following: • 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. Note that MD_SEL contains the value that is presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3
8	SET_REF	Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit is set by software and cleared by hardware once the command has been issued. 0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.
9	SET_PRE	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 is set by software and cleared by hardware once the command has been issued. 0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.
10–11	CKE_CNTL	Clock enable control. Allows software to globally clear or set all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value continues to be forced until software clears the CKE_CNTL bits. At that time, the DDR controller continues to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (such as, self refresh entry/exit, power down entry/exit). 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
12–15	—	Reserved
16–31	MD_VALUE	Mode register value. This field, which specifies the value that is 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

Table 9-18 shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 9-18. 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	—

Table 9-18. Settings of DDR_SDRAM_MD_CNTL Fields (continued)

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
CS_SEL	Chooses chip select (CS)			—
MD_SEL	Select mode register. See Table 9-17.	—	Selects logical bank	—
MD_VALUE	Value written to mode register	—	Only bit five is significant. See Table 9-17.	—
CKE_CNTL	0	0	0	See Table 9-17.

9.4.1.13 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 9-14, 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.

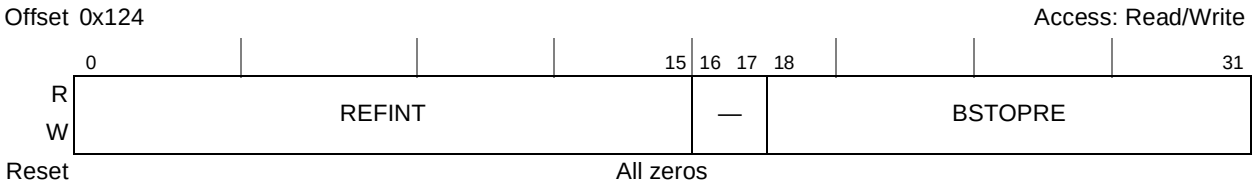


Figure 9-14. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 9-19 describes the DDR_SDRAM_INTERVAL fields.

Table 9-19. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	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 are not issued when the REFINT is set to all 0s.
16–17	—	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.

9.4.1.14 DDR SDRAM Data Initialization (DDR_DATA_INIT)

The DDR SDRAM data initialization register, shown in Figure 9-15, provides the value that is used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

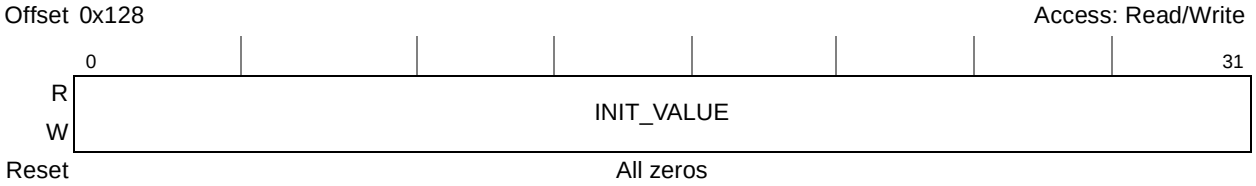


Figure 9-15. DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT)

Table 9-20 describes the DDR_DATA_INIT fields.

Table 9-20. DDR_DATA_INIT Field Descriptions

Bits	Name	Description
0–31	INIT_VALUE	Initialization value. Represents the value that DRAM is initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

9.4.1.15 DDR SDRAM Clock Control (DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register, shown in Figure 9-16, provides a 1/8-cycle clock adjustment.

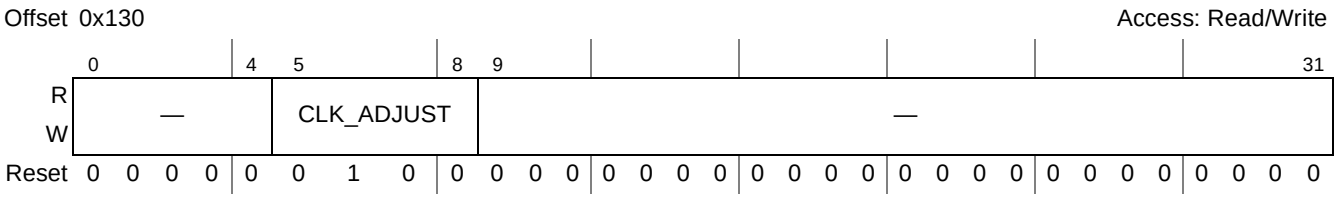


Figure 9-16. DDR SDRAM Clock Control Configuration Register (DDR_SDRAM_CLK_CNTL)

Table 9-21 describes the DDR_SDRAM_CLK_CNTL fields.

Table 9-21. DDR_SDRAM_CLK_CNTL Field Descriptions

Bits	Name	Description
0–4	—	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	—	Reserved

9.4.1.16 DDR Initialization Address (DDR_INIT_ADDR)

The DDR SDRAM initialization address register, shown in Figure 9-17, provides the address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

NOTE

After the skew adjustment, this address contains 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]).

If an $\overline{\text{HRESET}}$ has been issued after the DRAM is in self-refresh mode, however, memory is not initialized, so this address should be written to using an 8- or 32-byte transaction to avoid possible ECC errors if this address could later be accessed.

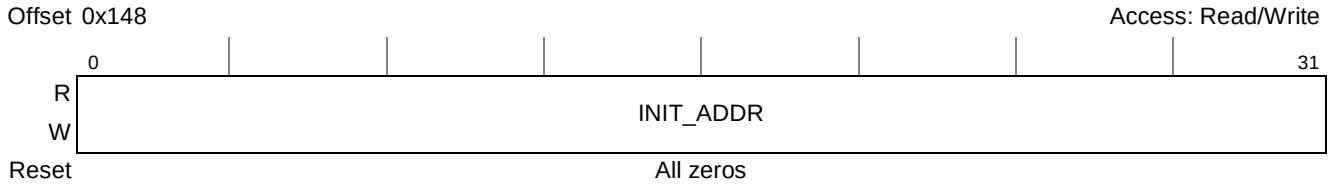


Figure 9-17. DDR Initialization Address Configuration Register (DDR_INIT_ADDR)

Table 9-22 describes the DDR_INIT_ADDR fields.

Table 9-22. DDR_INIT_ADDR Field Descriptions

Bits	Name	Description
0–31	INIT_ADDR	Initialization address. Represents the address that is used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at POR. This address is written to during the initialization sequence.

9.4.1.17 DDR Initialization Enable Extended Address (DDR_INIT_EXT_ADDR)

The DDR SDRAM initialization extended address register, shown in Figure 9-18, provides the extended address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration after POR.

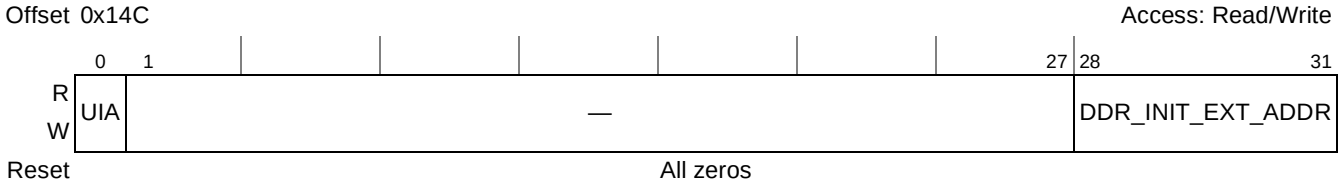


Figure 9-18. DDR Initialization Extended Address Configuration Register (DDR_INIT_EXT_ADDR)

Table 9-23 describes the DDR_INIT_EXT_ADDR fields.

Table 9-23. DDR_INIT_EXT_ADDR Field Descriptions

Bits	Name	Description
0	UIA	Use initialization address. 0 Use the default address for training sequence as calculated by the controller. This is the first valid address in the first enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR and DDR_INIT_EXT_ADDR.
1–27	—	Reserved, should be cleared.
28–31	INIT_EXT_ADDR	Initialization extended address. Represents the extended address that is used for the data strobe to data skew adjustment and automatic $\overline{\text{CAS}}$ to preamble calibration at POR. This extended address is written to during the initialization sequence.

9.4.1.18 DDR SDRAM Timing Configuration 4 (TIMING_CFG_4)

The DDR SDRAM timing configuration 4 register, shown in Figure 9-19, provides additional timing fields required to support DDR3 memories.

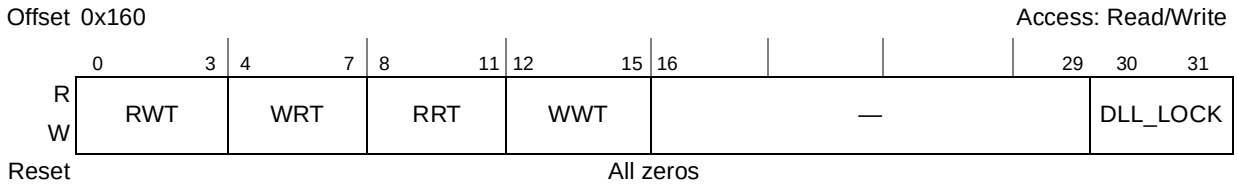


Figure 9-19. DDR SDRAM Timing Configuration 4 Register (TIMING_CFG_4)

Table 9-24 describes the TIMING_CFG_4 fields.

Table 9-24. TIMING_CFG_4 Field Descriptions

Bits	Name	Description
0–3	RWT	Read-to-write turnaround for same chip select. Specifies how many cycles are 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 uses 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] also is met before issuing a write command. 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

Table 9-24. TIMING_CFG_4 Field Descriptions (continued)

Bits	Name	Description
4–7	WRT	Write-to-read turnaround for same chip select. Specifies how many cycles are 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 uses 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 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] also is met before issuing a read command. 0000 Default 1000 8 clocks 0001 1 clock 1001 9 clocks 0010 2 clocks 1010 10 clocks 0011 3 clocks 1011 11 clocks 0100 4 clocks 1100 12 clocks 0101 5 clocks 1101 13 clocks 0110 6 clocks 1110 14 clocks 0111 7 clocks 1111 15 clocks
8–11	RRT	Read-to-read turnaround for same chip select. Specifies how many cycles are added between reads to the same chip select. If a value of 0000 is chosen, then 2 cycles are required between read commands to the same chip select if 4-beat bursts are used (4 cycles are 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. 0000 BL/2 clocks 1000 BL/2 + 8 clocks 0001 BL/2 + 1 clock 1001 BL/2 + 9 clocks 0010 BL/2 + 2 clocks 1010 BL/2 + 10 clocks 0011 BL/2 + 3 clocks 1011 BL/2 + 11 clocks 0100 BL/2 + 4 clocks 1100 BL/2 + 12 clocks 0101 BL/2 + 5 clocks 1101 BL/2 + 13 clocks 0110 BL/2 + 6 clocks 1110 BL/2 + 14 clocks 0111 BL/2 + 7 clocks 1111 BL/2 + 15 clocks
12–15	WWT	Write-to-write turnaround for same chip select. Specifies how many cycles are added between writes to the same chip select. If a value of 0000 is chosen, then 2 cycles are required between write commands to the same chip select if 4-beat bursts are used (4 cycles are 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. 0000 BL/2 clocks 1000 BL/2 + 8 clocks 0001 BL/2 + 1 clock 1001 BL/2 + 9 clocks 0010 BL/2 + 2 clocks 1010 BL/2 + 10 clocks 0011 BL/2 + 3 clocks 1011 BL/2 + 11 clocks 0100 BL/2 + 4 clocks 1100 BL/2 + 12 clocks 0101 BL/2 + 5 clocks 1101 BL/2 + 13 clocks 0110 BL/2 + 6 clocks 1110 BL/2 + 14 clocks 0111 BL/2 + 7 clocks 1111 BL/2 + 15 clocks
16–29	—	Reserved, should be cleared.
30–31	DLL_LOCK	DDR SDRAM DLL Lock Time. This provides the number of cycles that it takes for the DRAMs DLL to lock at POR and after exiting self refresh. The controller waits the specified number of cycles before issuing any commands after exiting POR or self refresh. 00 200 clocks 10 Reserved 01 512 clocks 11 Reserved

9.4.1.19 DDR SDRAM Timing Configuration 5 (TIMING_CFG_5)

The DDR SDRAM timing configuration 5 register, shown in Figure 9-20, provides additional timing fields required to support DDR3 memories.

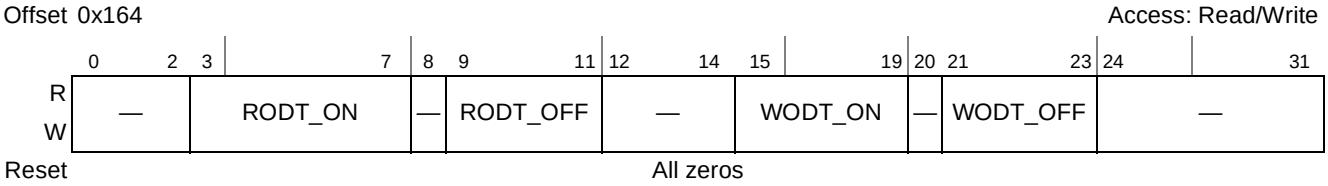


Figure 9-20. DDR SDRAM Timing Configuration 5 Register (TIMING_CFG_5)

Table 9-25 describes the TIMING_CFG_5 fields.

Table 9-25. TIMING_CFG_5 Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3–7	RODT_ON	Read to ODT on. Specifies the number of cycles that passes 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 is automatically added to the value selected in this field. 00000RL - 3 clocks — 1000015 clocks 000010 clocks — 1000116 clocks 000101 clocks — 1001017 clocks 000112 clocks — 1001118 clocks . — . . — . . — . 01111114 clocks — 1111130 clocks
8	—	Reserved, should be cleared.
9–11	RODT_OFF	Read to ODT off. Specifies the number of cycles that the relevant ODT signal(s) remains asserted for each read transaction. The default case (000) leaves the ODT signal(s) asserted for 3 DRAM cycles. 000 3 clocks — 100 4 clocks 001 1 clock — 101 5 clocks 010 2 clocks — 110 6 clocks 010 3 clocks — 111 7 clocks
12–14	—	Reserved, should be cleared.

Table 9-25. TIMING_CFG_5 Field Descriptions (continued)

Bits	Name	Description
15–19	WODT_ON	Write to ODT On Specifies the number of cycles that passes 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 is automatically added to the value selected in this field. 00000 WL – 3 clocks 10000 15 clocks 00001 0 clocks 10001 16 clocks 00010 1 clocks 10010 17 clocks 00011 2 clocks 10011 18 clocks . . . . . . 01111 14 clocks 11111 30 clocks
20	—	Reserved, should be cleared.
21–23	WODT_OFF	Write to ODT Off. Specifies the number of cycles that the relevant ODT signal(s) remains asserted for each write transaction. The default case (000) leaves the ODT signal(s) asserted for 3 DRAM cycles. 000 3 clocks 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 010 3 clocks 111 7 clocks
24–31	—	Reserved, should be cleared.

9.4.1.20 DDR ZQ Calibration Control (DDR_ZQ_CNTL)

The DDR ZQ Calibration Control register, shown in Figure 9-21, 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:

[((DDR_ZQ_CNTL[ZQOPER] + DDR_ZQ_CNTL[ZQCS])* (# enabled chip selects)) +
TIMING_CFG_1[PRETOACT] +
TIMING_CFG_1[REFREC] + $2t_{CK}$] > (DDR_SDRAM_INTERVAL[REFINT]),

then it is possible that one refresh is 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 do not allow for a refresh to be missed.

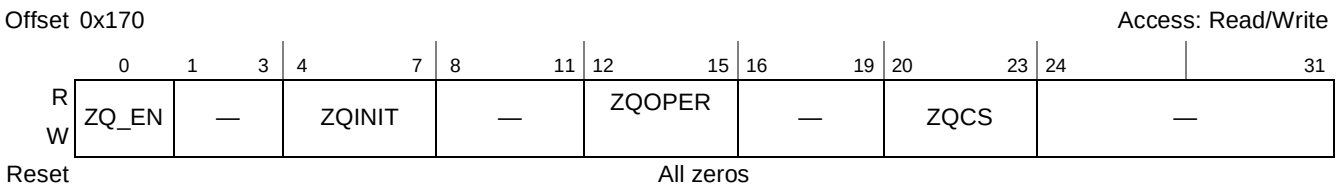


Figure 9-21. DDR ZQ Calibration Control Register (DDR_ZQ_CNTL)

Table 9-26 describes the DDR_ZQ_CNTL fields.

Table 9-26. DDR_ZQ_CNTL Field Descriptions

Bits	Name	Description
0	ZQ_EN	ZQ Calibration Enable. This bit determines if ZQ calibrating is used. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111). 0 ZQ Calibration is not used. 1 ZQ Calibration is used. A ZQCL command is issued by the DDR controller after POR and anytime the DDR controller is exiting self refresh. A ZQCS command is issued every 32 refresh sequences to account for VT variations.
1–3	—	Reserved, should be cleared.
4–7	ZQINIT	POR ZQ Calibration Time (t_{ZQinit}). Determines the number of cycles that must be allowed for DRAM ZQ calibration at POR. Each chip select is calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued. 0000–0110 Reserved 0111 128 clocks 1000 256 clocks 1001 512 clocks 1010 1024 clocks 1011–1111 Reserved
8–11	—	Reserved, should be cleared.
12–15	ZQOPER	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 is calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued. 0000-0110Reserved 0111 128 clocks 1000 256 clocks 1001 512 clocks 1010 1024 clocks 1011-1111Reserved
16–19	—	Reserved, should be cleared.

Table 9-26. DDR_ZQ_CNTL Field Descriptions (continued)

Bits	Name	Description
20–23	ZQCS	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 is calibrated separately, and this time must elapse after the ZQCS command is issued for each chip select before a separate command may be issued. 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	—	Reserved, should be cleared.

9.4.1.21 DDR Write Leveling Control (DDR_WRLVL_CNTL)

The DDR Write Leveling Control register, shown in Figure 9-22, provides controls for write leveling, as it is supported for DDR3 memory devices.

Offset 0x174 Access: Read/Write

	0	1	4	5 7	8	9 11	12	13 15	16 19	20	21 23	24 26	27	28 31
R W	WRLVL_EN	—		WRLVL_MRD	—	WRLVL_ODTEN	—	WRLVL_DQSEN	WRLVL_SMPL	—	WRLVL_WLR	—		WRLVL_START
Reset	All zeros													

Figure 9-22. DDR Write Leveling Control Register (DDR_WRLVL_CNTL)

Table 9-27 describes the DDR_WRLVL_CNTL fields.

Table 9-27. DDR_WRLVL_CNTL Field Descriptions

Bits	Name	Description
0	WRLVL_EN	Write Leveling Enable. This bit determines if write leveling is used. If this bit is set, then the DDR controller performs 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). In addition, write leveling is not supported for DDR3 mirrored DIMMs. 0 Write leveling is not used 1 Write leveling is used
1–4	—	Reserved, should be cleared.

Table 9-27. DDR_WRLVL_CNTL Field Descriptions (continued)

Bits	Name	Description
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 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
8	—	Reserved, should be cleared.
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	—	Reserved, should be cleared.
13–15	WRLVL_DQSEN	DQS/$\overline{\text{DQS}}$ 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

Table 9-27. DDR_WRLVL_CNTL Field Descriptions (continued)

Bits	Name	Description
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 Reserved (if DDR_WRLVL_CNTL[WRLVL_EN] is set) 0001 1 clocks 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 1010 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1010 14 clocks 1111 15 clocks
20	—	Reserved, should be cleared.
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

Table 9-27. DDR_WRLVL_CNTL Field Descriptions (continued)

Bits	Name	Description
24–27	—	Reserved, should be cleared.
28–31	WRLVL_START	Write leveling start time. 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 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 10101-11111Reserved

9.4.1.22 DDR Debug Status Register 1 (DDRDSR_1)

The DDRDSR_1 register, shown in Figure 9-23, 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.

Offset 0xB20 Access: Read only

	0 1	2 5	6 9	10 15	16 19	20 23	24 27	28 31
R	DDRDC	MDICPZ	MDICNZ	—	CPZ	CNZ	DPZ	DNZ
W								
Reset	All zeros							

Figure 9-23. DDR Debug Status Register 1 (DDRDSR_1)

Table 9-28 describes the DDRDSR_1 fields.

Table 9-28. DDRDSR_1 Field Descriptions

Bits	Name	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	—	Reserved, should be cleared.

Table 9-28. DDRDSR_1 Field Descriptions (continued)

Bits	Name	Description
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
28–31	DNZ	Current setting of NFET driver data impedance

9.4.1.23 DDR Debug Status Register 2 (DDRDSR_2)

The DDRDSR_2 register, shown in Figure 9-24, contains the current settings of the P and N FET impedance for the DDR drivers for clocks.

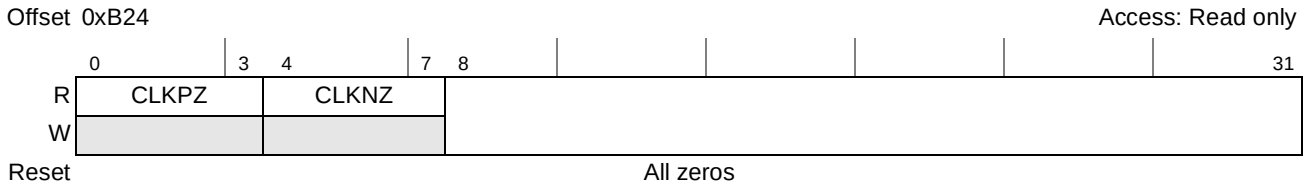


Figure 9-24. DDR Debug Status Register 2 (DDRDSR_2)

Table 9-29 describes the DDRDSR_2 fields.

Table 9-29. DDRDSR_2 Field Descriptions

Bits	Name	Description
0–3	CLKPZ	Current setting of PFET driver clock impedance
4–7	CLKNZ	Current setting of NFET driver clock impedance
8–31	—	Reserved

9.4.1.24 DDR Control Driver Register 1 (DDRCDR_1)

DDRCDR_1, shown in Figure 9-25, 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. This can be used to calibrate the DDR drivers to 18 Ω. However, this should only be used for full-strength driver applications.

The hardware DDR driver calibration is enabled via DDRCDR_1[DHC_EN]. If this is used, then it should be set 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, 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, 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]

Note that the legal impedance values (from highest impedance to lowest impedance) for DDR2 (1.8 V) are:

- 0000
- 0001
- 0011
- 0010
- 0110
- 0111
- 0101
- 0100
- 1100
- 1101
- 1110
- 1010 (default full-strength impedance)
- 1011
- 1001

A value of 1111 provides the target for half-strength mode when driver calibration is not used.

Note that the legal impedance values (from highest impedance to lowest impedance) for DDR3 (1.5 V) are:

- 0000
- 0001
- 0011
- 0010
- 0110

- 0111 (default full-strength impedance)
- 0101
- 0100
- 1100
- 1101

A value of 0000 should be used for default half-strength mode when driver calibration is not used.

Note that the drivers may either be calibrated to full-strength or half-strength.

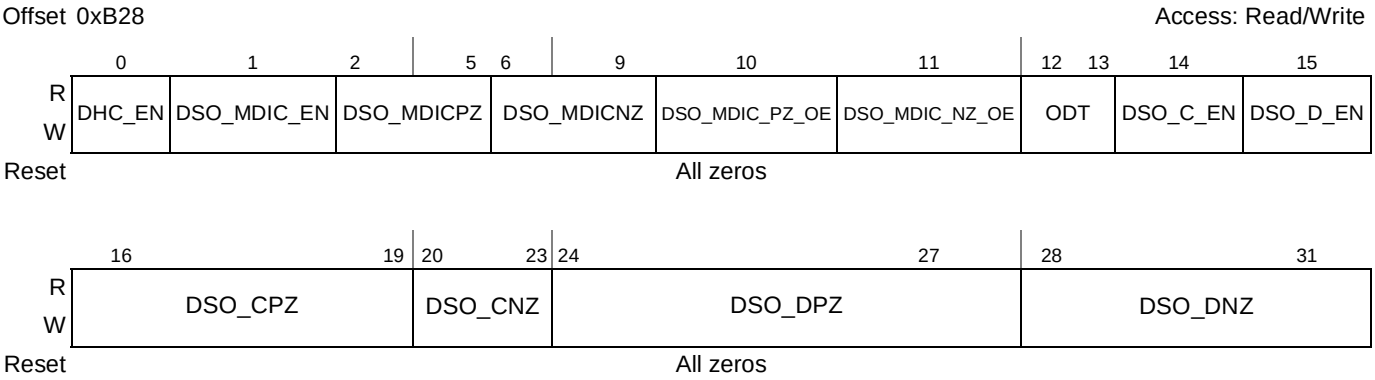


Figure 9-25. DDR Control Driver Register 1 (DDRCDR_1)

Table 9-30 describes the DDRCDR_1 fields.

Table 9-30. DDRCDR_1 Field Descriptions

Bits	Name	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 2 fields 000 75 Ω 001 46 Ω 010 60 Ω 011 43 Ω 100 150 Ω 101 33 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

Table 9-30. DDRCDR_1 Field Descriptions (continued)

Bits	Name	Description
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
28–31	DSO_DNZ	Driver software data n-impedance override

9.4.1.25 DDR Control Driver Register 2 (DDRCDR_2)

The DDRCDR_2, shown in Figure 9-26, sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

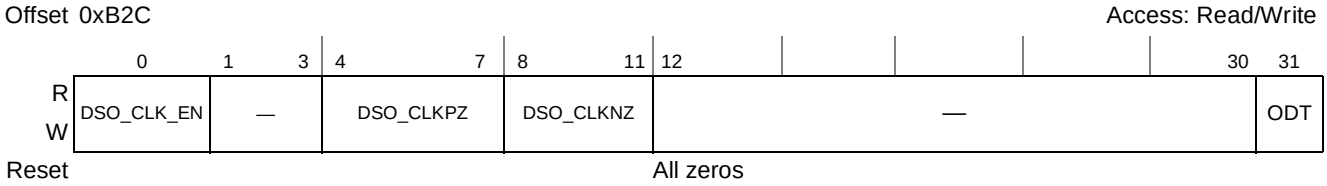


Figure 9-26. DDR Control Driver Register 2 (DDRCDR_2)

Table 9-31 describes the DDRCDR_2 fields.

Table 9-31. DDRCDR_2 Field Descriptions

Bits	Name	Description
0	DSO_CLK_EN	Driver software override enable for clocks
1–3	—	Reserved
4–7	DSO_CLKPZ	Driver software clocks p-impedance override
8–11	DSO_CLKNZ	Driver software clocks n-impedance override
12–30	—	Reserved
31	ODT	ODT termination value for IOs. This is combined with DDRCDR_1[ODT] to determine the termination value. Below is the termination based on concatenating these 2 fields 000 75 Ω 001 46 Ω 010 60 Ω 011 43 Ω 100 150 Ω 101 33 Ω 110 120 Ω 111 Reserved

9.4.1.26 DDR IP Block Revision 1 (DDR_IP_REV1)

The DDR IP block revision 1 register, shown in Figure 9-27, provides read-only fields with the IP block ID, along with major and minor revision information.

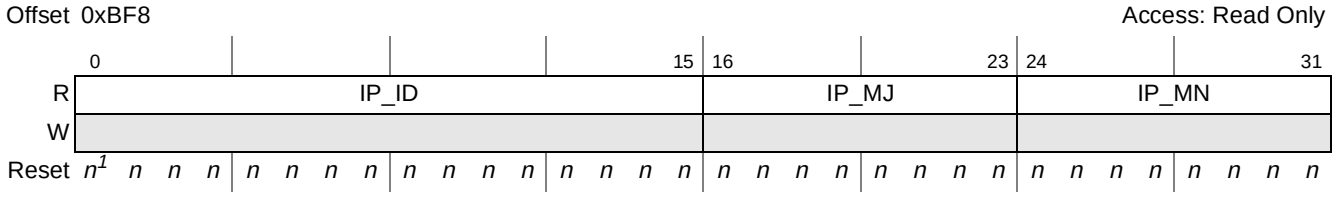


Figure 9-27. DDR IP Block Revision 1 (DDR_IP_REV1)

1 For reset values, see Table 9-32.

Table 9-32 describes the DDR_IP_REV1 fields.

Table 9-32. DDR_IP_REV1 Field Descriptions

Bits	Name	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 0x00.

9.4.1.27 DDR IP Block Revision 2 (DDR_IP_REV2)

The DDR IP block revision 2 register, shown in Figure 9-28, provides read-only fields with the IP block integration and configuration options.

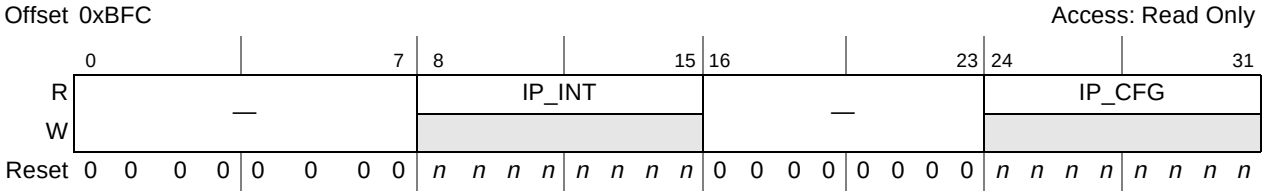


Figure 9-28. DDR IP Block Revision 2 (DDR_IP_REV2)

Table 9-33 describes the DDR_IP_REV2 fields.

Table 9-33. DDR_IP_REV2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	IP block integration options
16–23	—	Reserved
24–31	IP_CFG	IP block configuration options

9.4.1.28 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in Figure 9-29.

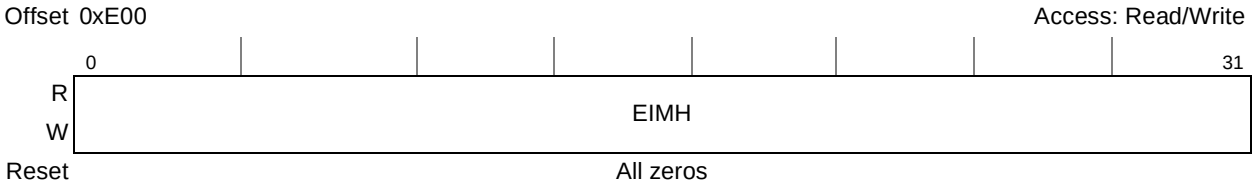


Figure 9-29. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

Table 9-34 describes the DATA_ERR_INJECT_HI fields.

Table 9-34. DATA_ERR_INJECT_HI Field Descriptions

Bits	Name	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.

9.4.1.29 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in Figure 9-30.

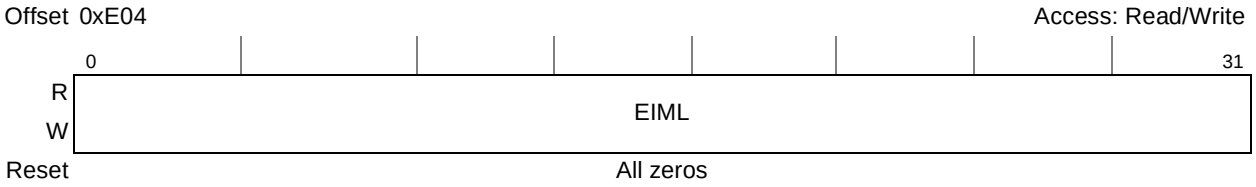


Figure 9-30. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

Table 9-35 describes the DATA_ERR_INJECT_LO fields.

Table 9-35. DATA_ERR_INJECT_LO Field Descriptions

Bits	Name	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.

9.4.1.30 Memory Data Path Error Injection Mask ECC (ERR_INJECT)

The memory data path error injection mask ECC register, shown in Figure 9-31, 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.

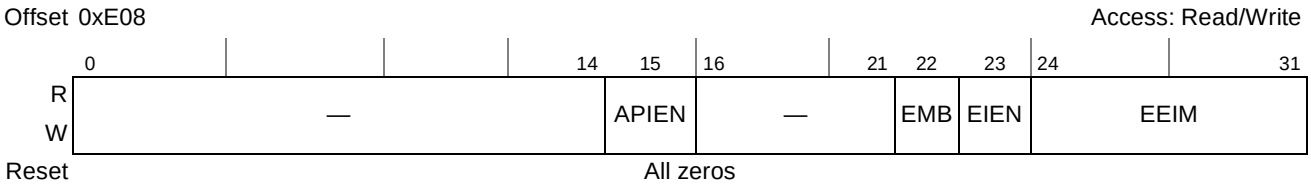


Figure 9-31. Memory Data Path Error Injection Mask ECC Register (ERR_INJECT)

Table 9-36 describes the ERR_INJECT fields.

Table 9-36. ERR_INJECT Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	APIEN	Address parity error injection enable. This bit is 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	—	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.

9.4.1.31 Memory Data Path Read Capture High (CAPTURE_DATA_HI)

The memory data path read capture high register, shown in Figure 9-32, stores the high word of the read data path during error capture.

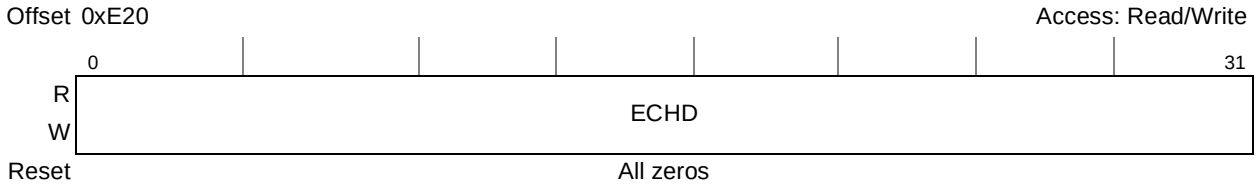


Figure 9-32. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 9-37 describes the CAPTURE_DATA_HI fields.

Table 9-37. CAPTURE_DATA_HI Field Descriptions

Bits	Name	Description
0–31	ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.

9.4.1.32 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in Figure 9-33, stores the low word of the read data path during error capture.

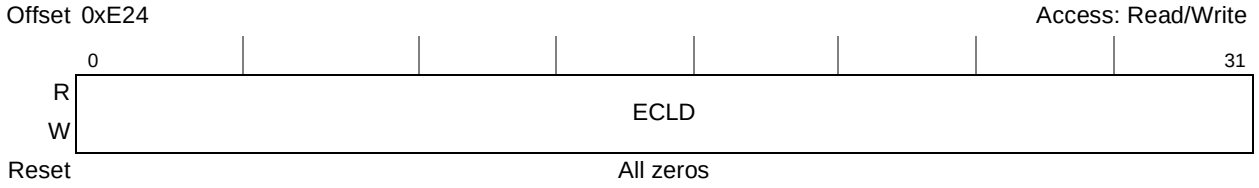


Figure 9-33. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 9-38 describes the CAPTURE_DATA_LO fields.

Table 9-38. CAPTURE_DATA_LO Field Descriptions

Bits	Name	Description
0–31	ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

9.4.1.33 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in Figure 9-34, stores the ECC syndrome bits that were on the data bus when an error was detected.

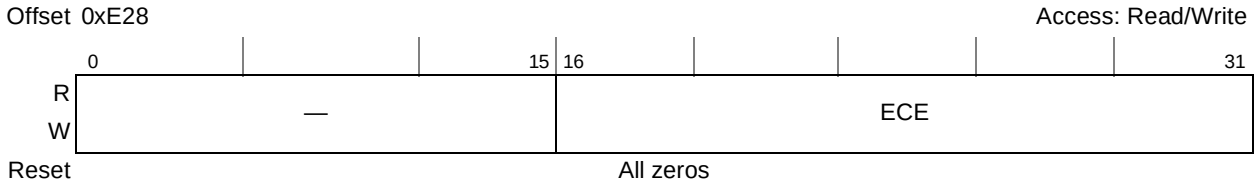


Figure 9-34. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-39 describes the CAPTURE_ECC fields.

Table 9-39. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 0:7—8-bit ECC for 1st 16 bits in 16-bit bus mode; should be ignored for 32-bit and 64-bit mode 8:15—8-bit ECC for 2nd 16 bits in 16-bit bus mode; 1st 32 bits in 32-bit bus mode; should be ignored for 64-bit bus mode 16:23—8-bit ECC for 3rd 16 bits in 16-bit bus mode; should be ignored for 32-bit and 64-bit mode 24:31—8-bit ECC for 4th 16 bits in 16-bit bus mode; 2nd 32 bits in 32-bit bus mode; all 64-bits in 64-bit bus mode

9.4.1.34 Memory Error Detect (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.

ERR_DETECT is shown in Figure 9-35.

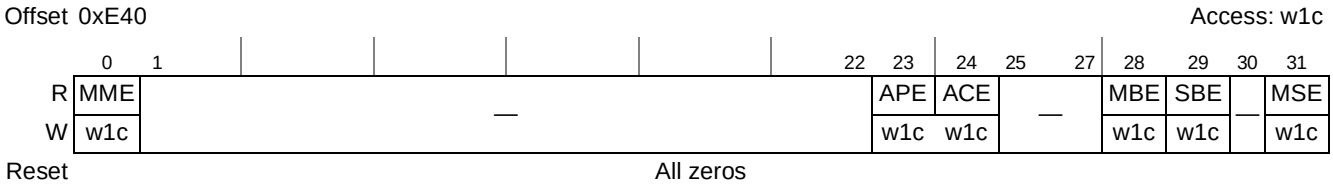


Figure 9-35. Memory Error Detect Register (ERR_DETECT)

Table 9-40 describes the ERR_DETECT fields.

Table 9-40. ERR_DETECT Field Descriptions

Bits	Name	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	—	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.
25–27	—	Reserved

Table 9-40. ERR_DETECT Field Descriptions (continued)

Bits	Name	Description
28	MBE	Multiple-bit error. This bit is cleared by software writing a 1. 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	—	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.

9.4.1.35 Memory Error Disable (ERR_DISABLE)

The memory error disable register, shown in Figure 9-36, allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

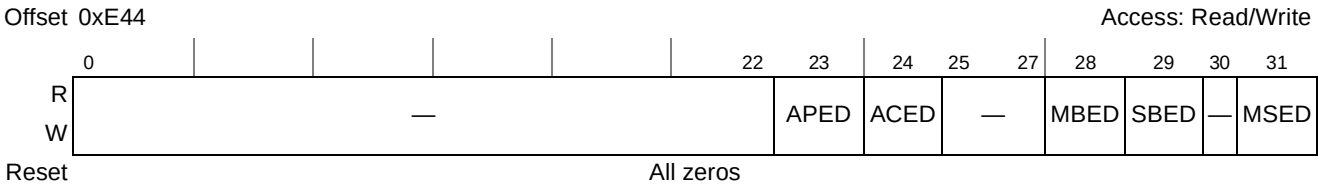


Figure 9-36. Memory Error Disable Register (ERR_DISABLE)

Table 9-41 describes the ERR_DISABLE fields.

Table 9-41. ERR_DISABLE Field Descriptions

Bits	Name	Description
0–22	—	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.
25–27	—	Reserved
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. Note that uncorrectable read errors cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED and ERR_INT_EN[MBEE] must be zero and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.

Table 9-41. ERR_DISABLE Field Descriptions (continued)

Bits	Name	Description
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

9.4.1.36 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 9-37, enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

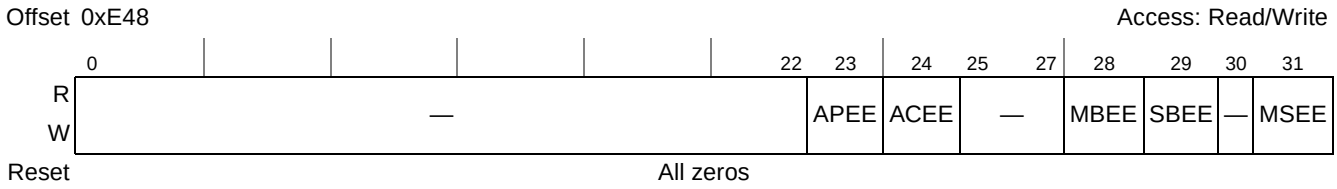


Figure 9-37. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 9-42 describes the ERR_INT_EN fields.

Table 9-42. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–22	—	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.
25–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBEE and ERR_DISABLE[MBED] must be zero and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 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.

Table 9-42. ERR_INT_EN Field Descriptions (continued)

Bits	Name	Description
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

9.4.1.37 Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)

The memory error attributes capture register, shown in Figure 9-38, sets attributes for errors including type, size, source, and others.

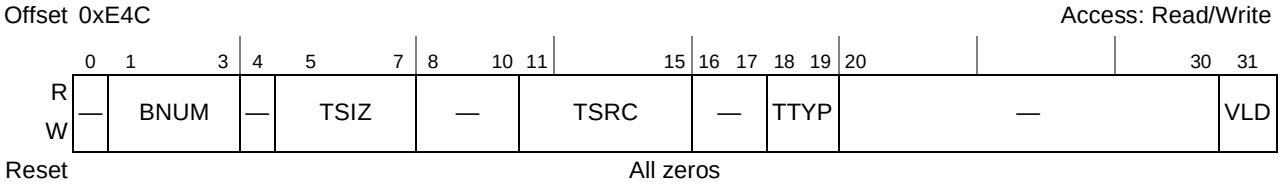


Figure 9-38. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 9-43 describes the CAPTURE_ATTRIBUTES fields.

Table 9-43. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4	—	Reserved
5–7	TSIZ	Transaction size for the error. Captures the transaction size in double words. 000 4 double words 001 1 double word 010 2 double words 011 3 double words Others Reserved
8–10	—	Reserved

Table 9-43. CAPTURE_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
11–15	TSRC	Transaction source for the error 00000 PCI Express 3 00001 PCI Express 2 00010 PCI Express 1 00011 Reserved 00100 PME 00101–00110 Reserved 00111 Security 01000 FEC 01001 Reserved 01010 Boot sequencer 01011 Reserved 01100 Serial RapidIO 01101 Reserved 01110 TLU 1 01111 TLU 2 10000 Processor 0 (instruction) 10001 Processor 0 (data) 10010 Processor 1 (instruction) 10011 Processor 1 (data) 10100 Reserved 10101 DMA 1 10110 DMA 2 10111 SAP 11000 eTSEC 1 11001 eTSEC 2 11010 eTSEC 3 11011 eTSEC 4 11100 RapidIO Message Unit 11101 RapidIO Doorbell Unit 11110 RapidIO Port-write Unit 11111 Reserved
		100xy are used by the QUICC Engine block as follows: x = TSRC[3] Least significant bit of SNUM y = TSCRC[4] CETM bit from RBMR/TBMR registers. RBMR/TBMR registers are described in protocol chapters. 101xx Reserved 11xxx Reserved **Note:** TSRC reflects the source of transaction and is used for debug purposes.
16–17	—	Reserved
18–19	TTYP	Transaction type for the error. 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VLD	Valid. Set as soon as valid information is captured in the error capture registers.

9.4.1.38 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 9-39, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

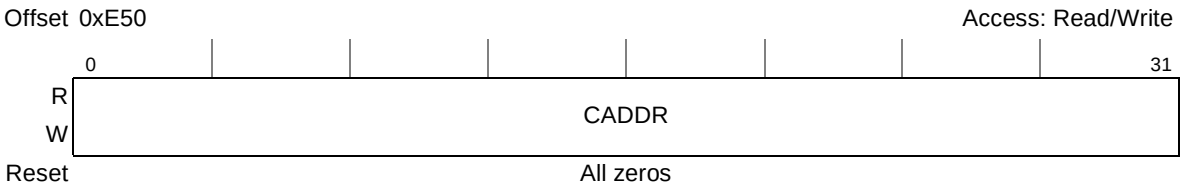


Figure 9-39. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 9-44 describes the CAPTURE_ADDRESS fields.

Table 9-44. CAPTURE_ADDRESS Field Descriptions

Bits	Name	Description
0–31	CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

9.4.1.39 Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS)

The memory error extended address capture register, shown in Figure 9-40, holds the four most significant transaction bits when an error is detected.

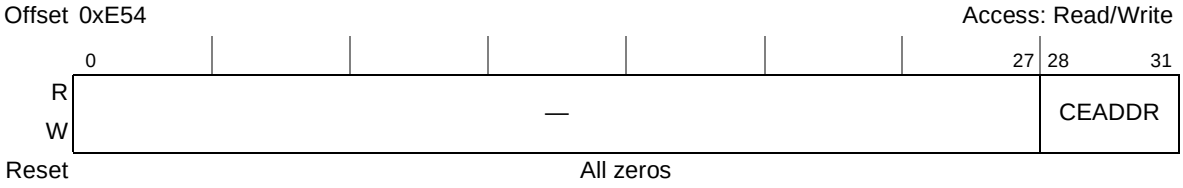


Figure 9-40. Memory Error Extended Address Capture Register (CAPTURE_EXT_ADDRESS)

Table 9-45 describes the CAPTURE_EXT_ADDRESS fields.

Table 9-45. CAPTURE_EXT_ADDRESS Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	CEADDR	Captured extended address. Captures the 4 msbs of the transaction address when an error is detected

9.4.1.40 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in Figure 9-41, 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.

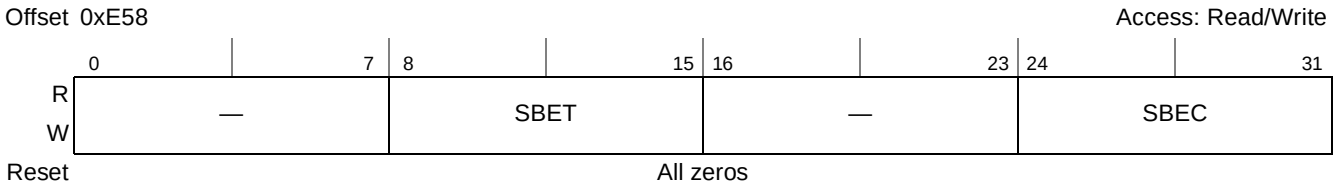


Figure 9-41. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 9-46 describes the ERR_SBE fields.

Table 9-46. ERR_SBE Field Descriptions

Bits	Name	Description
0–7	—	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	—	Reserved
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 a machine check or critical interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR3 and DDR2 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. In addition, DDR3 DIMM module specifications allow for vendors to use mirrored DIMMs, where some address and bank address lines are mirrored on the DIMM. The memory controller only supports these if the DDR_SDRAM_MD_CNTL register is used to initialize memory with DDR_SDRAM_CFG[BI] set. However, write leveling is not supported if these DIMMs are used.

Figure 9-42 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-/40bit wide memory. Bank sizes up to 4 Gbytes are supported, providing up to a maximum of 16 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].

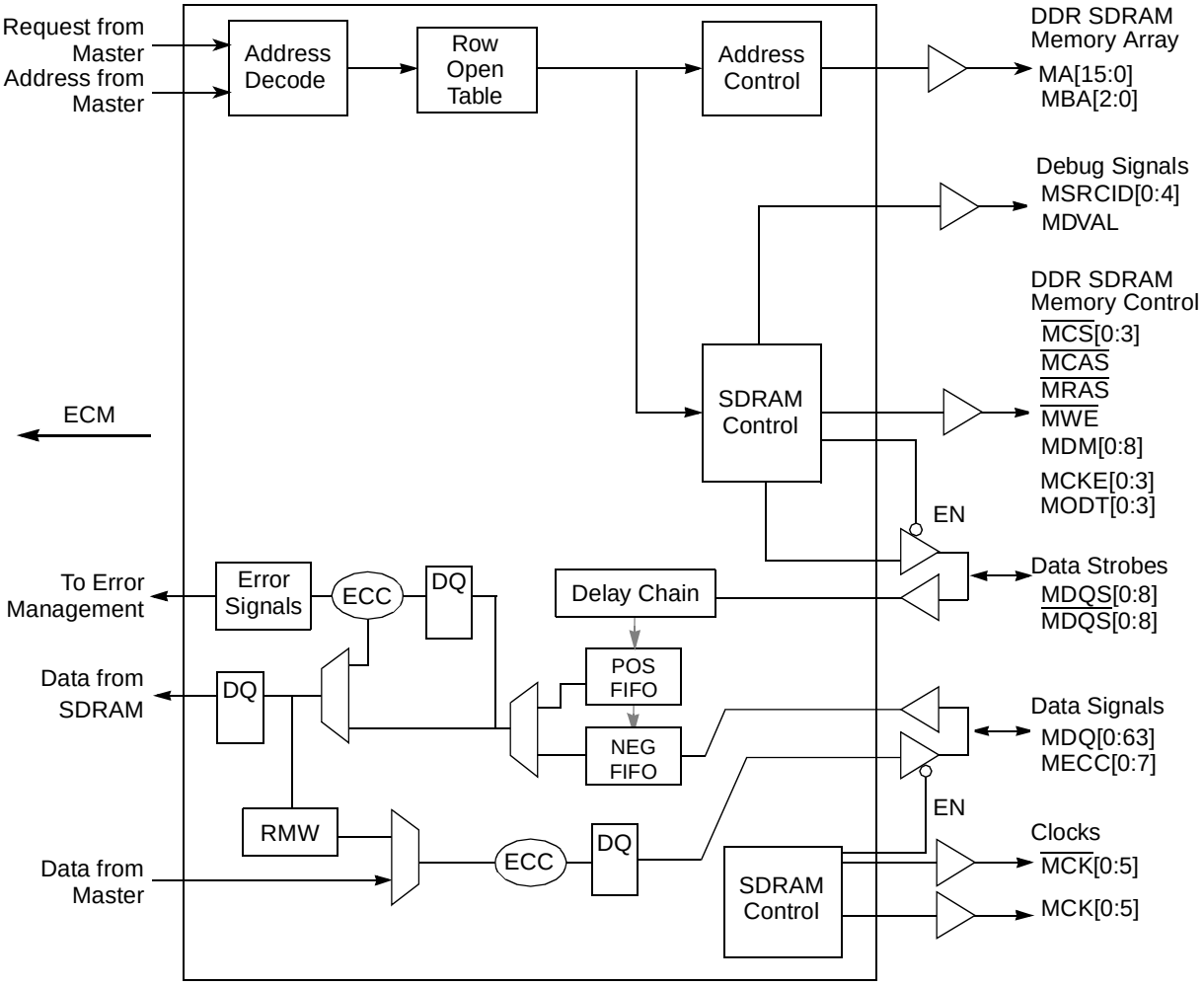


Figure 9-42. DDR Memory Controller Block Diagram

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[0:8]) 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 9-43 shows an example DDR SDRAM configuration with four logical banks.

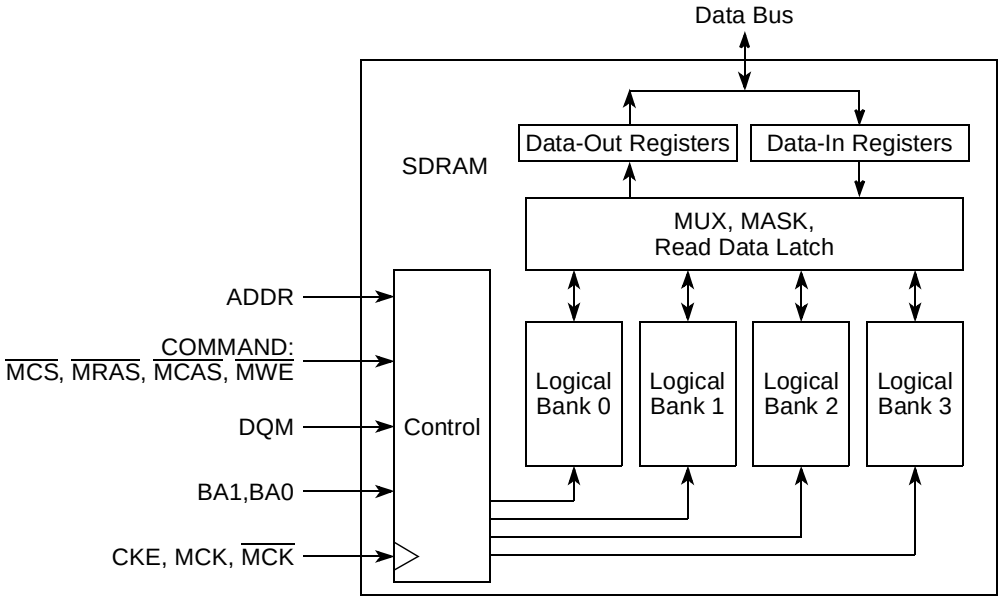


Figure 9-43. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-44 shows some typical signal connections.

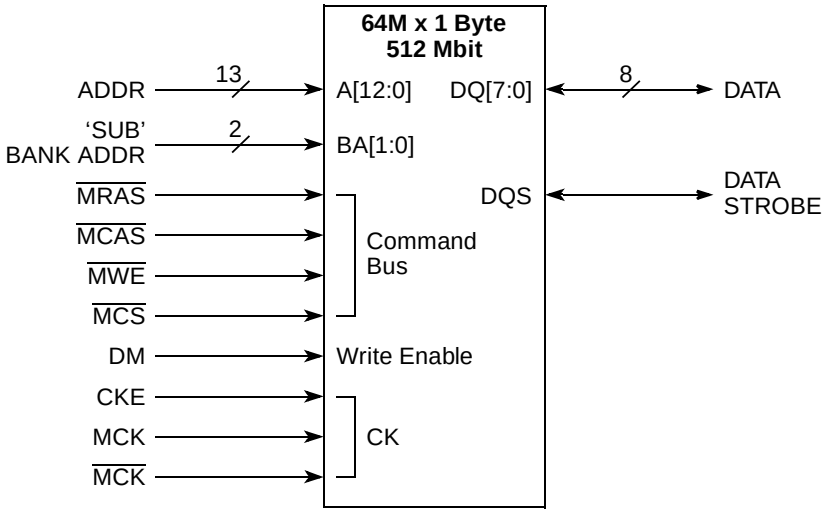
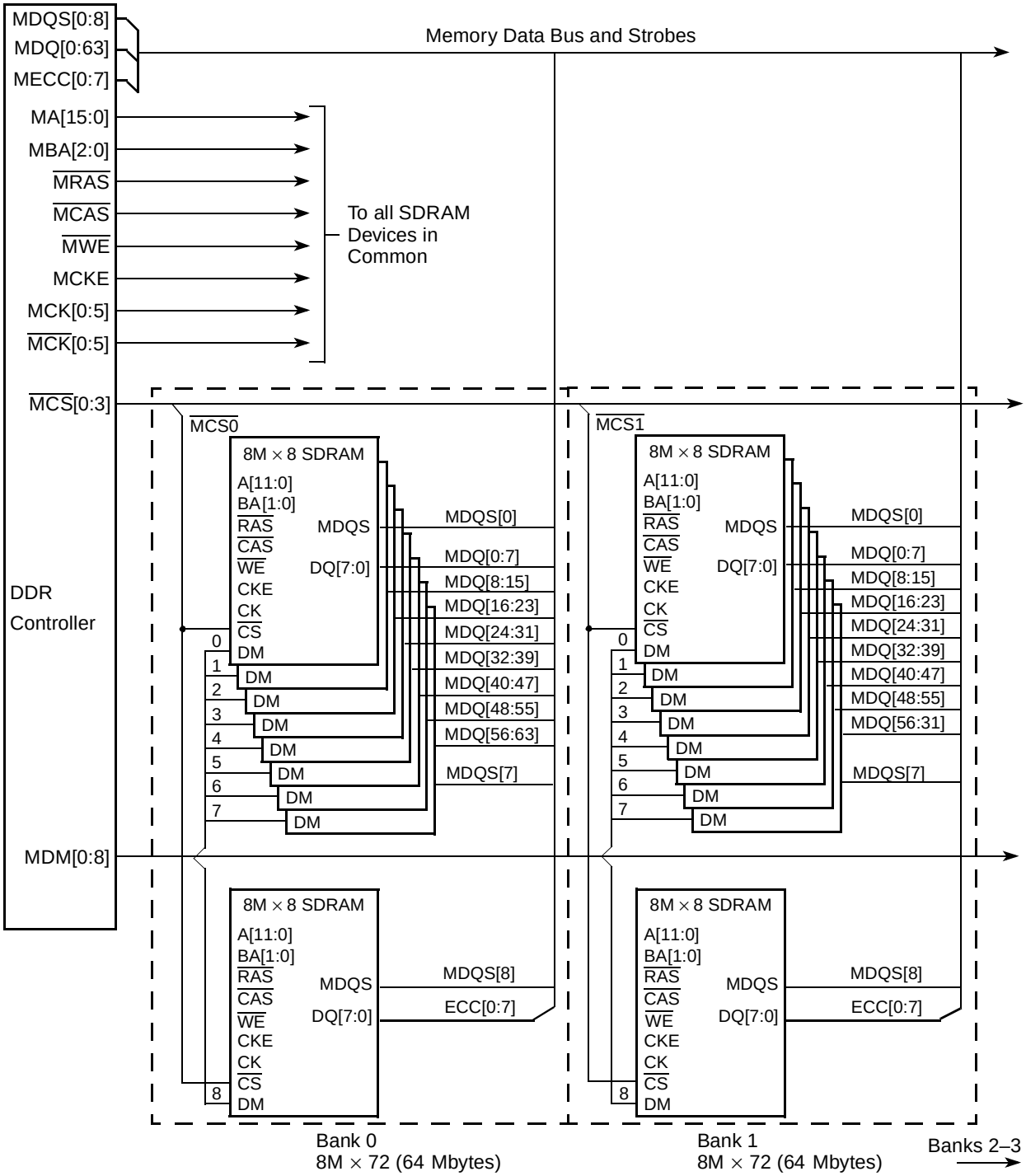


Figure 9-44. Typical DDR SDRAM Interface Signals

Figure 9-45 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M × 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.



1. All signals are connected in common (in parallel) except for MCS[0:3], MCK[0:5], MDM[0:8], and the data bus signals.
2. Each of the MCS[0:3] signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0:5] may be apportioned among all memory devices. Complementary bus is not shown.

Figure 9-45. Example 256-Mbyte DDR SDRAM Configuration With ECC

Section 9.5.12, "Error Management," explains how the DDR memory controller handles errors.

9.5.1 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 ($\overline{CS}$) 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 x8, x16, or x32 memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit, 2 Gbits, and 4 Gbits. 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 9-47 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 9-47. 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]

9.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 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.

Table 9-49 and Table 9-50 describe DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

Table 9-48. Supported DDR1 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-Bank Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits x 8	12 x 9 x 2	64 Mbytes	256 Mbytes
64 Mbits[1]	4 Mbits x 16	12 x 8 x 2	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits x 8	12 x 10 x 2	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits x 16	12 x 9 x 2	64 Mbytes	256 Mbytes
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	13 x 11 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 11 x 2	1 Gbyte	4 Gbytes
1 Gbit	64 Mbits x 16	14 x 10 x 2	512 Mbytes	2 Gbytes
2 Gbits	256 Mbits x 8	15 x 11 x 2	2 Gbytes	—
2 Gbits	128 Mbits x 16	15 x 10 x 2	1 Gbyte	4 Gbytes
4 Gbits	512 Mbits x 8	16 x 11 x 2	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	16 x 10 x 2	2 Gbytes	8 Gbytes

1 This configuration is not supported in 16-bit bus mode.

Table 9-49. 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	—
2 Gbits	128Mbits 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	—

Table 9-50. 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	256Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes
2 Gbits	128Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	512Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes

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 Section 9.5.12, “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 $\overline{\text{MCS}}n$ 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.

9.5.2 DDR SDRAM Address Multiplexing

Table 9-51, and Table 9-52 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 modes for reads and writes, so the column address can never use MA[10].

Table 9-51. 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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33–35
16 x 10 x 3	$\overline{\text{MRAS}}$	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	$\overline{\text{MRAS}}$		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 9-51. DDR2 Address Multiplexing for 64-Bit Data Bus with Interleaving and 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	33–35
14 x 10 x 2	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	

Table 9-52. DDR2 Address Multiplexing for 32-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	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34–35
16 x 10 x 3	$\overline{\text{MRAS}}$		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	$\overline{\text{MRAS}}$			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		2	1	0											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	

Table 9-52. 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
		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	33	34–35
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$							12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	

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 9-53 illustrates examples of address decode when interleaving between two chip selects, and Table 9-54 shows examples of address decode when interleaving between four chip selects.

Table 9-53. Example of Address Multiplexing for 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	33–35
14 x 10 x 3	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$			12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Table 9-54. 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	33–35
14 x 10 x 3	$\overline{\text{MRAS}}$	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	$\overline{\text{MRAS}}$		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$			12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL														
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	

Partial Array Self Refresh (PASR) can be enabled for any chip select using the CS*n*_CONFIG_2[PASR_CFG] fields. If PASR is enabled for a given chip select, then the sub-bank and row decode is swapped, and the sub-bank is decoded as the most significant portion of the DRAM address, as shown in Table 9-55. 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 9-56.

Table 9-55. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled

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	33	34–35
16 x 10 x 3	$\overline{\text{MRAS}}$					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA		2	1	0																											
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	$\overline{\text{MRAS}}$						14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA			2	1	0																										
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	$\overline{\text{MRAS}}$							13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA				2	1	0																									
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 2	$\overline{\text{MRAS}}$							13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA					1	0																									
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	

Table 9-55. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled (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	33	34–35
13 x 10 x 3	$\overline{\text{MRAS}}$								12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA					2	1	0																								
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 10 x 2	$\overline{\text{MRAS}}$								12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA						1	0																								
	$\overline{\text{MCAS}}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9 x 2	$\overline{\text{MRAS}}$									12	11	10	9	8	7	6	5	4	3	2	1	0										
	MBA							1	0																							
	$\overline{\text{MCAS}}$																						8	7	6	5	4	3	2	1	0	

Table 9-56. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled

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	33–35
14 x 10 x3	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA		2	1	0															CS SEL											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
14 x 10 x 2	$\overline{\text{MRAS}}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA			1	0															CS SEL											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
13 x 10 x 3	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA			2	1	0														CS SEL											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		
13 x 10 x 2	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA				1	0														CS SEL											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0		

9.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR3 or DDR2 modes.

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 9-57) 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 $\overline{\text{MCAS}}$ before $\overline{\text{MRAS}}$)—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: $\overline{\text{MCAS}}$ latency, additive latency (for DDR2), write recovery (for DDR2), burst type, and burst length. $\overline{\text{MCAS}}$ latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide $\overline{\text{MCAS}}$ latency {1,2,3}, some provide $\overline{\text{MCAS}}$ latency {1,2,3,4,5}, etc.). 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 DDR3memory 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, $\overline{\text{MCAS}}$ 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 places all logical banks in a precharged state.

Table 9-57. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	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

9.5.4 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 9-58 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-58. 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}.

Table 9-58. DDR SDRAM Interface Timing Intervals (continued)

Timing Intervals	Definition
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.
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 Section 9.4.1.5, “DDR SDRAM Timing Configuration 0 (TIMING_CFG_0),” Section 9.4.1.6, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” Section 9.4.1.7, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2),” and Section 9.4.1.4, “DDR SDRAM Timing Configuration 3 (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.

Figure 9-46 through Figure 9-49 show DDR SDRAM timing for various types of accesses; see Figure 9-46 for a single-beat read operation, Figure 9-47 for a single-beat write operation, and Figure 9-49 for a burst-write operation. 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.

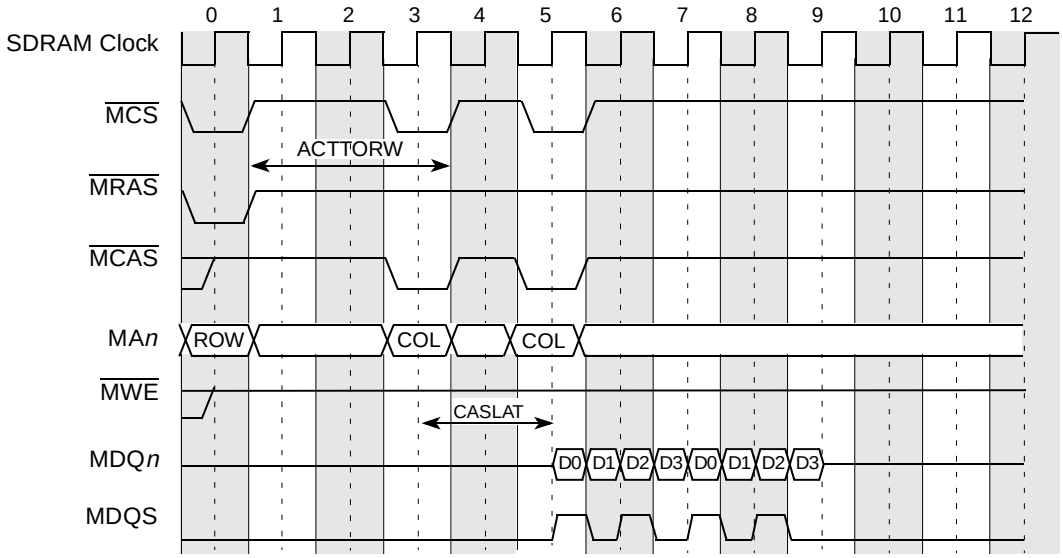


Figure 9-46. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

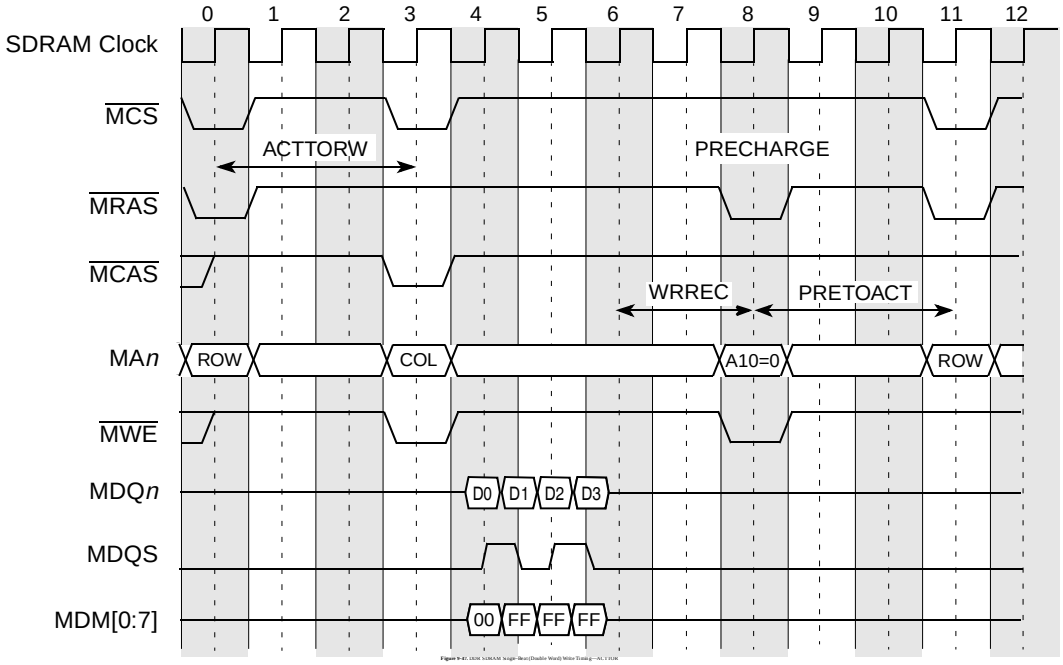


Figure 9-48. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

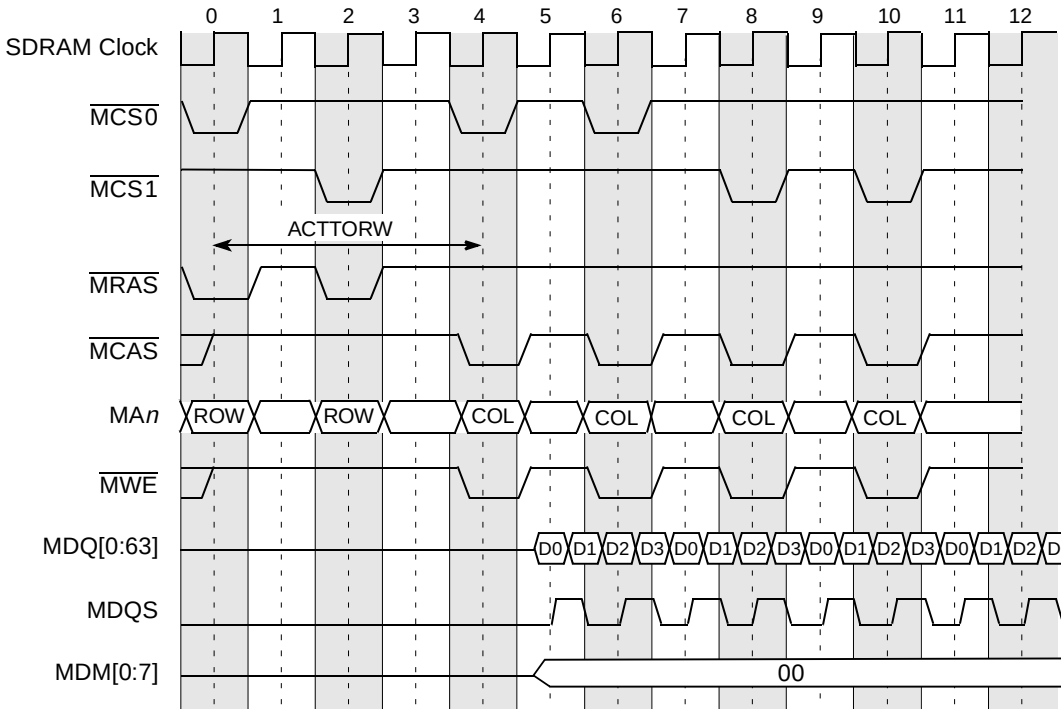


Figure 9-49. DDR SDRAM 4-Beat Burst Write Timing—ACTTORW = 4

9.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

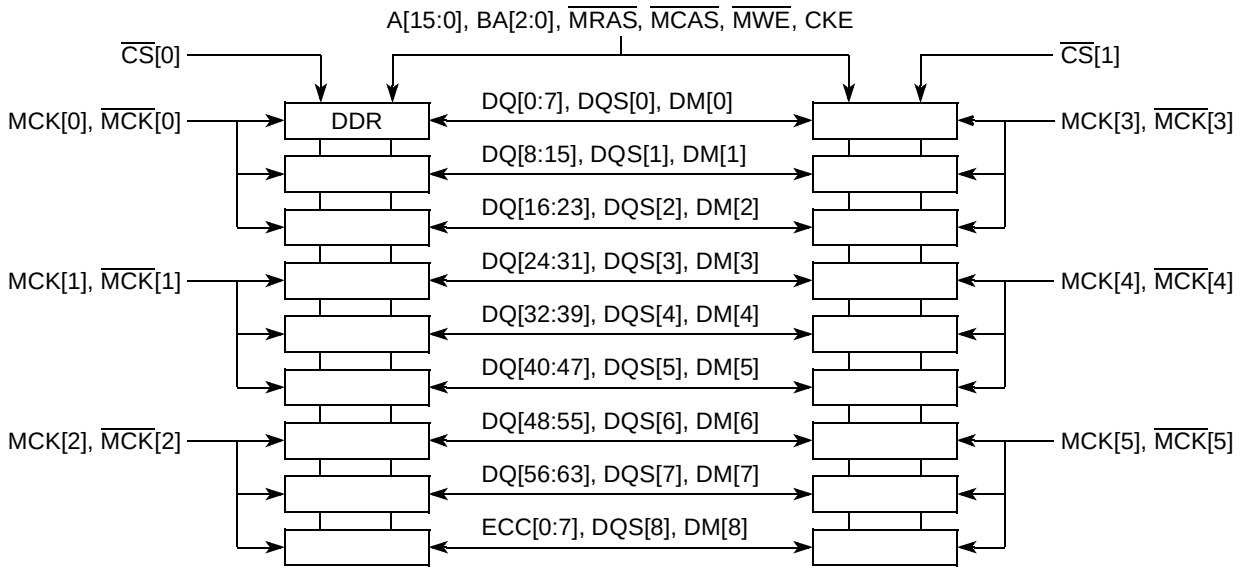


Figure 9-50. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

9.5.5 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 9-51 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.

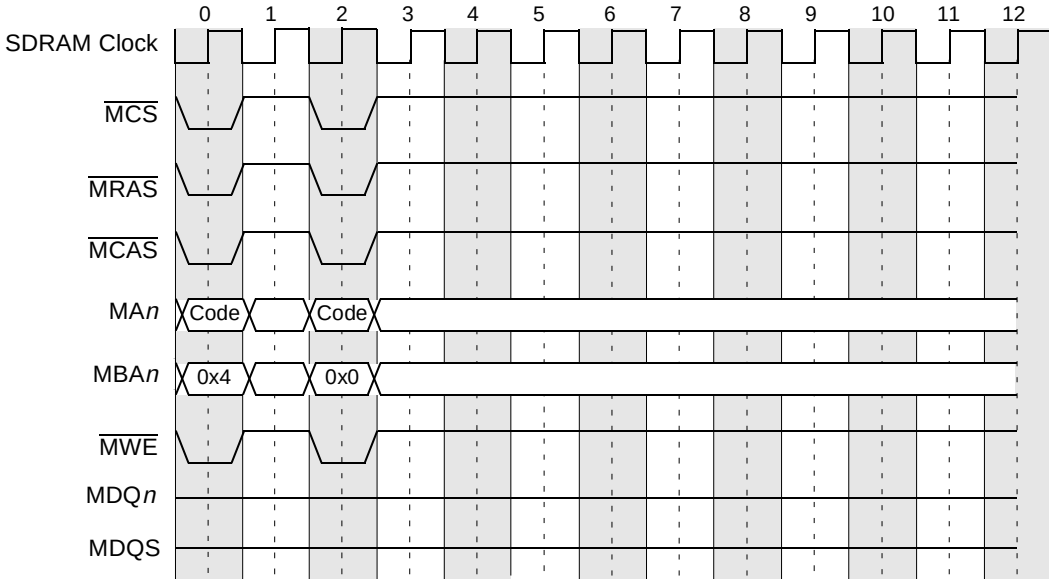


Figure 9-51. DDR SDRAM Mode-Set Command Timing

9.5.6 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.

Figure 9-52 shows the registered DDR SDRAM DIMM single-beat write timing.

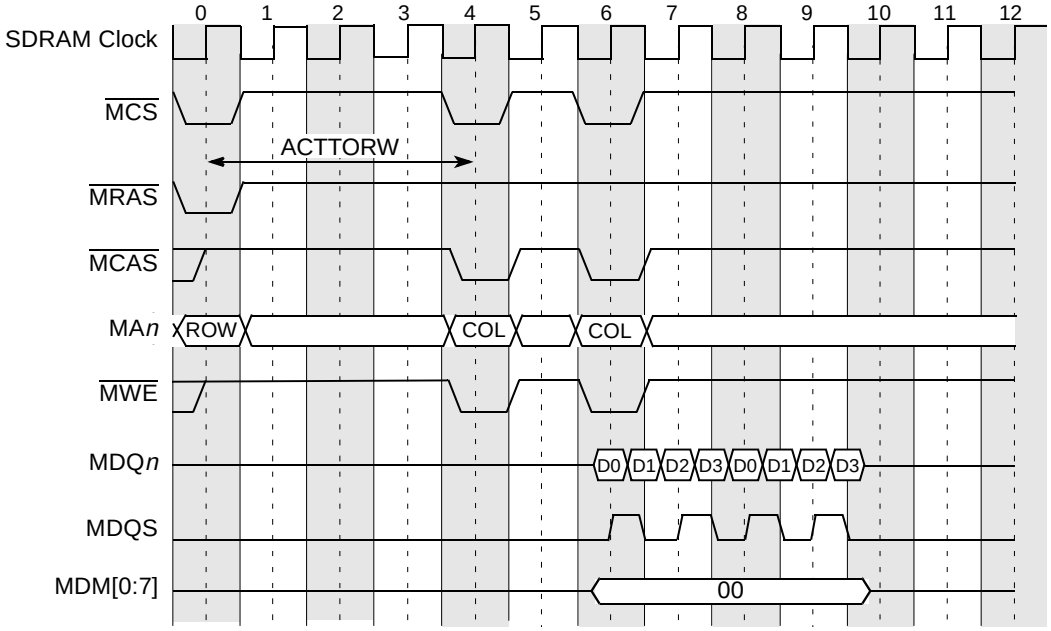


Figure 9-52. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 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 DIMM. 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 9-53 shows the use of the WR_DATA_DELAY parameter.

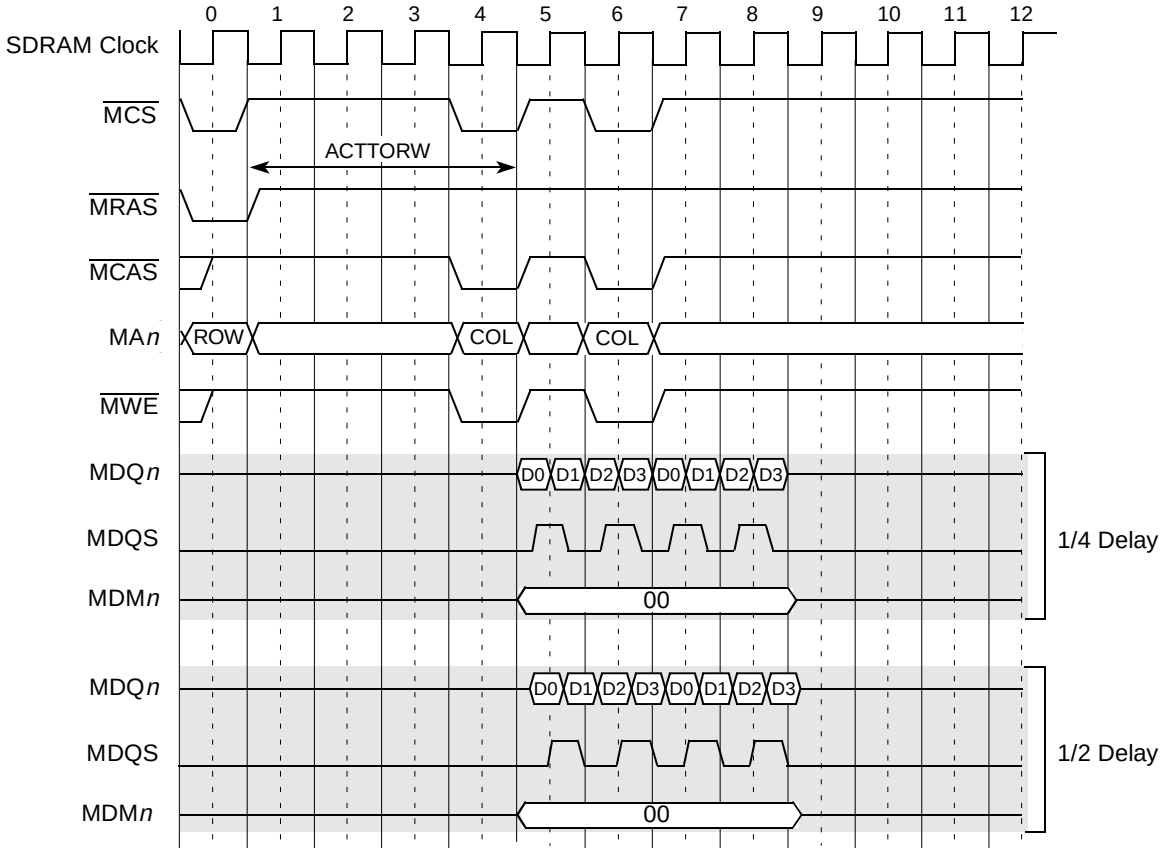


Figure 9-53. Write Timing Adjustments Example for Write Latency = 1

9.5.8 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. Three sets of auto refresh commands are issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in three 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.

9.5.8.1 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 Figure 9-54 (TIMING_CFG_1 [REFREC] = 10 in this example).

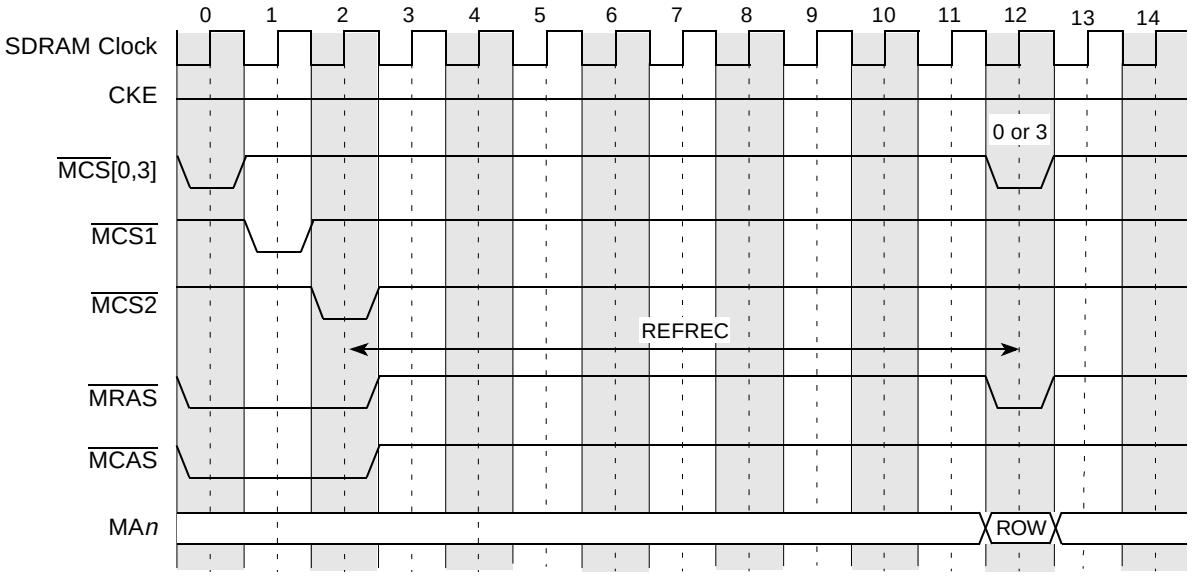


Figure 9-54. DDR SDRAM Bank Staggered Auto Refresh Timing

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.

9.5.8.2 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 9-59 summarizes the refresh types available in each power-saving mode.

Table 9-59. 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 9-55.

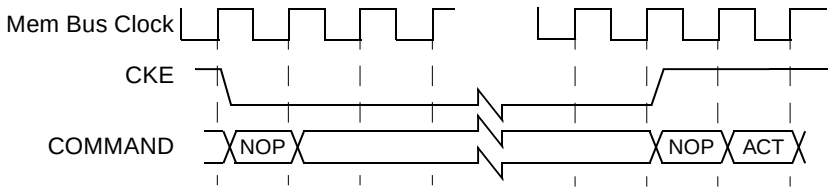


Figure 9-55. DDR SDRAM Power-Down Mode

9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-56 and Figure 9-57.

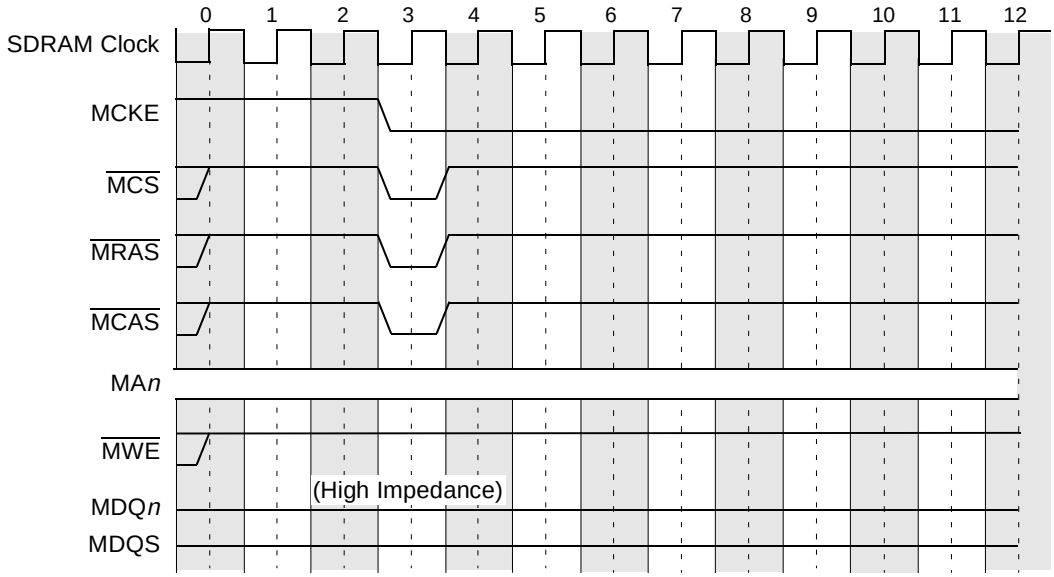


Figure 9-56. DDR SDRAM Self-Refresh Entry Timing

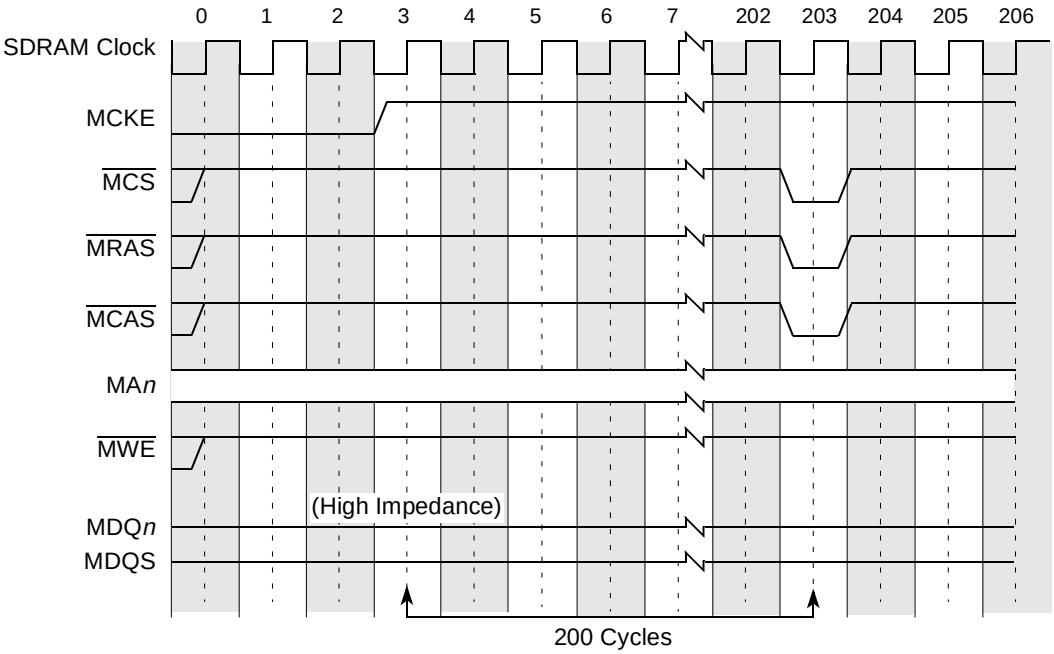


Figure 9-57. DDR SDRAM Self-Refresh Exit Timing

9.5.9 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.

Table 9-60 lists the data beat sequencing to and from the DDR SDRAM and the data queues for each of the possible transfer sizes with each of the possible starting double-word offsets. All underlined double-word offsets are valid for the transaction.

Table 9-60. Memory Controller–Data Beat Ordering

Transfer Size	Starting Double-Word Offset	Double-Word Sequence[1] to/from DRAM and Queues
1 double word	0 1 2 3	**0** - 1 - 2 - 3 **1** - 2 - 3 - 0 **2** - 3 - 0 - 1 **3** - 0 - 1 - 2
2 double words	0 1 2	**0 - 1** - 2 - 3 **1 - 2** - 3 - 0 **2 - 3** - 0 - 1
3 double words	0 1	**0 - 1 - 2** - 3 **1 - 2 - 3** - 0

[1] All underlined **Double**-word offsets are valid for the transaction. All writes are aligned to double-word 0 for DDR3 memories.

9.5.10 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 opens 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 CSn_CONFIG[AP_nEN].

9.5.11 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. Multiple-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 9-61 and Table 9-62.

Table 9-61. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•
3	•				•			•
4	•	•				•		
5	•		•			•		
6	•			•		•		
7	•				•	•		
8	•	•					•	
9	•		•				•	
10	•			•			•	
11	•				•		•	
12	•	•				•	•	•
32			•	•				•
33			•		•			•
34	•		•		•			
35		•	•		•			
36			•	•		•		
37			•		•	•		
38	•		•		•	•		•
39		•	•		•	•		•
40			•	•			•	
41			•		•		•	
42	•		•		•		•	•
43		•	•		•		•	•
44			•	•		•	•	•

Table 9-61. 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
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				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 9-62. 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								•

9.5.12 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 Section 9.4.1.36, “Memory Error Interrupt Enable (ERR_INT_EN),” Section 9.4.1.35, “Memory Error Disable (ERR_DISABLE),” and Section 9.4.1.34, “Memory Error Detect (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 a critical 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 the machine check or critical interrupt (if enabled, as described in Section 9.4.1.35, “Memory Error Disable (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 a critical interrupt (if enabled, as described in Section 9.4.1.34, “Memory Error Detect (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 9-63 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 9-63. 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.	The error is reported via machine check or critical interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

9.6 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 $\overline{\text{MCS}n}$ 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 Section 9.4.1.2, "Chip Select Configuration (CSn_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 Section 9.4.1, "Register Descriptions," for more detailed descriptions of the configuration registers. These parameters are shown in Table 9-64.

Table 9-64. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter		Section/page
CSn_BNDS	Chip select memory bounds	SAn EAn		9.4.1.1/9-13
CSn_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	9.4.1.2/9-14
CS*n*_CONFIG_2	Chip select configuration 2	PASR_CFG		9.4.1.3/9-17
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_REFREC EXT_ACTTOPRE EXT_CASLAT CNTL_ADJ		9.4.1.4/9-17
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT ODT_PD_EXIT MRS_CYC	9.4.1.5/9-19
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	9.4.1.6/9-21

Table 9-64. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/page
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE WR_DATA_DELAY CKE_PLS FOUR_ACT	9.4.1.7/9-23
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 8_BE DBW NCAP 2T_EN BA_INTLV_CTL x32_EN HSE BI	9.4.1.8/9-25
DDR_SDRAM_CFG_2	Control configuration	SR_IE DLL_RST_DIS DQS_CFG ODT_CFG NUM_PR OBC_CFG AP_EN D_INIT	9.4.1.9/9-28
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE	9.4.1.10/9-30
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	9.4.1.11/9-31
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	9.4.1.13/9-34
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	9.4.1.14/9-35
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	9.4.1.15/9-35
DDR_INIT_ADDR	Initialization address	INIT_ADDR	9.4.1.16/9-36
TIMING_CFG_4	Timing configuration	RWT WRT RRT WWT DLL_LOCK	9.4.1.18/9-37
TIMING_CFG_5	Timing configuration	RODT_ON RODT_OFF WODT_ON WODT_OFF	9.4.1.19/9-39
DDR_ZQ_CNTL	ZQ calibration control	ZQ_EN ZQINIT ZQOPER ZQCS	9.4.1.20/9-40
DDR_WRLVL_CNTL	Write leveling control	WRLVL_EN WRLVL_MRD WRLVL_ODTEN WRLVL_DQSEN WRLVL_SMPL WRLVL_WLR WRLVL_START	9.4.1.21/9-42

Table 9-64. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/page
DDRCDR_1	Driver control	DHC_EN ODT DSO_C_EN DSO_D_EN DSO_CPZ DSO_CNZ DSO_DPZ DSO_DNZ	9.4.1.24/9-46
DDRCDR_2	Driver control	DSO_CLK_EN DSO_CLKPZ DSO_CLKNZ	9.4.1.25/9-49
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR	9.4.1.17/9-36

9.6.1 Programming Differences Between Memory Types

Depending on the memory type used, certain fields must be programmed differently. Table 9-65 illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 9-65. Programming Differences Between Memory Types

Parameter	Description	Differences		Section/page
AP*n*_EN	Chip Select *n* Auto Precharge Enable	DDR2	Can be used to place chip select *n* in auto precharge mode	9.4.1.2/9-14
		DDR3	Can be used to place chip select *n* 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.	9.4.1.2/9-14
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select typically does 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).	9.4.1.2/9-14
		DDR3	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT typically is set to assert for the chip select that is getting written to (value would be set to 001).	

Table 9-65. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
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}.	9.4.1.5/9-19
		DDR3	Should be set to 0001 for DDR3. The powerdown times (t_{XP} and t_{XPDLL}) 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})	9.4.1.6/9-21
		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})	9.4.1.6/9-21
		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})	9.4.1.6/9-21
		DDR3	Should be set according to the specifications for the memory used (t_{RCD})	
CASLAT	CAS Latency	DDR2	Should be set, along with the Extended CAS Latency, to the desired CAS latency	9.4.1.6/9-21
		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})	9.4.1.6/9-21
		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})	9.4.1.6/9-21
		DDR3	Should be set according to the specifications for the memory used (t_{WR}). If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{WR} + 2 DRAM cycles.	
ACTTOACT	Activate *A* to Activate *B*	DDR2	Should be set according to the specifications for the memory used (t_{RRD})	9.4.1.6/9-21
		DDR3	Should be set according to the specifications for the memory used ($t_{RRD)}$	

Table 9-65. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
WRTORD	Write to Read Timing	DDR2	Should be set according to the specifications for the memory used (t_{WTR})	9.4.1.6/9-21
		DDR3	Should be set according to the specifications for the memory used (t_{WTR}) If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{WTR} + 2 DRAM cycles.	
ADD_LAT	Additive Latency	DDR2	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	9.4.1.7/9-23
		DDR3	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	DDR2	Should be set to CAS latency – 1 cycle. For example, if the CAS latency if 5 cycles, then this field should be set to 100 (4 cycles).	9.4.1.7/9-23
		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.	
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.	9.4.1.7/9-23
		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. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{RTP} + 2 DRAM cycles.	
CKE_PLS	Minimum CKE Pulse Width	DDR2	Should be set according to the specifications for the memory used (t_{CKE})	9.4.1.7/9-23
		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.	9.4.1.7/9-23
		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	9.4.1.8/9-25
		DDR3	If registered DIMMs are used, then this field should be set to 1	

Table 9-65. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
8_BE	8-beat burst enable	DDR2	Should be set to 0	9.4.1.8/9-25
		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 is needed to ensure t_{CCD} 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.	9.4.1.8/9-25
		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.	9.4.1.9/9-28
		DDR3	Should be set to 1	
DQS_CFG	DQS Configuration	DDR2	Can be set to either 00 or 01, depending on if differential strobes are used	9.4.1.9/9-28
		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.	9.4.1.9/9-28
		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.	
OBC_CFG	On-The-Fly Burst Chop Configuration	DDR2	Should be set to 0	9.4.1.9/9-28
		LPDDR	Should be set to 0	
		DDR3	Can be set to 1 if on-the-fly burst chop is used. This is expected to give the best performance in DDR3 mode. This feature can only be used if a 64-bit data bus is used.	
RWT	Read-to-write turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	9.4.1.18/9-37
		LPDDR	Should typically be set to 0000	
		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.	

Table 9-65. Programming Differences Between Memory Types (continued)

Parameter	Description	Differences		Section/page
WRT	Write-to-read turnaround for same chip select (in TIMING_CFG_4)	DDR2	Should typically be set to 0000	9.4.1.18/9-37
		LPDDR	Should typically be set to 0000	
		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] is 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	9.4.1.18/9-37
		LPDDR	Should typically be set to 0000	
		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	9.4.1.18/9-37
		LPDDR	Should typically be set to 0000	
		DDR3	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	
ZQ_EN	ZQ Calibration Enable	DDR2	Should be set to 0	9.4.1.20/9-40
		LPDDR	Should be set to 0	
		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	9.4.1.21/9-42
		LPDDR	Should be set to 0	
		DDR3	Can be set to 1 if write leveling is desired. Otherwise the value used in TIMING_CFG_2[WR_DATA_DELAY] is used to shift all bytes during writes to DRAM. If write leveling is used, all other fields in DDR_WRLVL_CNTL should be programmed appropriately based on the DRAM specifications.	
BSTOPR	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.	9.4.1.13/9-34
		DDR3	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

9.6.2 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 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 conducts an automatic initialization sequence to the memory, which follows the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

9.6.3 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.

9.6.3.1 Hardware Based Self-Refresh

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 this device's programmable interrupt controller (PIC) to the IRQ_OUT signal. Note that the IRQ_OUT signal must remain high until power is removed.

If DDR_SDRAM_CFG_2[SR_IE] is set, the IRQ_OUT signal from the interrupt controller is then automatically detected by the DDR controller, which immediately causes main memory to enter self-refresh mode. See Section 9.4.1.9, "DDR SDRAM Control Configuration 2 (DDR_SDRAM_CFG_2)," for further information on this bit.

These fields in the appropriate registers in the PIC must be set for self refresh to function:

- EIVPR*n*[PRIORITY] should be set to 0xF (highest priority)
- EIDR*n*[EP] should be set in order to route the incoming signal to IRQ_OUT

See Section 10.3.7.1, "External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)," and Section 10.3.7.2, "External Interrupt Destination Registers (EIDR0–EIDR11)," for descriptions of these registers.

Note that this application precludes any other simultaneous use of IRQ_OUT.

9.6.3.2 Software Based Self-Refresh

The DDR controller also has a software-programmable bit, DDR_SDRAM_CFG_2[FRC_SR], that immediately puts main memory into self-refresh mode. See Section 9.4.1.9, "DDR SDRAM Control Configuration 2 (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 has time to set this bit.

9.6.3.3 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 Section 9.4.1.8, “DDR SDRAM Control Configuration (DDR_SDRAM_CFG),” for information on this bit and Section 9.4.1.16, “DDR Initialization Address (DDR_INIT_ADDR),” for a discussion of avoiding possible ECC errors in this mode.

Note that the DDR controller automatically waits 200 DRAM cycles before issuing any command after the assertion of MCKE[0:3] when this mode is used.

Chapter 10
Programmable Interrupt Controller (PIC)

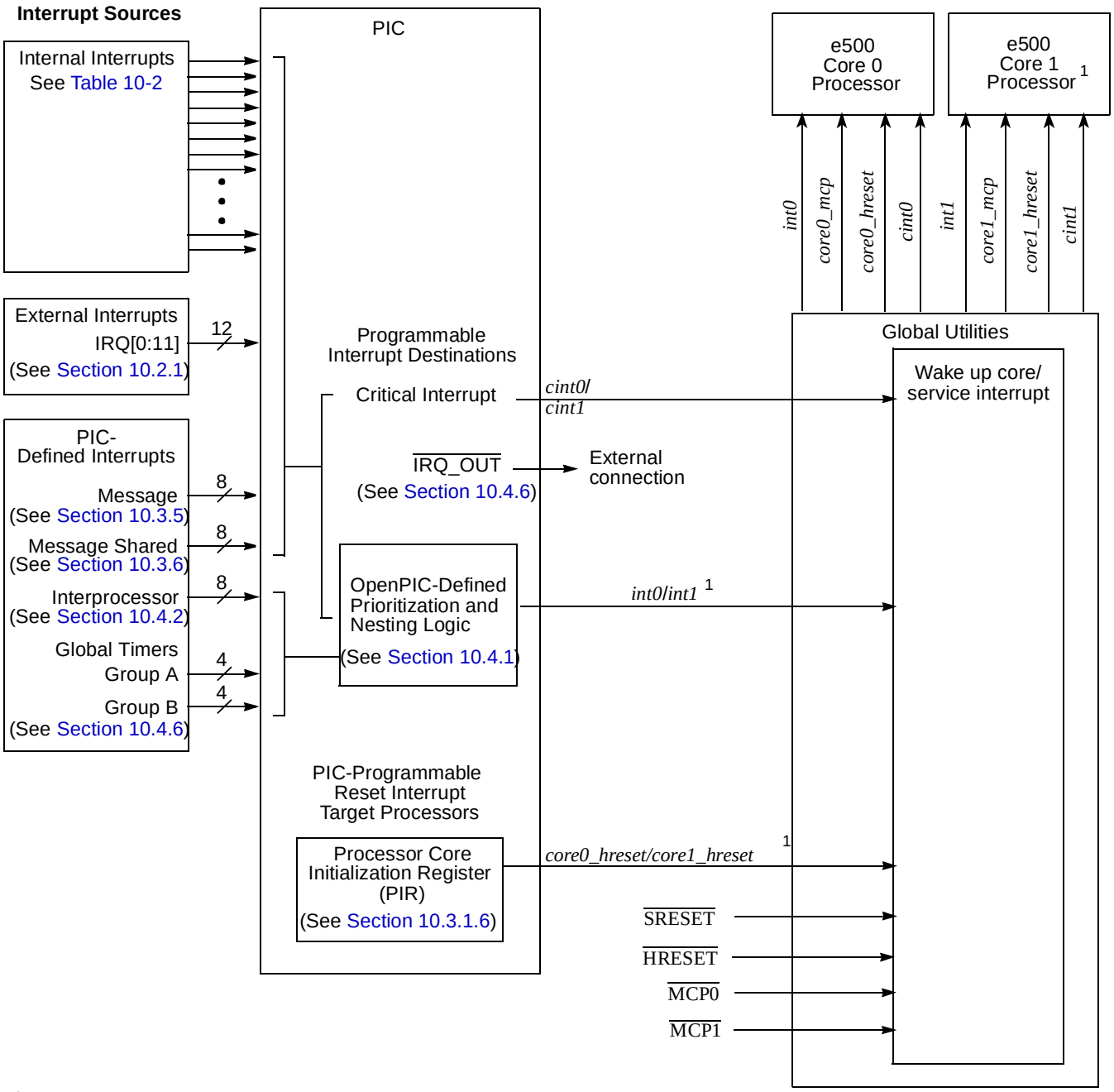
This chapter describes the programmable interrupt controller (PIC) interrupt protocol, various types of interrupt sources controlled by the PIC, and the PIC registers with some programming guidelines.

10.1 Introduction

The PIC conforms to the OpenPIC architecture. The interrupt controller provides multiprocessor interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to a CPU for servicing.

10.1.1 Overview

Figure 10-1 is a block diagram showing the relationship of the various functional blocks and how the signals external to the PIC are connected to other blocks on the device, including the cores.



1 Not supported in single-processor implementations.

Figure 10-1. Interrupt Sources Block Diagram Features

The PIC has the following features:

- Support for the following interrupt sources:
- External—Off-chip signals, IRQ[0:11]
- Internal interrupt sources; see Table 10-3. These are on-chip sources from peripheral logic within the integrated device signalling error conditions that need to be addressed by software.
- Interrupts generated from within the PIC itself, which are as follows:
  - Global timers A and B internal to the PIC
  - Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt in single-core devices.)
  - Message registers—From within the PIC. Triggered on register write, cleared on read. Used for interprocessor communication.
  - Shared message signaled registers—From within the PIC. Triggered on register write, cleared on read. Used for cross-program communication.
    - Eight 32-bit message interrupt channels.
    - Two groups of four global 32-bit timers clocked with the CCB clock or the RTC input. Timers within each group can be concatenated to time longer durations.
- Three types of programmable interrupt outputs:
  - External interrupt (*int0* and *int1*). Any of the PIC interrupt sources can be programmed to direct interrupt requests to *int0* and *int1*. Handling of such interrupt requests follows the OpenPIC specification, which guarantees that the highest priority interrupt supersedes lower priority interrupts. Section 10.4.1.2, “Interrupts Routed to *int*,” describes how the PIC logic handles these interrupts.
  - Critical interrupt (*cint0* and *cint1*). Connected to the respective core’s critical interrupt input.
  - $\overline{\text{IRQ\_OUT}}$.

    Section 10.4.1.1, “Interrupts Routed to *cint* or IRQ_OUT,” describes how the PIC logic supports this interrupt.
- Programming model compliant with the OpenPIC architecture.
  - Message, interprocessor and global timer interrupts. (Note that the interprocessor and global timer interrupts can only be routed to *int*.)
  - The following OpenPIC-defined features support only interrupts routed to the *int* signal:
    - 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 two processors.
  - Interrupts can be routed to processor core 0 or 1
  - Multi-cast delivery mode for interprocessor and global timer interrupts allowing these interrupts to be routed to either core 0 or 1, or both cores.
- Processor core initialization control

- Programmable resetting of the PIC 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.
- Pass-through mode (PIC disabled) in which the PIC directs interrupts off-chip for external servicing. See Section 10.1.4.2, “Pass-Through Mode (GCR[M] = 0).”

10.1.2 The PIC in Multiple-Processor Implementations

In a multiprocessor implementation, the PIC is replicated for each core, and where necessary, the duplicated resources are indicated with a 0 or a 1. For example, *int0* identifies the *int* signal to processor 0 and *int1* identifies the *int* signal to processor 1. Other resources associated with the cores are identified by the number. For example, setting PIR[P1] triggers a reset of core 1, and setting PIR[P0] triggers a reset of core 0.

However, where the distinction is not necessary, none is made and such resources are referred to generically. For example, the *int* signal refers to the behavior of either *int0* or *int1*.

In a multiprocessor system, it is possible for the PIC to direct interrupts to the other processor. This functionality is supported by certain multiprocessor aspects to the programming model. For example, an interrupt source in processor 0 can be programmed to target *int1* by setting the P1 bit in its destination register, *x*IDR*n*.

Note that although they are part of the programming model, such resources are reserved in a single-processor device.

10.1.3 Interrupts to the Processor Core

The external interrupt signal, *int*, is the main interrupt output from the PIC to the processor core.

The interrupt sources can also specify the critical interrupt output, c*int0* or c*int1*, if the corresponding *x*IDR*n*[CI0] or *x*IDR*n*[CI1] is set.

The PIC also defines the PIR, described in Section 10.3.1.6, “Processor Core Initialization Register (PIR),” which can be used to reset the core. Processor core interrupts generated by the PIC are described in Table 10-1.

Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources

Core Interrupt Type	Signaled by (Input to Core)	Sources
PIC-Programmable Interrupts		
External interrupt	$\overline{int}$	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
Critical interrupt	$\overline{cint}$	Generated by the PIC, as described in Section 10.1.5, “Interrupt Sources.”
Other Interrupts Generated Outside the Core		

Table 10-1. Processor Interrupts Generated Outside the Core—Types and Sources (continued)

Core Interrupt Type	Signaled by (Input to Core)	Sources
Machine check	*coren_mcp*	• $\overline{\text{MCP}}$ • $\overline{\text{SRESET}}$ • Assertion of *core_mcp* by global utilities block
Unconditional debug event	*coren_ude*	$\overline{\text{UDE}}$. Asserting $\overline{\text{UDE}}$ generates an unconditional debug exception type debug interrupt and sets a bit in the debug status register, DBSR[UDE], as described in Section 6.13.2, “Debug Status Register (DBSR).”
Reset	*coren_hreset*	• $\overline{\text{HRESET}}$ assertion (and negation) • *core_hreset_req*. Output from core—caused by writing to the core DBCR0[RST]. This condition is additionally qualified with MSR[DE] and DBCR0[IDM] bits. Note that assertion of this signal causes a hard reset of the core only. *core_hreset_req* can also be caused by a second timer timeout condition as described in Section 10.3.2.6, “Timer Control Registers (TCRA–TCRB).” • *core_reset*. Output from PIC. See Section 10.3.1.6, “Processor Core Initialization Register (PIR).”

10.1.4 Modes of Operation

Mixed or pass-through mode of operation is chosen by setting or clearing GCR[M], as described in Section 10.3.1.4, “Global Configuration Register (GCR).”

10.1.4.1 Mixed Mode (GCR[M] = 1)

In mixed mode, external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in Section 10.4.1, “Flow of Interrupt Control.”

10.1.4.2 Pass-Through Mode (GCR[M] = 0)

The PIC provides a mechanism to support alternate external interrupt controllers such as the PC/AT-compatible 8259 interrupt controller architecture. After a hard reset, the PIC defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to core 0 as shown in Figure 10-2; 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 PIC does not perform a vector fetch from an 8259 interrupt controller.

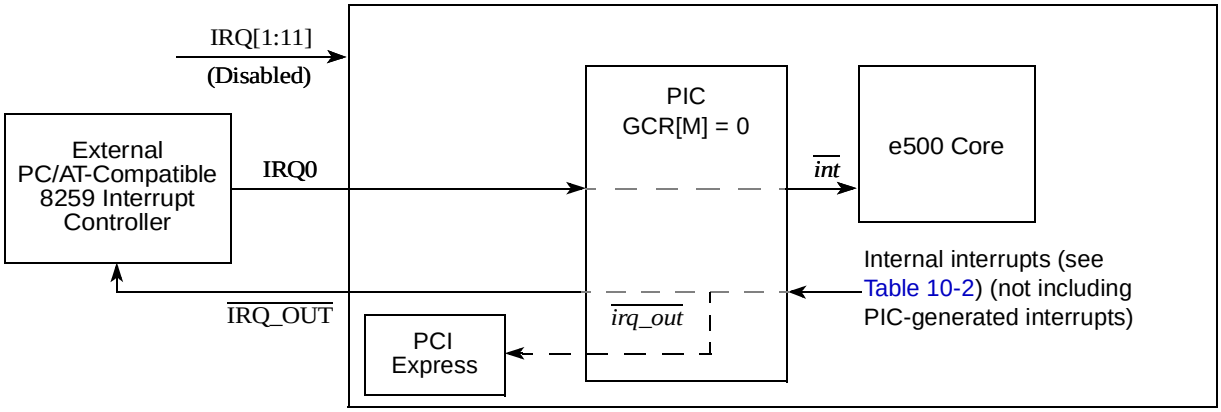


Figure 10-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts shown in Table 10-3 are not forwarded to core 0. Instead, the PIC passes the raw interrupts from the internal sources to $\overline{\text{IRQ\_OUT}}$. Note that when the PCI Express controller is configured as an endpoint (EP) device, the $\overline{irq\_out}$ signal from the PIC may used to automatically generate an outbound PCI Express MSI transaction toward the remote interrupt controller resource on the root complex (RC). See Section 21.4.2.1.2, "Hardware MSI Generation."

Note that in pass-through mode, interrupts generated within the PIC (global timers, interprocessor, and message register interrupts) are disabled. If internal or PIC-generated interrupts must be reported internally to the processor, mixed mode must be used.

10.1.5 Interrupt Sources

The PIC can receive separate interrupts from the following sources:

- External—Off-chip signals, IRQ[0:11]
- Internal—On-chip sources from peripheral logic within the integrated device. See Table 10-2.
- Global timers A and B internal to the PIC
- Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt in single-core implementations.)
- Message registers—From within the PIC. Triggered on register write, cleared on read. Used for interprocessor communication.
- Shared message signaled registers—From within the PIC. Triggered on register write, cleared on read. Used for cross-program communication.

10.1.5.1 Interrupt Routing—Mixed Mode

When an interrupt request is delivered to the PIC, the corresponding interrupt destination register is checked to determine where the request should be routed, as follows:

- If *x*IDR*n*[EP] = 1 (and all other destination bits are zero), the interrupt is routed off-chip to the external $\overline{\text{IRQ\_OUT}}$ signal. Or if the PCI Express controller is in EP mode and automatically generates a PCI Express MSI transaction. See Section 21.4.2.1.2, "Hardware MSI Generation."

- If either, but not both, *x*IDR*n*[CI0] or *x*IDR*n*[CI1] is set (and all other destination bits are zero), the interrupt is routed to *cint0* or *cint1*.
- If *x*IDR*n*[P0] is set (and all other destination bits are zero) the interrupt is routed to *int0*. Setting *x*IDR*n*[P1] likewise routs the interrupt to *int1* In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow is as described in Section 10.4.1, "Flow of Interrupt Control." Note that multicasting interrupts (global timer and interprocessor interrupts) can set both P0 and P1; other interrupt sources cannot.

10.1.5.2 Interrupt Destinations

Following its reset (by default), the PIC directs all timer, shared message signaled, and interrupts from external and internal sources to *int* output (connected to the *int* signal of the processor core).

Interprocessor and global timers interrupts can be programmed to be routed to either core's *int* signal or to both cores (multi-casting).

All other interrupts have more destination options, but only one destination can be chosen for a single non–multi-casting interrupt. Instead of being routed to *int*, these interrupts can be routed to the core through $\overline{\text{IRQ\_OUT}}$ or *cint*. These options are selected by writing to the EP or CI fields in the appropriate destination register.

10.1.5.3 Internal Interrupt Sources

Table 10-2 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 10-2. Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source
0	L2 cache
1	ECM
2	DDR DRAM controllers 1 and 2
3	LBC controller
4	DMA channel 1
5	DMA channel 2
6	DMA channel 3
7	DMA channel 4
8	PCI Express Port3
9	PCI Express Port2
10	PCI Express Port1
12	Unused
13	eTSEC 1 transmit
14	eTSEC 1 receive
15	eTSEC 3 transmit

Table 10-2. Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
16	eTSEC 3 receive
17	eTSEC 3 error
18	eTSEC 1 error
19	eTSEC 2 transmit
20	eTSEC 2 receive
21	eTSEC 4 transmit
22	eTSEC 4 receive
23	eTSEC 4 error
24	eTSEC 2 error
25	Fast Ethernet controller (FEC)
26	DUART
27	I^2C controllers
28	Performance monitor
29	Security interrupt 1
31	GPIO interrupt
32	SRIO error/write-port unit
33	SRIO outbound doorbell
34	SRIO inbound doorbell
37	SRIO outbound message unit1
38	SRIO inbound message unit 1
39	SRIO outbound message unit 2
40	SRIO inbound message unit 2
41	PME general
42	Security interrupt 2
45	TLU1
48	PME channel 1
49	PME channel 2
50	PME channel 3
51	PME channel 4
52	IEEE Std 1588™ TSEC1
53	IEEE 1588 TSEC 2
54	IEEE 1588 TSEC 3
55	IEEE 1588 TSEC 4
59	TLU2
60	DMA 2 channel 1
61	DMA 2 channel 2

Table 10-2. Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
62	DMA 2 channel 3
63	DMA 2 channel 4

10.2 External Signal Descriptions

The following sections describe the PIC signals.

10.2.1 Signal Overview

The PIC external interface signals are described in Table 10-3. There are 12 distinct external interrupt request input signals and 1 interrupt request output signal $\overline{\text{IRQ\_OUT}}$. As Table 10-3 shows, the IRQ inputs are also used for delivering INTx signals for the PCI Express root complexes.

10.2.2 Detailed Signal Descriptions

Table 10-3 provides detailed descriptions of the external PIC signals.

Table 10-3. Interrupt Signals—Detailed Signal Descriptions

Signal	I/O	Description	
IRQ[0:11]	I	Interrupt request 0–11. The polarity and sense of each of these signals is programmable. All of these inputs can be driven asynchronously. **Note:** Some interrupt request signals IRQ*n* may share PIC external interrupt registers with PCI Express INTx signaling. See Table 10.4.5.	
		State Meaning	Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the PIC 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 *Hardware Specifications*.
$\overline{\text{IRQ\_OUT}}$	O	Interrupt request out. When the PIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See Section 10.1.4, “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 $\overline{\text{IRQ\_OUT}}$.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{IRQ\_OUT}}$ occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 CCB clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Message interrupts: 2 cycles after write to message register. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 CCB clock cycles External interrupt: 4 cycles. Message interrupts: 2 cycles after message register cleared.

Table 10-3. Interrupt Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{MCP\_0}}$ and $\overline{\text{MCP\_1}}$	I	Machine check processor *n*. Assertion causes a machine check interrupt to the corresponding core. Note that if the core is not configured to process machine check interrupts (MSR[ME] = 0), assertion of $\overline{\text{MCP}n}$ causes a core checkstop condition. Note that internal sources for the internal *core_mcp n* can also cause a machine check interrupt to the processor core as described in Section 23.4.1.11, "Machine Check Summary Register (MCPSUMR)," Table 10-1 and Table 10-7.	
		State Meaning	Asserted—Integrated logic should direct the corresponding core to take a machine check interrupt or enter the checkstop state as directed by the MSR. Negated—Machine check handling is not being requested by the external system.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Because $\overline{\text{MCP}n}$ is edge-triggered, it can be negated one clock after its assertion.

10.3 Memory Map/Register Definition

The PIC programmable register map occupies 256 Kbytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect. All PIC registers are 32 bits wide and, although located on 128-bit address boundaries, should be accessed only as 32-bit quantities.

The PIC address offset map, shown in Table 10-4, is divided into three areas:

- 0x*nn*4_0000–0x*nn*4_FFF0—Global registers
- 0x*nn*5_0000–0x*nn*5_FFF0—Interrupt source configuration registers
- 0x*nn*6_0000–0x*nn*7_FFF0—Per-CPU registers

Table 10-4. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers—Block Base Address: 0x4_0000				
0x0000	BRR1—Block revision register 1	R	0x0040_0301	10.3.1.1/10-20
0x0010	BRR2—Block revision register 2	R	0x0000_0001	10.3.1.2/10-21
0x0020–0x0030	Reserved	—	—	—
0x0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.8.1/10-50
0x0050	IPIDR1—IPI 1 dispatch register			
0x0060	IPIDR2—IPI 2 dispatch register			
0x0070	IPIDR3—IPI 3 dispatch register			
0x0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.8.2/10-51
0x0090	WHOAMI—Who am I register	R	—	10.3.8.3/10-52
0x00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.8.4/10-52
0x00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.8.5/10-53
0x00C0–0x0FF0	Reserved	—	—	—
0x1000	FRR—Feature reporting register	R	0x006B_0*n*020 x0067_0002	10.3.1.3/10-21

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1010	Reserved	—	—	—
0x1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.4/10-22
0x1030	Reserved	—	—	—
0x1040–0x1070	Vendor reserved	—	—	—
0x1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.5/10-23
0x1090	PIR—Processor core initialization register	R/W	0x0000_0000	10.3.1.6/10-23
0x10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.7/10-24
0x10B0	IPIVPR1—IPI 1 vector/priority register			
0x10C0	IPIVPR2—IPI 2 vector/priority register			
0x10D0	IPIVPR3—IPI 3 vector/priority register			
0x10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.8/10-25
Global Timer Group A Registers				
0x10F0	TFRRA—Timer frequency reporting register (Group A)	R/W	0x0000_0000	10.3.2.1/10-25
0x1100	GTCCRA0—Global timer 0 current count register (Group A)	R	0x0000_0000	10.3.2.2/10-26
0x1110	GTBCRA0—Global timer 0 base count register (Group A)	R/W	0x8000_0000	10.3.2.3/10-26
0x1120	GTVPRA0—Global timer 0 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2.4/10-27
0x1130	GTDRA0—Global timer 0 destination register (Group A)	R/W	0x0000_0001	10.3.2.5/10-28
0x1140	GTCCRA1—Global timer 1 current count register (Group A)	R	0x0000_0000	10.3.2.2/10-26
0x1150	GTBCRA1—Global timer 1 base count register (Group A)	R/W	0x8000_0000	10.3.2.3/10-26
0x1160	GTVPRA1—Global timer 1 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2.4/10-27
0x1170	GTDRA1—Global timer 1 destination register (Group A)	R/W	0x0000_0001	10.3.2.5/10-28
0x1180	GTCCRA2—Global timer 2 current count register (Group A)	R	0x0000_0000	10.3.2.2/10-26
0x1190	GTBCRA2—Global timer 2 base count register (Group A)	R/W	0x8000_0000	10.3.2.3/10-26
0x11A0	GTVPRA2—Global timer 2 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2.4/10-27
0x11B0	GTDRA2—Global timer 2 destination register (Group A)	R/W	0x0000_0001	10.3.2.5/10-28
0x11C0	GTCCRA3—Global timer 3 current count register (Group A)	R	0x0000_0000	10.3.2.2/10-26
0x11D0	GTBCRA3—Global timer 3 base count register (Group A)	R/W	0x8000_0000	10.3.2.3/10-26
0x11E0	GTVPRA3—Global timer 3 vector/priority register (Group A)	R/W	0x8000_0000	10.3.2.4/10-27
0x11F0	GTDRA3—Global timer 3 destination register (Group A)	R/W	0x0000_0001	10.3.2.5/10-28
0x1200–0x12F0	Reserved	—	—	—
0x1300	TCRA—Timer control register (Group A)	R/W	0x0000_0000	10.3.2.6/10-28
0x1308	ERQSR—External interrupt summary register	R	0x0000_0000	10.3.3.1/10-31
0x1310	IRQSR0—IRQ_OUT summary register 0	R	0x0000_0000	10.3.3.2/10-31
0x1320	IRQSR1—IRQ_OUT summary register 1	R	0x0000_0000	10.3.3.3/10-32
0x1324	IRQSR2—IRQ_OUT summary register 2	R	0x0000_0000	10.3.3.4/10-33
0x1330	CISR0—Critical interrupt summary register 0	R	0x0000_0000	10.3.3.5/10-33

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1340	CISR1—Critical interrupt summary register 1	R	0x0000_0000	10.3.3.6/10-34
0x1344	CISR2—Critical interrupt summary register 2	R	0x0000_0000	10.3.3.7/10-34
0x1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0xFFFF_FFFF	10.3.4.1/10-35
0x1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x1364	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0xFFFF_FFFF	10.3.4.1/10-35
0x1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x1384	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0xFFFF_FFFF	10.3.4.1/10-35
0x13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x13A4	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0xFFFF_FFFF	10.3.4.1/10-35
0x13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x13C4	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_FFFF	10.3.4.2/10-36
0x13D0–0x13F0	Reserved	—	—	—
0x1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-37
0x1410	MSGR1—Message register 1	R/W	0x0000_0000	10.3.5.1/10-37
0x1420	MSGR2—Message register 2	R/W	0x0000_0000	10.3.5.1/10-37
0x1430	MSGR3—Message register 3	R/W	0x0000_0000	10.3.5.1/10-37
0x1440–0x14F0	Reserved	—	—	—
0x1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-37
0x1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-38
0x1520–0x15F0	Reserved	—	—	—
0x1600	MSIR0—Shared message signaled interrupt register 0	RC	0x0000_0000	10.3.6.1/10-39
0x1610	MSIR1—Shared message signaled interrupt register 1	RC	0x0000_0000	10.3.6.1/10-39
0x1620	MSIR2—Shared message signaled interrupt register 2	RC	0x0000_0000	10.3.6.1/10-39
0x1630	MSIR3—Shared message signaled interrupt register 3	RC	0x0000_0000	10.3.6.1/10-39
0x1640	MSIR4—Shared message signaled interrupt register 4	RC	0x0000_0000	10.3.6.1/10-39
0x1650	MSIR5—Shared message signaled interrupt register 5	RC	0x0000_0000	10.3.6.1/10-39
0x1660	MSIR6—Shared message signaled interrupt register 6	RC	0x0000_0000	10.3.6.1/10-39
0x1670	MSIR7—Shared message signaled interrupt register 7	RC	0x0000_0000	10.3.6.1/10-39
0x1680–0x1700	Reserved	—	—	—
0x1720	MSISR—Shared message signaled interrupt status register	R	0x0000_0000	10.3.6.2/10-39
0x1740	MSIIR—Shared message signaled interrupt index register	W	0x0000_0000	10.3.6.3/10-40

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1750–0x20E0	Reserved	—	—	—
Global Timer Group B Registers				
0x20F0	TFRRB—Timer frequency reporting register group B	R/W	0x0000_0000	10.3.2.1/10-25
0x2100	GTCCRB0—Global timer current count register group B 0	R	0x0000_0000	10.3.2.2/10-26
0x2110	GTBCRB0—Global timer base count register group B 0	R/W	0x8000_0000	10.3.2.3/10-26
0x2120	GTVPRB0—Global timer vector/priority register group B 0	R/W	0x8000_0000	10.3.2.4/10-27
0x2130	GTDRB0—Global timer destination register group B 0	R/W	0x0000_0001	10.3.2.5/10-28
0x2140	GTCCRB1—Global timer current count register group B 1	R	0x0000_0000	10.3.2.2/10-26
0x2150	GTBCRB1—Global timer base count register group B 1	R/W	0x8000_0000	10.3.2.3/10-26
0x2160	GTVPRB1—Global timer vector/priority register group B 1	R/W	0x8000_0000	10.3.2.4/10-27
0x2170	GTDRB1—Global timer destination register group B 1	R/W	0x0000_0001	10.3.2.5/10-28
0x2180	GTCCRB2—Global timer current count register group B 2	R	0x0000_0000	10.3.2.2/10-26
0x2190	GTBCRB2—Global timer base count register group B 2	R/W	0x8000_0000	10.3.2.3/10-26
0x21A0	GTVPRB2—Global timer vector/priority register group B 2	R/W	0x8000_0000	10.3.2.4/10-27
0x21B0	GTDRB2—Global timer destination register group B 2	R/W	0x0000_0001	10.3.2.5/10-28
0x21C0	GTCCRB3—Global timer current count register group B 3	R	0x0000_0000	10.3.2.2/10-26
0x21D0	GTBCRB3—Global timer base count register group B 3	R/W	0x8000_0000	10.3.2.3/10-26
0x21E0	GTVPRB3—Global timer vector/priority register group B 3	R/W	0x8000_0000	10.3.2.4/10-27
0x21F0	GTDRB3—Global timer destination register group B 3	R/W	0x0000_0001	10.3.2.5/10-28
0x2200–0x22F0	Reserved	—	—	—
0x2300	TCRB—Timer control register (Group B)	R/W	0x0000_0000	10.3.2.6/10-28
0x2310–0x23F0	Reserved	—	—	—
0x2400	MSGR4—Message register 4	R/W	0x0000_0000	10.3.5.1/10-37
0x2410	MSGR5—Message register 5			
0x2420	MSGR6—Message register 6			
0x2430	MSGR7—Message register 7			
0x2440–0x24F0	Reserved	—	—	—
0x2500	MER—Message enable register (for MSGR4–7)	R/W	0x0000_0000	10.3.5.2/10-37
0x2510	MSR—Message status register (for MSGG4–7)	R/W	0x0000_0000	10.3.5.3/10-38
0x2514–0xFFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers—Block Base Address: 0x5_0000				
0x0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register or PEX1-INTA vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0010	EIDR0—External interrupt 0 (IRQ0) destination register or PEX1-INTA destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register or PEX1-INTB vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0030	EIDR1—External interrupt 1 (IRQ1) destination register or PEX1-INTB destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register or PEX1-INTC vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0050	EIDR2—External interrupt 2 (IRQ2) destination register or PEX1-INTC destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register or PEX1-INTD vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0070	EIDR3—External interrupt 3 (IRQ3) destination register or PEX1-INTD destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register or PEX2-INTA vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0090	EIDR4—External interrupt 4 (IRQ4) destination register or PEX2-INTA destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register or PEX2-INTB vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x00B0	EIDR5—External interrupt 5 (IRQ5) destination register or PEX2-INTB destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register or PEX2-INTC vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x00D0	EIDR6—External interrupt 6 (IRQ6) destination register or PEX2-INTC destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register or PEX2-INTD vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x00F0	EIDR7—External interrupt 7 (IRQ7) destination register or PEX2-INTD destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register or PEX3-INTA vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0110	EIDR8—External interrupt 8 (IRQ8) destination register or PEX3-INTA destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register or PEX3-INTB vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0130	EIDR9—External interrupt 9 (IRQ9) destination register or PEX3-INTB destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register or PEX3-INTC vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43
0x0150	EIDR10—External interrupt 10 (IRQ10) destination register or PEX3-INTC destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register or PEX3-INTD vector/priority register	R/W	0x8000_0000	10.3.7.1/10-43

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0170	EIDR11—External interrupt 11 (IRQ11) destination register or PEX3-INTD destination register	R/W	0x0000_0001	10.3.7.2/10-44
0x0180– 0x01F0	Reserved	—	—	—
0x0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.7.4/10-46

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0550	IIDR26—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0600	IIVPR32—Internal interrupt 32 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0610	IIDR32—Internal interrupt 32 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0620	IIVPR33—Internal interrupt 33 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0630	IIDR33—Internal interrupt 33 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0640	IIVPR34—Internal interrupt 34 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0650	IIDR34—Internal interrupt 34 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0660	IIVPR35—Internal interrupt 35 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0670	IIDR35—Internal interrupt 35 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0680	IIVPR36—Internal interrupt 36 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0690	IIDR36—Internal interrupt 36 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x06A0	IIVPR37—Internal interrupt 37 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x06B0	IIDR37—Internal interrupt 37 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x06C0	IIVPR38—Internal interrupt 38 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x06D0	IIDR38—Internal interrupt 38 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x06E0	IIVPR39—Internal interrupt 39 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x06F0	IIDR39—Internal interrupt 39 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0700	IIVPR40—Internal interrupt 40 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0710	IIDR40—Internal interrupt 40 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0720	IIVPR41—Internal interrupt 41 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0730	IIDR41—Internal interrupt 41 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0740	IIVPR42—Internal interrupt 42 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0750	IIDR42—Internal interrupt 42 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0760	IIVPR43—Internal interrupt 43 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0770	IIDR43—Internal interrupt 43 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0780	IIVPR44—Internal interrupt 44 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0790	IIDR44—Internal interrupt 44 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x07A0	IIVPR45—Internal interrupt 45 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x07B0	IIDR45—Internal interrupt 45 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x07C0	IIVPR46—Internal interrupt 46 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x07D0	IIDR46—Internal interrupt 46 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x07E0	IIVPR47—Internal interrupt 47 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x07F0	IIDR47—Internal interrupt 47 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0800	IIVPR48—Internal interrupt 48 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0810	IIDR48—Internal interrupt 48 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0820	IIVPR49—Internal interrupt 49 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0830	IIDR49—Internal interrupt 49 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0840	IIVPR50—Internal interrupt 50 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0850	IIDR50—Internal interrupt 50 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0860	IIVPR51—Internal interrupt 51 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0870	IIDR51—Internal interrupt 51 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0880	IIVPR52—Internal interrupt 52 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0890	IIDR52—Internal interrupt 52 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x08A0	IIVPR53—Internal interrupt 53 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x08B0	IIDR53—Internal interrupt 53 destination register	R/W	0x0000_0001	10.3.7.4/10-46

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x08C0	IIVPR54—Internal interrupt 54 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x08D0	IIDR54—Internal interrupt 54 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x08E0	IIVPR55—Internal interrupt 55 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x08F0	IIDR55—Internal interrupt 55 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0900	IIVPR56—Internal interrupt 56 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0910	IIDR56—Internal interrupt 56 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0920	IIVPR57—Internal interrupt 57 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0930	IIDR57—Internal interrupt 57 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0940	IIVPR58—Internal interrupt 58 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0950	IIDR58—Internal interrupt 58 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0960	IIVPR59—Internal interrupt 59 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0970	IIDR59—Internal interrupt 59 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x0980	IIVPR60—Internal interrupt 60 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x0990	IIDR60—Internal interrupt 60 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x09A0	IIVPR61—Internal interrupt 61 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x09B0	IIDR61—Internal interrupt 61 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x09C0	IIVPR62—Internal interrupt 62 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x09D0	IIDR62—Internal interrupt 62 destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x09E0	IIVPR63—Internal interrupt 63 vector/priority register	R/W	0x8080_0000	10.3.7.3/10-45
0x09F0	IIDR63—Internal interrupt 63 destination register	R/W	0x0000_0001	10.3.7.3/10-45
0x0A00–0x15F0	Reserved	—	—	—
0x1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.7.4/10-46
0x1680	MIVPR4—Messaging interrupt 4 (MSG 4) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x1690	MIDR4—Messaging interrupt 4 (MSG 4) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x16A0	MIVPR5—Messaging interrupt 5 (MSG 5) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x16B0	MIDR5—Messaging interrupt 5 (MSG 5) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x16C0	MIVPR6—Messaging interrupt 6 (MSG 6) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47
0x16D0	MIDR6—Messaging interrupt 6 (MSG 6) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x16E0	MIVPR7—Messaging interrupt 7 (MSG 7) vector/priority register	R/W	0x8000_0000	10.3.7.5/10-47

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x16F0	MIDR7—Messaging interrupt 7 (MSG 7) destination register	R/W	0x0000_0001	10.3.7.6/10-48
0x1700–0x1BF0	Reserved	—	—	—
0x1C00	MSIVPR0—Shared message signaled interrupt vector/priority register 0	R/W	0x8000_0000	10.3.6.4/10-40
0x1C10	MSIDR0—Shared message signaled interrupt destination register 0	R/W	0x0000_0001	10.3.6.5/10-41
0x1C20	MSIVPR1—Shared message signaled interrupt vector/priority register 1	R/W	0x8000_0000	10.3.6.4/10-40
0x1C30	MSIDR1—Shared message signaled interrupt destination register 1	R/W	0x0000_0001	10.3.6.5/10-41
0x1C40	MSIVPR2—Shared message signaled interrupt vector/priority register 2	R/W	0x8000_0000	10.3.6.4/10-40
0x1C50	MSIDR2—Shared message signaled interrupt destination register 2	R/W	0x0000_0001	10.3.6.5/10-41
0x1C60	MSIVPR3—Shared message signaled interrupt vector/priority register 3	R/W	0x8000_0000	10.3.6.4/10-40
0x1C70	MSDIR3—Shared message signaled interrupt destination register 3	R/W	0x0000_0001	10.3.6.5/10-41
0x1C80	MSIVPR4—Shared message signaled interrupt vector/priority register 4	R/W	0x8000_0000	10.3.6.4/10-40
0x1C90	MSIDR4—Shared message signaled interrupt destination register 4	R/W	0x0000_0001	10.3.6.5/10-41
0x1CA0	MSIVPR5—Shared message signaled interrupt vector/priority register 5	R/W	0x8000_0000	10.3.6.4/10-40
0x1CB0	MSIDR5—Shared message signaled interrupt destination register 5	R/W	0x0000_0001	10.3.6.5/10-41
0x1CC0	MSIVPR6—Shared message signaled interrupt vector/priority register 6	R/W	0x8000_0000	10.3.6.4/10-40
0x1CD0	MSIDR6—Shared message signaled interrupt destination register 6	R/W	0x0000_0001	10.3.6.5/10-41
0x1CE0	MSIVPR7—Shared message signaled interrupt vector/priority register 7	R/W	0x8000_0000	10.3.6.4/10-40
0x1CF0	MSDIR7—Shared message signaled interrupt destination register 7	R/W	0x0000_0001	10.3.6.5/10-41
0x1D00–0xFFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers Block Base Address: 0x6_0000				
0x0000–0x0030	Reserved	—	—	—
0x0040	IPIDR0—Processor core 0 interprocessor 0 dispatch register	W	all zeros	10.3.8.1/10-50
0x0050	IPIDR1—Processor core 0 interprocessor 1 dispatch register			
0x0060	IPIDR2—Processor core 0 interprocessor 2 dispatch register			
0x0070	IPIDR3—Processor core 0 interprocessor 3 dispatch register			
0x0080	CTPR0—Processor core 0 current task priority register	R/W	0x0000_000F	10.3.8.2/10-51
0x0090	WHOAMI0—Processor core 0 who am I register	R	n/a	10.3.8.3/10-52
0x00A0	IACK0—Processor core 0 interrupt acknowledge register	R	all zeros	10.3.8.4/10-52
0x00B0	EOI0—Processor core 0 end of interrupt register	W	all zeros	10.3.8.5/10-53
0x00C0–0x0FF0	Reserved	—	—	—
0x1000–0x1030	Reserved	—	—	—

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1040	IPIDR0—Processor core 1 interprocessor 0 dispatch register [1]	W	all zeros	10.3.8.1/10-50
0x1050	IPIDR1—Processor core 1 interprocessor 1 dispatch register [1]			
0x1060	IPIDR2—Processor core 1 interprocessor 2 dispatch register [1]			
0x1070	IPIDR3—Processor core 1 interprocessor 3 dispatch register [1]			
0x1080	CTPR1—Processor core 1 current task priority register [1]	R/W	0x0000_000F	10.3.8.2/10-51
0x1090	WHOAMI1—Processor core 1 who am I register [1]	R	n/a	10.3.8.3/10-52
0x10A0	IACK1—Processor core 1 interrupt acknowledge register [1]	R	all zeros	10.3.8.4/10-52
0x10B0	EOI1—Processor core 1 end of interrupt register [1]	W	all zeros	10.3.8.5/10-53
0x10C0–0xFFFC	Reserved	—	—	—

[1] Not supported for single-processor implementations.

10.3.1 Global Registers

Although most PIC 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 ID. A copy of the per-CPU registers is available to each processor core at the same physical address, that is, in a private access address space that acts like an alias to a processor's own copy of the per-CPU registers. As shown in Figure 10-44, the ID of the core initiating the read/write transaction determines which processor's per-CPU registers to access. For more information, see Section 10.3.8, "Per-CPU (Private Access) Registers."

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

10.3.1.1 Block Revision Register 1 (BRR1)

BRR1, shown in Figure 10-3, provides information about the PIC IP block.

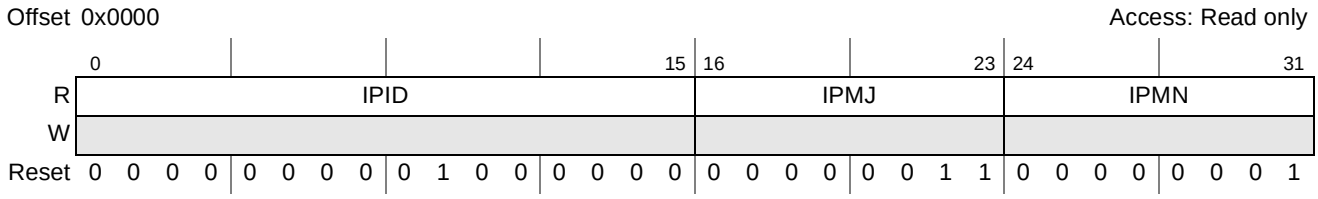


Figure 10-3. Block Revision Register 1 (BRR1)

Table 10-7 describes the BRR1 fields.

Table 10-5. BRR1 Field Descriptions

Bits	Name	Description
0–15	IPID	IP block ID.
16–23	IPMJ	The major revision of the IP block.
24–31	IPMN	The minor revision of the IP block.

10.3.1.2 Block Revision Register 2 (BRR2)

BRR2, shown in Figure 10-4, provides information about the IP block integration option and IP block configuration options.

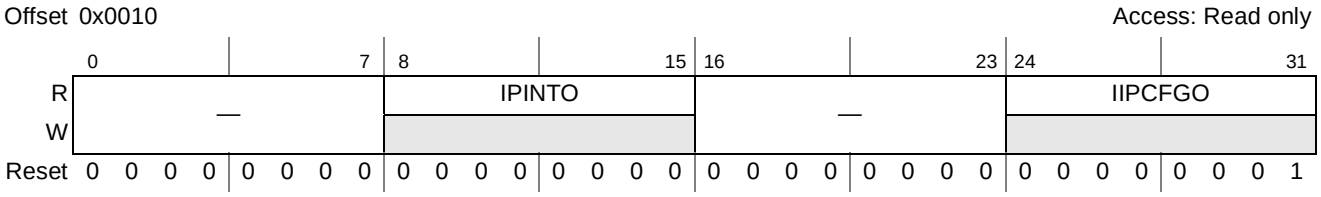


Figure 10-4. Block Revision Register 2 (BRR2)

Table 10-6 describes the BRR2 fields.

Table 10-6. BRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–15	IPINTO	IP block integration options
16–23	—	Reserved, should be cleared.
24–31	IPCFGO	IP block configuration options

10.3.1.3 Feature Reporting Register (FRR)

FRR, shown in Figure 10-5, provides information about interrupt and processor core configurations. It also informs the programming environment of the controller version.

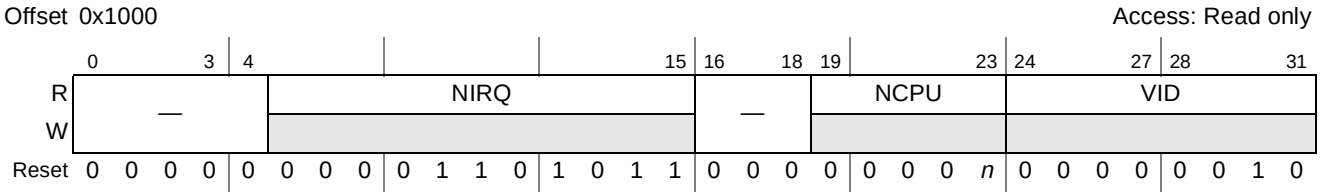


Figure 10-5. Feature Reporting Register (FRR)

Table 10-7 describes the FRR fields.

Table 10-7. FRR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–15	NIRQ	Number of interrupts. Holds the binary value of the number of the highest interrupt source supported minus one. The value is 107 (0x6B or 0b000_0110_1011).A zero in this field corresponds to one source.
16–18	—	Reserved, should be cleared
19–23	NCPU	Number of CPUs. The number of the highest physical CPUs (or processor cores) supported minus one. 00000 Single core—core0 00001 Two cores—core0 and core1
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.

10.3.1.4 Global Configuration Register (GCR)

GCR, shown in Figure 10-6, controls the PIC's operating mode, and allows software to reset the PIC.

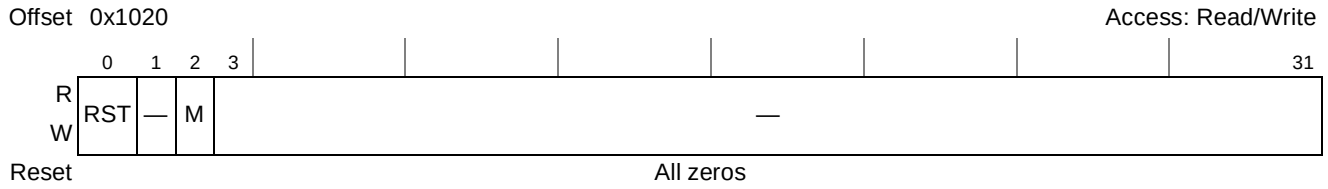


Figure 10-6. Global Configuration Register (GCR)

Table 10-8 describes the GCR fields.

Table 10-8. GCR Field Descriptions

Bits	Name	Description
0	RST	Reset. Setting RST forces the PIC to be reset. Cleared automatically when the reset sequence is complete. See Section 10.4.8, "Resetting the PIC," for more information.
1	—	Reserved, should be cleared.
2	M	Mode. PIC operating mode. Section 10.1.4, "Modes of Operation," provides details about these modes. 0 Pass-through mode. On-chip PIC is disabled and interrupts detected on IRQ0 are passed directly to core 0. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the PIC.
3–31	—	Reserved, should be cleared.

10.3.1.5 Vendor Identification Register (VIR)

VIR, shown in Figure 10-7, is defined by the OpenPIC specifications and is provided for compliance. The zero value for VIR[VENDORID] indicates a generic OpenPIC-compliant device, which makes the other VIR fields meaningless.

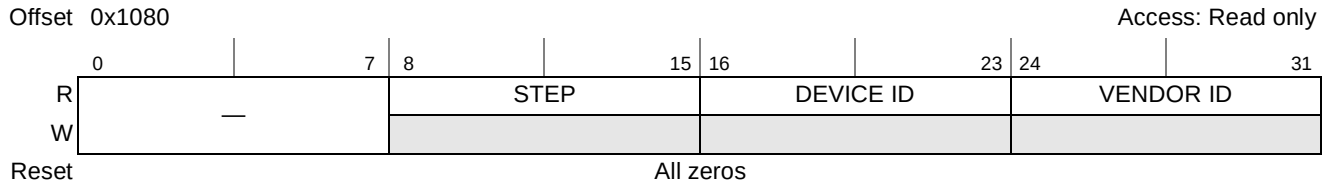


Figure 10-7. Vendor Identification Register (VIR)

Table 10-9 describes the VIR fields.

Table 10-9. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
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.

10.3.1.6 Processor Core Initialization Register (PIR)

PIR, shown in Figure 10-8, provides a way for software to generate a core reset. Setting P1 or P0 causes the respective *core0_hreset* or *core1_hreset* signal to assert. Note that after requesting a core reset using this register the applicable bit should not be cleared until the requested core reset has occurred. However, if one core is used to reset another one, the core being reset can effectively be held off indefinitely from issuing its initial boot vector fetch to the platform by leaving its appropriate PIR[Px] bit asserted. Clearing it releases the core to fetch.

Note that although the OpenPIC architecture was defined to support up to 32 processing cores, only fields corresponding to processors on the device are implemented.

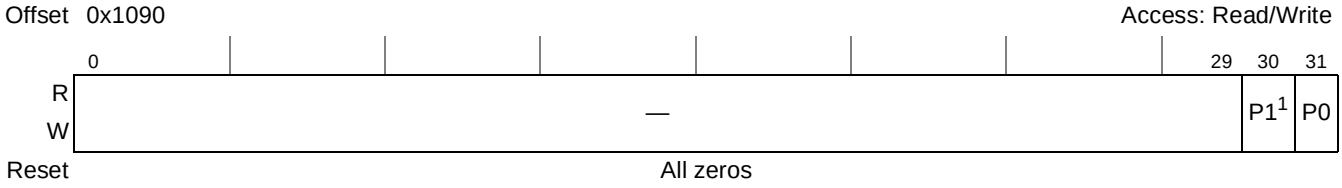


Figure 10-8. Processor Core Initialization Register (PIR)

Table 10-10 describes the PIR fields.

Table 10-10. PIR Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1 reset. Setting this bit causes the PIC to assert the *core1_hreset* signal. Reserved in single-processor implementations.
31	P0	Processor core 0 reset. Setting this bit causes the PIC to assert the *core0_hreset* signal.

10.3.1.7 Interprocessor Interrupt Vector/Priority Registers (IPIVPR0–IPIVPR3)

IPIVPRs, shown in Figure 10-9, 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.

Offset IPIVPR0: 0x10A0; IPIVPR1: 0x10B0; IPIVPR2: 0x10C0; IPIVPR3: 0x10D0 Access: Read/Write

	0	1	2 … 11	12 … 15	16 … 31
R	MSK	A	—	PRIORITY	VECTOR
W					
Reset	1	0	0 … 0	0 … 0	0 … 0

Figure 10-9. Interprocessor Interrupt Vector/Priority Register (IPIVPR*n*)

Table 10-11 describes the IPIVPR*n* fields.

Table 10-11. IPIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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	—	Reserved, should be cleared.
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 10-50.

10.3.1.8 Spurious Vector Register (SVR)

SVR, shown in Figure 10-10, contains the 16-bit vector returned to the processor core when the corresponding IACK register is read for a spurious interrupt.

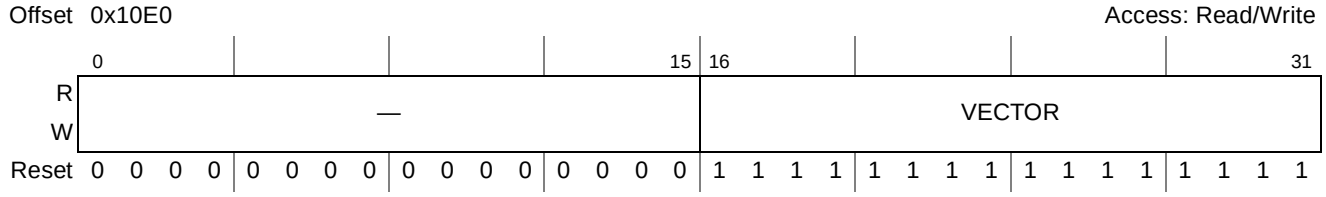


Figure 10-10. Spurious Vector Register (SVR)

Table 10-12 describes the SVR fields.

Table 10-12. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Spurious interrupt vector. Value returned when IACK is read during a spurious vector fetch. Section 10.4.1.2.4, "Spurious Vector Generation," gives information about the conditions that may cause a spurious vector fetch.

10.3.2 Global Timer Registers

The two independent groups of global timer registers, group A and group B, are identical in their functionality, except that they appear at different locations within the PIC register map. Note that each of the four timers within an *x* group have four individual configuration registers (GTCCR*xn*, GTBCR*xn*, GTVPR*xn*, GTDR*xn*), but they are only shown once in this section. These two groups of timers cannot be cascaded together.

10.3.2.1 Timer Frequency Reporting Register (TFRRA–TFRRB)

The TFRRs, shown in Figure 10-11, are written by software to report the clocking frequency of the PIC timers. Note that although TFRRs are read/write, the PIC ignores the register values.

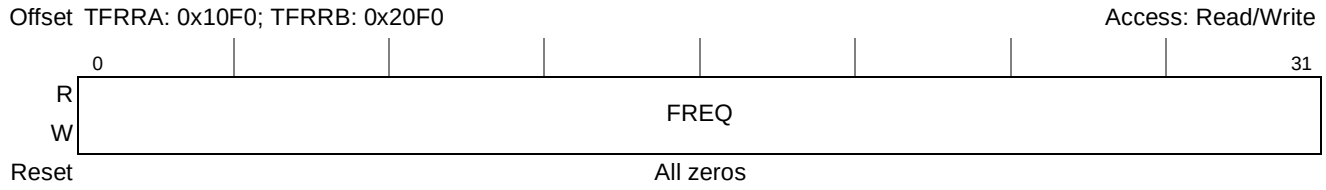


Figure 10-11. Timer Frequency Reporting Registers (TFRR*x*)

Table 10-13 describes the TFRR*x* registers.

Table 10-13. TFRR*x* Field Descriptions

Bits	Name	Description
0–31	FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source, (either the CCB clock or the frequency of the RTC signal), to user software. TFRR*x* 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 TCR*x*[CLKR]. See Section 10.3.2.6, "Timer Control Registers (TCRA–TCRB)."

10.3.2.2 Global Timer Current Count Registers (GTCCRA0–GTCCRA3, GTCCRB0–GTCCRB3)

The GTCCRs, shown in Figure 10-12, contain the current count for each of the four PIC timers in each of the two groups.

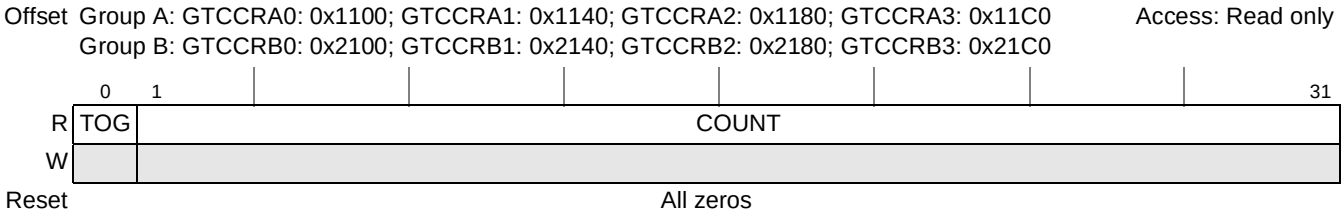


Figure 10-12. Global Timer Current Count Registers (GTCCR*xn*)

Table 10-14 describes the GTCCR*xn* fields.

Table 10-14. GTCCR*xn* Field Descriptions

Bits	Name	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 Section 10.3.2.6, "Timer Control Registers (TCRA–TCRB)," for more details.

10.3.2.3 Global Timer Base Count Registers (GTBCRA0–GTBCRA3, GTBCRB0–GTBCRB3)

The GTBCRs contain the base counts for each of the four PIC timers in each of the two groups, as shown in Figure 10-13. 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.

Offset Group A: GTBCRA0: 0x1110; GTBCRA1: 0x1150; GTBCRA2: 0x1190; GTBCRA3: 0x11D0 Access: Read/write
Group B: GTBCRB0: 0x2110; GTBCRB1: 0x2150; GTBCRB2: 0x2190; GTBCRB3: 0x21D0

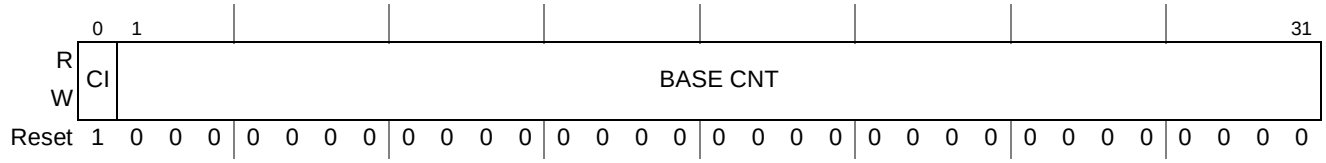


Figure 10-13. Global Timer Base Count Register (GTBCR*xn*)

Table 10-15 describes the GTBCR*xn* fields.

Table 10-15. GTBCR*xn* Field Descriptions

Bits	Name	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.

10.3.2.4 Global Timer Vector/Priority Registers (GTVPRA0–GTVPRA3, GTVPRB0–GTVPRB3)

The GTVPRs contain the interrupt vector and the interrupt priority values for the timers as shown in Figure 10-14. They also contain the mask and activity fields for all the timers.

Offset Group A: GTVPRA0: 0x1120; GTVPRA1: 0x1160; GTVPRA2: 0x11A0; GTVPRA3: 0x11E0; Access: Read/Write
Group B: GTVPRB0: 0x2120; GTVPRB1: 0x2160; GTVPRB2: 0x21A0; GTVPRB3: 0x21E0

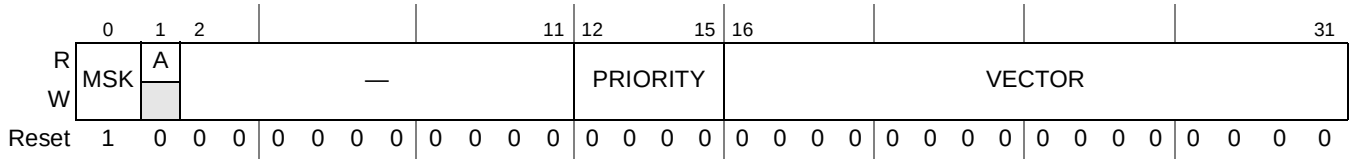


Figure 10-14. Global Timer Vector/Priority Register (GTVPR*xn*)

Table 10-16 describes the GTVPR*xn* fields.

Table 10-16. GTVPR*xn* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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	—	Reserved, should be cleared.

Table 10-16. GTVPR*xn* Field Descriptions (continued)

Bits	Name	Description
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 10-50.

10.3.2.5 Global Timer Destination Registers (GTDRA0–GTDRA3, GTDRB0–GTDRB3)

The GTDR*xn* registers, shown in Figure 10-15, 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 that either P1 or P0 or both can be set for this type of interrupt.

Offset Group A: GDTRA0: 0x1130; GDTRA1: 0x1170; GDTRA2: 0x11B0; GDTRA3: 0x11F0; Access: Read/Write
Group B: GDTRB0: 0x2130; GDTRB1: 0x2170; GDTRB2: 0x21B0; GDTRB3: 0x21F0

	0																								29	30	31
R W	—																									P1[1]	P0
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] Reserved in single-processor implementations.

Figure 10-15. Global Timer Destination Registers (GTDR*xn*)

Table 10-17 describes the GTDR*xn* fields.

Table 10-17. GTDR*xn* Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1. This interrupt is multicasting, so both P0 and P1 can be set. Reserved in single-processor implementations. 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 PIC is reset. Both P0 and P1 can be set. 0 Processor core 0 does not receive this interrupt. 1 Directs the timer interrupt to processor core 0.

10.3.2.6 Timer Control Registers (TCRA–TCRB)

The TCR registers, shown in Figure 10-17, 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 CCB bus 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, TCR*x*[ROVR]. Setting TCR*x*[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 TCR*x*[ROVR] settings.

For example, timers 0–2 can be cascaded to generate one interrupt per hour. As shown in Table 10-18, given an CCB clock frequency of 333 MHz, letting the timer clock frequency default to $1/8^{th}$ the system clock, (TCR*x*[CLKR] = 0 sets a clock ratio of 8), provides a basic input of 41.625 MHz to timer 0. Setting timer 0 to count 41,625,000 (0x27B_25A8) timer clock cycles generates one output per second. Setting both timers 1 and 2 to 59, and cascading all three timers, generates one interrupt every hour from timer 2.

Table 10-18. Parameters for Hourly Interrupt Timer Cascade Example

System Clock	Clock Ratio	Timer Clock	Timer 0 Count	Timer 1 Count	Timer 2 Count
333 MHz	1 / 8	41.625 MHz	41.625×10^6 (0x027B_25A8)	59[1] (0x0000_0036)	59 (0x0000_0036)

[1] Counting down from 59 through 0 requires 60 ticks.

$$(41.625 \times 106 \text{ ticks/sec}) \times (60 \text{ sec/min}) \times (60 \text{ min/hr}) = \text{total ticks/hr generating 1 interrupt/hr}$$

Figure 10-16. Example Calculation for Cascaded Timers

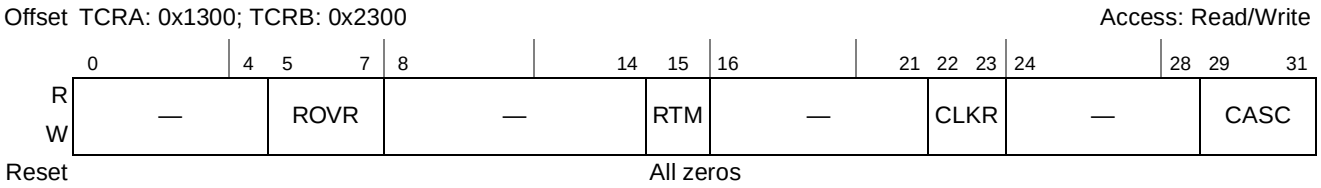


Figure 10-17. Timer Control Registers (TCR*x*)

Table 10-19 describes the TCR*x* fields.

Table 10-19. TCR*x* Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
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	—	Reserved, should be cleared.
15	RTM	Real time mode. Specifies the clock source for the PIC timers. 0 Timer clock frequency is a ratio of the frequency of the CCB clock as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the PIC timers. If this bit is set, the CLKR field has no meaning.
16–21	—	Reserved, should be cleared.
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the CCB clock. The following are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28	—	Reserved, should be cleared.
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

10.3.3 IRQ_OUT and Critical Interrupt Summary Registers

The summary registers indicate the specific interrupt sources routed to the $\overline{\text{IRQ\_OUT}}$ or *cint0/cint1*. PIC outputs. Summary register bits are cleared when the corresponding interrupt that caused a bit to be set is negated. Note that only level-sensitive interrupts can be routed to $\overline{\text{IRQ\_OUT}}$ or *cint0* and *cint1*.

The IRQ_OUT summary registers, shown in Figure 10-19 through Figure 10-21 contain one bit for each interrupt source that can be routed to $\overline{\text{IRQ\_OUT}}$. The corresponding bit is set if the interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR*n*[EP] is set).

The critical interrupt summary registers, shown in Figure 10-22 through Figure 10-24, contain one bit for each interrupt source that can be designated as a critical interrupt. The corresponding bit is set if the interrupt is active and is routed to either the *cint0* or *cint1* outputs of the PIC (if *x*IDR*n*[CI*n*] = 1 in its corresponding destination register).

10.3.3.1 External Interrupt Summary Register (ERQSR)

NOTE

ERQSR fields report only the current state of IRQ0–IRQ11. These fields were designed to work with level-sensitive interrupts; values returned for edge-sensitive interrupts may be unreliable.

Figure 10-18 shows the ERQSR fields.

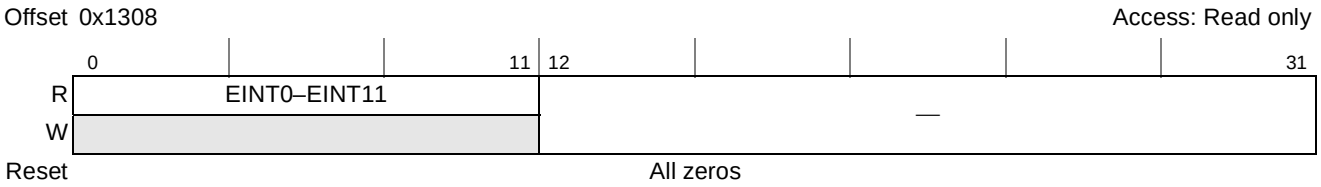


Figure 10-18. External Interrupt Summary Register (ERQSR)

Table 10-20 describes the ERQSR fields.

Table 10-20. ERQSR Field Descriptions

Bits	Name	Description
0–11	EINT*n*	External interrupts signal 0–11 status. Bit 0 represents EINT0. Bit 11 represents EINT11. 0 The corresponding external interrupt signal is not active. 1 The corresponding external interrupt signal is active.
12–31	—	Reserved, should be cleared.

10.3.3.2 IRQ_OUT Summary Register 0 (IRQSR0)

Figure 10-19 shows the IRQSR0 fields.

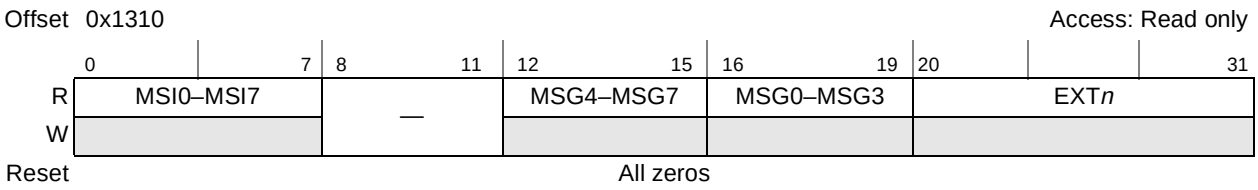


Figure 10-19. IRQ_OUT Summary Register 0 (IRQSR0)

Table 10-21 describes the IRQSR0 fields.

Table 10-21. IRQSR0 Field Descriptions

Bits	Name	Description
0–7	MSI*n*	Shared message signaled interrupt *n* status 0 Interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is active and is routed to the $\overline{\text{IRQ\_OUT}}$ signal (that is, if the corresponding *x*IDR*n*[EP] is set).
8–11	—	Reserved, should be cleared.
12–15	MSG*n*	Message interrupt *n* status 0 Interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is active and is routed to the $\overline{\text{IRQ\_OUT}}$ signal (that is, if the corresponding *x*IDR*n*[EP] is set).
16–19	MSG*n*	Message interrupt *n* status 0 Interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is active and is routed to the $\overline{\text{IRQ\_OUT}}$ signal (that is, if the corresponding *x*IDR*n*[EP] is set).
20–31	EXT*n*	External interrupts 0–11. Each bit corresponds to a unique interrupt according to the following: Bit Interrupt 20 IRQ0 21 IRQ1 … 31 IRQ11 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and routed to $\overline{\text{IRQ\_OUT}}$ (if the corresponding *x*IDR*n*[EP] is set).

10.3.3.3 IRQ_OUT Summary Register 1 (IRQSR1)

Figure 10-20 shows the IRQSR1 fields.

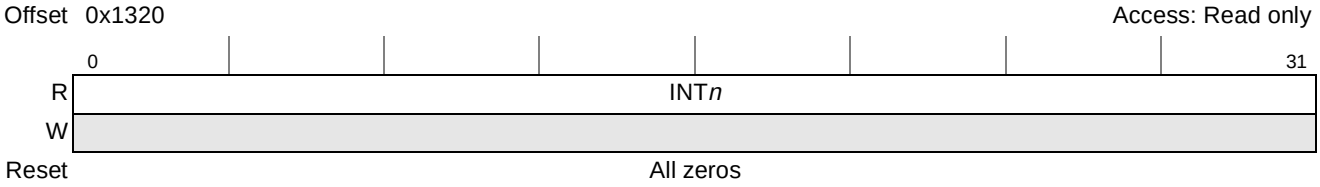


Figure 10-20. IRQ_OUT Summary Register 1 (IRQSR1)

Table 10-22 describes the IRQSR1 fields.

Table 10-22. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR*n*[EP] is set).

10.3.3.4 IRQ_OUT Summary Register 2 (IRQSR2)

Figure 10-21 shows the IRQSR2 fields.

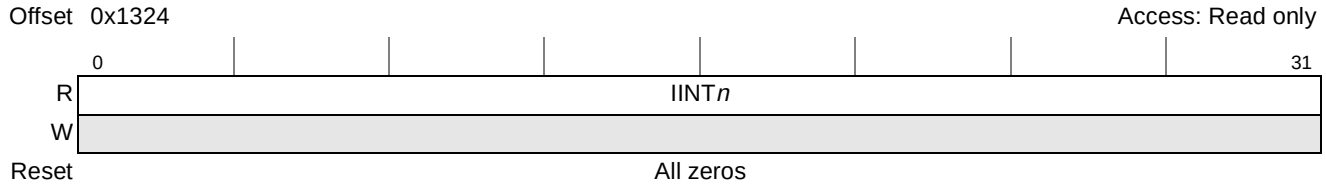


Figure 10-21. IRQ_OUT Summary Register 2 (IRQSR2)

Table 10-23 describes the IRQSR2 fields.

Table 10-23. IRQSR2 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 32–63 status. Bit 0 represents INT32. Bit 31 represents INT63. 0 The corresponding interrupt is not active or not routed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to $\overline{\text{IRQ\_OUT}}$, if the corresponding *x*IDR*n*[EP] is set.

10.3.3.5 Critical Interrupt Summary Register 0 (CISR0)

Figure 10-22 shows CISR0.

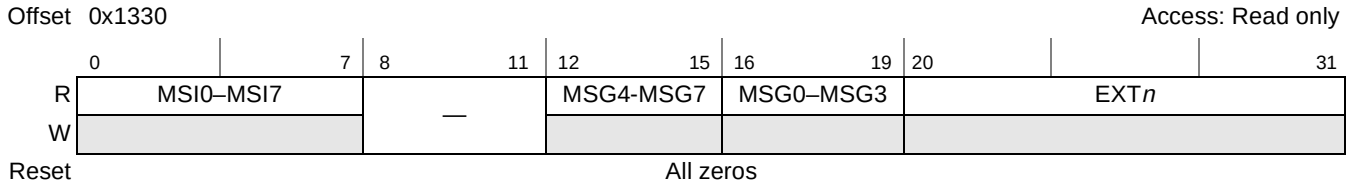


Figure 10-22. Critical Interrupt Summary Register 0 (CISR0)

Table 10-24 describes CISR0 fields.

Table 10-24. CISR0 Field Descriptions

Bits	Name	Description
0–7	MSI*n*	Shared message signaled interrupts 0–7. Bit 0 represents MSI0; bit 7 represents MSI7. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1*, if the corresponding *x*IDR*n*[CI] is set.
8–11	—	Reserved, should be cleared.
12–15	MSG*n*	Message interrupts 4–7. Bit 16 represents MSG4; bit 19 represents MSG7. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).
16–19	MSG*n*	Message interrupts 0–3. Bit 16 represents MSG0; bit 19 represents MSG3. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).
20–31	EXT*n*	External interrupts 0–11. Bit 20 represents IRQ0. Bit 31 represents IRQ11. 0 The corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).

10.3.3.6 Critical Interrupt Summary Register 1 (CISR1)

Figure 10-23 shows the CISR1.

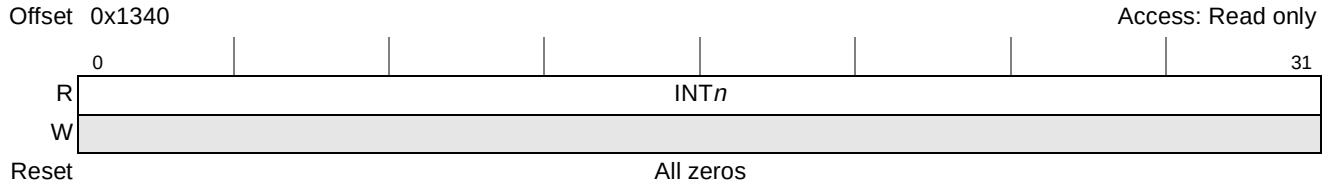


Figure 10-23. Critical Interrupt Summary Register 1 (CISR1)

Table 10-25 describes CISR1.

Table 10-25. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. Bit 31 represents INT31. 0 Corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to the *cint0* or *cint1* (if the corresponding *x*IDR*n*[CI] is set).

10.3.3.7 Critical Interrupt Summary Register 2 (CISR2)

Figure 10-24 shows the CISR2.

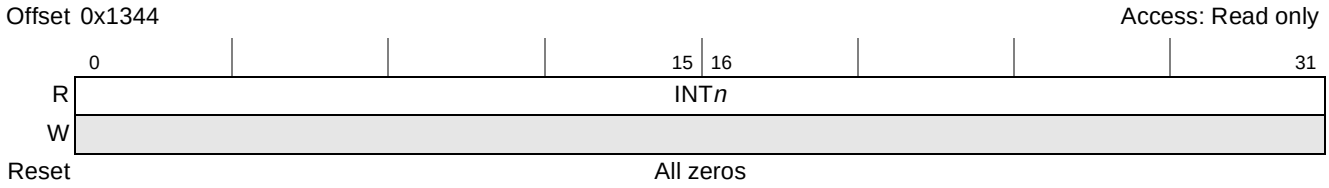


Figure 10-24. Critical Interrupt Summary Register 2 (CISR2)

Table 10-26 describes CISR2.

Table 10-26. CISR2 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 32–63. Bit 0 represents INT32. Bit 31 represents INT63. 0 Corresponding interrupt is not active or not routed to *cint0* or *cint1*. 1 The corresponding interrupt is active and is routed to the *cint0* or *cint1*, if the corresponding *x*IDR*n*[CI] is set.

10.3.4 Performance Monitor Mask Registers (PMMRs)

The twelve performance monitor mask registers consist of four sets of three 32-bit registers, PM*n*MR0, PM*n*MR1, and PM*n*MR2. Each set can be configured to select one interrupt source (interprocessor, timer, message, shared message signaled, external, or internal) to generate a performance monitor event. The performance monitor can be configured to track this event in the performance monitor local control registers. See Section 24.3.2.2, “Performance Monitor Local Control Registers (PMLCAn, PMLCBn).”

10.3.4.1 Performance Monitor Mask Registers 0 (PM0MR0–PM3MR0)

Each PM*n*MR0 register, shown in Figure 10-25, is matched with a PM*n*MR1 and a PM*n*MR2 register. Because each unreserved bit in the 96-bit vector (PM*n*MR0/1/2) specifies a different interrupt, only one bit in the 96-bit vector can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

Offset PM0MR0: 0x1350; PM1MR0: 0x1370; PM2MR0: 0x1390; PM3MR0: 0x13B0 Access: Read/Write

	0				7	8			11	12			15	16			19	20							31
R W	MShI					IPI				TIMER				MSG				EXT							
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	1	1	1	1

Figure 10-25. Performance Monitor Mask Registers 0 (PM*n*MR0)

Table 10-27 describes the PM*n*MR0 fields.

Table 10-27. PM*n*MR0 Field Descriptions

Bits	Name	Description
0–7	MShI	Shared message signaled interrupts 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.
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 0–7 Bit 0 is used for MSG0 and MSG4 Bit 1 is used for MSG1 and MSG5 Bit 2 is used for MSG2 and MSG6 Bit 3 is used for MSG3 and MSG7 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[0:11] 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.

10.3.4.2 Performance Monitor Mask Registers 1 (PM0MR1–PM3MR1)

Figure 10-26 shows the PM*n*MR1 registers.

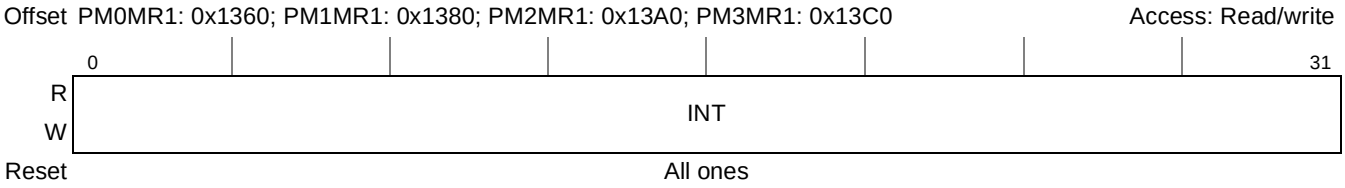


Figure 10-26. Performance Monitor Mask Registers 1 (PM*n*MR1)

Table 10-28 describes the PM*n*MR1 registers.

Table 10-28. PM*n*MR1 Field Descriptions

Bits	Name	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.

10.3.4.3 Performance Monitor Mask Registers 2 (PM0MR2–PM3MR2)

Figure 10-27 shows the PM*n*MR2 registers.

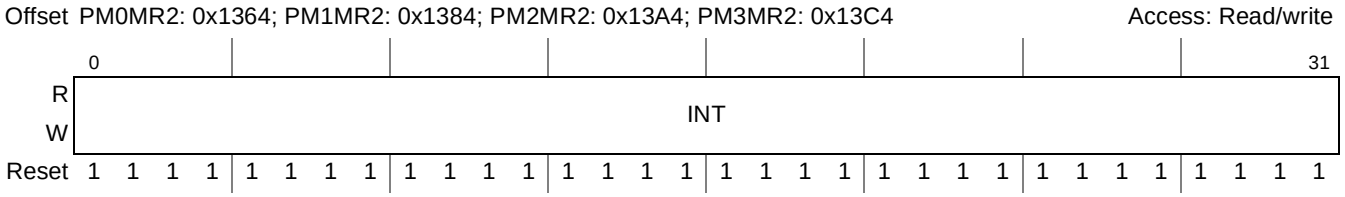


Figure 10-27. Performance Monitor Mask Registers 2 (PM*n*MR2)

Table 10-29 describes the PM*n*MR2 registers.

Table 10-29. PM*n*MR2 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 32–64 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.

10.3.5 Message Registers

The following registers support the message register interrupts:

- Section 10.3.5.1, “Message Registers (MSGR0–MSGR7)”
- Section 10.3.5.2, “Message Enable Register (MER)”
- Section 10.3.5.3, “Message Status Register (MSR)”
- Section 10.3.7.5, “Messaging Interrupt Vector/Priority Registers (MIVPRn)”

- Section 10.3.7.6, “Messaging Interrupt Destination Registers (MIDR0–MIDR7)”

Writing to one of the four message registers (MSGR0–MSGR7) causes a messaging interrupt as directed by the other message registers listed above. Reading a message register clears the messaging interrupt. Note that a messaging interrupt can also be cleared by writing a one to the corresponding status field of the PIC message status register (MSR), shown in Figure 10-30.

10.3.5.1 Message Registers (MSGR0–MSGR7)

The message registers (MSGR0–MSGR7), shown in Figure 10-28, can contain a 32-bit message.

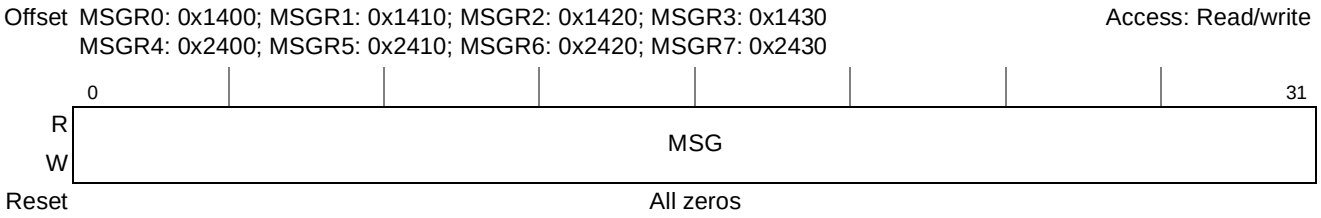


Figure 10-28. Message Registers (MSGRs)

Table 10-30 describes the MSGR registers.

Table 10-30. MSGR*n* Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data.

10.3.5.2 Message Enable Register (MER)

The MER, shown in Figure 10-29, contains the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

When bits in MER are set to mask message interrupts, an interrupt is not generated if the message register is written while it is masked in MER and the MER bit is then cleared. To mask the interrupt without loss, set MIVPR*n*[MSK]. (See Section 10.3.7.5, “Messaging Interrupt Vector/Priority Registers (MIVPRn).”) MER should be set to 0x0000_000F at reset and then left unchanged during normal operation.

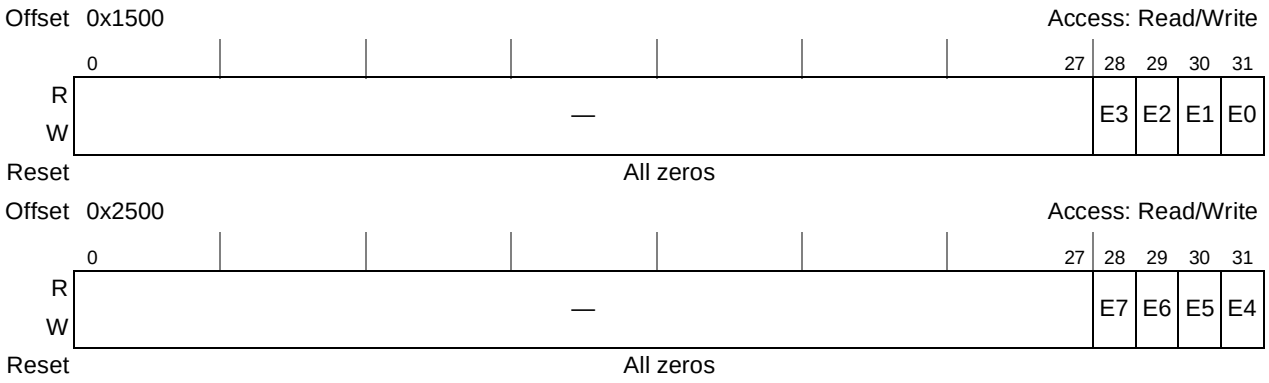


Figure 10-29. Message Enable Register (MER)

Table 10-31 describes the MER fields.

Table 10-31. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–32	E*n*	Enable 3–enable 0 or enable 7–enable 0. Used to enable interrupt generation for MSGR*n* (where *n* = 0–7). 0 Interrupt generation for MSGR*n* disabled. 1 Interrupt generation for MSGR*n* enabled.

10.3.5.3 Message Status Register (MSR)

The message status register (MSR) shown in Figure 10-30 contains 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.

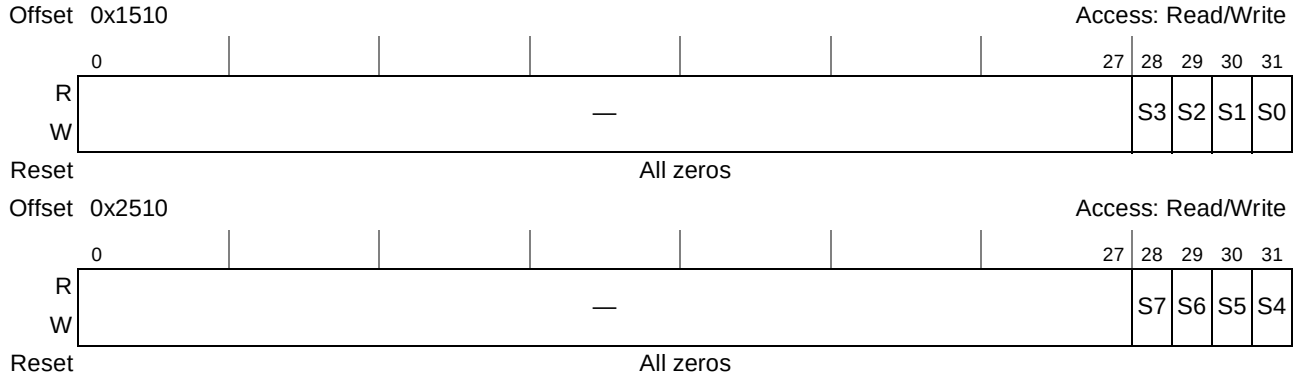


Figure 10-30. Message Status Register (MSR)

Table 10-32 describes the MSR fields.

Table 10-32. MSR Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28	S*n*	Status 3–status 0 or status 7–status 4. 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.

10.3.6 Shared Message Signaled Registers

This section contains description the shared message signaled interrupt registers (MSIRs). The shared message signaled interrupt structure allows programs to interrupt each other by simply writing to these shared memory-mapped registers in the PIC. Each of the eight MSIRs can be thought of as collecting interrupts from 32 different memory-mapped writes that can cause interrupts.

10.3.6.1 Shared Message Signaled Interrupt Registers (MSIR0–MSIR7)

The eight MSIRs indicate which of the up to 32 interrupt sources sharing the message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Offset MSIR0: 0x1600; MSIR1: 0x1610; MSIR2: 0x1620; MSIR3: 0x1630; MSIR4: 0x1640; MSIR5: 0x1650; MSIR6: 0x1660; MSIR7: 0x1670 — Access: Read only

	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	SH19	SH18	SH17	SH16
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH15	SH14	SH13	SH12	SH11	SH10	SH9	SH8	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W																
Reset	All zeros (Note: After soft reset, read the MSIRs to ensure that any interrupts previously pending are cleared).															

Figure 10-31. Message Shared Interrupt Registers (MSIR*n*)

Table 10-33 describes the bits of the MSIRs.

Table 10-33. MSIR*n* Field Descriptions

Bits	Name	Description
n	SH*n*	Message sharer *n* has a pending interrupt.

10.3.6.2 Shared Message Signaled Interrupt Status Register (MSISR)

MSISR, shown in Figure 10-32, contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIR has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIR.

Offset 0x1720 — Access: Read only

	0	23	24	25	26	27	28	29	30	31
R	—		S7	S6	S5	S4	S3	S2	S1	S0
W										
Reset	All zeros									

Figure 10-32. Shared Message Signaled Interrupt Status Register (MSISR)

Table 10-34 describes the bits of the MSISR.

Table 10-34. MSISR Field Descriptions

Bits	Name	Description
0–23	—	Reserved, should be cleared.
24–31	S*n*	Status *n*. 0 MSIR*n* is not active. 1 MSIR*n* has an active interrupt.

10.3.6.3 Shared Message Signaled Interrupt Index Register (MSIIR)

MSIIR, shown in Figure 10-32, provides the mechanism for setting an interrupt in the MSIRs. When MSIIR is written, MSIIR[SRS] selects the register in which an interrupt bit is to be set; MSIIR[IBS] selects the shared interrupt field in the selected MSIR register to be set. MSIIR is primarily intended to support PCI Express MSIs.

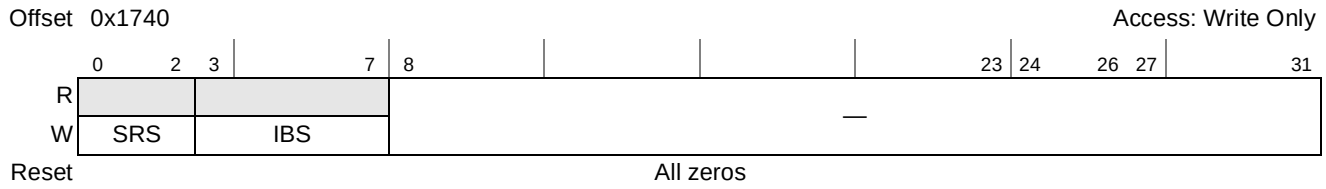


Figure 10-33. Shared Message Signaled Interrupt Index Register (MSIIR)

Table 10-35 describes the bits of the MSIIR.

Table 10-35. MSIIR Field Descriptions

Bits	Name	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	—	Reserved, should be cleared.

10.3.6.4 Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)

The MSIVPRs, shown in Figure 10-34, have the same fields and format as the GTVPRs.

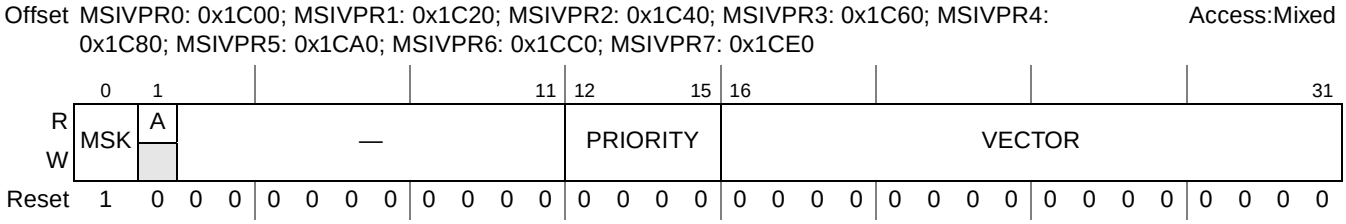


Figure 10-34. Shared Message Signaled Interrupt Vector/Priority Register (MSIVPRs)

Table 10-36 describes the bits of the MSIVPRs.

Table 10-36. MSIVPRn Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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	—	Reserved, should be cleared.
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 10-49.

10.3.6.5 Shared Message Signaled Interrupt Destination Registers 0–7 (MSIDR*n*)

The MSIDRs, shown in Figure 10-35, contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Offset MSIDR0: 0x1C10; MSIDR1: 0x1C30; MSIDR2: 0x1C50; MSIDR3: 0x1C70; MSIDR4: 0x1C90; MSIDR5: 0x1CB0; MSIDR6: 0x1CD0; MSIDR7: 0x1CF0 — Access: Read/Write

	0	1	2																								29	30	31
R W	EP	CI0	CI1[1]	—																								P1[1]	P0
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] Reserved in single-processor implementations.

Figure 10-35. Shared Message Signaled Interrupt Destination Registers (MSIDR*n*)

Table 10-37 describes MSIDR*n* fields.

Table 10-37. MSIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows interrupt to be serviced externally. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external servicing.
1	CI0	Critical interrupt 0. 0 Processor core 0 does not receive this interrupt. 1 Directs the shared message signaled interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 10.1.3, “Interrupts to the Processor Core.”
2	CI1	Critical interrupt 1. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the shared message signaled interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 10.1.3, “Interrupts to the Processor Core.”

Table 10-37. MSIDR*n* Field Descriptions (continued)

Bits	Name	Description
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int.* Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this shared message signaled interrupt after the PIC is reset.

10.3.7 Interrupt Source Configuration Registers

The interrupt source configuration registers control the source and destinations of interrupts, specifying parameters such as the interrupting event, signal polarity, and relative priority.

Figure 10-36 shows the destination register differences.

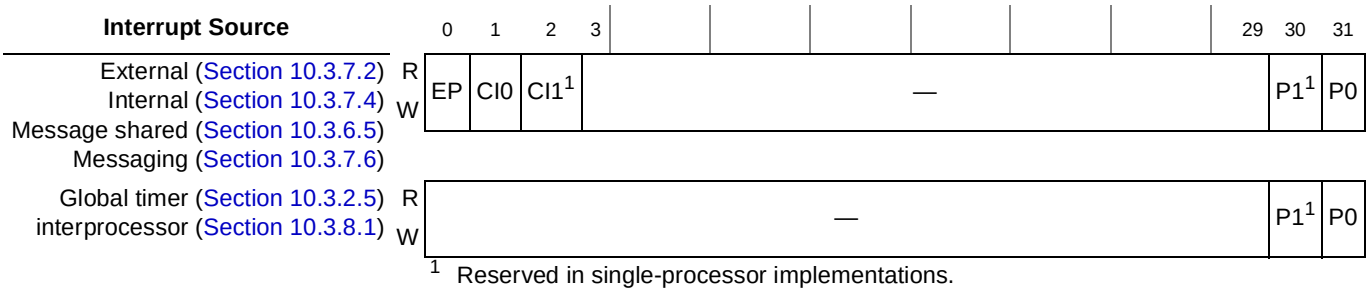


Figure 10-36. Destination Register Summary

Note the following:

- The global timer and interprocessor destination register support only the P0 and P1 options. That is, they cannot be routed to *cint* or to $\overline{\text{IRQ\_OUT}}$.
- Only the global timer and interprocessor interrupts are multicasting, so only these interrupts allow more than one destination bit to be specified.

Figure 10-36 shows the vector/priority register differences.

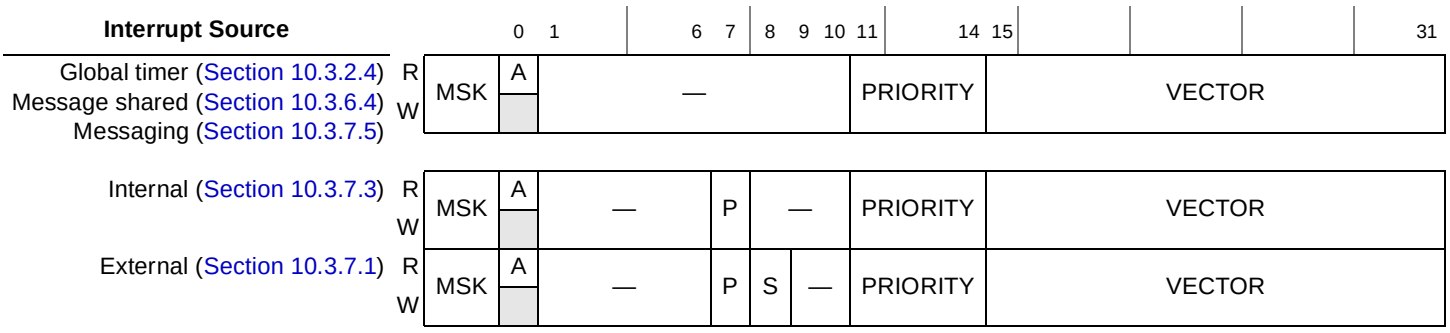


Figure 10-37. Vector/Priority Register Summary

Note the following:

- The MSK, A, PRIORITY, and VECTOR fields are defined by the OpenPIC specification and have meaning only for interrupts routed to the *int* signal.
- The polarity field, P, is provided to indicate whether the signals from the corresponding source are active high or low.
- The sense field, S, is provided to allow external interrupt sources to be configured as level-sensitive so they can be routed to either *cint* or $\overline{\text{IRQ\_OUT}}$.

10.3.7.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The EIVPRs, shown in Figure 10-38, contain polarity and sense fields for the external interrupts, that is, those caused by the assertion of any of IRQ[0:11].

Offset EIVPR0: 0x0000; EIVPR1: 0x0020; EIVPR2: 0x0040; EIVPR3: 0x0060; EIVPR4: 0x0080; EIVPR5: 0x00A0; EIVPR6: 0x00C0; EIVPR7: 0x00E0; EIVPR8: 0x0100; EIVPR9: 0x0120; EIVPR10: 0x0140; EIVPR11: 0x0160 — Access: Read/write

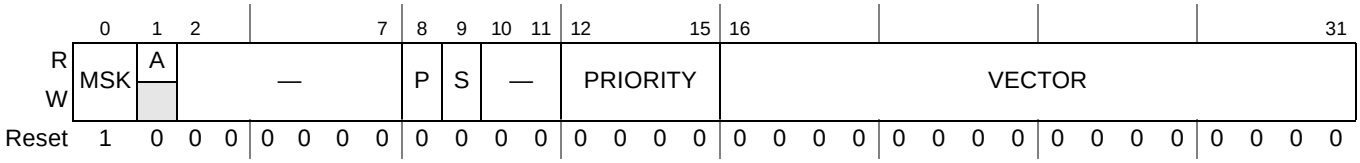


Figure 10-38. External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

Table 10-38 describes the EIVPR fields.

Table 10-38. EIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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.

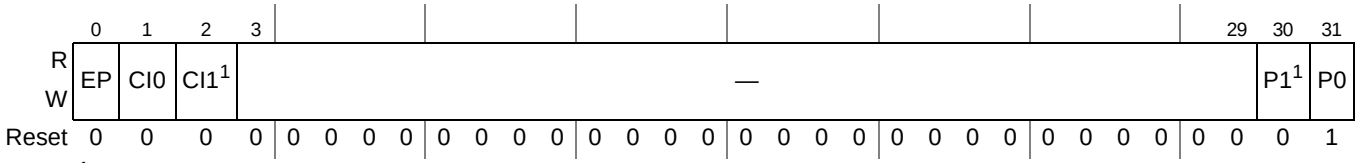
Table 10-38. EIVPR*n* Field Descriptions (continued)

Bits	Name	Description
2–7	—	Reserved, should be cleared.
8	P	Polarity. Specifies the polarity for the external interrupt. 0 Polarity is active-low or negative edge-triggered. 1 Polarity is active-high or positive edge-triggered.
9	S	Sense. Specifies the sense for external interrupts. 0 The external interrupt is edge sensitive. 1 The external interrupt is level sensitive. This setting must be used to direct the interrupt to $\overline{\text{IRQ\_OUT}}$ or *cint*. Note: If an IRQ*n* signal is used to receive INT*x* signals from one of the PCI Express ports as a root complex, S must be set to be level-sensitive.
10–11	—	Reserved, should be cleared.
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 10-50.

10.3.7.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The EIDRs, shown in Figure 10-39, control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. Only one destination bit may be set; otherwise, behavior is undefined.

Offset EIDR0: 0x0010; IDR1: 0x0030; EIDR2: 0x0050; EIDR3: 0x0070; EIDR4: 0x0090; EIDR5: 0x00B0; EIDR6: 0x00D0; EIDR7: 0x00F0; EIDR8: 0x0110; EIDR9: 0x0130; EIDR10: 0x0150; EIDR11: 0x0170 — Access: Read/write



[1] Reserved in single-processor implementations.

Figure 10-39. External Interrupt Destination Registers (EIDRs)

Table 10-39 describes the EIDR fields.

Table 10-39. EIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows interrupt to be serviced externally. EP should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting for edge-sensitive does not provide reliable interrupt response. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external service.
1	CI0	Critical interrupt 0. Ci*n* fields should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting them for edge-sensitive does not provide reliable interrupt response. 0 Processor core 0 does not receive this interrupt. 1 Directs the external interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 10.1.3, “Interrupts to the Processor Core.”

Table 10-39. EIDR*n* Field Descriptions (continued)

Bits	Name	Description
2	CI1	Critical interrupt 1. Ci*n* fields should be set only for level-sensitive external interrupts (EIVPR*n*[S]= 1). Setting them for edge-sensitive does not provide reliable interrupt response. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the external interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 10.1.3, "Interrupts to the Processor Core."
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int.* Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

10.3.7.3 Internal Interrupt Vector/Priority Registers (IIVPR*n*)

The IIVPRs, shown in Figure 10-40, have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in Table 10-3. These interrupts are all level-sensitive.

NOTE

Because all internal interrupts are active-high, clearing the polarity field, IIVPR*n*[P], disables that interrupt. Care should be taken to ensure this field is set during initialization and that it is not inadvertently corrupted when loading or reloading IIVPRs with priority, mask, or vector data.

Offset IIVPR0–7 0x0200, 0x0220, 0x0240, 0x0260, 0x0280, 0x02A0, 0x02C0, 0x02E0 Access: Read/Write
IIVPR8–15 0x0300, 0x0320, 0x0340, 0x0360, 0x0380, 0x03A0, 0x03C0, 0x03E0
IIVPR16–23 0x0400, 0x0420, 0x0440, 0x0460, 0x0480, 0x04A0, 0x04C0, 0x04E0
IIVPR24–31 0x0500, 0x0520, 0x0540, 0x0560, 0x0580, 0x05A0, 0x05C0, 0x05E0
IIVPR32–39 0x0600, 0x0620, 0x0640, 0x0660, 0x0680, 0x06A0, 0x06C0, 0x06E0
IIVPR40–47 0x0700, 0x0720, 0x0740, 0x0760, 0x0780, 0x07A0, 0x07C0, 0x07E0
IIVPR48–55 0x0800, 0x0820, 0x0840, 0x0860, 0x0880, 0x08A0, 0x08C0, 0x08E0, IIVPR56–63 0x0900, 0x0920, 0x0940, 0x0960, 0x0980, 0x09A0, 0x09C0, 0x09E0

	0	1	2	7	8	9	11	12	15	16	31
R W	MSK	A	—		P	—		PRIORITY		VECTOR	
Reset	1	0	0 0 0 0 0 0		0	0 0 0		0 0 0 0		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Figure 10-40. Internal Interrupt Vector/Priority Registers (IIVPRs)

Table 10-40 describes the IIVPR fields.

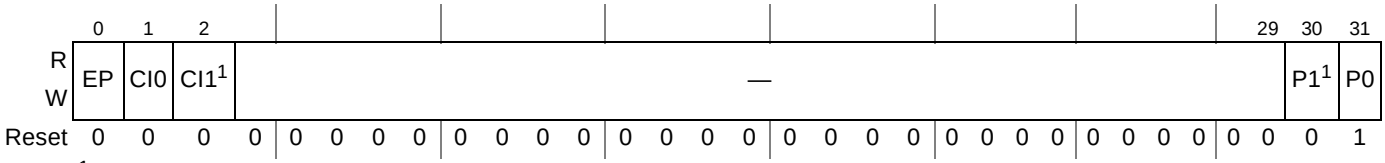
Table 10-40. IIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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.
2–7	—	Reserved, should be cleared.
8	P	Polarity. Specifies the polarity for the internal interrupt. Note: Because all internal interrupts are active-high, clearing this bit disables the interrupt. 0 Interrupt polarity is active-low. This value disables the interrupt. 1 Interrupt polarity is active-high.
9–11	—	Reserved, should be cleared.
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 10-50.

10.3.7.4 Internal Interrupt Destination Registers (IIDR*n*)

The IIDRs, shown in Figure 10-41, have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources listed in Table 10-3. Only one destination bit may be set; otherwise, behavior is undefined.

Offset IIDR0–70x0210, 0x0230, 0x0250, 0x0270, 0x0290, 0x02B0, 0x02D0, 0x02F0 Access: Read/Write
IIDR8–15 0x0310, 0x0330, 0x0350, 0x0370, 0x0390, 0x03B0, 0x03D0, 0x03F0
IIDR16–23 0x0410, 0x0430, 0x0450, 0x0470, 0x0490, 0x04B0, 0x04D0, 0x04F0
IIDR24–31 0x0510, 0x0530, 0x0550, 0x0570, 0x0590, 0x05B0, 0x05D0, 0x05F0
IIDR32–39 0x0610, 0x0630, 0x0650, 0x0670, 0x0690, 0x06B0, 0x06D0, 0x06F0
IIDR40–47 0x0710, 0x0730, 0x0750, 0x0770, 0x0590, 0x07B0, 0x07D0, 0x07F0
IIDR48–53 0x0810, 0x0830, 0x0850, 0x0870, 0x0890, 0x08B0, 0x08D0, 0x08F0
IIDR54–63 0x0910, 0x0930, 0x0950, 0x0970, 0x0990, 0x09B0, 0x09D0, 0x09F0



[1] Reserved in single-processor implementations.

Figure 10-41. Internal Interrupt Destination Registers (IIDRs)

Table 10-41 describes the IIDR fields.

Table 10-41. IIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows internal interrupt to be serviced externally. 0 Interrupt is not routed to IRQ_OUT. 1 Interrupt is routed to IRQ_OUT for external service.
1	CI0	Critical interrupt 0. See Section 10.1.3, "Interrupts to the Processor Core," for more information. 0 Processor core 0 does not receive this interrupt. 1 Directs the internal interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert.
2	CI1	Critical interrupt 1. See Section 10.1.3, "Interrupts to the Processor Core," for more information. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the internal interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert.
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int*. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

10.3.7.5 Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

The MIVPRs have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

Offset MIVPR0: 0x1600; MIVPR1: 0x1620; MIVPR2: 0x1640; MIVPR3: 0x1660
MIVPR4: 0x1680; MIVPR5: 0x16A0; MIVPR6: 0x16C0; MIVPR7: 0x16E0 Access: Read/Write

	0	1	2–11	12–15	16–31
R	MSK	A	—	PRIORITY	VECTOR
W					
Reset	1	0	0 0 0 0 0 0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 10-42. Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

Table 10-42 describes the MIVPR*n* fields.

Table 10-42. MIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects only interrupts routed to *int*. 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.

Table 10-42. MIVPR*n* Field Descriptions (continued)

Bits	Name	Description
2–11	—	Reserved, should be cleared.
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 10-50.

10.3.7.6 Messaging Interrupt Destination Registers (MIDR0–MIDR7)

The messaging interrupt destination registers (MIDRs), shown in Figure 10-43, control the destination for the messaging interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Offset MIDR0: 0x1610; MIDR1: 0x1630; MIDR2: 0x1650; MIDR3: 0x1670
MIDR0: 0x1690; MIDR1: 0x16B0; MIDR2: 0x16D0; MIDR3: 0x16F0 Access: Read/Write

	0	1	2	3–29	30	31
R W	EP	CI0	CI1[1]	—	P1[1]	P0
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] Reserved in single-processor implementations.

Figure 10-43. Messaging Interrupt Destination Registers (MIDR*n*)

Table 10-43 describes the MIDR*n* fields.

Table 10-43. MIDR*n* Field Descriptions

Bits	Name	Description
0	EP	External signal. Allows messaging interrupt to be serviced externally. 0 Interrupt is not routed to $\overline{\text{IRQ\_OUT}}$. 1 Message interrupt is routed to $\overline{\text{IRQ\_OUT}}$ for external service.
1	CI0	Critical interrupt 0. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 by causing the *cint0* output signal from the PIC to assert. See Section 10.1.3, “Interrupts to the Processor Core,” for more information.
2	CI1	Critical interrupt 1. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 by causing the *cint1* output signal from the PIC to assert. See Section 10.1.3, “Interrupts to the Processor Core,” for more information.
3–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt through *int*. Reserved in single-processor implementations. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1 through the assertion of *int1*.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0 through the assertion of *int0*. The default destination is for processor core 0 to receive this external interrupt after the PIC is reset.

10.3.8 Per-CPU (Private Access) Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To comply with the OpenPIC specification, the PIC incorporates several of these multiprocessor capabilities.

NOTE

Note that these registers are meaningful only for interrupts routed to *int*.

The registers in Table 10-44 are called per-CPU registers because they are duplicated for each core in a multi-core 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 10-44. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
Interprocessor 0 dispatch register (IPIDR0)	0x0040
Interprocessor 1 dispatch register (IPIDR1)	0x0050
Interprocessor 2 dispatch register (IPIDR2)	0x0060
Interprocessor 3 dispatch register (IPIDR3)	0x0070
Current task priority register (CTPR)	0x0080
Who am I register (WHOAMI0)	0x0090
Interrupt acknowledge register (IACK)	0x00A0
End of interrupt register (EOI)	0x00B0

These addresses, shown in Table 10-44, appear in the memory map at the same offset for every processor in what is called the private access space. This duplication allows user code to execute correctly in an multiprocessor environment without needing to know which core it is running on. On a single-core device, each register has two addresses, one in the normal address space and one in the private access space. It is included on even single-core devices to simplify the porting of such code.

Figure 10-44 shows how the duplicated registers are addressed in a four-core device Note that when accessing a register normally, each core sources a different address. However, when accessing the same register using the per-CPU address space, each core sources the same address.

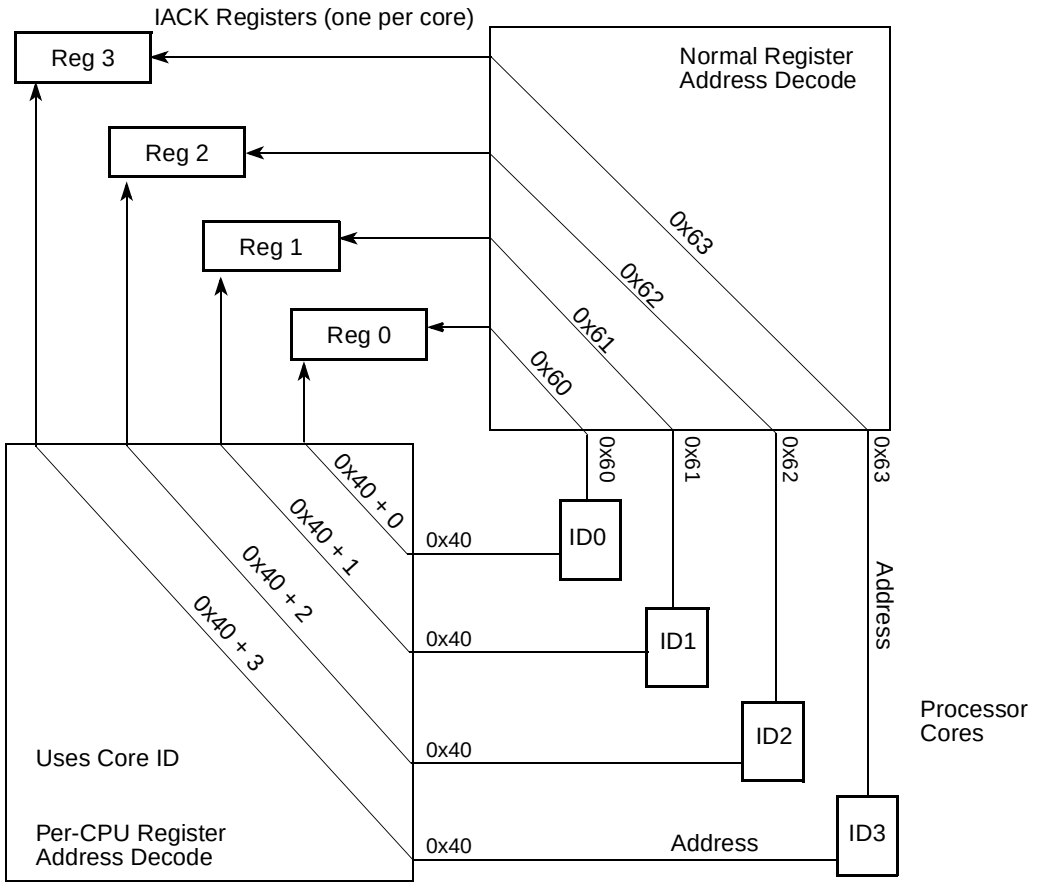


Figure 10-44. Per-CPU Register Address Decoding in a Four-Core Device

10.3.8.1 Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)

Figure 10-45 shows the four IPIDRs, one for each interprocessor interrupt channel. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device. Because external bus masters can write to these registers, this feature can serve as a doorbell type interrupt.

Offset Processor core 0: IPIDR0: 0x0040; IPIDR1: 0x0050; IPIDR2: 0x0060; IPIDR3: 0x0070 Access: Write only
Processor core 1[1]:IPIDR0: 0x1040; IPIDR1: 0x1050; IPIDR2: 0x1060; IPIDR3: 0x1070
Pre-CPU offsets: IPIDR0: 0x0040; IPIDR1: 0x0050; IPIDR2: 0x0060; IPIDR3: 0x0070

	0	30	31
R	—		
W		P1[1]	P0
Reset	All zeros		

[1] Reserved in single-processor implementations.

Figure 10-45. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 10-40 describes the IPIDR*n* fields.

Table 10-45. IPIDR*n* Field Descriptions

Bits	Name	Description
0–29	—	Reserved, should be cleared.
30	P1	Processor core 1. Specifies if processor core 1 receives the interrupt. Reserved in single-processor implementations. This interrupt is multicasting, so both P0 and P1 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.

10.3.8.2 Processor Core Current Task Priority Registers 0–1 (CTPR0–CTPR1)

There is one CTPR per processor core on this device as shown in Figure 10-46.

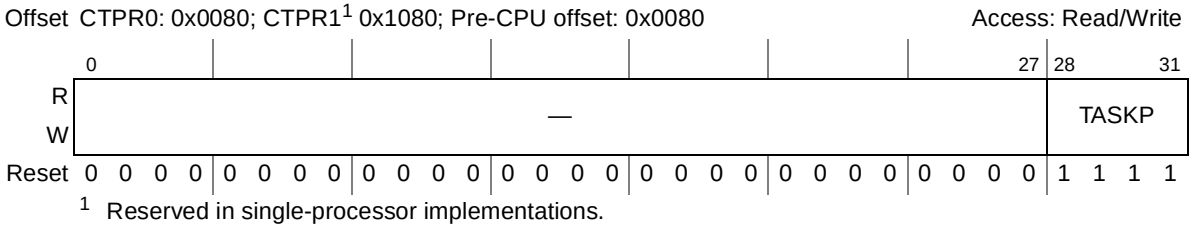


Figure 10-46. Processor Core Current Task Priority Registers (CTPR*n*)

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 PIC 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.

Table 10-46 describes the CTPR task priority field.

Table 10-46. CTPR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	TASKP	Task priority. Indicates the threshold that individual interrupt priorities must exceed for the interrupt request to be serviced. 0000–1111 *x*VPR*n*[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[P*n*] is set.

10.3.8.3 Who Am I Registers 0–1 (WHOAMI0–WHOAMI1)

The processor core WHOAMI*n* register, shown in Figure 10-47, can be read by a processor core to determine its physical connection to the PIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

Offset WHOAMI0: 0x0090; WHOAMI1[1]: 0x1090; Per-CPU offset: 0x0090 Access: Read only

	0 – 26	27 – 31
R	—	ID
W		
Reset	n/a	

[1] Reserved in single-processor implementations.

Figure 10-47. Processor Core Who Am I Registers (WHOAMI*n*)

Table 10-47 describes the WHOAMI*n* fields.

Table 10-47. WHOAMI*n* Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27–31	ID	Returns the ID of the processor core reading this register. 0_0000 Processor core 0 0_0001 Processor core 1. (Value not supported in single-processor implementations.) 1_1111 Other devices

10.3.8.4 Processor Core Interrupt Acknowledge Registers 0–1 (IACK0–IACK1)

NOTE

IACK has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint* or $\overline{\text{IRQ\_OUT}}$.

In systems based on processors built on Power Architecture™ technology, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK), shown in Figure 10-48. 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 PIC is negated.

Reading IACK when no interrupt is pending returns the spurious vector value, as described in Section 10.3.1.8, “Spurious Vector Register (SVR).”

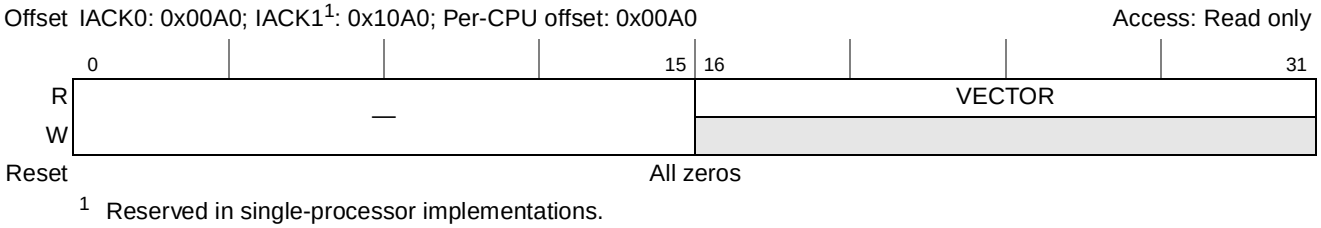


Figure 10-48. Processor Core Interrupt Acknowledge Registers (IACK*n*)

Table 10-48 describes the IACK*n* fields.

Table 10-48. IACK*n* Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only)

10.3.8.5 Processor Core End of Interrupt Registers (EOI0–EOI1)

NOTE

EOI has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint* or $\overline{\text{IRQ\_OUT}}$.

Each core is assigned an EOI register, shown in Figure 10-49. 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.

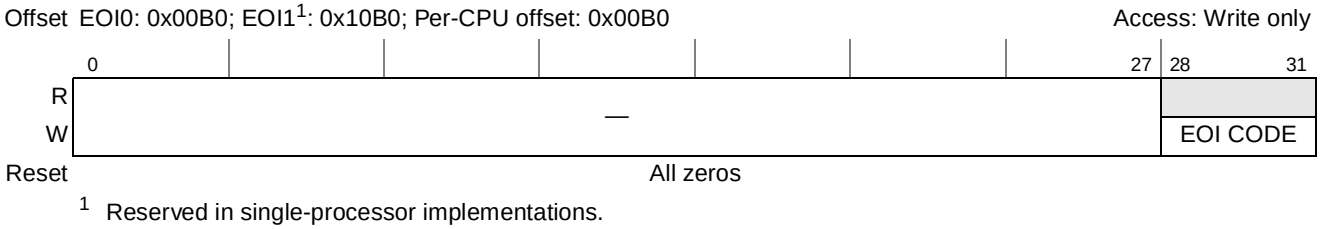


Figure 10-49. End of Interrupt Registers (EOI*n*)

Table 10-49 describes the EOI*n* fields.

Table 10-49. EOI*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	EOI CODE	0000 (write only)

10.4 Functional Description

This section is a functional description of the PIC.

10.4.1 Flow of Interrupt Control

Figure 10-50 shows the flow of interrupts directed by the PIC to the *int*, *cint*, and $\overline{\text{IRQ\_OUT}}$ outputs. Note that this diagram describes a conceptual model of an PIC 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 focusses especially on the OpenPIC-defined logic and shows how the PIC controls interrupt requests that target the *int* signal. The flow in Figure 10-50 is from the bottom to the top, and shows at the bottom how the destination register associated with each source determines the path.

10.4.1.1 Interrupts Routed to *cint* or $\overline{\text{IRQ\_OUT}}$

Interrupt requests routed to *cint* or $\overline{\text{IRQ\_OUT}}$ bypass the logic that is dedicated to interrupt sources that compete for *int*. That is, if *x*IDR*n*[CI*n*] or *x*IDR*n*[EP] = 1, corresponding *x*IVPR field settings have no hardware effects; however, an interrupt handler may be able to make use of some of those fields.

cint signals are connected to the respective core's critical interrupt input.

NOTE

Because interrupt sources routed to *cint* or $\overline{\text{IRQ\_OUT}}$ must be level sensitive, EIVPR[S] should be set. See Section 10.3.7.1, "External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)."

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.

10.4.1.2 Interrupts Routed to *int*

As shown in Figure 10-50, the PIC receives interrupt requests from external and internal sources and from within the PIC itself. As Figure 10-50 shows, all of these interrupt sources can be routed to *int*; the global timer and timer processor interrupts can be directed only to *int*.

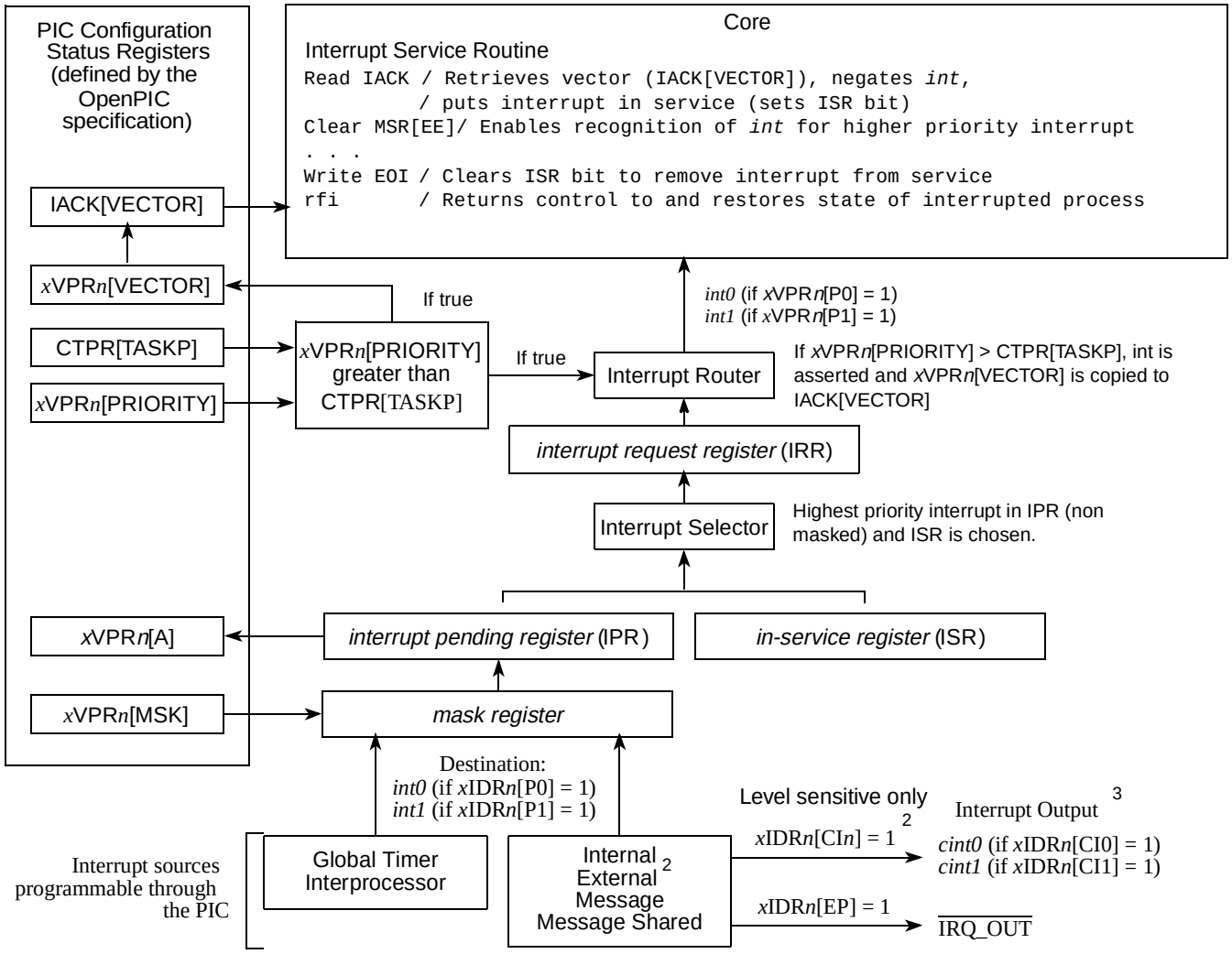
The sources' mask bits (*x*VPR*n*[MSK]) are tracked in the internal mask register. If a source's MSK bit is set, the mask register prevents the PIC from asserting *int* on its behalf.

Unmasked interrupt requests are qualified and latched in the interrupt pending register (IPR), an internal interrupt summary register with a bit for each source. If an interrupt request is multi-cast, 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 (*x*VPR*n*[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 (*x*VPR*n*[VECTOR]) is copied to IACK[VECTOR], which the interrupt handler retrieves by reading IACK.

If the priority (*x*VPR*n*[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.



1 If *cint* or IRQ_OUT is the destination, EIVPR*n*[S] must be set to configure the source as level sensitive.

2 If multiple destination register bits are set, PIC behavior is undefined.

3 Although setting CI*n* directs the interrupt request to the critical interrupt output (*cint0*/*cint1*), integrated logic may connect this signal to a different interrupt input to the core.

Figure 10-50. PIC Interrupt Processing Flow Diagram for Each Core (*n*)

The interrupt handler must acknowledge the interrupt by explicitly reading the corresponding IACK register, described in Section 10.3.8.4, "Processor Core Interrupt Acknowledge Registers 0–1 (IACK0–IACK1)." The PIC interprets this read as an interrupt acknowledge (IACK) cycle. 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 PIC considers an EOI signal.

Figure 10-50 shows required elements in the interrupt handler

10.4.1.2.1 Nesting of Interrupts

While an interrupt is being handled, if an interrupt request arrives with a higher *x*VPR*n*[PRIORITY] value, the interrupt being serviced is superseded. As described in Section 10.4.1.2, "Interrupts Routed to *int*," the PIC 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 PIC 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.

10.4.1.2.2 Interrupt Source Priority

Each interrupt source routed to *int* is assigned a value through its *x*VPR*n*[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 *x*VPR*n*[PRIORITY] to zero inhibits that interrupt. Likewise, setting TASKP to 15 prevents the PIC from delivering interrupts to that core through the *int* signal. Note that this is the reset value, preventing the PIC from asserting *int* before the PIC is configured.

The PIC services simultaneous interrupts occurring with the same priority according to the following order:

1. MSG0–MSG7
2. MSI0–MSI7
3. IPI0–IPI3
4. Group A timer0–timer3
5. Group B timer0–timer3
6. IRQ[0:11]/PCI INT*x*
7. Internal0–internal63

For example, if MSG0, MSG2, and IPI0 are all assigned the same priority and receive simultaneous interrupts, they are serviced in the following order:

1. MSG0
2. MSG2
3. IPI0

10.4.1.2.3 Interrupt Acknowledge

When the PIC causes *int* to be asserted, the external interrupt service routine acknowledges the request by reading that core's IACK register, 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 *x*VPR*n*[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 PIC to signal a higher-priority interrupt, as described in Section 10.4.1.2.1, "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.

10.4.1.2.4 Spurious Vector Generation

Under certain circumstances, the PIC 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* 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* 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* 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 int signal has not been asserted by the PIC.

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.

10.4.2 Interprocessor Interrupts

Processors 0 and 1 can generate interprocessor interrupts that target either or both processors. A self interrupt occurs when a core dispatches an interprocessor interrupt event to itself. Interrupts are initiated

by writing either or both of the PO*n* bits in an interprocessor interrupt dispatch register (IPIDR0–IPIDR3) of one of the four IPI channels. 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.

10.4.3 Message Interrupts

The eight MSGRs, described in Section 10.3.5.1, "Message Registers (MSGR0–MSGR7)," 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.

10.4.4 Shared Message Signaled Interrupts

There are eight shared MSIRs, described in Section 10.3.6.1, "Shared Message Signaled Interrupt Registers (MSIR0–MSIR7)," 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 inbound PCI Express message signaled interrupts (MSIs) when the PCI Express controller is configured as a root complex (RC).

MSIIR[SRS] selects the associated MSIR and MSIIR[IBS] selects the interrupt flag/bit in that register that is to be set. The corresponding interrupt needs to be unmasked for the interrupt to occur. A read to an MSIR clears the all of its flags.

10.4.5 PCI Express INTx

Whenever the PCI Express controller is in root complex mode and it receives an inbound INTx asserted or negated message transaction, it asserts or negates an equivalent internal INTx signal to the PIC. This INTx virtual-wire interrupt signaling mechanism replaces the PCI standard sideband interrupts (INTA, INTB, INTC, and INTD) that historically were connected to the IRQ*n* external interrupt inputs. The internal INTx signals from the PCI Express controller are logically combined with the interrupt request (IRQ*n*) signals so that they share the same OpenPIC external interrupt controlled by the associated EIVPR*n* and EIDR*n* registers.

Table 10-50details the association of INTx signals to IRQ*n* signals.

Table 10-50. PCI Express INTx/IRQ*n* Sharing

PCI Express Number	INTx	IRQ*n*
PCI Express 1	INTA	IRQ0
	INTB	IRQ1
	INTC	IRQ2
	INTD	IRQ3

Table 10-50. PCI Express INTx/IRQ*n* Sharing (continued)

PCI Express Number	INTx	IRQ*n*
PCI Express 2	INTA	IRQ4
	INTB	IRQ5
	INTC	IRQ6
	INTD	IRQ7
PCI Express 3	INTA	IRQ8
	INTB	IRQ9
	INTC	IRQ10
	INTD	IRQ11

In general, these signals should be considered mutually exclusive. If a PCI Express INTx signal is being used, the PIC must be configured so that external interrupts are level sensitive (EIVPRn[S] = 1). If an IRQ*n* signal is being used as edge-triggered (EIVPRn[S] = 0), the system must not allow inbound PCI Express INTx transactions.

Note that it is possible to share IRQ*n* and INTx if the external interrupt is level sensitive; however, if an interrupt occurs, the interrupt service routine must poll both the external sources connected to the IRQ*n* input and the PCI Express INTx sources to determine from which path the external interrupt came. In any case, IRQ*n* should be pulled to the negated state as determined by the associated polarity setting in EIVPR*n*[P].

10.4.6 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the internal PIC timers. These 8 timers are organized as 2 groups of 4 timers each. 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. Each timer has the following four configuration and control registers:

- Global timer current count register (GTCCR*xn*)
- Global timer base count register (GTBCR*xn*)
- Global timer vector-priority register (GTVPR*xn*)
- Global timer destination register (GTDR*xn*)

The timer frequency should be written to the TFRR*xn*, described in Section 10.3.2.1, “Timer Frequency Reporting Register (TFRRA–TFRRB).”

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 Section 10.3.2.6, “Timer Control Registers (TCRA–TCRB).”

10.4.7 Resets

This section describes the behavior of the PIC at reset and the PIC's ability to initiate processor resets.

10.4.8 Resetting the PIC

The PIC 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, MSR, and MSGR0–MSGR7 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 PIC defaults to the pass-through mode (GCR[M] = 0).
- All other registers remain at their pre-reset programmed values.

GCR[RST] is automatically cleared when the reset sequence is complete.

10.4.8.1 Processor Core Initialization

A software reset can be routed to either of the cores by writing to the processor core initialization register (PIR). This causes the assertion of the corresponding *core_hreset* output signal from the PIC. When this occurs, the corresponding CTPR also gets written to 0x000F to prevent delivery of any interrupts to *int*.

10.5 Initialization/Application Information

This section contains initialization and application information for the PIC.

10.5.1 Programming Guidelines

The following subsections contain information about programming PIC registers.

10.5.1.1 PIC Registers

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- Interprocessor dispatch and EOI registers, 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 PIC is in mixed mode (GCR[M] = 1), the following guidelines are recommended:

- All PIC registers must be located in a cache-inhibited, guarded area (configured through the core's MMU).
- The PIC 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 = 0x0000_0000).
3. Program the PIC to mixed mode by setting GCR[M].
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.
7. Read the MSIRs to clear any pending message shared interrupts that may have been pending before a soft reset.
8. Set MER to 0x0000_000F. See Section 10.3.5.2, "Message Enable Register (MER)," for more information.

Depending on the interrupt system configuration, the PIC 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 PIC 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 PIC 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.

10.5.1.2 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* or $\overline{\text{IRQ\_OUT}}$ makes the A, MSK, and PRIORITY fields meaningless.

Chapter 11
Security Engine (SEC) 3.0

This chapter describes the functionality of the device's integrated security engine (SEC 3.0). It addresses the following topics:

- Section 11.1, "Architecture Overview"
- Section 11.2, "Configuration of Internal Memory Space"
- Section 11.3, "Descriptor Overview"
- Section 11.5, "Polychannel"
- Section 11.6, "Controller"
- Section 11.6.1, "Bus Transfers"
- Section 11.7, "Execution Units"

The SEC 3.0 is designed to off-load computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the processor core of the device. It is optimized to process all cryptographic algorithms associated with IPsec, IKE, SSL/TLS, iSCSI, SRTP, 802.11i, 802.16 (WiMAX), 802.1AE (MACSec), 3GPP, A5/3 for GSM and EDGE, and GEA3 for GPRS. The SEC 3.0 is derived from integrated security cores found in other members of the PowerQUICC family, particularly the SEC 2.x used in several PowerQUICC II Pro and PowerQUICC III devices.

The security engine includes eight different execution units (EUs). Where data flows in and out of an EU, each has buffer FIFOs of at least 256 bytes. EU types and features include the following:

- AESU—Advanced Encryption Standard unit
  - Implements the Rijndael symmetric key cipher
  - Modes providing data confidentiality: ECB, CBC, CCM, Counter, GCM, CBC-RBP, OFB-128, and CFB-128.
  - Modes providing data authentication: CCM, GCM, CMAC (OMAC1), and XCBC-MAC.
  - 128-, 192-, 256-bit key lengths (only 128-bit keys in XCBC-MAC)
  - ICV checking in CCM, GCM, CMAC (OMAC1), and XCBC-MAC mode
  - XOR operations on 2 - 6 sources for RAID
- AFEU—ARC Four execution unit
  - Implements a stream cipher compatible with the RC4 algorithm
  - 8- to 128-bit programmable key
- CRCU—Cyclical redundancy check unit
  - Implements CRC32C as required for iSCSI header and payload checksums, CRC32 as required for IEEE 802 packets, and programmable CRC modes
  - ICV checking

- DEU—Data Encryption Standard execution unit
  - DES, 3DES
  - Two key (K1, K2, K1) or Three Key (K1, K2, K3)
  - ECB, CBC, CFB-64 and OFB-64 modes for both DES and 3DES
- KEU—Kasumi execution unit
  - Implements cipher and authentication modes f8 and f9 used in 3GPP, A5/3 for GSM and EDGE, and GEA3 for GPRS
  - 128-bit confidentiality key and 128-bit integrity key
  - ICV checking for f9
- MDEU—Message digest execution unit
  - Implements SHA-1 with 160-bit, 224-bit, 256-bit, 384-bit, and 512-bit message digest (as specified by the FIPS 180-2 standard)
  - Implements MD5 with 128-bit message digest (as specified by RFC 1321)
  - Implements HMAC computation with either message digest algorithm (as specified in RFC 2104 and FIPS-198)
  - Implements SSL MAC computation
  - ICV checking
- PKEU—Public key execution unit that support the following:
  - RSA and Diffie-Hellman
    - Programmable field size up to 4096 bits
  - Elliptic curve cryptography
    - F_2m and F_p modes
    - Programmable field size up to 1023 bits
  - Run time equalization to protect against timing and power attacks
- RNG—Random number generator
  - Combines a True Random Number Generator (TRNG) and NIST-approved Pseudo-Random Number Generator (PRNG) (as described in Annex C of FIPS140-2 and ANSI X9.62).

In addition to the execution units, SEC 3.0 also includes:

- A context switching polychannel, permitting operation of up to four virtual channels, where each channel:
  - Can be configured for either of the hosts
  - Supports a queue of commands (descriptor pointers) to be executed
  - Provides dynamic arbitration for needed crypto-execution units
  - Manages up to two execution units (one ciphering and one hashing), and configures for any required data transfers from one to another
  - Performs flow-control management of buffer FIFOs on the inputs and outputs of execution units

- Supports scatter/gather of input and output data (where the term data is used loosely, and includes keys, context, ICV values, etc.), enabling concatenation of multiple segments of memory when reading or writing data
- bursts on 32-byte boundaries to optimize bus bandwidth

- Master and slave interfaces, with DMA capability
  - 32- or 36-bit address/64 -bit data
  - Master interface allows multiple pipelined requests
  - DMA data blocks can start and end on any byte boundary

11.1 Architecture Overview

An example position of SEC 3.0 (henceforth referred to as SEC) is shown in, which is only an example and different PowerQUICC platforms may vary. The SEC can act as a master on the internal system bus, allowing it to off-load the data movement bottleneck normally associated with slave-only cores. The host processor accesses the SEC through its device drivers using system memory for data storage. The SEC resides in the peripheral memory map of the processor. When an application requires cryptographic functions, it simply creates descriptors for the SEC, which define the cryptographic function to be performed and the locations of the data. With a single 64-bit write, the host processor can enqueue a descriptor pointer in the SEC. The SEC's bus-mastering capability then enables it to execute the entire cryptographic task, performing reads and writes on system memory as needed.

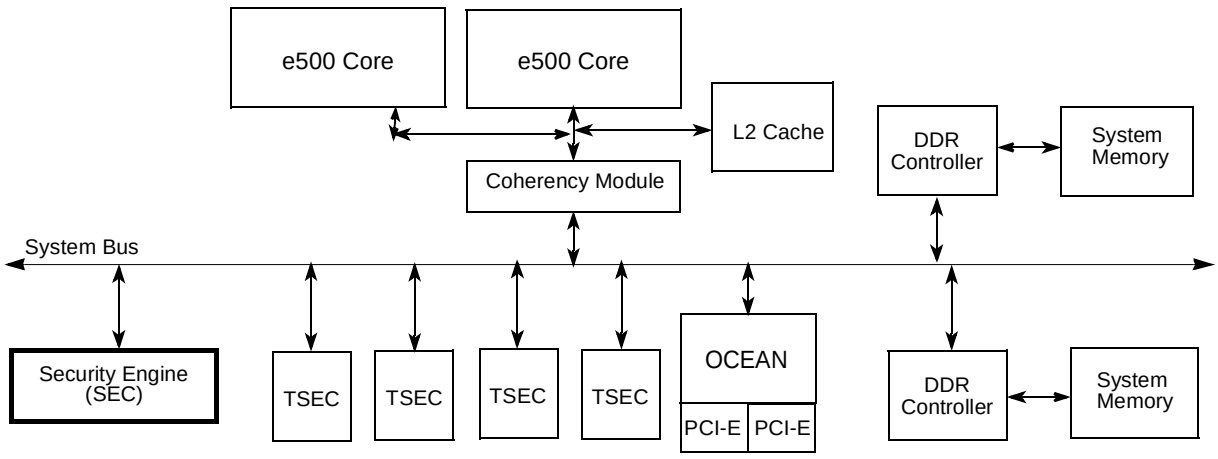


Figure 11-1. SEC Connected to MPC8572E System Bus

A block diagram of the SEC internal architecture is shown in Figure 11-2.

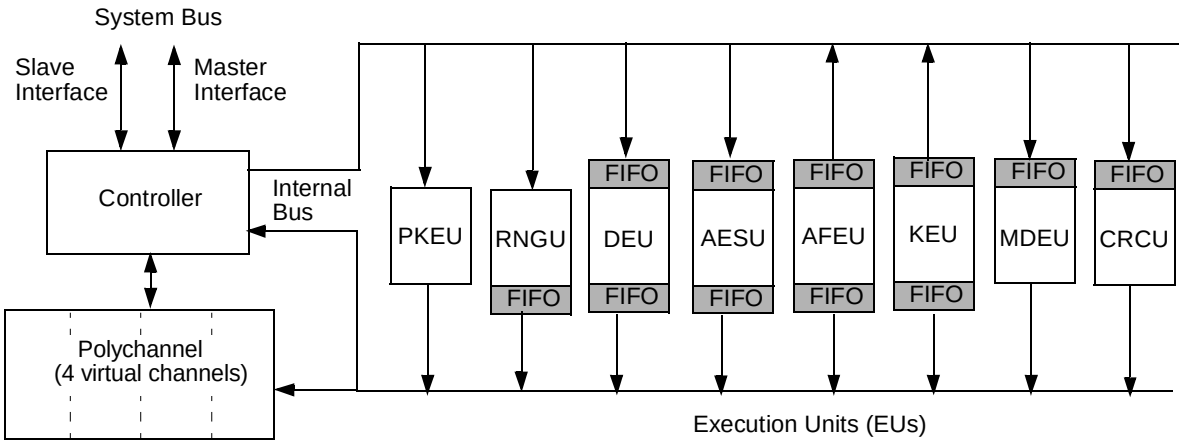


Figure 11-2. SEC Functional Modules

The SEC interfaces with the system buses through the controller. The Slave Interface permits an external device to perform 32- or 64-bit writes on any register or FIFO inside the SEC Some locations permit byte writes. Reads may be of any length. Using the Master Interface, the controller can transfer blocks of 64-bit words between system memory and SEC FIFOs or registers.

A typical SEC operation begins when the host writes a descriptor pointer to the Fetch FIFO in one of the four SEC virtual channels. This write operation uses the slave interface (where the host is master and SEC is the slave). From this point on, the channel directs the sequence of operations, using the master interface (where SEC is master). The channel uses the descriptor pointer to read the descriptor, then decodes the first word of the descriptor to determine the operation to be performed and the crypto-execution units needed to perform it. If necessary, the channel waits for the needed crypto-execution units to be free. Next, the channel requests that the controller transfer keys, context and data from memory locations specified in the descriptor to the appropriate execution units. The controller satisfies the requests through its master interface. Data is fed into the execution units through their registers and input FIFOs. The execution units read from their input FIFOs and write processed data to their output FIFOs and registers. The channel requests the controller to write data from the output FIFOs and registers back to system memory.

The channel can signal to the host that it is done with a descriptor via interrupt or by a writeback of the descriptor header into host memory. For more about this signalling, see Section 11.1.2, “Polychannel.”

Upon completion of a descriptor, the channel checks the next entry in its Fetch FIFO, and, if non-zero, the channel requests a read of the next descriptor.

For most packets, the entire payload is too long to fit in an execution unit’s input or output FIFO, so the channel uses a flow control scheme for reading and writing data. The channel directs the controller to read bursts of input as necessary to keep refilling the input FIFO, until the entire payload has been fetched. Similarly, the channel directs the controller to write bursts of output whenever enough accumulates in the execution unit’s output FIFO.

The Polychannel can process up to four descriptors concurrently by implementing four virtual channels. Channels arbitrate for use of execution units, and wait if the needed execution unit is currently reserved

by another channel. Each channel has its own FIFO of descriptor pointers to fetch and execute, and its own internal storage. The four channels, however, time-share a single control and datapath unit, and hence they are referred to as virtual channels. A programmable priority scheme allows for round-robin or weighted priorities among these channels.

11.1.1 Descriptors

The SEC controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through descriptors. A descriptor specifies a cryptographic function to be performed, and contains reference address pointers to all necessary input data and to the places where output data is to be written. Some descriptor types perform multiple functions to facilitate particular protocols. A sample descriptor is diagrammed in Table 11-1.

Each descriptor contains eight dwords (64 bits each), consisting of the following:

- One dword of header—The header describes the required services and encodes information that indicates which EUs to use and which modes to set. It also indicates whether notification should be sent to the host when the descriptor operation is complete.
- Seven dwords containing pointers and lengths used to locate input or output data. Each pointer can either point directly to the data, or can point to a link table that lists a set of data segments to be concatenated.

Table 11-1. Example Descriptor

Field Name	Value	Description
Header	2053_1E08_0000_0000 (hex)	Example header for IPsec ESP outbound using DES and MD-5
Length0 Extent0 Pointer0	16 0 (32- or 36-bit pointer)	Number of bytes in authenticate key Unused Pointer to authentication key
Length1 Extent1 Pointer1	16 0 (32- or 36-bit pointer)	Number of bytes in authentication-only data Unused Pointer to authentication-only data
Length2 Extent2 Pointer2	8 0 (32- or 36-bit pointer)	Length of input context (IV) Unused Pointer to input context
Length3 Extent3 Pointer3	8 0 (32- or 36-bit pointer)	Number of bytes in cipher key Unused Pointer to cipher key
Length4 Extent4 Pointer4	1500 0 (32- or 36-bit pointer)	Number of bytes of data to be ciphered Unused Pointer to input data to perform ciphering upon

Table 11-1. Example Descriptor (continued)

Field Name	Value	Description
Length5 Extent5 Pointer5	1500 12 (32- or 36-bit pointer)	Number of bytes of data after ciphering Number of bytes in authentication result (ICV) Pointer to location where cipher output is to be written, followed by ICV
Length6 Extent6 Pointer6	8 0 (32- or 36-bit pointer)	Length of output Context (IV) Unused Pointer to location where altered Context is to be written

For more information, refer to Section 11.3, “Descriptor Overview.”

11.1.2 Polychannel

The Polychannel block implements four channels for processing descriptors. Each channel contains the following addressable structures:

- A Fetch FIFO, which holds a queue of pointers to descriptors waiting to be processed
- A Configuration Register, which allows the user a number of options for SEC event signalling
- A Status Register containing an indication of the last unfulfilled bus request
- A Descriptor Buffer memory used to store the active descriptor and other temporary

Whenever a channel is idle and its Fetch FIFO is non-empty, the channel reads the next descriptor pointer from the Fetch FIFO. Using this pointer, the channel fetches the descriptor and places it in its descriptor buffer. Typical operations performed by a channel to process a descriptor are:

1. Analyze the descriptor header to determine the cryptographic services required, and arbitrate for the appropriate EUs. If required EUs are already reserved by another channel, wait for the EUs to be available. When available, reserve them.
2. Set the mode register in each reserved EU(s) for the required EU function.
3. Fetch “parcels” (up to 64K-1 bytes long) from system memory using pointers from the descriptor buffer, and place them in either an EU input FIFO or EU registers, as appropriate. The term “parcel” refers here to any input or output of an EU algorithm, such as a key, hash result, input context, output context, or text-data. “Context” refers to either an IV (initialization vector) or other internal EU state that can be read out or loaded in. “Text-data” refers to plaintext or ciphertext to be operated on. Each parcel transfer may involve using link tables to gather input data that has been split into multiple segments in system memory.
4. Take data accumulated in the EU output FIFO and write it to system memory using pointers from the descriptor buffer. This may again involve using link tables to scatter output data into multiple segments in system memory.
5. If the data size is greater than EU FIFO size, continue fetching input data and writing output data to memory as needed.
6. After writing the last input data to each EUs input FIFO, write to the end_of_message register in the EU.

7. Wait for EU(s) to complete processing of text-data.
8. Unload final results from output FIFOs and context registers and write them to external memory using pointers from the descriptor buffer. This may again involve using link tables to scatter output data into multiple segments in system memory.
9. Reset and release the EUs.
10. If “done notification” is enabled, perform this notification to notify the host of descriptor completion.

A channel can signal to the host that it is done with a descriptor via interrupt and/or by a writeback of the descriptor header into host memory. In the case of writeback, the value written back is identical to the header that was read, with the exception that a DONE byte is set to all 1s. The user can opt to do this signalling at end of every descriptor, or at end of selected descriptors.

An EU operation can include generating an ICV and then comparing it against a received ICV. The result of the ICV checking can be signalled to the host either via interrupt or by a writeback of the descriptor header. If both are enabled, note that the occurrence of an error interrupt prevents the writeback from occurring. In the case of writeback, the user can opt to do it at end of every descriptor, or only at the end of descriptors that call for ICV checking.

In case of an error condition in a channel or its reserved EUs, the channel issues an interrupt to the host. The channel can be configured to either abort the current descriptor and proceed to the next one, or halt and wait for host intervention.

For more about configuring signalling see Section 11.5.4.1, “Channel Configuration Register (CCR),” and for detail on the writeback fields see Section 11.3.4, “Link Table Format.”

Many security protocols involve both encryption and hashing of packet payloads. To accomplish this without requiring two passes through the data, channels can configure data flows through two EUs. In such cases, one EU is designated the “primary EU”, and the other as the “secondary EU”. The primary EU receives its data from memory via the controller, and the secondary EU receives its data by “snooping” the SEC buses.

There are two types of snooping:

- Input data can be fed to the primary EU and the same input data snooped by the secondary EU. This is called “in-snooping”.
- Output data from the primary EU can be snooped by the secondary EU. This is called “out-snooping”.

In the SEC, only MDEU and CRCU are used as secondary EUs.

For more information, refer to Section 11.5, “Polychannel.”

11.1.3 Controller

The controller manages the master and slave interfaces to the PowerQUICC system bus and the internal buses that connect all the various modules. It receives service requests from the host (via the slave interface) and from the channels, and schedules the required data transfers. The system bus interface and

access to system memory are critical factors in performance, and the 64-bit master and slave interfaces of the SEC controller enable it to achieve performance unattainable on secondary buses.

The controller provides for two ways of operating the execution units, channel-controlled access and host-controlled access.

NOTE

Host-controlled access of execution units is provided primarily for system debug purposes. SEC contains no mechanism to prevent the Host from accessing an execution unit assigned to a Channel, and it provides no mechanism for preventing assigning an execution unit being used by the host to a Channel. Simultaneous use of an execution unit by a Channel and a Host is a very effective way to force the execution unit into an error condition.

11.1.3.1 Channel-Controlled Access to EUs

This is the normal way to operate SEC. In channel-controlled access, all interactions with EUs are directed by a channel executing a descriptor. The host is involved only in initially supplying the descriptor pointer and in handling results once descriptor processing is complete. The typical steps for descriptor processing are given in Section 11.1.2, “Polychannel.” In these steps, all the reads and writes of system memory and other register transfers are performed by the channel making requests to the controller, and the controller actually making the transfer. When transfers involve data in system memory, the controller acts as a master on the system bus.

11.1.3.2 Host-Controlled Access to EUs

In host-controlled access, the host moves data into and out of execution units directly through memory-mapped EU registers. No descriptor is involved. This type of operation is recommended only for debug purposes.

Under host control, the SEC operates as a slave, and the host must write the information typically provided through the descriptor into the appropriate registers and FIFOs of the EUs. This is much more CPU intensive than channel-controlled access, and requires a great deal of familiarity with the EU and controller registers and procedures. If host-controlled access is used, it is recommended that only a single EU be operated at a time. Snooping is not available through this interface.

NOTE

Host-controlled access of execution units is provided primarily for system debug purposes. SEC contains no mechanism to prevent the Host from accessing an execution unit assigned to a Channel, and it provides no mechanism for preventing assigning an execution unit being used by the host to a Channel. Simultaneous use of an execution unit by a Channel and a Host is a very effective way to force the execution unit into an error condition.

For more information about the controller, refer to Section 11.6, “Controller.”

11.1.4 Execution Units (EUs)

"Execution unit" (EU) is the generic term for a functional block that performs the cryptographic computations. The EUs are compatible with many protocols, and can work together to perform high-level cryptographic tasks. The SEC's execution units are as follows:

- PKEU for computing asymmetric key operations, including modular exponentiation (and other modular arithmetic functions) or ECC point arithmetic
- DEU for performing block cipher, symmetric key cryptography using DES and 3DES
- AESU for performing the Advanced Encryption Standard algorithm in various modes
- AFEU for performing RC-4 compatible stream cipher symmetric key cryptography
- MDEU for performing security hashing using MD-5, SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512
- KEU for performing 3GPP confidentiality and integrity (f8 and f9) algorithms.
- CRCU for generating cyclical redundancy check values
- RNGU for random number generation

The following sections give an overview of the EUs. Operational details of each EU are given in Section 11.7, "Execution Units."

11.1.4.1 Common EU Interface

All EUs use a standardized interface to the channel and controller. As shown in Figure 11-2, the main access to the EUs is through common input and output buses. EU FIFOs and registers are memory-mapped on these buses such that each EU is allocated 4 Kbytes. Within each EU memory segment, standard addresses are used for common register types. In addition to this bus interface, each EU supplies three interrupt signals to the controller. Two of these, done interrupt and error interrupt, are routed by the controller to the channel currently using that EU.

11.1.4.2 Public Key Execution Unit (PKEU)

The PKEU is capable of performing many advanced mathematical functions to support both RSA and ECC public key cryptographic algorithms. ECC is supported in both F_2m (polynomial field) and F_p (prime field) modes.

To assist the host in performing its desired cryptographic functions, the PKEU supports functions with various levels of complexity. For example, at the highest level, the accelerator performs modular exponentiations to support RSA and performs point multiplies to support ECC. At a lower level, the PKEU can perform simple operations such as modular adds and multiplies. For more information, refer to Section 11.7.7, "Public Key Execution Units (PKEU)."

11.1.4.2.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 33 bits and 1024 bits in programmable increments of 8, with each programmable value *i* supporting all

actual field sizes from 8*i*–7 to 8*i*. The result is hardware supporting a wide range of cryptographic security. Larger field / modulus sizes result in greater security but lower performance.

Compared to RSA, elliptic curve cryptography provides greater security with smaller field sizes. For example, an elliptic curve field size of 160 is roughly equivalent to the security provided by 1024 bit RSA. A field size set to 224 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations which are addition, multiplication, inverse, and squaring. Point add and double each use of all four finite field operations. Similarly, point multiplication uses all elliptic curve point operations as well as the finite field operations. All these functions are supported both in prime fields and polynomial fields.

11.1.4.2.2 Modular Exponentiation Operations

The PKEU is also capable of performing integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures (including ECDSA) and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the SEC's PKEU include the following (refer to Table 11-68 for a complete list):

- R^2 mod N
- (A × B) R^{-1} mod N
- (A × B) R^{-2} mod N
- (A + B) mod N
- (A – B) mod N

(where N is the modulus vector, A and B are input vectors, E is the exponent vector, and R is $2^{Sz'(N)}$, where Sz'(N) is the bit length of the N vector rounded up to the nearest multiple of 32)

The PKEU can perform modular arithmetic on operands up to 4096 bits in length. The modulus must be larger than or equal to 33 bits (5 bytes). This is not seen as a limitation since for sizes smaller than this, no useful cryptographic application exists. Furthermore, for small data sizes the overhead of using a hardware accelerator would not be justified. The PKEU uses the Montgomery modular multiplication algorithm to perform core functions. The addition and subtraction functions help support known methods of the Chinese Remainder Theorem (CRT) for efficient implementation of the RSA algorithm.

11.1.4.3 Data Encryption Standard Execution Unit (DEU)

The DES Execution Unit (DEU) performs bulk data encryption/decryption, in compliance with the Data Encryption Standard algorithm (NIST FIPS 46-3). The DEU can also compute 3DES, an extension of the DES algorithm in which each 64-bit input block is processed three times. The SEC supports 2 key (K1=K3) or 3 key 3DES.

The DEU operates by permuting 64-bit data blocks with a shared 56-bit key and an initialization vector (IV). The SEC supports four modes of operation:

- Electronic Code Book (ECB)
- Cipher Clock Chaining (CBC)
- 64-bit Cipher Feedback Mode (CFB-64)
- 64-bit Output Feedback Mode (OFB-64).

For more information, refer to Section 11.7.4, “Data Encryption Standard Execution Unit (DEU).”

11.1.4.4 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the Advanced Encryption Standard algorithm Rijndael specified by NIST standard FIPS-197. The AESU executes on 128 bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AESA is a symmetric key algorithm, meaning the sender and receiver use the same key for encryption and decryption. The session key and IV are supplied to the AESU module prior to encryption. The processor supplies data to the module that is processed as 128 bit input.

AESU implements the following confidentiality Modes from NIST Recommendation 800-38A:

- Electronic Codebook mode (ECB)
- Cipher Block Chaining mode (CBC)
- Output Feedback mode (OFB)
- 128-bit Cipher Feedback mode (CFB-128)
- Counter mode (CTR)

AESU also implements other NIST recommended modes providing Authentication (two of which also provide confidentiality):

- Counter with CBC-MAC (CCM) per NIST recommendation 800-38C
- Galois Counter Mode (GCM) per NIST draft recommendation 800-38D
- Cipher-based MAC (CMAC) per NIST recommendation 800-38B. Note that CMAC is identical to OMAC1.

AESU modes also implement the following modes not sanctioned by NIST:

- CBC-RBP (Residual Block Processing from 802.16)
- XCBC-MAC as specified by IETF RFC-3566

In all modes supporting Authentication, the AESU hashes data to produce an integrity check vector (ICV). If a reference ICV is supplied to the AESU, it can do a bitwise check of this supplied ICV against the one computed by the AESU (ICV checking).

For more information, refer to Section 11.7.1, “Advanced Encryption Standard Execution Unit (AESU).”

11.1.4.5 Arc Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning the data to be ciphered can be any number of bytes. The AFEU supports key lengths from 8 to 128 bits (in byte increments), providing a wide range of security strengths.

For more information, refer to Section 11.7.2, “ARC Four Execution Unit (AFEU).”

11.1.4.6 Message Digest Execution Unit (MDEU)

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using either the MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 algorithms for bulk data hashing.

With any hash algorithm, the larger message is mapped onto a smaller output “digest”, so it is possible for two messages to produce the same digest. The hash digest lengths (128 bits or more) are sufficiently large, however, that such collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computationally infeasible to construct two distinct but similar messages that produce the same hash output. The MDEU is designed to support the following cryptographic algorithms:

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.
- SHA-1 is a 160-bit hash function, specified by the NIST FIPS 180-1 standard.
- SHA-224 is a 224-bit hash function, specified by the NIST FIPS 180-2 standard.
- SHA-256 is a 256-bit hash function that provides 256 bits of security against collision attacks, specified by the NIST FIPS 180-2 standard.
- SHA-384 is a 384-bit hash function, specified by the NIST FIPS 180-2 standard.
- SHA-512 is a 512-bit hash function, specified by the NIST FIPS 180-2 standard.
- The MDEU also supports HMAC computations, as specified by the NIST FIPS-198 standard.

If a digest is supplied to the MDEU, it can do a bitwise check of this supplied digest against the one computed by the MDEU (ICV checking).

For more information, refer to Section 11.7.6, “Message Digest Execution Unit (MDEU).”

11.1.4.7 Kasumi Execution Unit (KEU)

The KEU (Kasumi Execution Unit) is a functional block capable of encrypting/decrypting and/or performing integrity checks on 64-bit blocks of data using a 128-bit key. The KEU is designed support the following cryptographic algorithms:

- f8 as defined in the ETSI/SAGE Specification Document 1 for the 3GPP standard
- f9 as defined in the ETSI/SAGE Specification Document 1 for the 3GPP standard
- A5/3 for GSM/EDGE
- GEA3 for GPRS

With the exception of f9, which is an authentication algorithm, KEU implements confidentiality algorithms. For f9, if the KEU is supplied with a MAC value, it is capable of performing a bitwise check of this original MAC against an f9 MAC generated by the KEU (ICV checking).

For more information, refer to Section 11.7.5, “Kasumi Execution Unit (KEU).”

11.1.4.8 Cyclical Redundancy Check Unit (CRCU)

The CRCU computes a single 32-bit cyclic redundancy code (checksum) from all data presented on the input bus.

The CRC algorithm treats a message stream of bits as coefficients of a massive polynomial and computes the remainder of the modulo two division by an order 32 divisor polynomial. The divisor polynomial is specific to the protocol and chosen to conform to certain mathematical properties to ensure that single bit errors can be detected. Cyclic redundancy codes are used to ensure data integrity over potentially unreliable channels. There are two major CRC protocol algorithms: CRC32 and CRC32C. IEEE 802 defines the CRC32 algorithm, while iSCSI defines the CRC32C algorithm. Both protocols bit swap, byte swap, and then complement the calculated remainder to generate the checksum. The CRCU is designed to support the following check algorithms:

- The CRC32 algorithm is specified in IEEE 802.1.
- The CRC32C algorithm is specified in RFC3385.
- The CRCU also supports a programmable polynomial mode with no remainder bit mangling. This can be used to implement proprietary protocols.

The CRCU can perform ICV checking by computing a raw CRC across a message *and* previously-calculated CRC. Integrity is verified if the result matches the polynomial specific residue.

For more information, refer to Section 11.7.3, “Cyclical Redundancy Check Unit (CRCU).”

11.1.4.9 Random Number Generator (RNGU)

The RNGU is a functional block capable of generating 64-bit random numbers. It is designed to comply with FIPS 140-1 standards for randomness and non-determinism.

Because many cryptographic algorithms use random numbers as a source for generating a secret value (a nonce), it is desirable to have a private RNG for use by the SEC. The anonymity of each random number must be maintained, as well as the unpredictability of the next random number. The FIPS-140 ‘common criteria’ compliant private RNG allows the system to develop random challenges or random secret keys. The secret key can thus remain hidden from even the high-level application code, providing an added measure of physical security.

For more information, refer to Section 11.7.8, “Random Number Generator (RNGU).”

11.2 Configuration of Internal Memory Space

Table 11-2 is the base address map, showing the blocks of addresses assigned to each SEC sub-block. Table 11-3 is the detailed address map, including all registers in each sub-block. The 18-bit SEC address bus value is shown. These address values are offsets from IMMRBAR.

Table 11-2. SEC Address Map

Byte Address Offset (AD 17–0)	Module	Description	Type	Reference
0x3_0100–0x3_01FF	Channel_1	Alternate location for Channel 1	Channels	See the Channel regions starting at address 0x3_1100 and the RCA bits in Table 11-21
0x3_0200–0x3_02FF	Channel_2	Alternate location for Channel 2		
0x3_0300–0x3_03FF	Channel_3	Alternate location for Channel 3		
0x3_0400–0x3_04FF	Channel_4	Alternate location for Channel 4		
0x3_1000–0x3_10FF	Controller	Arbiter/controller control register space	Controller	11.6/11-49
0x3_1100–0x3_11FF	Channel_1	Channel 1	Channels	11.5/11-35 Also see RCA bits in Table 11-21
0x3_1200–0x3_12FF	Channel_2	Channel 2		
0x3_1300–0x3_13FF	Channel_3	Channel 3		
0x3_1400–0x3_14FF	Channel_4	Channel 4		
0x3_1500–0x3_16FF	PolyChn	PolyChannel		11.5.2/11-38
0x3_1BF8	Controller	IP block revision register	Read only	11.6.3.6/11-59
0x3_2000–0x3_2FFF	DEU	DES/3DES execution unit	Crypto EU	11.7.4/11-112
0x3_4000–0x3_4FFF	AESU	AES execution unit		11.7.1/11-62
0x3_6000–0x3_6FFF	MDEU	Message digest execution unit		11.7.6/11-135
0x3_8000–0x3_8FFF	AFEU	Arc Four execution unit		11.7.2/11-92
0x3_A000–0x3_AFFF	RNGU	Random number generator		11.7.8/11-159
0x3_C000–0x3_CFFF	PKEU	Public key execution unit		11.7.7/11-150
0x3_D000 - 0x3_DFFF	STEU	SNOW3G execution unit		11.7.7/11-150
0x3_E000–0x3_EFFF	KEU	Kasumi execution unit		11.7.5/11-121
0x3_F000–0x3_FFFF	CRCU	Cyclical Redundancy Check Unit		11.7.3/11-102

All regions not listed are reserved for future use.

Table 11-3 shows the system address map showing all functional registers.

All SEC registers are allocated 8 bytes (one dword), and the addresses listed in the table are all at 8-byte boundaries (for example, ending in 0 or 8). It is possible, however, to access the registers by 4-byte word, or in some cases by byte. The “Write by” column distinguishes these cases. Possible entries in this column are:

- byte: This register can be written by byte (using any address), by word (using an address ending in 0, 4, 8, or C), or by dword (using an address ending in 0 or 8).
- word: This register can be written by dword (using an address ending in 0 or 8), or by word (using an address ending in 0, 4, 8, or C), but not by byte.

Reads can always be done by byte, word, or dword.

Table 11-3. SEC Detailed Address Map

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_0100–0x3_01FF	Channel_1	Alternate location for Channel 1	See the Channel regions starting at address 0x3_1108 and the RCA bits in Table 11-21		
0x3_0200–0x3_02FF	Channel_2	Alternate location for Channel 2			
0x3_0300–0x3_03FF	Channel_3	Alternate location for Channel 3			
0x3_0400–0x3_04FF	Channel_4	Alternate location for Channel 4			
0x3_1008	Controller	Interrupt enable	R/W	byte[1]	11.6.3.2/1111-53
0x3_1010		Interrupt status	R	—	11.6.3.3/1111-56
0x3_1018		Interrupt clear	R/W	byte	11.6.3.4/1111-57
0x3_1020		Identification	R	—	11.6.3.5/1111-59
0x3_1028		EU assignment status	R	—	11.6.3.1/1111-52
0x3_1030		Master control	R/W	byte	11.6.3.7/1111-60
0x3_1108	Channel_1	Configuration register	R/W	word	11.5.4.1/1111-41
0x3_1110		Pointer status	R/W	word	11.5.4.2/1111-44
0x3_1140		Current descriptor pointer	R	—	11.5.4.3/1111-47
0x3_1148		Fetch FIFO	W	word	11.5.4.4/1111-47
0x3_1180–0x3_11BF		Descriptor buffer	R	—	11.5.4.5/1111-48
0x3_11C0–0x3_11DF		Gather Link Table	R	—	11.5.4.6/1111-48
0x3_11E0–0x3_11FF		Scatter Link Table	R	—	11.5.4.6/1111-48
0x3_1208	Channel_2	Config register	R/W	word	11.5.4.1/1111-41
0x3_1210		Pointer status	R/W	word	11.5.4.2/1111-44
0x3_1240		Current descriptor pointer	R	—	11.5.4.3/1111-47
0x3_1248		Fetch FIFO	W	word	11.5.4.4/1111-47
0x3_1280–0x3_12BF		Descriptor buffer	R	—	11.5.4.5/1111-48
0x3_12C0–0x3_12DF		Gather Link Table	R	—	11.5.4.6/1111-48
0x3_12E0–0x3_12FF		Scatter Link Table	R	—	11.5.4.6/1111-48
0x3_1308	Channel_3	Config register	R/W	word	11.5.4.1/1111-41
0x3_1310		Pointer status	R/W	word	11.5.4.2/1111-44
0x3_1340		Current descriptor pointer	R	—	11.5.4.3/1111-47
0x3_1348		Fetch FIFO	W	word	11.5.4.4/1111-47
0x3_1380–0x3_13BF		Descriptor buffer	R	—	11.5.4.5/1111-48
0x3_13C0–0x3_13DF		Gather Link Table	R	—	11.5.4.6/1111-48
0x3_13E0–0x3_13FF		Scatter Link Table	R	—	11.5.4.6/1111-48

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_1408	Channel_4	Config register	R/W	word	11.5.4.1/1111-41
0x3_1410		Pointer status	R/W	word	11.5.4.2/1111-44
0x3_1440		Current descriptor pointer	R	—	11.5.4.3/1111-47
0x3_1448		Fetch FIFO	W	word	11.5.4.4/1111-47
0x3_1480–0x3_14BF		Descriptor buffer	R	—	11.5.4.5/1111-48
0x3_14C0–0x3_14DF		Gather Link Table	R	—	11.5.4.6/1111-48
0x3_14E0–0x3_14FF		Scatter Link Table	R	—	11.5.4.6/1111-48
0x3_1500	Poly-Channel	Fetch FIFO Enqueue Count	R/W	word	11.5.2.1/1111-38
0x3_1508		Descriptor Finished Count	R/W	word	11.5.2.1/1111-38
0x3_1510		Data Bytes In Count	R/W	word	11.5.2.1/1111-38
0x3_1518		Data Bytes Out Count	R/W	word	11.5.2.1/1111-39
0x3_1BF8	Controller	IP block revision	R	—	11.6.3.6/1111-59
0x3_2000	DEU	Mode register	R/W	word	11.7.4.1/1111-112
0x3_2008		Key size register	R/W	word	11.7.4.2/1111-113
0x3_2010		Data size register	R/W	word	11.7.4.3/1111-114
0x3_2018		Reset control register	R/W	word	11.7.4.4/1111-115
0x3_2028		Status register	R	—	11.7.4.5/1111-116
0x3_2030		Interrupt status register	R/W	word	11.7.4.6/1111-117
0x3_2038		Interrupt mask register	R/W	word	11.7.4.7/1111-118
0x3_2050		EU-Go	W	word	11.7.4.8/1111-120
0x3_2100		IV register	R/W	word	11.7.4.9/1111-121
0x3_2400		Key 1 register	W	byte	11.7.4.10/1111-121
0x3_2408		Key 2 register	W	byte	11.7.4.10/1111-121
0x3_2410		Key 3 register	W	byte	11.7.4.10/1111-121
0x3_2800–0x3_2FFF		Input FIFO / Output FIFO	R/W[2]	byte	11.7.4.11/1111-121

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_4000	AESU	Mode register	R/W	word	11.7.1.1/1111-63
0x3_4008		Key size register	R/W	word	11.7.1.2/1111-67
0x3_4010		Data size register	R/W	word	11.7.1.3/1111-68
0x3_4018		Reset control register	R/W	word	11.7.1.4/1111-68
0x3_4028		Status register	R	—	11.7.1.5/1111-70
0x3_4030		Interrupt status register	R/W	word	11.7.1.6/1111-71
0x3_4038		Interrupt mask register	R/W	word	11.7.1.7/1111-73
0x3_4040		ICV Size register	R/W	word	11.7.1.8/1111-74
0x3_4050		End of message register	W	word	11.7.1.10/1111-75
0x3_4100–0x3_415F		Context	R/W	byte	11.7.1.11/1111-75
0x3_4400–0x3_441F		Key memory	R/W	byte	11.7.1.11/1111-92
0x3_4800–0x3_4FFF		Input FIFO / Output FIFO	R/W [1]	byte	11.7.1.11/1111-92
0x3_6000	MDEU	Mode register	R/W	word	11.7.6.2/1111-136
0x3_6008		Key size register	R/W	word	11.7.6.4/1111-140
0x3_6010		Data size register	R/W	word	11.7.6.5/1111-140
0x3_6018		Reset control register	R/W	word	11.7.6.6/1111-141
0x3_6028		Status register	R	—	11.7.6.7/1111-142
0x3_6030		Interrupt status register	R/W	word	11.7.6.8/1111-143
0x3_6038		Interrupt mask register	R/W	word	11.7.6.9/1111-145
0x3_6040		ICV size register	W	word	11.7.6.10/1111-146
0x3_6050		End_of_message	W	word	11.7.6.11/1111-146
0x3_6100–0x3_6147		Context registers	R/W	byte	11.7.6.12/1111-147
0x3_6400–0x3_647F		Key registers	W	byte	11.7.6.13/1111-150
0x3_6800–0x3_6FFF		Input FIFO	W [1]	byte	11.7.6.14/1111-150

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_8000	AFEU	Mode register	R/W	word	11.7.2.1/1111-93
0x3_8008		Key size register	R/W	word	11.7.2.2/1111-94
0x3_8010		Data size register	R/W	word	11.7.2.3/1111-94
0x3_8018		Reset control register	R/W	word	11.7.2.4/1111-95
0x3_8028		Status register	R	—	11.7.2.5/1111-96
0x3_8030		Interrupt status register	R/W	word	11.7.2.6/1111-97
0x3_8038		Interrupt mask register	R/W	word	11.7.2.7/1111-98
0x3_8050		End_of_message register	W	word	11.7.2.8/1111-100
0x3_8100–0x3_81FF		Context memory	R/W	byte	11.7.2.9/1111-100
0x3_8200		Context memory pointers	R/W	byte	11.7.2.9/1111-100
0x3_8400–0x3_840F		Key registers	W	byte	11.7.2.10/1111-101
0x3_8800–0x3_8FFF (3_8E00)		Input FIFO / Output FIFO (special context address)	R/W [1]	byte	11.7.2.10/1111-101
0x3_A000	RNGU	Mode register	R/W	word	11.7.8.1/1111-160
0x3_A010		Data size register	R/W	word	11.7.8.2/1111-161
0x3_A018		Reset control register	R/W	word	11.7.8.3/1111-161
0x3_A028		Status register	R	—	11.7.8.4/1111-162
0x3_A030		Interrupt status register	R/W	word	11.7.8.5/1111-163
0x3_A038		Interrupt mask register	R/W	word	11.7.8.6/1111-164
0x3_A050		End_of_message	W	word	11.7.8.7/1111-165
0x3_A400–0x3_A43F		Entropy Registers	W	word	11.7.8.8/1111-165
0x3_A800–0x3_AFFF		Output FIFO	R [1]	—	11.7.8.8/1111-165

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_C000	PKEU	Mode register	R/W	word	11.7.7.1/1111-151
0x3_C008		Key size register	R/W	word	11.7.7.2/1111-151
0x3_C010		Data size register	R/W	word	11.7.7.4/1111-153
0x3_C018		Reset control register	R/W	word	11.7.7.5/1111-154
0x3_C028		Status register	R	—	11.7.7.6/1111-155
0x3_C030		Interrupt status register	R/W	word	11.7.7.7/1111-156
0x3_C038		Interrupt mask register	R/W	word	11.7.7.8/1111-157
0x3_C040		ABSize	R/W	word	11.7.7.3/1111-153
0x3_C050		End_of_message	W	word	11.7.7.9/1111-158
0x3_C200–0x3_C27F		Parameter memory A0	R/W	byte	11.7.7.10/1111-159
0x3_C280–0x3_C2FF		Parameter memory A1	R/W	byte	
0x3_C300–0x3_C37F		Parameter memory A2	R/W	byte	
0x3_C380–0x3_C3FF		Parameter memory A3	R/W	byte	
0x3_C400–0x3_C47F		Parameter memory B0	R/W	byte	
0x3_C480–0x3_C4FF		Parameter memory B1	R/W	byte	
0x3_C500–0x3_C57F		Parameter memory B2	R/W	byte	
0x3_C580–0x3_C5FF		Parameter memory B3	R/W	byte	
0x3_C800–0x3_C9FF		Parameter memory N	R/W	byte	
0x3_CA00–0x3_CBFF		Parameter memory E	W	byte	

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_E000	KEU	Mode register	R/W	word	11.7.5.1/1111-122
0x3_E008		Key size register	R/W	word	11.7.5.2/1111-124
0x3_E010		Data size register	R/W	word	11.7.5.3/1111-124
0x3_E018		Reset control register	R/W	word	11.7.5.4/1111-125
0x3_E028		Status register	R	—	11.7.5.5/1111-126
0x3_E030		Interrupt Status register	R/W	word	11.7.5.6/1111-127
0x3_E038		Interrupt Mask register	R/W	word	11.7.5.7/1111-129
0x3_E048		Data out register (f9 MAC)	R	—	11.7.5.8/1111-130
0x3_E050		End_of_message register	W	word	11.7.5.9/1111-131
0x3_E100		IV_1 register	R/W	byte	11.7.5.10/1111-131
0x3_E108		ICV_In register	R/W	byte	11.7.5.11/1111-132
0x3_E110		IV_2 register (Fresh)	R/W	byte	11.7.5.12/1111-133
0x3_E118		Context_1 register	R/W	byte	11.7.5.13/1111-133
0x3_E120		Context_2 register	R/W	byte	11.7.5.13/1111-133
0x3_E128		Context_3 register	R/W	byte	11.7.5.13/1111-133
0x3_E130		Context_4 register	R/W	byte	11.7.5.13/1111-133
0x3_E138		Context_5 register	R/W	byte	11.7.5.13/1111-133
0x3_E140		Context_6 register	R/W	byte	11.7.5.13/1111-133
0x3_E400		Key data register_1 (CK-high)	R/W	byte	11.7.5.14/1111-133
0x3_E408		Key data register_2 (CK-low)	R/W	byte	11.7.5.14/1111-133
0x3_E410		Key dataregister_3 (IK-high)	R/W	byte	11.7.5.15/1111-134
0x3_E418		Key data register_4 (IK-low)	R/W	byte	11.7.5.15/1111-134
0x3_E800–0x3_EFFF		Input FIFO / Output FIFO	R/W [1]	byte	11.7.5.16/1111-135

Table 11-3. SEC Detailed Address Map (continued)

Byte Address Offset (AD 17–0)	Module	Register	Access	Write by	Reference
0x3_F000	CRCU	Mode register	R/W	word	11.7.3.2/1111-103
0x3_F008		Key size register	R/W	word	11.7.3.3/1111-104
0x3_F010		Data size register	R/W	word	11.7.3.4/1111-104
0x3_F018		Reset control register	R/W	word	11.7.3.5/1111-104
0x3_F020		Control	R/W	word	11.7.3.6/1111-105
0x3_F028		Status register	R	—	11.7.3.7/1111-106
0x3_F030		Interrupt status register	R/W	word	11.7.3.8/1111-107
0x3_F038		Interrupt mask register	R/W	word	11.7.3.9/1111-108
0x3_F040		ICV Size	R/W	word	11.7.1.8/1111-74
0x3_F050		End_of_message register	W	word	11.7.3.11/1111-110
0x3_F108		Context register	R/W	byte	11.7.3.13/1111-110
0x3_F400		Key register	R/W	byte	11.7.3.14/1111-111
0x3_F800–0x3_FFFF		Input FIFO	W [1]	byte	11.7.3.15/1111-112

1 Byte accessibility is controlled by internal logic, particularly at FIFOs, to prevent unintended overwrites of partial words during writes, and to prevent unintended duplicate reads of partial data during reads. In addition, these bytes must be presented on the correct byte lanes for the intended destination.

2 For the EU FIFOs, write operations anywhere in the address range enqueue to the input FIFO, and read operations anywhere in the address range dequeue from the output FIFO. See the referenced section for more detailed information.

11.3 Descriptor Overview

The host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host has determined that a security operation is required, it creates a “descriptor” containing all the information the SEC needs to perform the security operation. The host creates the descriptor in main memory, then writes a pointer to the descriptor into the Fetch FIFO of one of the SEC channels. The channel uses this pointer to read the descriptor into its descriptor buffer. Once it obtains the descriptor, the SEC uses its bus mastering capability to obtain inputs and write results, thus off-loading data movement and encryption operations from the host processor.

For test purposes, it is also possible for the host to write keys, context, and text-data directly to execution units, using SEC’s host-controlled access; and to read the results once the execution unit has completed processing. This method avoids use of descriptors.

NOTE

Host-controlled access of execution units is provided primarily for system debug purposes. SEC contains no mechanism to prevent the Host from accessing an execution unit assigned to a Channel, and it provides no mechanism for preventing assigning an execution unit being used by the host to a Channel. Simultaneous use of an execution unit by a Channel and a Host is a very effective way to force the execution unit into an error condition.

11.3.1 Descriptor Structure

SEC descriptors are designed to support the cryptographic computation of a single packet using a single descriptor. SEC descriptors have a fixed length of 64 bytes, that is, eight 64-bit words (referred to as dwords). A descriptor consists of one header dword and seven “pointer dwords”, as seen in Figure 11-3.

	0 15	16	17 23	24 27	28 31	32 63
Header Dword	Descriptor Control					Descriptor Feedback
Pointer Dword 0	Length0	J0	Extent0	—	Eptr0	Pointer0
Pointer Dword 1	Length1	J1	Extent1	—	Eptr1	Pointer1
Pointer Dword 2	Length2	J2	Extent2	—	Eptr2	Pointer2
Pointer Dword 3	Length3	J3	Extent3	—	Eptr3	Pointer3
Pointer Dword 4	Length4	J4	Extent4	—	Eptr4	Pointer4
Pointer Dword 5	Length5	J5	Extent5	—	Eptr5	Pointer5
Pointer Dword 6	Length6	J6	Extent6	—	Eptr6	Pointer6

Figure 11-3. Descriptor Format

The first half of the header dword is a Descriptor Control field, and the other half is the Descriptor Feedback field.

The Descriptor Control field of the header dword specifies the security operation to be performed, the execution unit(s) needed, and the modes for each execution unit. The pointer dwords, all of which have the same format, contain pointer and length information for locating input or output parcels (such as keys, context, or text-data). The large number of pointers provided in the descriptor allows for multi-algorithm operations that require fetching of multiple keys, as well as fetch and return of contexts. Any pointer dword that is not needed can be given a length of zero.

SEC descriptors include scatter/gather capability, which means that each pointer in a descriptor can be either a direct pointer to a contiguous parcel of data, or can be a pointer to a “link table” which is a list of pointers and lengths used to assemble the parcel. When a link table is used to read input data, this is referred to as a “gather” operation; when used to write output data, it is referred to as a “scatter” operation.

11.3.2 Descriptor Format: Header Dword

Descriptors are created by the host to guide the SEC through required cryptographic operations. The header dword provides the primary indication of the operations to be performed, the mode for each operation, and internal addressing used by the controller and channel for internal data movement. The fields that must be supplied to SEC are shown in the “Field” rows of Figure 11-4, and described in Table 11-4. The SEC device drivers allow the host to create proper headers for each cryptographic operation.

SEC processing of a descriptor sometimes includes performing a header writeback, that is, writing the original header dword back to system memory with certain fields modified. The modified fields are shown in the “Writeback” rows of Figure 11-4, and described in Table 11-5.

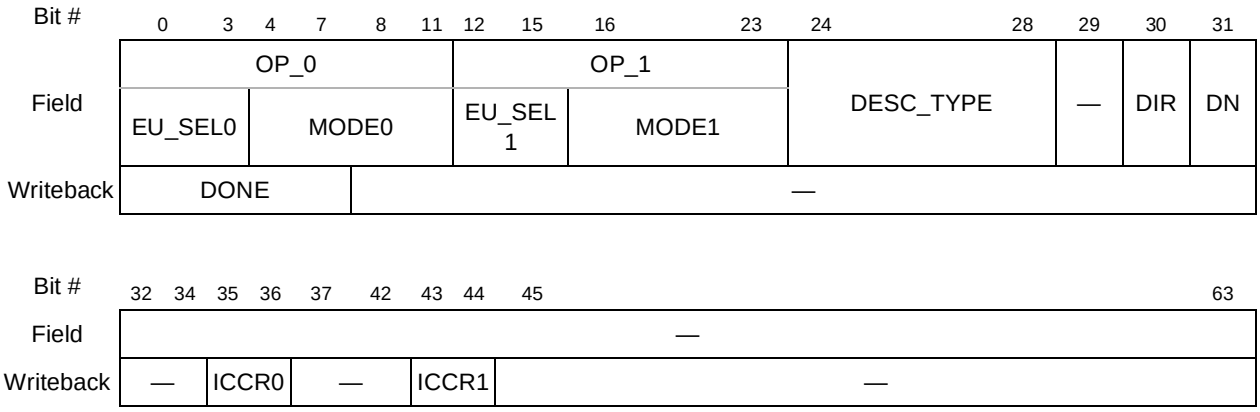


Figure 11-4. Header Dword

Table 11-4. Header Dword Bit Definitions

Bits	Name	Description
	OP_0:	
0–3	EU_SEL0	Primary EU select: See Section 11.3.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,” for possible values.
4–11	MODE0	Primary mode: Mode data used to program the primary EU. The mode data is specific to the chosen EU. This field is passed directly to bits 56–63 of the mode register in the selected EU. Refer to the EU-specific Mode register sections (Section 11.7.1.1, “AESU Mode Register,” Section 11.7.2.1, “AFEU Mode Register,” Section 11.7.3.2, “CRCU Mode Register,” Section 11.7.4.1, “DEU Mode Register,” Section 11.7.5.1, “KEU Mode Register (KEUMR),” Section 11.7.6.2, “MDEU Mode Register,” Section 11.7.7.1, “PKEU Mode Register,” and Section 11.7.8.1, “RNGU Mode Register”) for further info. Any bits of any use in any mode register beyond bits 56–63 are under control of the channel and not the MODE0 field.
	OP_1:	
12–15	EU_SEL1	Secondary EU select: See Section 11.3.2.1, “Selecting Execution Units—EU_SEL0 and EU_SEL1,” for possible values.
16–23	MODE1	Secondary mode: Mode data used to program the primary EU. The mode data is specific to the chosen EU. This field is passed directly to bits 56–63 of the mode register in the selected EU. Refer to the EU-specific Mode register sections (sections Section 11.7.3.2, “CRCU Mode Register,” and Section 11.7.6.2, “MDEU Mode Register”) for further info.
24–28	DESC_TYPE	Descriptor Type: This, along with the DIR field, determines the sequence of actions to be performed by the channel and selected EUs using the blocks of data listed in the rest of the descriptor. The attributes determined include the direction of data flow for each data block, which EU (primary or secondary) is accessed, what snooping options are used, and address offsets for internal EU accesses. See Section 11.3.2.2, “Selecting Descriptor Type—DESC_TYPE,” for possible values.
29	—	Reserved
30	DIR	Direction: direction of overall data flow: 0 Outbound 1 Inbound This, along with the DESC_TYPE field, helps determine the sequence of actions to be performed by the channel and selected EUs.
31	DN	Done notification: 0 No done notification. 1 Signal “done” to the host on completion of this descriptor. This enables done notification if the NT field is 1 in the Channel Configuration Register (see Table 11-11). The done notification can take the form of an interrupt, a writeback in the DONE field of this header dword (see Table 11-5), or both, depending upon the states of the CDIE (Channel Done Interrupt Enable) and CDWE (Channel Done Writeback Enable) bits in the Channel Configuration Register.

Table 11-5. Header Dword Writeback Bit Definitions

Bits	Name	Description
0–7	DONE	When done writeback is used, then at the completion of descriptor processing this byte is written with the value 0xFF. To determine when done writeback is used, see the CDWE, NT, and CDIE fields in the Channel Configuration Register (Table 11-11).
8–34	—	Reserved
35–36	ICCR0	Integrity check comparison result from primary: These bits are supplied by the primary EU when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
37–42	—	Reserved
43–44	ICCR1	Integrity check comparison result from secondary: These bits are supplied by the secondary EU (if any) when descriptor processing is complete. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
45–47	—	Reserved
48–63	ID_TAG	Identification Tag. This value is copied from the ID_TAG field written by the Host into the Fetch FIFO (see Section 11.5.4.4, “Fetch FIFO (FF)”)

11.3.2.1 Selecting Execution Units—EU_SEL0 and EU_SEL1

Table 11-6 shows the values for EU_SEL0 and EU_SEL1 in the descriptor header. The following rules govern the choices for these fields:

1. EU_SEL0 values of “No EU selected” or “Reserved” results in an “Unrecognized Header Error” condition during processing of the descriptor header.
2. The only valid choices for EU_SEL1 are “No EU selected”, CRCU, or MDEU. Any other choice results in an “Unrecognized Header” error condition.
3. If EU_SEL1 is CRCU or MDEU, then EU_SEL0 must be DEU, AESU, AFEU, or KEU. All other values of EU_SEL0 results in an “Unrecognized header” error condition.

Table 11-6. EU_SEL0 and EU_SEL1 Values

Value (Binary)	Selected EU
0000	No EU selected
0001	AFEU
0010	DEU
0011	MDEU-A
1011	MDEU-B
0100	RNGU

Table 11-6. EU_SEL0 and EU_SEL1 Values (continued)

Value (Binary)	Selected EU
0101	PKEU
0110	AESU
0111	KEU
1000	CRCU
others	Reserved
1111	Reserved for header writeback

MDEU uses two different designators to refer to the same Message Digest Execution Unit. If MDEU-B is selected, then the Channel configures MDEU to perform SHA-224, SHA-256, SHA-384, and SHA-512. If MDEU-A is selected, then the Channel configures MDHA to perform SHA-160, SHA-224, SHA-256, or MD5. This configuration is achieved automatically; the channel sets bit 51 of the MDEU mode register as it inserts the MODE0 (or MODE1) value into the MDEU mode register. For more information, see Section 11.7.6.2, “MDEU Mode Register.”

11.3.2.2 Selecting Descriptor Type—DESC_TYPE

Table 11-7 shows the permissible values for the DESC_TYPE field in the descriptor header. Descriptor types from the SEC1.0, which have “0” in the last bit, are listed first, followed by new SEC 2.x/3.x types, which have “1” in the last bit.

Table 11-7. Descriptor Types

Value (Binary)	Descriptor Type	Notes
0000_0	aesu_ctr_nonsnoop	AESU CTR nonsnoooping
0001_0	common_nonsnoop	Common, nonsnooping, non-PKEU, non-AFEU
0010_0	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011_0	—	Reserved
0100_0	—	Reserved
0101_0	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110_ 0	—	Reserved
0111_0	—	Reserved
1000_0	pkeu_mm	PKEU-Montgomery Multiplication
1001_0	—	Reserved
1010_0	—	Reserved
1011_0	—	Reserved
1100_0	hmac_snoop_aesu_ctr	AESU CTR hmac snooping [2]
1101_0	—	Reserved

Table 11-7. Descriptor Types (continued)

Value (Binary)	Descriptor Type	Notes
1110_0	—	Reserved
1111_0	—	Reserved
0000_1	ipsec_esp	IPsec ESP mode encryption and hashing
0001_1	802.11i_aes_ccmp	CCMP encryption and hashing, suitable for 802.11i
0010_1	srtp	SRTP encryption and hashing
0011_1	pkeu_build	pkeu_build Elliptic Curve Cryptography
0100_1	pkeu_ptmul	pkeu_ptmul Elliptic Curve Cryptography
0101_1	pkeu_ptadd_dbl	pkeu_ptadd_dbl Elliptic Curve Cryptography
0110_1	—	Reserved
0111_1	—	Reserved
1000_1	tls_ssl_block	TLS/SSL generic block cipher
1001_1	tls_ssl_stream	TLS/SSL generic stream cipher
1010_1	raid_xor	XOR 2-6 sources together
1011_1	ipsec_aes_gcm	IPsec ESP mode using AES GCM encryption and hashing
1100_1	dbl_crc	Do two CRC operations
others	—	Reserved

[1] Type 0000_0 is for AES-CTR operations. Type 0001_0 also supports AES-CTR, however to use AES-CTR with 0001_0, the user must pre-pend zeros to the AES-Context before loading the AES Context Registers.

[2] Type 1100_0 is for AES-CTR operations with HMAC. Type 0010_0 also supports AES-CTR with HMAC, however to use AES-CTR with 0010_0, the user must pre-pend zeros to the AES-Context before loading the AES Context Registers.

For more about descriptor types and the data used for each type, see Section 11.4, “Descriptor Types.”

11.3.3 Descriptor Format: Pointer Dwords

The descriptor contains seven “pointer dwords” which define where in memory the SEC should access its input and output parcels. The pointer dwords are numbered 0 to 6 as shown in Figure 11-3. The channel determines how it uses each of the pointer dwords based on the “Descriptor Type” and “Direction” fields in the header. The channel accesses the first parcel by starting at a location given by a POINTER value, and accessing a number of bytes given by a LENGTH or EXTENT value. Subsequent parcels may be accessed by starting where a previous parcel ended, or by starting at a different POINTER. The LENGTH or EXTENT used with any POINTER may be from the same pointer dword or from a different pointer dword in the same descriptor.

If Extend Address Enable is high (see the EAE bit in Table 11-11), then the four EPTR bits are concatenated with the POINTER field to form a 36-bit pointer address.

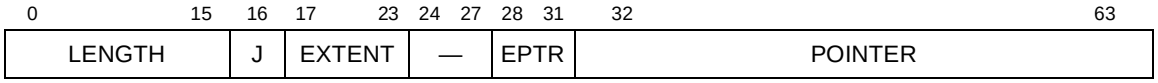


Figure 11-5. Pointer Dword

Table 11-8. Pointer Dword Field Definitions

Bits	Name	Description
0–15	LENGTH	Length: A number of bytes in the range 0 to 65535. The use of this field depends on the “Descriptor Type” and “Direction” in the header dword. A value of zero may cause the channel to skip this dword.
16	J	Jump: Determines whether to “jump” to a link table whenever the POINTER field in this same dword is used. 0 The POINTER field points to data. 1 The POINTER field points to a link table, and scatter/gather is enabled.
17–23	EXTENT	Extent: A number of bytes in the range 0 to 127. The use of this field depends on the “Descriptor Type” and “Direction” in the header dword.
24–27	—	Reserved
28–31	EPTR	Extended Pointer: Concatenated as the top 4 bits of the pointer when EAE is high (see the EAE bit in Table 11-11).
32–63	POINTER	Pointer: A memory address.

On occasion, a descriptor field may not be applicable to the requested service. With seven pointer dwords, it is possible that not all these dwords are required to specify the input and output parameters. (Some operations, for example, don’t require context.) Wherever a particular field is not used, it should be set to all 0s.

Some descriptors involve more than seven parcels of input and output data. In these cases, it is necessary to use one POINTER field to address a sequence of parcels.

LENGTH and EXTENT fields normally specify the sizes of parcels -- often, but not always the size of a parcel located at the address contained in the matching POINTER field[1]. In some cases, however, the POINTER field is zero, and the LENGTH and/or EXTENT fields simply specify values to be written to an EU.

The J bit in each pointer dword is used to enable the scatter/gather feature. If a parcel to be read or written by SEC is in one contiguous block of memory locations, then the scatter/gather feature is not needed. In this case the POINTER should be set to point directly at the first byte of the parcel, and the J bit should be 0. On the other hand, if the parcel is stored in several separate segments of memory, then the scatter/gather capability is needed to assemble or distribute the complete parcel. In this case the POINTER should be set to point to a “link table”, and the J bit should be 1. For link table format, see Section 11.3.4, “Link Table

1. Sometimes an EXTENT field refers to data in a pointer not in the same DWORD. For example, with the CCMP descriptor type, The length of the CRC check field appears in E0, but the field that is E0 bytes in length is referred to either by P4 or P5, depending on the direction bit in the Header Control Word.

Format.” Scatter/gather capability is available for all pointer dwords of all descriptor types, with the following exception:

- Raid-xor descriptor type does not allow scatter/gather.

11.3.4 Link Table Format

Link tables implement scatter/gather capability. For “gather” operations, a link table specifies a list of “memory segments” that are to be concatenated in the process of assembling parcels. For “scatter” operations, a link table specifies a list of memory segments into which the output data should be written. Scatter or gather of a parcel may be specified by a single link table or by a chain of link tables that are linked together with pointers (see Figure 11-7).

The link table or chain of link tables accessed through some descriptor POINTER must specify enough memory segments to hold precisely *all the data that is accessed through that pointer.* In most cases, only a single parcel is accessed through a given POINTER, and the chain of link tables specifies just that parcel. In other cases, the descriptor POINTER is used multiple times to access a sequence of parcels, and the chain of link tables must supply data for the entire sequence. An example of construction of link tables is illustrated in Figure 11-7.

A link table may contain any number of dword entries. There are two kinds of entries, “regular” entries and “next” entries. Each “regular” entry specifies a memory segment by means of a 36-bit starting address (SEGPTR) and a 16-bit length (SEGLEN). A “next” entry is used at the end of a link table to specify that the list of memory segments is continued in another link table. In a “next” entry, the N bit is set, the SEGPTR field gives the address of the next link table, and the SEGLEN field must be 0. A chain of link tables may contain any number of link tables.

Whether the list of memory segments is in a single link table or split into several link tables, the last entry in the last link table is a “regular” entry with the R (return) bit set. The R bit signifies the end of link table operations so that the channel returns to the descriptor for its next pointer (if any). Link tables are illustrated in Figure 11-7. A single link table entry is shown in Figure 11-6.

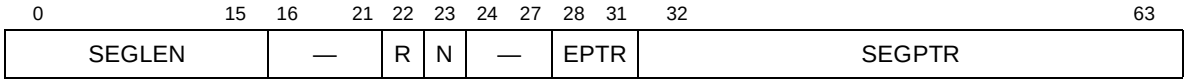


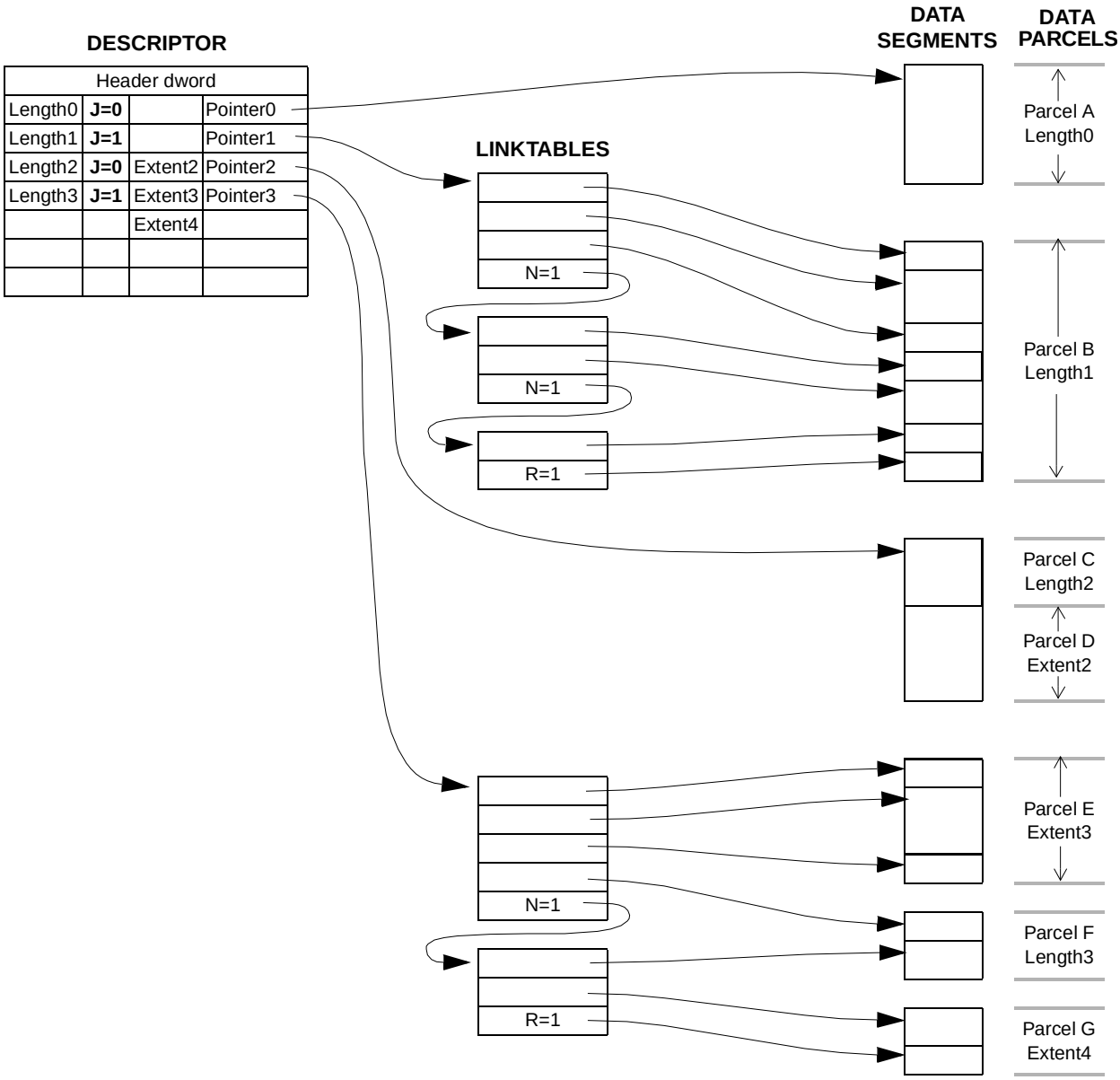
Figure 11-6. Link Table Entry

Table 11-9. Link Table Field Definitions

Bits	Name	Description
0–15	SEGLEN	Length: 0-15 When N=0, a number in the range 1 to 65535, specifying the number of bytes in the memory segment pointed to by SEGPTR. A value of 0 causes the SGZL error bit to be set in the Channel Status (see Section 11.5.4.2, “Channel Status Register (CSR)”). When N=1, must be 0.
16–21	—	Reserved

Table 11-9. Link Table Field Definitions (continued)

Bits	Name	Description
22	R	Return: When N=0: 0 No special action. 1 This is the last entry in the chain of link tables. If this entry does not specify the right number of bytes to complete the last parcel, a G-STATE or S-STATE error is set in the Channel Status Register (see Section 11.5.4.2, “Channel Status Register (CSR)”). When N=1, ignored.
23	N	Next: 0 No special action. 1 This is the last dword in the current link table. The SEGPTR field is the address of the next link table in the chain.
24–27	—	Reserved
28–31	EPTR	Extended Pointer: Concatenated as the top 4 bits of the segment pointer when EAE is high (see the EAE bit in Table 11-11).
32–63	SEGPTR	Segment pointer: A memory address.



This figure illustrates various ways that a descriptor may specify parcels:

The first pointer dword in the descriptor specifies Parcel A using the simplest method—the parcel is specified directly through Pointer0 and Length0.

The next pointer dword uses a chain of link tables to specify Parcel B. Since J=1, Pointer1 is used as the address of a link table. The link table specifies several “regular” entries specifying data segments to be concatenated. The last word of the link table is a “next” entry indicating that the list continues in the next link table. The last entry in the last link table of the chain has the R bit set.

The last cases illustrate how one pointer in a descriptor can be used to specify multiple parcels. Pointer2 and Length2 specify Parcel C, then Parcel D follows immediately afterwards, with length specified by Extent2. Pointer3 is used for three parcels (E, F, and G), this time using link tables.

Figure 11-7. Descriptors, Link Tables, and Parcels

For any sequence of parcels accessed by a link table or chain of link tables, the combined lengths of the parcels (the sum of their LENGTH and/or EXTENT fields) must equal the combined lengths of the link table memory segments (SEGLEN fields). Otherwise the channel sets the error state in the SGLM bit of the Channel Status Register (see Section 11.5.4.2, “Channel Status Register (CSR)”).

Example (from Figure 11-7): To demonstrate use of a link table, assume that the current descriptor type calls for the channel to read a parcel using Pointer3 and Extent3 fields, and assume that J3=1. Due to the J3 value, Pointer3 is not used as a data address but instead used as the address of a link table. The channel begins by reading the first four dwords starting at Pointer3 into an internal “gather table buffer”.

Using the first entry of the gather table buffer, the channel starts accessing the parcel by reading SEGLEN bytes beginning at SEGPTR. If the required parcel size (Extent3) is greater than this first SegLen, the channel moves on to the next entry of the gather table buffer, and reads SEGLEN bytes starting at SEGPTR. While there are more bytes to be read in the parcel, this process continues. If the channel’s gather table buffer is exhausted, the channel reads the next four dwords of the link table into its gather table buffer. If a gather table buffer entry is encountered in which the N bit is set, the channel uses the SEGPTR field in that word to find the next link table in the chain.

Now assume that the channel accesses its next parcel using Pointer3 again, this time with length given by Length3. In this case the channel continues to the next line of the link table, and begins reading the memory segment specified there. As before, the channel concatenates memory segments from as many link table entries as necessary to obtain the required number of bytes (Length3).

Similarly, the next parcel is obtained by using Pointer3 yet again, this time with length given by Extent4.

Assume that for the current descriptor type, the Extent4 parcel is the last one to be accessed through Pointer3. Then the link table entry that supplies the last memory segment for Extent4 has the R bit set, signifying that this is the last entry in the chain of link tables.

11.4 Descriptor Types

Table 11-10 shows in summary form how the pointer dwords are used with the various descriptor types. Detailed information about each descriptor type is given in the remainder of this chapter. Additional explanation of the use of certain descriptor types, can be found in the SEC 3.0 Descriptor Programmer’s Guide.

As usual, older descriptor types which end in 0 are listed first, followed by newer types that end in 1.

Table 11-10. Descriptor Format Summary

Descriptor Type	field type	Pointer Dword0	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Pointer Dword5	Pointer Dword6
0000_0 aesu_ctr_ nosnoop	Length	reserved	Cipher Context In	Cipher Key	Main Data In	Data Out	Cipher Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved

Table 11-10. Descriptor Format Summary (continued)

Descriptor Type	field type	Pointer Dword0	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Pointer Dword5	Pointer Dword6
0001_0 common_ nosnoop for DES, KEU f8, RNGU, AES-CCM	Length	reserved	Context In	Key	Main Data In	Data Out	Context Out (incl. ICV out)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nosnoop for MDEU	Length	reserved	Context In	Key	Main Data In	ICV In	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0001_0 common_ nosnoop for KEU f9, AES-XCBC, AES-CMAC	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0001_0 common_ nonsnoop for CRCU	Length	reserved	Context In	Key	Main Data In	reserved	Context Out	reserved
	Extent	reserved	reserved	reserved	reserved	ICV In	reserved	reserved
0010_0 hmac_snoop _no_afeu	Length	Hash Key	Hash-only Header	Cipher Key	Cipher Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0101_0 common_ nosnoop_ afeu	Length	reserved	Context In (via In FIFO)	Cipher Key	Main Data In	Data Out	Context Out (via Out FIFO)	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1000_0 pkeu_mm	Length	“N” In	“B” In	“A” In	“E” In	“B” Out	reserved	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1100_0 hmac_snoop_ aesu_ctr	Length	Hash Key	Hash-only Header	AES Key	AES Context In	Main Data In	Data Out	ICV Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0000_1 ipsec_esp	Length	HMAC Key	Hash-only Header	Cipher IV In	Cipher Key	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	reserved	ICV In	ICV Out	reserved

Table 11-10. Descriptor Format Summary (continued)

Descriptor Type	field type	Pointer Dword0	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Pointer Dword5	Pointer Dword6
0001_1 802.11i AES ccmp	Length	CRC-only Header	AES Context In	AES Key	Hash-only Header	Main Data In	Data Out	AES Context Out
	Extent	CRC In/Out (FCS)	reserved	reserved	reserved	MIC In	MIC Out	reserved
0010_1 srtp with ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	Data Out	HMAC Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	reserved	reserved
0010_1 srtp without ICV Check	Length	HMAC Key	AES Context In	AES Key	Main Data In	HMAC In	Data Out	AES Context Out
	Extent	reserved	reserved	reserved	Hash-only Header	Hash-only Trailer	HMAC Out	reserved
0011_1 pkeu_build	Length	“A0” In	“A1” In	“A2” In	“A3” In	“B0” In	“B1” In	“Build” Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0100_1 pkeu_ptmul	Length	“N” In	“E” In	“Build” In	“B1” Out	“B2” Out	“B3” Out	reserved
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
0101_1 pkeu_ptadd_dbl	Length	“N” In	“Build” In	“B2” In	“B3” In	“B1” Out	“B2” Out	“B3” Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1000_1 outbound tls_ssl_block	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	Cipher-only Trailer	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved
1000_1 inbound tls_ssl_block	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved
1001_1 outbound tls_ssl_stream	Length	MAC Key	Cipher IV In	Cipher Key	Main Data In	Cipher-only Trailer	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV Out	reserved	reserved

Table 11-10. Descriptor Format Summary (continued)

Descriptor Type	field type	Pointer Dword0	Pointer Dword1	Pointer Dword2	Pointer Dword3	Pointer Dword4	Pointer Dword5	Pointer Dword6
1001_1 inbound tls_ssl_ stream	Length	MAC Key	Cipher IV In	Cipher Key	reserved	Main Data In	Data Out	Cipher IV Out
	Extent	reserved	reserved	reserved	Hash-only Header	ICV In	ICV Out	reserved
1010_1 raid_xor	Length	Source F Data In	Source E Data In	Source D Data In	Source C Data In	Source B Data In	Source A Data In	Data Out
	Extent	reserved	reserved	reserved	reserved	reserved	reserved	reserved
1011_1 ipsec_aes_ gcm	Length	AES Ctx In	AAD In	Nonce Part 2 In	AES Key	Main Data In	Data Out	Cipher Ctx Out
	Extent	reserved	reserved	reserved	Nonce Part 1 In	AES ICV In	ICV Out	CRC ICV In/Out
1100_1 dbl_crc	Length	Header In	Payload In	reserved	reserved	reserved	reserved	reserved
	Extent	Header ICV	Payload ICV	Header ICV Out	Payload ICV Out	reserved	reserved	reserved
others		Reserved						

11.5 Polychannel

The polychannel is the main control unit in the SEC. It implements four independent channels.

In the SEC, each cryptographic task is managed by a channel. Control information and data pointers for a given task are stored in the form of a descriptor (see Section 11.3.1, “Descriptor Structure”) placed in system memory. A descriptor determines what EUs are used, how they are configured, where to fetch needed data, and where to store the results. To invoke cryptographic tasks, the host constructs a descriptor, selects a channel, and writes a pointer to the descriptor into the selected channel’s Fetch FIFO. Each Fetch Fifo can store up to 24 pointers. Typical operations performed by channels to process a descriptor are listed in Section 11.1.2, “Polychannel.”

When a channel completes operation on a descriptor, it can notify the host that it is done in several ways—through interrupt and/or through a writeback of the descriptor header dword. The done interrupt is enabled by the CDIE bit and the “done writeback” is enabled by the CDWE bit of the Channel Configuration Register (Table 11-11). Table 11-5 shows the DONE field that is written back if writeback is enabled.

The selected done notification can be performed at the end of processing of every descriptor, or only on selected descriptors. If the NT field is 0 in the Channel Configuration Register, then done notification is performed after every descriptor. If the NT field is 1, then done notification is only performed on descriptors in which the DN bit is set in the packet header (Table 11-4).

In addition to the “done” notification, the results of ICV checking can also be communicated back to the host through either interrupt or writeback as follows: writeback is the last step by the channel before the channel signals it is done to the Controller. Any error detected by the channel prevents the done signal, and also prevents writeback.

To use the interrupt method, the integrity check error interrupt must be unmasked in the EU’s interrupt mask register (see the sections on the individual EUs).

To use the writeback method for integrity check comparison results, set either the AWSE or the IWSE bit of the Channel Configuration Register (Table 11-11). If AWSE is set, then the integrity check results are written back for every descriptor. If IWSE is set then the writeback only occurs on descriptors that call for ICV checking. In either case, whenever the integrity check result is written back to the header, the DONE field is also written back.

CAUTION

The done and status writebacks are not performed should the channel signal any error during processing. This detected error includes, but is not limited to a failing, unmasked ICV Check in an EU.

11.5.1 Arbitration Among Channels

All channels share a set of common resources, namely, use of EUs, use of the controller to perform data transfers, and use of the polychannel itself to implement channel activity. This section discusses the arbitration mechanisms for these shared facilities.

11.5.1.1 Arbitration for Use of the Controller

No arbitration for use of the controller is necessary. Since channels execute one at a time, a channel experiences no contention when sending a request to the controller. In effect, when a channel wins arbitration for use of the polychannel, it wins use of the controller as well.

11.5.1.2 Arbitration for Use of Execution Units

While one channel has a particular EU reserved, it is possible for multiple other channels to request use of that same EU. Arbitration is necessary to determine which channel gets use of the EU next. To accomplish this, there is an arbiter for each type of execution unit.

EU arbitration can be either round-robin or a weighted priority-based scheme, depending on the values of CHN3_EU_PR_CNT and CHN4_EU_PR_CNT in the Master Control Register (Section 11.6.3.7, “Master Control Register (MCR)”). For more about this, see Section 11.5.1.3, “Arbitration Algorithms.”

If a channel needs two EUs, a primary and a secondary, it requests them one at a time. Sometimes a channel reserves one EU and then have to wait for some other channel(s) to finish before obtaining the second requested EU. Though such waiting may occur, the requests are always eventually satisfied. Deadlock is avoided through the following design rules:

1. The channel always requests the secondary EU first.

2. In cases where both a primary and secondary are used, the choices for primaries and secondaries are distinct sets. Primaries are AESA, AFHA, DES, and KFHA, and the secondaries are MDHA and CRCU.

Since primaries and secondaries are distinct sets, and primary and secondary requests are strictly ordered, no deadlock is possible.

11.5.1.3 Arbitration Algorithms

This section applies to both arbitration for use of the polychannel, and arbitration for use of execution units. Control fields for both are in the Master Control Register (Section 11.6.3.7, “Master Control Register (MCR)”):

- CHN3_BUS_PR_CNT and CHN4_BUS_PR_CNT control polychannel arbitration
- CHN3_EU_PR_CNT and CHN4_EU_PR_CNT control EU arbitration

In this section we refer to generic control fields CHN3_xx_PR_CNT and CHN4_xx_PR_CNT, where the “xx” refers to either “BUS” or “EU”.

If both CHN3_xx_PR_CNT and CHN4_xx_PR_CNT are zero (the default), arbitration is round-robin, described in Section 11.5.1.3.1, “Round-Robin Arbitration. If they are set to non-zero values, the arbiter implements a weighted priority scheme, described in Section 11.5.1.3.2, “Priority Arbitration.”

The SEC does not dynamically adjust its own transaction priorities. System software, however, can adjust SEC transaction priority in realtime, with the change in priority taking effect immediately.

11.5.1.3.1 Round-Robin Arbitration

In round-robin arbitration, requesting channels are granted access in rotating numerical order: 1, 2, 3, 4, 1, 2, ... etc.

11.5.1.3.2 Priority Arbitration

When arbitrating on the priority scheme, the priority is as follows:

- Channel 1—Highest priority
- Channel 2—Second highest priority, unless CHN3_xx_PR_CNT or CHN4_xx_PR_CNT has expired
- Channel 3—Third priority, unless CHN4_xx_PR_CNT expired
- Channel 4—Lowest priority, until CHN4_xx_PR_CNT expired

Initially, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, a weight-based scheme is used. When channel 3 has lost arbitration the number of times specified in CHN3_xx_PR_CNT, channel 3 replaces channel 2 as the second-highest priority in the next round of arbitration. Likewise, when channel 4 has lost arbitration the number of times specified in CHN4_xx_PR_CNT, channel 4 replaces channel 2 as the second-highest priority in the next round of arbitration. Channel 1 always has the highest priority, but it cannot make back to back requests, so the second highest priority channel wins arbitration either immediately, or after one win from channel 1.

11.5.2 Polychannel Registers

11.5.2.1 Traffic Counters

11.5.2.1.1 Fetch FIFO Enqueue Counter

The Fetch FIFO Enqueue Counter, shown in Figure 11-8, indicates the total number of descriptor addresses that have been enqueued to the channel fetch FIFOs. If this counter reaches all 1s, the next count causes it to roll over to all 0s and set the FFE_CNT bit of the Interrupt Enable Register (see Section 11.6.3.2, “Interrupt Enable Register (IER)”).

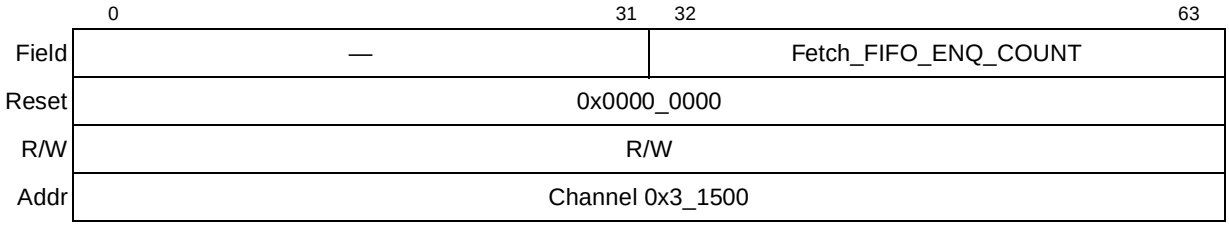


Figure 11-8. Fetch FIFO Enqueue Counter

11.5.2.1.2 Descriptor Finished Counter

The Descriptor Finished Counter, shown in Figure 11-9, indicates the total number of descriptors that have successful completed processing. It does not count descriptors that halt due to error. If this counter reaches all 1s, the next count causes it to roll over to all 0s and set the DF_CNT bit of the Interrupt Enable Register (see Section 11.6.3.2, “Interrupt Enable Register (IER)”).

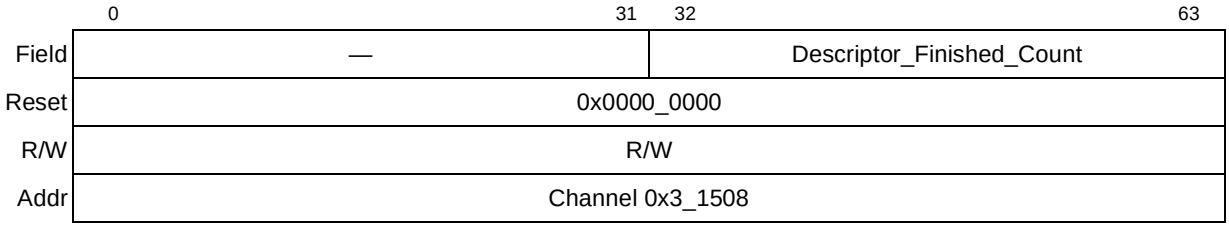


Figure 11-9. Descriptor Finished Counter

11.5.2.1.3 Data Bytes In Counter

The Data Bytes In Counter, shown in Figure 11-10, indicates the total number of bytes written into a primary EU input FIFO. If other parcels such as context or ICV are placed in the input FIFO, they are not counted. For cases where a secondary EU is used, data going only to the secondary EU (such as a hash-only region or authentication data) is counted. In no case is data counted twice by the same counter.

If this counter reaches all 1s, the next count causes it to roll over to all 0s and set the DI_CNT bit of the Interrupt Enable Register (see Section 11.6.3.2, “Interrupt Enable Register (IER)”).

If this counter is read by software in 32-bit increments, then the least significant 32 bits must be read first, followed by the most significant 32 bits. If this counter is written by software in 32 bit increments, then

the most significant 32 bits must be written first, followed by the least significant 32 bits. Note that 32 bit reads and writes must not be interleaved (that is, read low, write low, read high, write high is not allowed). These restrictions are required to maintain counter coherency.

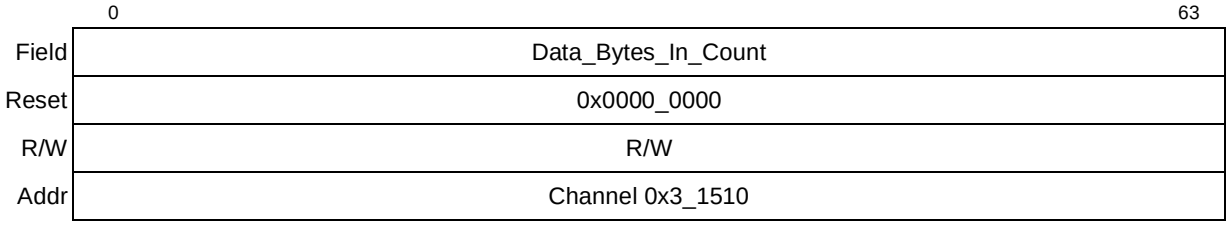


Figure 11-10. Data Bytes In Counter

11.5.2.1.4 Data Bytes Out Counter

The Data Bytes Out Counter, shown in Figure 11-11, indicates the total number of payload bytes read from an EU output FIFO. If other parcels such as context or ICV are read from the output FIFO, they are not counted. In no case is data counted twice by the same counter.

If this counter reaches all 1s, the next count causes it to roll over to all 0s and set the DO_CNT bit of the Interrupt Enable Register (see Section 11.6.3.2, “Interrupt Enable Register (IER)”).

If this counter is read by software in 32-bit increments, then the least significant 32 bits must be read first, followed by the most significant 32 bits. If this counter is written by software in 32 bit increments, then the most significant 32 bits must be written first, followed by the least significant 32 bits. Note that 32 bit reads and writes must not be interleaved (that is, read low, write low, read high, write high is not allowed). These restrictions are required to maintain counter coherency.

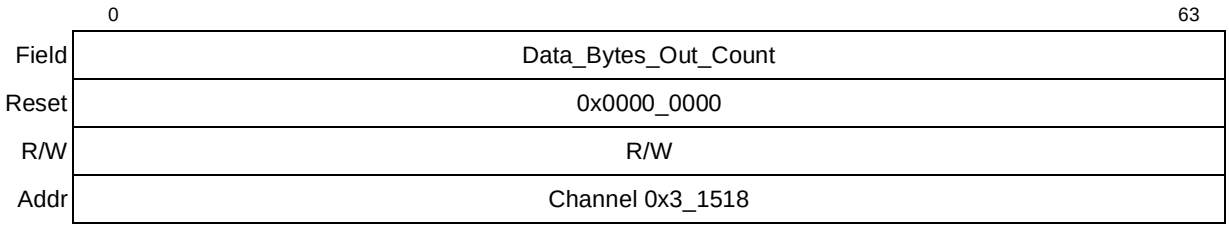


Figure 11-11. Descriptor Finished Counter

11.5.3 Channel Interrupts

The channel can assert done and error interrupts to the controller. The status of the registered channel interrupts is available in the controller interrupt status register. The channel does not have an internal interrupt mask, but the controller can be programmed to block channel interrupts via its interrupt enable register (see Section 11.6.3.2, “Interrupt Enable Register (IER)”). Any asserted interrupt bit results in the assertion of the interrupt pin appropriate to the Channel, per the Remap Channel Address bits in the Controller’s Master Control Register.

11.5.3.1 Channel Done Interrupt

Whether and when a channel done interrupt is generated depends on the setting of the Channel Configuration Register NT and CDIE bits in the CCR (see Figure 11-12). If the CDIE (Channel Done Interrupt Enable) is set, the channel generates an interrupt event after every successfully completed descriptor (Notification Type set to Global), or after each successfully completed descriptor with the DN (Done Notification) bit set in the header word of the descriptor. If the EU(s) signal any error during processing, the channel done interrupt is not generated.

Even if multiple channel done interrupt events are generated by a channel before the first can be cleared by the host, the interrupt events are not lost. The controller keeps count of the backlog of channel done interrupts from each channel (see Section 11.6.2, “Controller Interrupts”).

11.5.3.2 Channel Error Interrupt

The Channel Error Interrupt is generated when an error condition occurs during descriptor processing. The error could be in one of the EUs reserved by the channel, a bus error for a transaction requested by the channel, or in the channel itself. The channel error interrupt is asserted as soon as the error condition is detected. The type of error condition is reflected in the ERROR field of the Channel Status Register (CSR).

For most error types, the error causes the corresponding channel to halt. Any EUs reserved by the halted channel continue to be reserved until the channel reset occurs. Other channels continue normal processing, though they may be held up if they need an EU that is reserved by a halted channel.

Handling of errors depends on the error type. Detail about each error type is given in Table 11-15. For some types, the host must clear the source of the error before restarting the channel. If the channel is halted, the host restarts it by setting the No-Pop-Reset, Continue or Reset bits of the CCR (see Section 11.5.4.1, “Channel Configuration Register (CCR)”).

11.5.4 Channel Registers

These registers are replicated for each of the 4 channels in the polychannel.

11.5.4.1 Channel Configuration Register (CCR)

The channel configuration register, shown in Figure 11-12, contains bits that allow the user to configure and reset the channel.

	0	29	30	31
Field	—	NPR	CON	R
Reset	0			
R/W	R/W			
Addr	Channel 1: 0x3_1108, Channel 2: 0x3_1208, Channel 3: 0x3_1308, Channel 4: 0x3_1408			

	32 50	51	52 54	55	56	57	58	59	60	61	62	63
Field	—	PBS	—	BS	IWSE	—	EAE	CDWE	AWSE	NT	CDIE	—
Reset	0											
R/W	R/W											
Addr	Channel 1: 0x3_110C, Channel 2: 0x3_120C, Channel 3: 0x3_130C, Channel 4: 0x3_140C											

Figure 11-12. Channel Configuration Register

Table 11-11 describes the CRR register fields.

Table 11-11. Channel Configuration Register Signals

Bits	Name	Description
0–28	—	Reserved, must be cleared.
29	NPR[1]	No-Pop-Reset. 0 No special action. 1 Causes the same channel reset actions as bit CON, except that the Fetch FIFO is left unchanged, such that the Channel picks up by re-fetching the previous descriptor. This permits debug of a descriptor in-place without having to rewrite the descriptor pointer into the Fetch FIFO. • If the NPR bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request. • If the NPR bit is set after the channel has been assigned one or more EUs, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s).
30	CON	Continue bit. 0 No special action. 1 Causes the same channel reset actions as bit R, except that the Fetch FIFO and the lower half of the CCR register (bits 32-63) are not cleared. After the reset sequence is complete, this bit automatically returns to 0 and the channel resumes normal operation, servicing the next descriptor pointer in the Fetch FIFO, if any. • If the CON bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request. • If the CON bit is set after the channel has been assigned one or more EUs, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s).

Table 11-11. Channel Configuration Register Signals (continued)

Bits	Name	Description
31	R	Reset channel. 0 No special action. 1 Causes a software reset of the channel, clearing all its registers. Some actions depend on what the channel is doing when the bit is set: • If the R bit is set while the channel is requesting an EU assignment from the controller, the channel cancels its request. • If the R bit is set after the channel has been assigned one or more EUs, the channel requests a write from the controller to set the software reset bit of each reserved EU. The channel then releases the EU(s). After the reset sequence is complete, the channel returns to the idle state and the R bit automatically returns to 0 and resumes normal operation.
32–48	—	Reserved, must be cleared.
49	FCC	Fast clock counting. 0 Watchdog Timer counts normally 1 Watchdog Timer counts in an accelerated fashion (force-assert several selected bits in timer) to assist with functional testing.
50	WGN	Watchdog go now. 0 Watchdog Timer disabled 1 Watchdog Timer enabled
51	PBS	Permit byte summing. 0 Bytes written to EU input FIFOs and read from EU output FIFOs are not counted in the Data Bytes Counters 1 Bytes written to EU input FIFOs and read from EU output FIFOs are counted in the Data Bytes Counters
52–54	—	Reserved, must be cleared.
55	BS	Burst size. The SEC accesses long text-parcels in main memory through bursts of programmable size. 0 Burst size is 64 bytes 1 Burst size is 128 bytes
56	IWSE	ICV writeback status enable. 0 No special action. 1 If the descriptor calls for ICV checking, then at the completion of descriptor processing, the channel writes back to the descriptor header all of the writeback information shown in Table 11-5, that is, the DONE, ICCR0, and ICCR1 fields.[2]
57	—	Reserved, must be cleared.
58	EAE	Extend address enable. This bit determines whether the channel uses a 36-bit address bus or a 32-bit address bus. 0 Channel's address bus is 32 bits. 1 Channel's address bus is 36 bits.
59	CDWE	Channel done writeback enable. 0 Channel Done Writeback disabled. 1 Channel Done Writeback enabled. Upon successful completion of descriptor processing, if the NT bit is reset (Global), or if the DN (Done Notification) bit is set in the header word of the descriptor, then the channel notifies the host by writing back the descriptor header with the DONE field shown in Table 11-5. This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed. [1]

Table 11-11. Channel Configuration Register Signals (continued)

Bits	Name	Description
60	AWSE	Always writeback status enable. 0 No special action. 1 At the completion of processing each descriptor, the channel writes back to the descriptor header all of the writeback information shown in Table 11-5, that is, the DONE, ICCR0, and ICCR1 fields. In this case, IWSE has no effect.[1]
61	NT	Notification type. This bit controls when the channel generates channel done notification. Channel done notification can take the form of an interrupt or modified header writeback or both, depending on the state of the CDIE and CDWE control bits. 0 Global notification: The channel generates channel done notification (if enabled) at the end of each descriptor. 1 Selected notification: The channel generates channel done notification (if enabled) at the end of every descriptor with the DN bit set in the descriptor header.
62	CDIE	Channel done interrupt enable. 0 Channel Done Interrupt disabled 1 Channel Done Interrupt enabled. Upon successful completion of descriptor processing, if the NT bit indicates Global, or if the DN (Done Notification) bit is set in the header word of the descriptor, then notify the host by asserting an interrupt. [1] Refer to Section 11.5.3, "Channel Interrupts," for complete description of channel interrupt operation.
63	—	Reserved, must be cleared.

1 WARNING: When using reset bits R, CON and NPR: the configuration register must be polled to confirm completion of a multi-cycle reset sequence. The length of time required for this reset sequence is dependent on several factors and should be considered indeterminate. Completion is indicated by the self-negating of the asserted reset bit. Failure to ensure completion of reset prior to writing to the channel may result in a channel hang condition.

2 WARNING: The done interrupt, done writeback, and status writeback does not occur if an EU produces an error interrupt to the channel. In particular, if the ICV check error interrupt is enabled in the EU (see the ICE bit in the EU's interrupt nask register), and the ICV check finds a mismatch, then the channel produces an error interrupt, but no channel done interrupt and no writebacks.

The following tables give further details as to how to use the Channel Configuration Register and the Descriptor Header Control Word to control done interrupt generation and header control word writeback.

Table 11-12. Writeback Options

AWSE Register bit 60	CDWE Register bit 59	IWSE Register bit 56	NT Register bit 61	DN Header bit 63	Writeback Action for a Descriptor Completing Without Error
1	x	x	x	x	Write back Header fields DONE, ICCR0, ICCR1
0	1	x	1	0	No write back performed
0	1	x	1	1	Write back Header field DONE
0	1	x	0	x	Write back Header field DONE
0	x	1	x	x	If the descriptor header indicated ICV checking in AESU, CRCU, KEU, or MDEU, then write back Header fields DONE, ICCR0, and ICCR1.

Table 11-13. Done Interrupt Options

NT Register bit 61	DN Header bit 63	CDIE Register bit 62	Done Interrupt action by Channel to Controller for a Descriptor Completing Without Error
x	x	0	Never assert done interrupt
0	x	1	Assert done interrupt
1	0	1	Never assert done interrupt
1	1	1	Assert done interrupt

11.5.4.2 Channel Status Register (CSR)

The channel status register, shown in Figure 11-13, contains status fields and counters that provide status information regarding the channel's processing of the current descriptor. This is intended for debug use.

In case of a channel error, the Error field indicates the type of error.

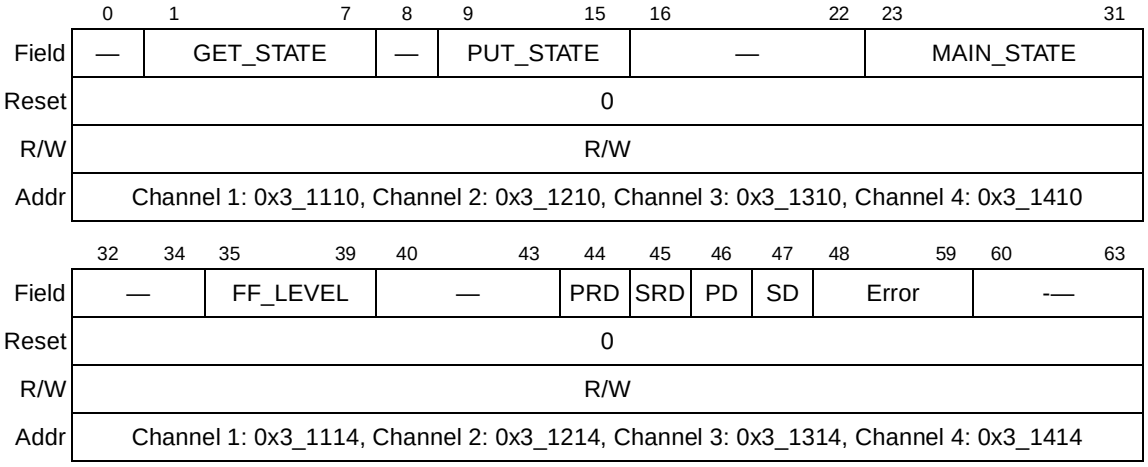


Figure 11-13. Channel Status Register

The channel's multiple state-machine architecture makes it difficult to determine the status of the channel. The channel can be truly be considered idle only if GET_STATE, PUT_STATE, and MAIN_STATE are all zero, and the FF_LEVEL is also zero.

Table 11-14 describes the channel status register fields.

Table 11-14. Channel Status Register Signals

Bits	Name	Description
0	—	Reserved
1–7	GET_STATE	Get state machine state. This field reflects the state of the Get State Machine when it last went to sleep, or the state captured when an error occurred. For debug purposes only.
8	—	Reserved
9–15	PUT_STATE	Put State Machine State. This field reflects the state of the Put State Machine when it last went to sleep, or the state captured when an error occurred. For debug purposes only.

Table 11-14. Channel Status Register Signals (continued)

Bits	Name	Description
16–22	—	Reserved
23–31	MAIN_STATE	Main state machine state. This field reflects the state of the Main State Machine when it last went to sleep, or the state captured when an error occurred. For debug purposes only.
32–34	—	Reserved, must be cleared.
35–39	FF_LEVEL	Fetch FIFO level. This 5 bit counter indicates how many pointers are currently stored in the Fetch FIFO.
40–43	—	Reserved, must be cleared.
44	PRD	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
45	SRD	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
46	PD	Primary EU done. The PRI_DONE bit reflects the state of the done interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing. This means that final values are available from EU registers. For EUs with output FIFOs, it means that all textdata output has been placed in the output FIFO. For EUs that provide context out through the output FIFO, the EU places the context in the output FIFO after asserting PRI_DONE.
47	SD	Secondary EU done. The SEC_DONE bit reflects the state of the done interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing. This means that final values are available from EU registers.
48–59	Error	Error bits for the channel. See below.
60–63	—	Reserved

Table 11-15 lists the types of errors in the Error field of the Channel Status Register. Multiple errors are possible. If any of these bits are set, a Channel Error Interrupt is generated. Most errors also halt the channel. For some error types, the host must take actions to clear the error bit before restarting the channel, as described in Table 11-15. See the R and CON bits of the CCR for information on restarting a channel.

Table 11-15. Channel Status Register Error Field Definitions

Value	Error
48	DOF–Double Fetch Fifo write overflow Error. This bit is set when the channel Fetch Fifo is full, SOF is set, and another write has been made to the fetch FIFO. This error halts the channel. To clear this error, the host must write a ‘1’ to this bit.
49	SOF–Single Fetch Fifo write overflow Error. This bit is set when the channel Fetch Fifo is full and another write has been made to the Fetch Fifo. The channel continues processing, but the descriptor pointer is lost. To clear this error, the host must write a ‘1’ to this bit.
50	MDTE–Master Data Transfer Error. When the SEC, while acting as a bus master, detects an error, the controller passes this error to the channel. This error halts the channel. Restarting the channel clears this bit.
51–52	Reserved
53	IDH–Illegal descriptor header. Possible causes of an illegal descriptor header are: • Invalid primary EU indicated by op0 field in descriptor header. • Invalid secondary EU indicated by op1 field in descriptor header. This error halts the channel. Restarting the channel clears this bit.
54	Reserved
55	EUE–EU error. An EU assigned t o t hi s channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register. This error halts the channel. To clear this error, the host must clear the error source in the EU that produced the error.
56	WDT–Watchdog timeout. The main state machine stayed asleep too long. This timer runs only after EUs have been reserved, and does not run if the primary EU is the RNGU or PKEU. The timeout interval is controlled by the FCC field of the Channel Configuration Register. This error halts the channel. Restarting the channel clears this bit.
57	SGLM–Scatter/Gather Length Mismatch. Indicates the total data size covered by a gather link table did not match the total data size from the main descriptor. This error halts the channel. Restarting the channel clears this bit.
58	RSI–Raid Size Incorrect. The channel was provided with a descriptor of type RAID_XOR with data sizes not permitted.
59	RSG–Raid Scatter Gather Error. The channel was provided with a descriptor of type RAID_XOR with a j bit set. Use of scatter/gather is not permitted with RAID_XOR type descriptors.

11.5.4.3 Current Descriptor Pointer Register (CDPR)

The CDPR, shown in Figure 11-14, reflects the value of the head-end of the Fetch FIFO, which contains the address of the descriptor which the channel is currently processing.

	0 — 27	28 31	32 — 63
Field	—	EPTR	CUR_DES_PTR_ADRS
Reset	(not reset)		
R/W	R		
Addr	Channel 1: 0x3_1140, Channel 2: 0x3_1240, Channel 3: 0x3_1340, Channel 4: 0x3_1440		

Figure 11-14. Current Descriptor Pointer Register

Table 11-16 describes the current descriptor pointer register fields.

Table 11-16. Current Descriptor Pointer Register Signals

Bits	Name	Description
0–27	—	Reserved, must be cleared.
28-31	EPTR	Extended Pointer: Concatenated as the top 4 bits of the CUR_DES_PTR_ADRS when EAE is high (see the EAE bit in Table 11-11).
32–63	CUR_DES_PTR_ADRS	Current Descriptor Pointer Address. Pointer to system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the channel requests a fetch of a descriptor from the controller. The value from the Fetch FIFO is transferred to the current descriptor pointer register immediately after the fetch is completed. This address is used as the destination for writeback of the modified header dword, if header writeback notification is enabled.

11.5.4.4 Fetch FIFO (FF)

Each channel contains a Fetch FIFO to store a queue of pointers to descriptors, which the channel processes.

A Fetch FIFO entry is displayed in Figure 11-15. Each entry contains the address of the first byte of a descriptor to be processed. In typical operation, the host CPU creates a descriptor in memory containing all relevant mode and location information for the SEC, then “launch” the descriptor by writing its address to the SEC’s Fetch FIFO.

The Fetch FIFO can hold up to 24 Descriptor Pointers at a time. When processing of the current descriptor is completes, the descriptor pointed to by the next location in the Fetch FIFO is read to launch the next descriptor.

The Fetch Address is written into the FIFO only if the write includes the least significant byte (bits 56–63). If Extend Address Enable is high (see the EAE bit in Table 11-11), then the Extended Fetch Address must be written before or concurrent with the Fetch Address.

	0–27	28–31	32–63
Field	—	EPTR	CUR_DES_PTR_ADRS
Reset	0x0000_0000		
R/W	W		
Addr	Channel 1: 0x3_01148, Channel 2: 0x3_01248, Channel 3: 0x3_01348, Channel 4: 0x3_01448		

Figure 11-15. Fetch FIFO

Table 11-17 describes the Fetch FIFO fields.

Table 11-17. Fetch FIFO Field Descriptions

Bits	Name	Description
0–27	—	Reserved, must be cleared.
28-31	EPTR	Extended Pointer: Concatenated as the top 4 bits of the FETCH_ADRS when EAE is high (see the EAE bit in Table 11-11).
32–63	FETCH_ADR	Fetch Address. Pointer to system memory location of a descriptor the host wants the SEC to fetch.

11.5.4.5 Descriptor Buffer (DB)

The descriptor buffer (DB) provides read-only access to the descriptor currently being processed by the channel. All descriptors are 8 dwords long. For descriptor format, see Figure 11-3.

Note that the Descriptor Buffer is working storage and the Channel may modify the contents of the Descriptor Buffer during processing.

11.5.4.6 Scatter and Gather Link Tables (SLT, GLT)

A pointer in the DB refers to a gather link table (GLT) or a scatter link table (SLT) if the J bit in the dword is set. As a channel works on a DB pointer entry, the GLT/SLT must be loaded into Channel memory. Readback of the GLT/SLT is provided for debug purposes.

Figure 11-16 shows the GLT.

	0–15	16–21	22	23	24–27	28–31	32–63
Field	SEGLEN	—	R	N	—	EPTR	SEGPTR
Reset	(not reset)						
R/W	R/W						
Addr	Channel 1: 0x3_11c0-0x3_11df; Channel 2: 0x3_12c0-0x3_12df; Channel 3: 0x3_13c0-0x3_13df; Channel 4: 0x3_14c0-0x3_14df						

Figure 11-16. Gather Link Table

Figure 11-17 shows the SLT.

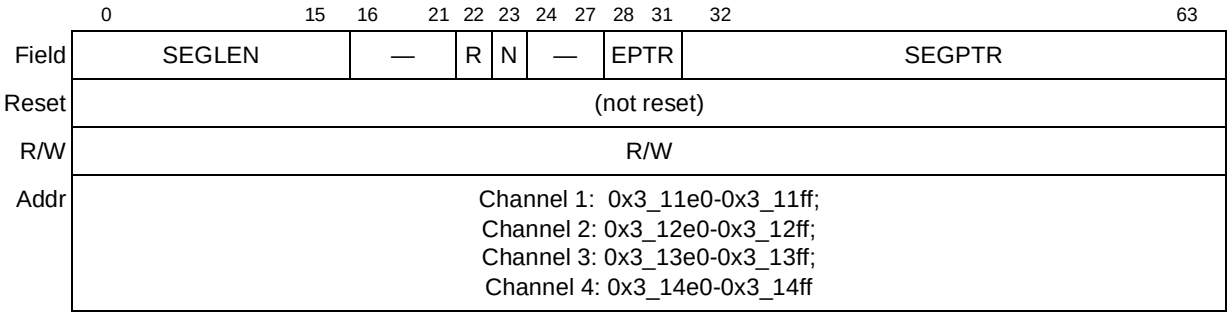


Figure 11-17. Scatter Link Table

11.6 Controller

The controller within the SEC is responsible for mastering bus transfers on behalf of the channels. The controller interfaces to the hosts via the master and slave bus interfaces and to the channels and EUs via internal buses. All transfers between the hosts and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Accept and execute commands from the slave system bus to read or write memory-mapped locations (up to 64 bits) anywhere in the SEC.
- Accept and execute requests from the polychannel to transfer blocks of bytes among system memory, EUs, and the channels.
- Realign read and write data to the proper byte alignment
- Monitor interrupts from channels and pass them to the host(s)

11.6.1 Bus Transfers

The controller is involved in transfers on the system bus and the internal bus. The system bus actually refers to two buses:

- Slave bus
- Master bus

The internal bus is a private 64-bit slave bus, with the controller block as the sole master. All accesses to SEC from the system bus go through the controller. The controller also directs transfers over the internal bus.

For host-controlled access, the host uses the external slave bus to access the controller as a slave, and the controller relays the read or write accesses to the proper block over the internal bus. When a write command is received from the system bus, the controller takes the data and sends it to whichever internal location is indicated by the address. For a read, the controller goes to the internal location, fetches the requested data from the specified address (if allowed), and returns it over the system bus.

For channel-controlled access, the channels make requests to the controller to perform data transfers. The channel specifies data lengths and addresses for the internal and system buses. The controller can queue

up to four requests. The controller dequeues requests and performs the required transfer. Most transfers involve not only the internal bus, but also the external master bus with the controller as bus master. Following are examples of the various types of bus-master transfers:

1. Obtaining a descriptor
   - channel makes request
   - controller arbitrates for use of the system bus and performs read from external memory
   - controller sends descriptor to channel over the internal bus
2. Transferring data length parameter from channel to EU
   - channel makes request
   - controller transfers data from channel to EU over the internal bus
3. Obtaining input data from external memory for input to an EU
   - channel makes request
   - controller arbitrates for use of the system bus and performs read from external memory
   - controller sends data to EU over the internal bus. If insnooping, data is sent to two EUs
4. Writing output data from an EU back to external memory
   - channel makes request
   - controller begins reading data from the EU into a controller FIFO. If outsnooping, the same data is also read by another EU. Meanwhile, the controller arbitrates for use of the system bus.
   - controller performs write to external memory
5. Status writeback
   - channel makes request
   - controller reads value from channel and arbitrates for use of the system bus
   - controller performs write to external memory, overwriting the writeback field of the descriptor’s header

11.6.1.1 System Bus Master Read

Following is more detail on the sequence of events for a system bus read with the controller as master:

1. Channel asserts bus read request to the controller
2. Channel furnishes external read address, internal write address, and transfer length
3. Controller asserts request to the system bus through the Magenta master interface
4. Controller waits for system bus read to begin
5. When bus read begins, controller receives data from the master interface and performs a write to the appropriate internal address supplied by the channel. Data may be realigned byte-wise by the controller if either:
   - the external read address was not on an 8-byte boundary, or
   - the internal write address was not on an 8-byte boundary

6. Transfer continues until the bus read is completed and the controller has written all data to the appropriate internal address. The master interface continues making bus requests until the full data length is read.

When the SEC performs a transaction as master, it is possible for the intended slave to terminate the transfer due to an error. The SEC's transaction requests are posted to the device's target queue, after which the device takes responsibility for completing the transaction or signalling error. An error in an SEC initiated transaction is also reported by the SEC via the channel interrupt status register. The host is able to determine which channel generated the interrupt by checking the ISR for the channel ERROR bit.

11.6.1.2 System Bus Master Write

More detail on the sequence of events for a system bus write with the controller as master is as follows:

1. Channel asserts its bus write request to the controller
2. Channel furnishes internal read address, external write address, and transfer length
3. Controller performs a read from the appropriate internal address supplied by the channel, loads the write data into its FIFO, asserts a request to the system bus through the Magenta master interface, and waits for the system bus to become available
4. When the system bus becomes available, controller writes data from its FIFO to the master interface

11.6.2 Controller Interrupts

The controller produces a single interrupt to each of the hosts—a primary interrupt and a secondary interrupt. Only the Channel Error, Channel Done and Channel Done Overflow status bits can cause the secondary interrupt to assert. All other interrupt types assert the primary interrupt.

11.6.2.1 Controller Primary Interrupt

All interrupt outputs from other SEC blocks are fed to the controller as interrupt conditions. In addition, the controller itself detects some interrupt conditions. The controller maintains an Interrupt Status Register (ISR) with bits corresponding to all of these possible interrupt conditions. If an interrupt condition occurs and the corresponding bit of the interrupt enable register (IER) is high, then the associated Interrupt Status Register bit is set, indicating the presence of a pending interrupt. Whenever any bits are set in the Interrupt Status Register, the controller asserts its primary interrupt output line to the host. One exception is that the Channel Error, Channel Done and Channel Done Overflow status bits only cause the primary interrupt output line to assert if the channel is not remapped to the alternative address by [RCA1, RCA2, RCA3 or RCA4]MCR.

To handle an interrupt, the host must read the Interrupt Status Register to determine the source. It may then need to do further reads of interrupt status registers of other blocks to get more detailed information about the cause. In some cases, the host may need to take action to clear the root cause of the interrupt. After that, the host can clear the desired bit of the Interrupt Status Register by writing a 1 to the corresponding bit of the Interrupt Clear Register (ICR). If the cause of the interrupt condition has not been cleared, or if there is some other interrupt condition from the same source, then the Interrupt Status Register bit clears for a cycle and go high again, and the interrupt output line to the host remains high. If the Interrupt Status

Register bit is successfully cleared and no other interrupt conditions are present, the controller de-asserts its interrupt output. If any interrupts are still pending in the Interrupt Status Register, the interrupt output remains asserted.

Note that EU interrupt conditions may be blocked at two different levels. There is an interrupt mask register in each EU which can block particular interrupt conditions before they reach the EU's interrupt status register. In addition, interrupts from EUs can be individually blocked by bits of the controller's interrupt enable register before they reach the controller's interrupt status register. For normal operation, interrupts from EUs are typically disabled in the controller's Interrupt Enable Register, but they still reach the channel, and the channel produces Done or Error interrupts to the host as needed.

Interrupt conditions from the channels and controller can only be blocked through the controller's Interrupt Enable Register.

A channel can generate frequent interrupts, especially if it is configured to interrupt at the completion of each descriptor. To make sure that the host receives the right number of interrupts, each channel Done interrupt has a special "queuing" feature. If multiple Channel Done interrupts are generated before the first is cleared, then the additional interrupts are counted by the controller. Each time the host clears a channel interrupt, the count is decremented. If the host clears the channel interrupt and the count reaches zero, the Channel Done interrupt is de-asserted. If the count does not reach zero, the controller de-asserts the interrupt for one cycle and then re-asserts it again.

Up to 15 interrupts can be queued for each channel. If the count of queued interrupts for any channel exceeds 15, that channel's "Done Overflow" bit is set (see Figure 11-18).

11.6.2.2 Controller Secondary Interrupt

Whenever the Channel Done, Channel Error or Channel Done Overflow bits are set in the Interrupt Status Register and the channel has been remapped to the alternative address by [RCA1, RCA2, RCA3 or RCA4]MCR, the controller asserts the secondary interrupt output line to the secondary host. No other Interrupt Status Register status bits can cause the secondary interrupt to assert.

11.6.3 Controller Registers

The controller registers are described in detail in the following sections.

11.6.3.1 EU Assignment Status Register (EUASR)

The EUASR, showed in Figure 11-18, indicates which EUs are reserved by a particular channel. When an EU is already assigned, it is inaccessible to any other channel.

	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
Field	—	AFEU	—	MDEU	—	AESU	—	DEU
Reset	0xF	0x0	0xF	0x0	0xF	0x0	0xF	0x0
R/W	R/W							
Addr	0x3_1028							

	32–35	36–39	40–43	44–47	48–51	52–55	56–59	60–63
Field	—	—	CRCU	KEU	—	PKEU	—	RNG
Reset	0xF	0xF	0x0	0x0	0xF	0x0	0xF	0x0
R/W	R							
Addr	0x3_102C							

Figure 11-18. EU Assignment Status Register (EUASR)

In Table 11-18, a 4-bit field indicates the channel to which an EU is assigned.

Table 11-18. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0xA–0xE	Undefined
0xF	Unavailable

11.6.3.2 Interrupt Enable Register (IER)

The SEC controller generates the interrupt outputs from all possible interrupt sources. These sources can be individually enabled by the interrupt enable register. If enabled, the interrupt source value, when active, is captured into the interrupt status register. Figure 11-19 shows the bit positions of each potential interrupt source. Each interrupt source is individually enabled by setting its corresponding bit. At reset, all bits are disabled.

For normal operation the Interrupt Enable Register should be programmed as follows: Leave channel interrupts enabled, while disabling interrupts from the EUs. In case of an EU error, the channel using that EU generates the appropriate interrupt to the host.

NOTE

The recommended value for this register is 0x0031_0fff_0000_0000.
Execution Unit interrupt bits are provided as a convenience during debug.

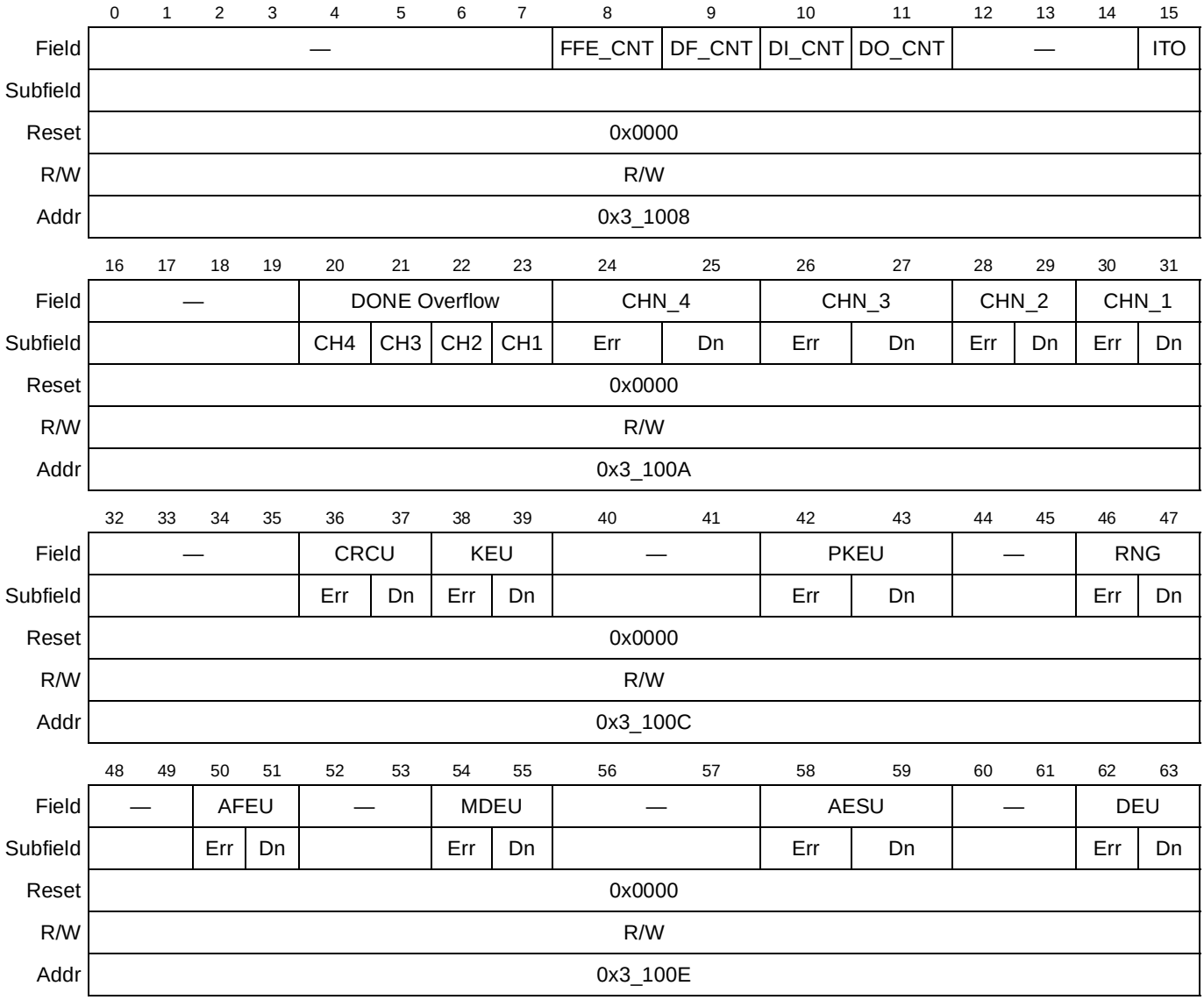


Figure 11-19. Interrupt Enable Register

Table 11-19 describes the register fields in the interrupt enable register, interrupt status register, and interrupt clear register.

Table 11-19. Field Names in Interrupt Enable, Interrupt Status, and Interrupt Clear Registers

Bits	Name	Description
0–7	—	Reserved
8	FFE_CNT	Fetch FIFO enqueue count rollover 0 No rollover. 1 The Fetch Fifo Enqueue Counter rolled over to zero (see Section 11.5.2.1.1, “Fetch FIFO Enqueue Counter”).

Table 11-19. Field Names in Interrupt Enable, Interrupt Status, and Interrupt Clear Registers (continued)

Bits	Name	Description
9	DF_CNT	Descriptor Finished Count Rollover 0 No rollover. 1 The Descriptor Finished Counter rolled over to zero (see Section 11.5.2.1.2, “Descriptor Finished Counter”).
10	DI_CNT	Data In Count Rollover 0 No rollover. 1 The Data In Counter rolled over to zero (see Section 11.5.2.1.2, “Descriptor Finished Counter”).
11	DO_CNT	Data Out Count Rollover 0 No rollover. 1 The Data Out Counter rolled over to zero (see Section 11.5.2.1.2, “Descriptor Finished Counter”).
12–14	—	Reserved
15	ITO	Internal Time Out 0 No Internal Time Out 1 An Internal Time Out was detected **Note:** Internal Time Out is an indication that a channel or EU has failed to respond to a slave read or write within 16 cycles, which would only occur in an impending hang condition. Assertion of this interrupt indicates the SEC Controller has completed the transaction to avoid a hang, however the ’completed’ transaction does not result in a successful read or write, and the interrupt advises the system that the slave transaction was unsuccessful.
20–23	Done Overflow	Overflow. Done Overflow (one bit for each channel—CH1 to CH4) 0 No Done Overflow 1 Done Overflow Error. Indicates that more than 15 Done interrupts were queued from the associated channel without a corresponding interrupt clear from the host.
24–31	Err and Dn bits for channels (CHN_1 to CHN_4)	Err 0 No error detected. 1 Error detected. Indicates that channel status register must be read to determine exact cause of the error. Dn 0 Not DONE. 1 DONE bit indicates that the corresponding channel has completed a descriptor.

Table 11-19. Field Names in Interrupt Enable, Interrupt Status, and Interrupt Clear Registers (continued)

Bits	Name	Description
36–39, 42–43, 46–47, 50–51, 54–55, 58–59, 62–63	Err and Dn bits for execution units (PKEU, etc.)	Err 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. Dn 0 Not Done 1 DONE bit indicates that the corresponding EU has completed its operation. This means that final values are available from EU registers. For EUs with output FIFOs, it means that all textdata output has been placed in the output FIFO. For EUs that provide context out through the output FIFO, the EU places the context in the output FIFO after asserting PRI_DONE.
0–9, 16–19, 32–35, 40–41, 44–45, 48–49, 52–53, 56–57, 60–61	—	Reserved, must be cleared.

11.6.3.3 Interrupt Status Register (ISR)

The interrupt status register, shown in Figure 11-20, contains fields representing all possible sources of interrupts. The interrupt status register is cleared either by a reset, or by setting the appropriate bits in the interrupt clear register. Figure 11-20 shows the bit positions of each potential interrupt source. The bit fields are described in Table 11-19.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—								FFE_CNT	DF_CNT	DI_CNT	DO_CNT	—			ITO
Subfield																
Reset	0x0000															
R/W	R															
Addr	0x3_1010															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—								CHN_4		CHN_3		CHN_2		CHN_1	
Subfield									Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000															
R/W	R															
Addr	0x3_1012															

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	—				CRCEU		KEU		—		PKEU		—		RNG	
Subfield					Err	Dn	Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R															
Addr	0x3_1014															

	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—		AFEU		—		MDEU		—		AESU		—		DEU	
Subfield			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R															
Addr	0x3_1016															

Figure 11-20. Interrupt Status Register

11.6.3.4 Interrupt Clear Register (ICR)

The interrupt clear register, shown in Figure 11-21, provides a means of clearing the interrupt status register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output pin $\overline{\text{IRQ}}$ (assuming the cleared bit in the ISR is the only interrupt source). If the input source to the ISR is a steady-state signal that remains active, the appropriate ISR bit, and subsequently $\overline{\text{IRQ}}$, is reasserted shortly thereafter. Figure 11-21 shows the bit positions of each interrupt source that can be cleared by this register. The complete bit definitions for the ICR can be found in Figure 11-21. The bit fields are described in Table 11-19.

When an ICR bit is written, it automatically clears itself 1 cycle later. That is, it is not necessary to write a 0 to a bit position that has been written with a 1.

NOTE

Interrupts are registered and sent based upon the conditions that cause them. If the cause of an interrupt is not removed, the interrupt returns a few cycles after it has been cleared using the ICR.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—								FFE_CNT	DF_CNT	DI_CNT	DO_CNT	—			ITO
Subfield																
Reset	0x0000															
R/W	W															
Addr	0x3_1018															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—				DONE Overflow				CHN_4		CHN_3		CHN_2		CHN_1	
Subfield					CH4	CH3	CH2	CH1	Err	Dn	Err	Dn	Err	Dn	Err	Dn
Reset	0x0000															
R/W	W															
Addr	0x3_101A															

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	—				CRCEU		KEU		—		PKEU		—		RNG	
Subfield					Err	Dn	Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	W															
Addr	0x3_101C															

	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—		AFEU		—		MDEU		—		AESU		—		DEU	
Subfield			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	W															
Addr	0x3_101E															

Figure 11-21. Interrupt Clear Register

11.6.3.5 ID Register

The Read-Only ID Register, shown in Figure 11-22, contains the same value as the IP Block Revision Register below. In updating existing software, use of this address is likely to be most convenient, since it is the same address used for version information in previous revisions of the SEC.

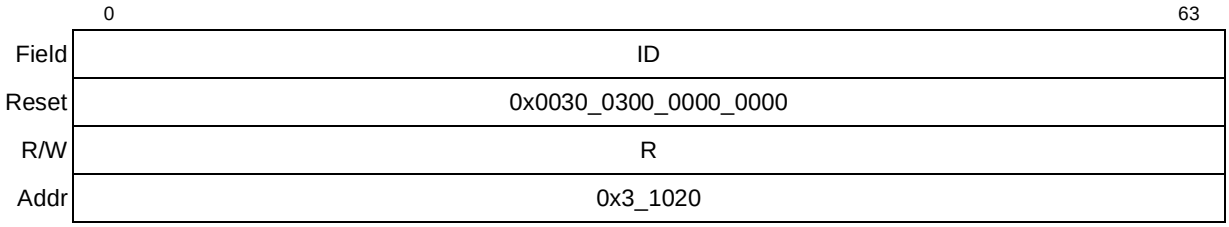


Figure 11-22. ID Register

11.6.3.6 IP Block Revision Register

The Read-Only IP Block Revision Register, shown in Figure 11-23, contains a 64-bit value that uniquely identifies the version of the SEC 3.0. The value of this register is 0x0030_0300_0000_0000.

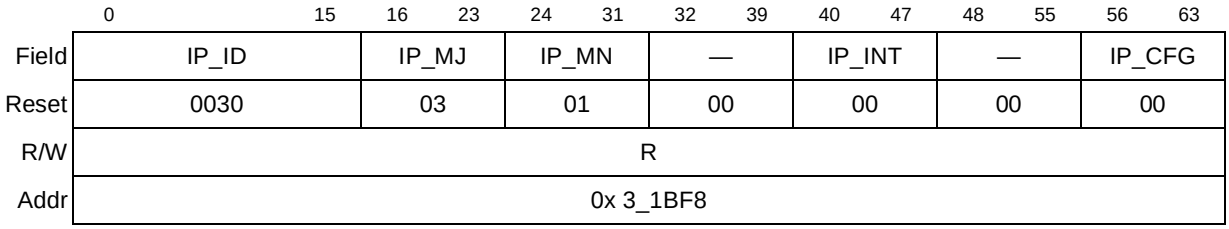


Figure 11-23. IP Block Revision Register

Table 11-20 describes the IP block revision register fields.

Table 11-20. IP Block Revision Register Fields

Bits	Name	Description
0–15	IP_ID	IP block identifier
16–23	IP_MJ	IP major revision number
24–31	IP_MN	IP minor revision number
32–39	—	Reserved
40–47	IP_INT	IP block integration options
48–55	—	Reserved
56–63	IP_CFG	IP block configuration options

11.6.3.7 Master Control Register (MCR)

The MCR, shown in Figure 11-24, controls certain functions in the controller and provides a means for software to reset the SEC.

	0 - 15	16	17	18	19	20	21	22	23	24 - 29	30	31
Field	0	RCA1	RCA2	RCA3	RCA4	—		PRIORITY		0	GIH	SWR
Reset	0x0000_0000											
R/W	R/W											
Addr	0x3_1030											

	32 39	40 47	48 55	56 63
Field	CHN3_EU_PR_CNT	CHN4_EU_PR_CNT	CHN3_BUS_PR_CNT	CHN4_BUS_PR_CNT
Reset	0x0000_0000			
R/W	R/W			
Addr	0x3_1034			

Figure 11-24. Master Control Register

Table 11-21 describes the master control register fields.

Table 11-21. Master Control Register Signals

Bits	Name	Description
0–15	—	Reserved
16	RCA1	Remap Channel Address 1. Writing a 1 to this bit remaps all Channel 1 host accessible registers to an alternate 4-Kbyte address range; in addition, Channel 1's Done and Error bits in the Interrupt Status Register causes the secondary interrupt to assert instead of the primary interrupt. 0 Channel 1 registers are accessible in the 0x11xx address range (default) 1 Channel 1 registers are accessible in the 0x01xx address range
17	RCA2	Remap Channel Address 2. Writing a 1 to this bit remaps all Channel 2 host accessible registers to an alternate 4-Kbyte address range; in addition, channel 2's Done and Error bits in the Interrupt Status Register causes the secondary interrupt to assert instead of the primary interrupt. 0 Channel 2 registers are accessible in the 0x12xx address range (default) 1 Channel 2 registers are accessible in the 0x02xx address range
18	RCA3	Remap Channel Address 3. Writing a 1 to this bit remaps all Channel 3 host accessible registers to an alternate 4-Kbyte address range; in addition, Channel 3's Done and Error bits in the Interrupt Status Register causes the secondary interrupt to assert instead of the primary interrupt. 0 Channel 3 registers are accessible in the 0x13xx address range (default) 1 Channel 3 registers are accessible in the 0x03xx address range
19	RCA4	Remap Channel Address 4. Writing a 1 to this bit remaps all Channel 4 host accessible registers to an alternate 4-Kbyte address range; in addition, Channel 4's Done and Error bits in the Interrupt Status Register causes the secondary interrupt to assert instead of the primary interrupt. 0 Channel 4 registers are accessible in the 0x14xx address range (default) 1 Channel 4 registers are accessible in the 0x04xx address range
20-21	—	Reserved

Table 11-21. Master Control Register Signals (continued)

Bits	Name	Description
22-23	Priority	Priority on Master Bus. The setting of these bits determines the transaction priority level the SEC asserts to the platform internal arbiter. The SEC does not dynamically alter its priority level based on system congestion or SEC utilization, however software may change the SEC priority level in realtime. 00 Lowest Priority (default) 01 Next Lowest Priority 10 Next Highest Priority 11 Highest Priority
24-29	—	Reserved
30	GIH	Global Inhibit. Writing 1 to this bit indicates that external master bus transfers are defined as not snoopable and results in lowering the snoop attribute of bus requests generated by the external gasket. 0 External master bus transfers are defined as snoopable (default) 1 External master bus transfers are defined as not snoopable
31	SWR	Software Reset. Writing 1 to this bit causes a global software reset. Upon completion of the reset, this bit is automatically cleared. 0 Do not reset 1 Global Reset
32–39	CHN3_EU_PR_CNT	Channel 3 EU Priority Counter. This counter is used by the controller to determine when Channel 3 has been denied access to a requested EU long enough to warrant immediate priority elevation. **Note:** If CHN3_EU_PR_CTR is zero and CHN4_EU_PR_CTR is zero, the controller assigns EUs on a pure round robin basis. If either of these counters is zero and the other is non-zero, then the zero is interpreted as 256.
40–47	CHN4_EU_PR_CNT	Channel 4 EU Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to a requested EU long enough to warrant immediate priority elevation. **Note:** If CHN3_EU_PR_CTR is zero and CHN4_EU_PR_CTR is zero, the controller assigns EUs on a pure round robin basis. If either of these counters is zero and the other is non-zero, then the zero is interpreted as 256.
48–55	CHN3_BUS_PR_CNT	Channel 3 Bus Priority Counter.This counter is used by the controller to determine when Channel 3 has been denied access to the polychannel long enough to warrant immediate priority elevation. **Note:** If CHN3_BUS_PR_CTR is zero and CHN4_BUS_PR_CTR is zero, the controller assigns EUs on a pure round robin basis. If either of these counters is zero and the other is non-zero, then the zero is interpreted as 256.
56–63	CHN4_BUS_PR_CNT	Channel 4 Bus Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to the polychannel long enough to warrant immediate priority elevation. **Note:** If CHN3_BUS_PR_CTR is zero and CHN4_BUS_PR_CTR is zero, the controller l assigns EUs on a pure round robin basis. If either of these counters is zero and the other is non-zero, then the zero is interpreted as 256.

11.6.4 Snooping by Caches

All SEC memory transactions are snooped by the device’s coherency module, which is part of the wiring of the SEC interface and requires no user intervention.

11.7 Execution Units

Execution unit (EU) is the term used for a functional block that performs the mathematical manipulations required by protocols used in cryptographic processing. The following execution units are used in the SEC (covered here in alphabetical order):

- Advanced Encryption Standard Execution Unit (AESU) implementing the Rijndael symmetric key cipher
- ARC Four Execution Unit (AFEU)
- Cyclical Redundancy Check Accelerator (CRCU)
- Data Encryption Standard Execution Unit (DEU)
- Kasumi (f8/f9) Execution Unit (KEU)
- Message Digest Execution Unit (MDEU)
- Public Key Execution Unit (PKEU)
- Random Number Generator (RNGU)

Working together, the EUs can perform high-level cryptographic tasks, such as IPsec Encapsulating Security Protocol (ESP) and digital signature. The remainder of this chapter provides details about the execution units themselves.

11.7.1 Advanced Encryption Standard Execution Unit (AESU)

This section contains details about the Advanced Encryption Standard Execution Unit (AESU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the ASEU is used through channel-controlled access, which means that most reads and writes of ASEU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

For CCM, GCM, CMAC (OMAC1), and XCBC-MAC cipher modes, this EU includes an ICV checking feature, that is, it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the host either via interrupt or by a writeback of EU status fields into host memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see Section 11.5.4.1, “Channel Configuration Register (CCR)”), and mask the ICE bit in the Interrupt Mask Register (Section 11.7.1.7, “AESU Interrupt Mask Register”). In this case the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an error interrupt to the host, but no channel done interrupt” or writeback.

11.7.1.1 AESU Mode Register

The AESU mode register, shown in Figure 11-25, contains 11 bits that are used to program the AESU. The mode register is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

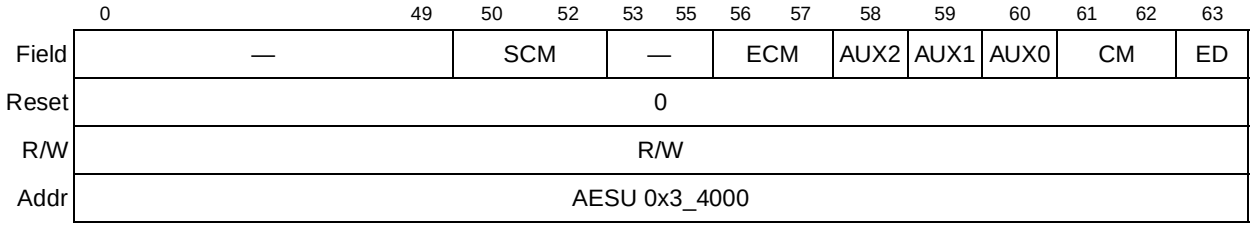


Figure 11-25. AESU Mode Register

Table 11-22 describes AESU mode register fields.

Table 11-22. AESU Mode Register Field Descriptions

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–49	—	Reserved
50–52	SCM	Sub-Cipher-Mode: Specifies additional options specific to particular cipher modes. • XOR cipher mode: specifies the number of sources to be XORed together. Valid values are 2-6. For all other cipher modes, this field must be 0.
53–55	—	Reserved, must be cleared.
The following bits are controlled through the MODE0 field of the descriptor header.		
56–57	ECM	Extended Cipher Mode: Used in combination with bits 61:62 (Cipher Mode) to select the cipher mode for AES operation. See Table 11-23 on page 11-66 for mode bit combinations.

Table 11-22. AESU Mode Register Field Descriptions (continued)

Bits	Name	Description
58	AUX2	AUX2 Mode has a definition that depends upon the value of the 4 Cipher Mode and Extended Cipher Mode bits:
	AUX2 = Finalize MAC for CCM and GCM modes	CCM and GCM Cipher Modes (ECM=1X, CM=0X): Generate Final MAC Bit - Processes final message block and generates final MAC tag at the end of message processing. 0 = Do not generate final MAC tag 1 = Generate final MAC tag after CCM/GCM processing is complete. Note that for GCM, when message processing is split into multiple descriptors, it must be AUX1=1 when AUX2=1.
	AUX2 = LRW Tweak Control	LRW Cipher Mode (ECM=10, CM=00): Context Switched Bit - Enables passing initial tweak value (T) to the cipher and skips its computation, thus allowing proper continuation of the interrupted processing after context switch by restoring the saved content of the Context Registers 1-2 (T - tweak) and 3-4 (I - Index). 1 = Skip computing tweak for the first data block and use the one provided in Context Registers 1-2. Also, use the index provided in Context Registers 3-4 for the computation of the next tweak. 0 = Ignore content of Context Registers 1-2 and compute the initial tweak using index provided in Context Registers 3-4
	AUX2 = ICV Bit	XCBC-MAC and CMAC Cipher Modes (ECM=10, 01, CM=10): ICV Bit - Enables XCBC-MAC w/ICV and CMAC w/ICV Cipher Modes 0 = XCBC-MAC or CMAC cipher mode 1 = XCBC-MAC w/ICV or CMAC w/ICV cipher mode
	AUX2 = Enable RBP	CBC, CBCRBP Cipher Modes (ECM=00, CM=01): RBP Bit - Enables CBC-RBP 0 = CBC cipher mode 1 = CBC-RBP cipher mode

Table 11-22. AESU Mode Register Field Descriptions (continued)

Bits	Name	Description
59	AUX1	AUX1 Mode has a definition that depends upon the value of the 4 Cipher Mode and Extended Cipher Mode bits:
	AUX1 = Initialize CCM	CCM Cipher Mode (ECM=10, CM=00): Initialize Mode Bit - Initializes AESU for new message 0 = Do not initialize (context is loaded by host) 1 = Initialize new message with nonce/initialization vector
	AUX1 = Generate Final GHASH	GCM Cipher Mode (ECM=10, CM=01): Generate Final GHASH Bit - Enables completion of GHASH computation by signaling that the last iteration of GHASH should be performed. This last iteration performs XOR of the current (intermediate) GHASH result with the concatenation of AAD and ciphertext bit lengths in case of GHASH(H, AAD, ciphertext), or with the concatenation of 0^{64} and the bit length of IV in case of GHASH(H, {}, IV). As an exception, this bit should be cleared if the whole message (IV+AAD+textdata) together with the generation of the final MAC is processed with one descriptor since in that case the generation of final GHASH is implied. Incidentally, whenever AUX1=1 in GCM cipher mode, the total bit lengths of AAD, textdata or IV must be provided in context registers 9–10. 0 = Do not perform the last iteration in GHASH(H, AAD, ciphertext) or GHASH(H, {}, IV) unless the message is processed and the final MAC computed in 1 descriptor. 1 = Generate the final result of GHASH(H, AAD, ciphertext) or GHASH(H, {}, IV) - implies that the message processing is split into multiple descriptors.
	AUX1 = Use Context for XCBC-MAC derived keys	XCBC-MAC Cipher Mode (ECM=10, CM=10): Load Keys - Do not compute K1, K2 and K3, but instead use the keys loaded in the Key Data Registers (K1), and Context Registers 5-6 (K2) and 7-8 (K3). 0 = Compute K1=E(K, 16{01}), K2=E(K, 16{02}), K3=E(K, {03}) and write K1 to Context Registers 3-4, K2 to 5-6, and K3 to 7-8. 1 = Load keys: K1= [Key Data Reg 1-2], K2= [Reg 5-6], K3=[Reg 7-8]
	AUX1 = Use Context for CMAC derived keys	CMAC Cipher Mode (ECM=01, CM=10): Load Keys - Do not compute E(K, 0^{128}) to derive K1 and K2, but instead use the value loaded in Context Registers 3-4. This is useful after a context switch. Deriving K1 and K2 does not incur any timing penalty. 0 = Compute E(K, 0^{128}) and write it to Context Registers 3-4 1 = Load E(K, 0^{128}) and preserve it in Context Registers 3-4
60	AUX0	AUX0 Mode has a definition that depends upon the value of the 4 Cipher Mode and Extended Cipher Mode bits, and Encrypt/Decrypt bit:
	AUX0 = Restore Decrypt Key	ECB, CBC, OFB, CFB Cipher Modes ((ECM=00, CM=XX) or (ECM=10, CM=11)) and Decrypt (ED=0): Restore Decrypt Key Bit - Specifies that key data write contains pre-expanded key (decrypt mode only). See note below on use of RDK bit. 0 = Expand user key prior to decrypting first block 1 = Do not expand key. Expanded decryption key is written following context switch.
	AUX0 = GCM GHASH Only	GCM Cipher Mode (ECM=10, CM=01) and Encrypt (ED=1): Specifies GHASH mode—performs GHASH on AAD and ciphertext 0 = Perform GCM encryption 1 = Compute GHASH(H, AAD, ciphertext)
	AUX0 = Finalize Mac for XCBC-MAC and CMAC modes	XCBC-MAC, CMAC Cipher Modes (ECM=10, 01, CM=10): Do Not Generate Final MAC Bit - Does not generate final MAC tag at the end of message processing (used only when splitting a message into multiple descriptors) 0 = Generate final MAC tag such as XOR the final data block with K2/K3 (XCBC-MAC), or K1/K2 (CMAC) before encryption 1 = Do not generate final MAC tag, for example, does not XOR final data block with K2 (XCBC-MAC) or K1 (CMAC) before encryption. This enables message processing to be interrupted on the block boundary and later continued after a context switch

Table 11-22. AESU Mode Register Field Descriptions (continued)

Bits	Name	Description
61–62	CM	Cipher Mode: Used in combination with bits 56:57 (Extended Cipher Mode) to select the cipher mode for AES operation. See Table 11-23 on page 11-66 for mode bit combinations.
63	ED	Encrypt/Decrypt. If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption Note: This bit is ignored in CTR, SRT, CMAC, and XCBC-MAC Cipher Modes.

Table 11-23. AES Cipher Modes

Cipher Mode	ECM (56:57)	AUX2 (58)	CM (61:62)
ECB	00	X	00
CBC	00	X	01
CBC-RBP	00	1	01
OFB	00	X	10
CTR	00	X	11
CMAC	01	X	10
CMAC w/ICV	01	1	10
SRT[1]	01	X	11
CCM	10	X	00
GCM	10	X	01
XCBC-MAC	10	0	10
XCBC-MAC w/ICV	10	1	10
CFB128	10	X	11
CCM w/ICV	11	X	00
GCM w/ICV	11	X	01
XOR	11	X	11
Reserved	all others		

1 SRT is not a new AES cipher mode, it is an AESU method of performing AES-CTR mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0 'srtp'. See Section 11.7.1.11.3, "Context for SRT Cipher Mode," for more information on how SRT cipher mode reduces context loading overhead.

NOTE

On Restore decrypt key (RDK)—In most networking applications, the decryption of an AES protected packet is performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.)

The use of RDK is completely optional, as the Input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is “0100_0- AESU Key Expand Output”. The AESU mode must be “Decrypt”. See Table 11-7 for more information. The descriptor causes the SEC to write the contents of the Context registers and the key registers (containing the expanded decrypt key) to memory.

To process the remainder of the message, use a “common” descriptor type (0001_0), and set the “restore decrypt key” mode bit. Load the context registers and the expanded decrypt key with previously saved key and context data from the first message. The key size is written as before (16, 24, or 32 bytes).

11.7.1.2 AESU Key Size Register

The AESU key size register, shown in Figure 11-26, is used to specify the number of bytes in the key (16, 24, or 32). Any key data beyond the number of bytes in the key size register is ignored. This register is cleared when the AESU is reset or re-initialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error is generated. The same occurs if XCBC-MAC cipher mode is specified and a key size other than 16 is specified. If the key size register is modified during processing, a context error is generated.

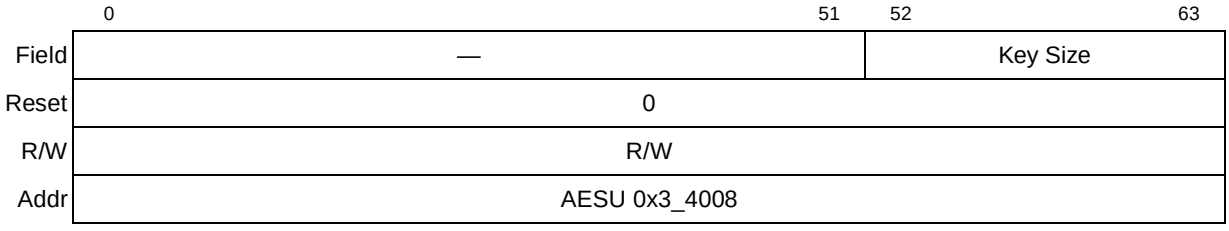


Figure 11-26. AESU Key Size Register

11.7.1.3 AESU Data Size Register

The AESU data size register, shown in Figure 11-27, is used to specify the number of bits (not bytes) of plaintext/ciphertext to be processed in the current descriptor. The number of data size register bits used by the SEC, and the acceptable values for these bits, vary depending on the AES cipher mode selected (see Table 11-24).

Table 11-24. Use of Data Size Register

AESU Cipher Mode	Register Bits Used by SEC (Others are Don't Cares)	Legal Values (Data Size in Bits)
ECB, CBC, LRW	Lowest 7 bits [57:63]	Must be a multiple of 128
OFB, LRW, CMAC, SRT, CCM, XCBC-MAC, CFB128		Must be a multiple of 8
GCM	All bits	Any value
XOR	All bits	Must be a multiple of 256

This register is cleared when the AESU is reset or re-initialized.

Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated; see Figure 11-27.

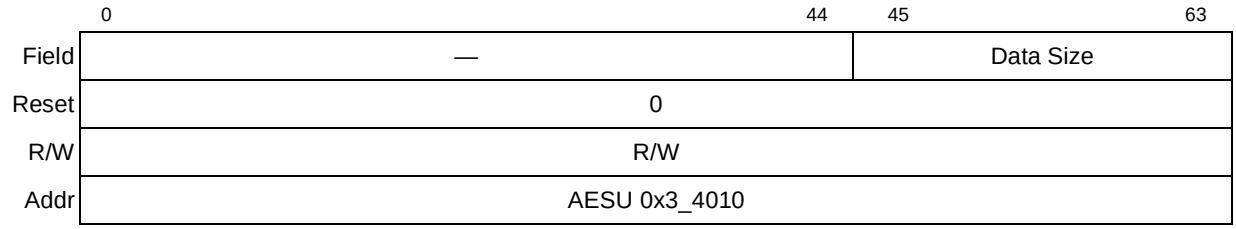


Figure 11-27. AESU Data Size Register

11.7.1.4 AESU Reset Control Register

This register allows three levels reset of just AESU, as defined by the three self-clearing bits in Figure 11-28.

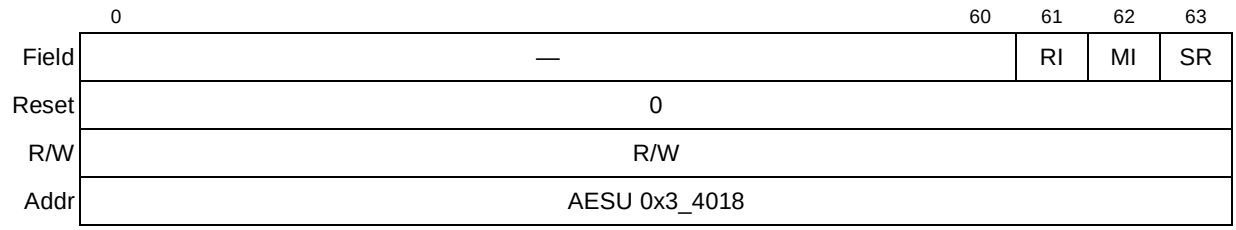


Figure 11-28. AESU Reset Control Register

Table 11-25 describes AESU reset control register fields.

Table 11-25. AESU Reset Control Register Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes AESU interrupts signalling done and error to be reset. It further resets the state of the AESU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt mask register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register. 0 Do not reset 1 Reset most of AESU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only resets AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU enters a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register indicates when this initialization routine is complete. 0 Do not reset 1 Full AESU reset

11.7.1.5 AESU Status Register

AESU status register is a read-only register that reflects the state of six status outputs. Writing to this location results in an address error being reflected in the AESU interrupt status register; see Figure 11-29.

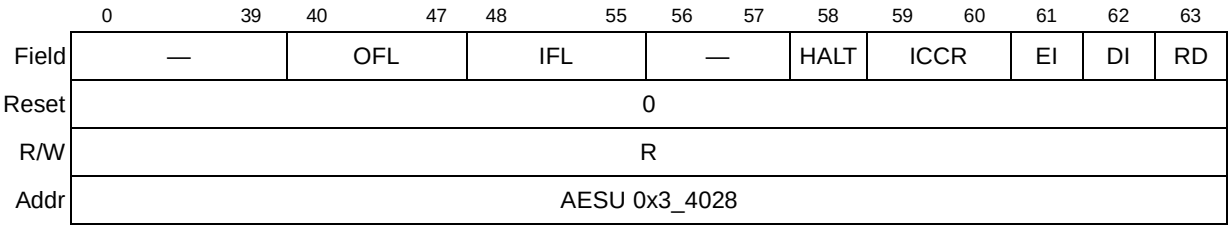


Figure 11-29. AESU Status Register

Table 11-26 describes AESU status register fields.

Table 11-26. AESU Status Register Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the AESU has halted due to an error. 0 AESU not halted 1 AESU halted **Note:** Because the error causing the AESU to stop operating may be masked before reaching the interrupt status register, the AESU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity Check Comparison Result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved A “passed” or “failed” result is generated only if the cipher mode with ICV checking is selected
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 AESU is not signaling error 1 AESU is signaling error

Table 11-26. AESU Status Register Field Descriptions (continued)

Bits	Name	Description
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 AESU is not signaling done 1 AESU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that AESU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.1.6 AESU Interrupt Status Register

The interrupt status register, shown in Figure 11-30, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AESU interrupt mask register is zero (see Section 11.7.1.7, “AESU Interrupt Mask Register”).

If the AESU interrupt status register is non-zero, the AESU halts and the AESU error interrupt signal is asserted to the controller (see Section 11.6.3.3, “Interrupt Status Register (ISR)”). In addition, if the AESU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the Channel Status Register (see Table 11-15) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the host, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the AESU Reset Control Register.

	0–48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	0															
R/W	R/W															
Addr	AESU 0x3_4030															

Figure 11-30. AESU Interrupt Status Register

Table 11-27 describes AESU interrupt status register fields.

Table 11-27. AESU Interrupt Status Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check Error: 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the ASEU.

Table 11-27. AESU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while the AESU was processing. 0 No error detected 1 Internal error **Note:** This bit is asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Mask Register or by resetting the AESU.
52	ERE	Early Read Error. An AESU Context Register was read while the AESU was processing. 0 No error detected 1 Early read error
53	CE	Context Error. An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while AESU was processing 0 No error detected 1 Context error
54	KSE	Key Size Error. An inappropriate value (not 16, 24 or 32bytes) was written to the AESU Key Size Register 0 No error detected 1 Key size error
55	DSE	Data Size Error (DSE): A value was written to the AESU Data Size Register that is not a proper size. See Section 11.7.1.3, “AESU Data Size Register,” on page 11-68. 0 No error detected 1 Data size error
56	ME	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Valid Data 1 Reserved or invalid mode selected
57	AE	Address Error. An illegal read or write address was detected within the AESU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO Error. The AESU output FIFO was detected non-empty upon write of AESU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO Error. The AESU input FIFO was detected non-empty upon generation of done interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO was pushed while full. 0 No error detected 1 Input FIFO has overflowed **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the AESU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO Underflow. The AESU Output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

11.7.1.7 AESU Interrupt Mask Register

The AESU interrupt mask register, shown in Figure 11-31, controls the result of detected errors. For a given error (as defined in Section 11.7.1.6, "AESU Interrupt Status Register"), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0–48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	1000															
R/W	R/W															
Addr	AESU 0x3_4038															

Figure 11-31. AESU Interrupt Mask Register

Table 11-28 describes the AESU interrupt mask register fields.

Table 11-28. AESU Interrupt Mask Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check Error. The supplied ICV did not match the one computed by the ASEU. 0 Integrity check error enabled. WARNING: Do not enable this if using EU status writeback (see bits IWSE and AWSE in Section 11.5.4.1, "Channel Configuration Register (CCR)"). 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while the AESU was processing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The AESU IV Register was read while the AESU was processing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while the AESU was processing. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Mode error enabled 1 Mode error disabled

Table 11-28. AESU Interrupt Mask Register Field Descriptions (continued)

Bits	Name	Description
57	AE	Address Error. An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
58	OFE	Output FIFO Error. The AESU Output FIFO was detected non-empty upon write of AESU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO Error. The AESU Input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO Overflow. The AESU Input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO Underflow The AESU Output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

11.7.1.8 ICV Size Register

The ICV size register, shown in Figure 11-32, is used in GCM cipher mode to specify the number of most significant bytes in the received MAC tag supplied in Context Registers 11–12 (GCM w/ICV). GCM truncates the computed MAC in Context Registers 3–4 to the same number of most significant bytes and write zeros in the remaining positions. Note that the received MAC can be padded to the full 16 bytes with any value, such as, not necessarily zeros when written into Context Registers 11–12. Acceptable values for ICV size are 8, 12, and 16 bytes. All other sizes are interpreted as 16.

A similar concept applies to CMAC cipher mode. The difference is that the received MAC must be provided in Context Registers 5-6 (CMAC w/ICV) and that the computed MAC is available in Registers 1–2. Acceptable values for ICV size are 8, 10, 12, 14 and 16 bytes. All other sizes are interpreted as 16.

ICV size register is not used in case of XCBC-MAC cipher mode. Even though full 16-bytes of the received MAC can be written to Context Registers 9–10 for comparison with the computed MAC, only most significant 12-bytes are considered as defined in the XCBC-MAC-96 for IPsec specification. The computed MAC written at the end of processing to Context Registers 1–2 are a full 16-byte MAC.

CCM w/ICV does not use the ICV Size Register.

11.7.1.9 AESU ICV Size Register

	0 — 56	57 — 63
Field	—	ICV Size
Reset	0	
R/W	R/W	
Addr	AESU 0x3_4040	

Figure 11-32. AESU ICV Size Register

11.7.1.10 AESU End_of_Message Register

The AESU end_of_message register, shown in Figure 11-33, is used to indicate an AES operation may be completed. After the final message block is written to the input FIFO, the end_of_message register must be written. The value in the data size register is used to determine how many bits of the final message block (always 128) is processed. Writing to this register causes the AESU to process the final block of a message, allowing it to signal done interrupt. A read of this register always returns a zero value.

	0 — 63
Field	AESU End of Message
Reset	0
R/W	W
Addr	AESU 0x3_4050

Figure 11-33. AESU End_of_Message Register

11.7.1.11 AESU Context Registers

There are twelve 64-bit context data registers that allow the host to read/write the contents of the context used to process a message. The context must be written prior to the key data. If the context registers are written during message processing, a context error is generated. All context registers are cleared when an initialization or a hard or soft reset is performed.

If a message is processed through the AESU in two separate operations (for example, using two descriptors), then the context must be read out of the SEC at the end of the first operation and then restored at the beginning of the second operation. Context is always read and restored as a contiguous subset of the twelve context registers ending with the highest numbered register used in that cipher mode, for example, if restoring context in CTR cipher mode, seven context registers (1–7) must be written where registers 1–4 must be filled with zeros.

The context registers are summarized in Table 11-29, Table 11-30, and Table 11-31.

Table 11-29. AESU Context Registers for Confidentiality Modes

Context Register (byte address)	Cipher Mode Providing Only Confidentiality				
	ECB	CBC / CBC-RBP / OFB / CFB128	CTR	LRW	SRT
1 (0x34100)	—	IV*	—	I	Counter*
2 (0x34108)					
3 (0x34110)	—	—	—	Key 2	Counter Modulus*
4 (0x34118)					—
5 (0x34120)	—	—	Counter*	Tweak	—
6 (0x34128)					
7 (0x34130)	—	—	Counter Modulus Exponent*	—	——

Notes:
Context Registers 8 through 12 unused for these modes
* Must be written at start of new message, except if zero
— don't care

11.7.1.11.1 Context for CBC, CBC-RBP, OFB, and CFB128 Cipher Modes

For use in CBC, CBC-RBP, OFB, and CFB128 cipher modes, there are two 64-bit context data registers within the Context registers that allow the host to read/write the contents of the initialization vector (IV), as follows:

- Context Register 1 holds the least significant bytes of the initialization vector (bytes 1–8).
- Context Register 2 holds the most significant bytes of the initialization vector (bytes 9–16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the mode is not set, a context error is generated.

The IV registers may only be read after processing has completed, as indicated by the assertion of DI (done interrupt) in the AESU status register as shown in Section 11.7.1.5, "AESU Status Register." If the IV registers are read prior to assertion of Interrupt Done, an early read error is generated.

11.7.1.11.2 Context for Counter Cipher Mode

In counter cipher mode, a random 128-bit initial counter value is incremented modulo 2^M with each block processed. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8. The value of the M is specified by writing to Context Register 7 as described in Table 11-29.

11.7.1.11.3 Context for SRT Cipher Mode

As was noted in the AESU Mode Register, SRT is not a new AES cipher mode; it is an AESU method of performing AES-CTR cipher mode with reduced context loading overhead specifically for performing SRTP. It should be used with descriptor type 0010_0 “srtp.” As with counter cipher mode, a random 128-bit initial counter value is incremented modulo 2^M with each block processed. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext. The modulus exponent M can be set between 8 and 128 in multiples of 8. The value of M is specified by writing to Context Register 3 as described in Table 11-29.

The only difference between SRT and CTR cipher modes is that in SRT mode, the AES Context is loaded and read via Context Registers 1-3, with no requirement to access Context Registers 4–7. In CTR mode, Context Registers 1–4 must be loaded with zeros, with the Counter and Modulus being loaded into and read from Context Registers 5–7.

Table 11-30. AESU Context Registers for Integrity Modes

Context Register # (Byte Address)	Cipher Mode Providing Only Data Integrity		
	CMAC	GCM–GHASH	XCBC–MAC
1 (0x34100)	Computed MAC	Computed MAC	Computed MAC
2 (0x34108)			
3 (0x34110)	Received MAC*	—	Received MAC*
4 (0x34118)			
5 (0x34120)	Key 1	—	$E(K, 0^{128})$
6 (0x34128)			
7 (0x34130)	Key 2	$len(AAD)^T$	—
8 (0x34138)		—	
9 (0x34140)	Key 3	H	—
10 (0x34148)			
11 (0x34150)	—	$len(AAD)^C$	—

Notes:

Context register 12 is unused for these modes

* Must be written at start of new message for ICV checking

^{C} length of data processed with current descriptor (in bits)

^{T} length of total data (in bits)

^{ICV} Needed only in ICV mode

— don’t care

11.7.1.11.4 Context and Operation for XCBC-MAC Cipher Mode

XCBC-MAC cipher mode is an authentication only mode of AES. Normal CBC-MAC runs AES in CBC cipher mode and assigns the final ciphertext result as the MAC. XCBC-MAC supports only 16-byte keys and extends the normal CBC-MAC as follows:

1. 3 keys are precomputed
   a) K1 = AES-Encrypt (K, $\{01\}^{16}$). [1]
   b) K2 = AES-Encrypt (K, $\{02\}^{16}$).
   c) K3 = AES-Encrypt (K, $\{03\}^{16}$).
2. Compute C_{n-1} = AES-CBC (P_1, 0, K1) ... AES-CBC (P_{n-1}, C_{n-2}, K1)
3. If $|P_n|$ = block size (128 bits)

then: MAC = AES-CBC ($P_n \oplus K2$, C_{n-1}, K1)

else: MAC = AES-CBC ($(P_n || 10^i) \oplus K3$, C_{n-1}, K1)

In XCBC-MAC cipher mode, AUX0=1 means that the final data block is not XORed with K2 or K3, so that message processing can be interrupted and later continued after a context switch. AUX1=1 disables computation of keys K1, K2, and K3, and instead expects these keys to be placed in Key Data Registers 5–6 (K1), Context Registers 7-8 (K2) and 9–10 (K3). If AUX1=0, computed keys are placed into Context Registers 5–10. AUX2=1 enables XCBC-MAC w/ICV. In XCBC-MAC w/ICV, the transmitted MAC is supplied in Context Registers 3–4 and compared to the computed MAC in Context Registers 1–2.

For host-controlled operation of the AESU in XCBC-MAC cipher mode, the following steps must be performed:

1. Reset
2. Write the Mode Register to:
   a) Set Cipher Mode to XCBC-MAC (enc/dec bit is ignored)
   b) Set AUX0 = 1 if processing of the message is going to be interrupted and later continued after a context switch. Set AUX0 = 0 if this is the last (or only) part of the message so that the final MAC can be generated.
   c) Set AUX1 = 1 if keys K1, K2 and K3 are loaded to Key Data Registers 5–6, Context Registers 7–8 and 9–10, respectively. Otherwise, set AUX1=0, and put K into Key Data Registers 1–2, so that keys K1, K2, and K3 can be computed and written to Context Registers 5–6, 7–8, and 9–10.
   d) Set AUX2 = 1 if desire XCBC-MAC w/ICV.
3. Load Key (K or K1—see above)
4. Load Context
5. Set Key size
6. Set Data size
7. While available:
   a) Load data blocks

1.Notation: $\{01\}^{16}$ means the byte 01 repeated 16 times

8. Write to the End of Message register
9. Read MAC from Context Registers 1–2
10. For XCBC-MAC w/ICV, check ICCR bits in the Status Register

11.7.1.11.5 Context and Operation for CMAC (OMAC1) Cipher Mode

CMAC cipher mode is an authentication only mode of AES. The following is a specification of CMAC:

1. 2 keys are precomputed
   a. K1 = xtime(AES-Encrypt (K, $\{0\}^{128}$)). [1]
   b. K2 = xtime(K1).
2. Compute C_{n-1} = AES-CBC (P_1, 0, K) ... AES-CBC (P_{n-1}, C_{n-2}, K)
3. If $|P_n|$ = block size (128 bits)
   then: MAC = AES-CBC ($P_n \oplus K1$, C_{n-1}, K)
   else: MAC = AES-CBC ($(P_n || 10^i) \oplus K2$, C_{n-1}, K)

Assuming L is a 128-bit vector, L[127] is its most significant bit, and << is a bit-wise shift, xtime(L) is defined as:

If L[127] = 0, then xtime(L) = L << 1;
Else xtime(L) = (L << 1) XOR 0x87;

CMAC cipher mode requires transmitted MAC to be placed in Context Registers 3–4, whereas computed MAC is in Registers 1–2 if AUX0=1. Context registers 5–6 are used to provide E(K, $\{0\}^{128}$) if AUX1=1 so that K1 and K2 can be computed after context switch without a time penalty. Computed value of E(K, $\{0\}^{128}$) always is stored in Context Registers 5–6 to be available for saving context in case of context switching.

For host-controlled operation of the AESU in CMAC cipher mode, the following steps must be performed:

1. Reset
2. Write the Mode Register to:
   a) Set Cipher Mode to CMAC (enc/dec bit is ignored)
   b) Set AUX0 = 1 if processing of the message is going to be interrupted and later continued after a context switch. Set AUX0 = 0 if this is the last (or only) part of the message so that the final MAC can be generated.
   c) Set AUX1 = 1 for keys K1 and K2 to be derived from E(K, $\{0\}^{128}$) that is loaded into Context Registers 3-4. Otherwise, set AUX1=0 and CMAC computes E(K, $\{0\}^{128}$).
   d) Set AUX2 = 1 if desire CMAC w/ICV.
3. Load Key
4. Load Context
5. Set Key size

1.Notation: $\{0\}^{128}$ means the bit 0 repeated 128 times

6. Set ICV Size for computed/received MAC (8, 10, 12, 14 or 16 bytes, default is 16) - ignored if AUX0 = 1
7. Set Data size
8. While available:
   a) Load data blocks
9. Write to the End of Message register
10. Read MAC from Context Registers 1-2
11. For CMAC w/ICV, check ICCR bits in the Status Register

Table 11-31. AESU Context Registers for Modes Providing Confidentiality and Integrity

Context Register # (Byte Address)	Cipher Mode Providing Confidentiality and Integrity	
	CCM	GCM
1 (0x34100)	IV* / MAC	Computed MAC
2 (0x34108)		
3 (0x34110)	Encrypted MAC** / Decrypted MAC / Encrypted Counter	Received MAC
4 (0x34118)		
5 (0x34120)	Counter*	Counter
6 (0x34128)		
7 (0x34130)	Counter Modulus Exponent* (header size/ MAC size)***	$len(AAD)^T$
8 (0x34138)	—	$len(IV)^T$
9 (0x34140)	—	Y_0
10 (0x34148)		
11 (0x34150)	—	$len(AAD)^C$
12 (0x34158)		$len(IV)^C$

* Must be written at start of new message, except if zero

** Must be written at start of new CCM decryption

*** The Header and MAC Sizes are internally constructed by the AES engine; then, that information is included inside Context Register 7 for context switching purposes

^{C} length of data processed with current descriptor (in bits)

^{T} length of total data (in bits)

^{ICV} Needed only in ICV mode

— don't care

11.7.1.11.6 Context for CCM Cipher Mode

The SEC AESU is capable of performing single pass encryption and MAC generation. The host is required to order the CCM context in such a way that the context can be fetched as a contiguous string into the context registers, prior to encryption/MAC generation or decryption/MAC validation. The context register contents for CCM cipher mode is summarized in Figure 11-34 and further described below.

		Context Registers						
		1	2	3	4	5	6	7
Encrypt (outbound)	Inputs	IV		0		Initial Counter		Counter Modulus Exponent
	Outputs	MAC	0	MIC	0	—		—
Decrypt (inbound)	Inputs	IV		MIC	0	Initial Counter		Counter Modulus Exponent
	Outputs	Computed MAC	0	Decrypted MAC	0	—		—

Figure 11-34. AESU CCM Context Registers

Context for CCM encryption/MAC generation:

Reg 1–2

Session specific 128-bit initialization vector (from memory)

Reg 3–4

128 bits of zero padding

Reg 5–6

Session specific counter (initial counter value) (from memory)

Reg 7

Counter modulus exponent (msb<--lsb) should be fixed at 0x0000_0080

NOTE

The counter modulus for CCM cipher mode is currently defined as 2^{128} making the exponent 128. This value has been made programmable in the SEC in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU Context Register prior to CCM encryption.

CCM encryption processing steps are as follows:

With the session specific key and context, the AESU performs the following operations:

1. Initialize the IV, and encrypt with the symmetric key.
2. In CBC fashion, take the output of step 1, hash with the first block of plaintext, and encrypt with the symmetric key.
3. Continue as in step 2 until the final block of plaintext has been processed. The result of the encryption of the final block of plaintext with the symmetric key is the MAC Tag. The full 128 bits of MAC data is written to Context Registers 1–2, for use in the next phase of CCM processing.

4. Once the MAC Tag has been generated (step 3), the MAC tag, along with the plaintext is encrypted with the AESU operating in Counter cipher mode.
5. The first item to be encrypted in Counter cipher mode is the Counter (Initial Counter Value) from Context Registers 5–6. The counter is encrypted with the symmetric key, and the result is hashed with the MAC Tag (retrieved from Context Reg 1–2) to produce the MIC (encrypted MAC), which is then stored in Context Registers 3–4. At the completion of CCM encrypt processing, this MIC is output to memory (per the descriptor pointer) for the host to append to the 802.11i frame.

   Note: The MIC written out to memory by the AESU is the full 128 bits. The host must only append the most significant 64 bits to the frame as the MIC.
6. The counter value is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of plaintext to produce the first block of cipher text. The ciphertext is placed in the AESU output FIFO.
7. The counter continues to be incremented, and encrypted with the symmetric key, with the result hashed with each successive block of plaintext, until all plaintext has been converted to ciphertext. The SEC controller manages FIFO reads and writes, fetching plaintext and writing ciphertext per the pointers provided in the descriptor. When all ciphertext and the MIC has been output, the CCM encrypt operation is complete.

Context for CCM decryption/MAC generation are as follows:

Reg 1–2

Session specific128-bit Initialization Vector (from memory)

Reg 3–4

MIC (from received frame) + 64 bits of zero padding

Reg 5–6

Session Specific Counter (Initial Counter Value) (from memory)

Reg 7

Counter Modulus Exponent (msb<--lsb) Should be fixed at 0x0000_0080.

NOTE

The counter modulus for CCM cipher mode is currently defined as 2^128, making the exponent 128. This value has been made programmable in the SEC to in case the final version of 802.11i uses a different counter modulus. Because this is a programmable field, it must be generated and stored along with other session specific information for loading into the AESU Context Register prior to CCM decryption.

CCM decryption processing steps are as follows:

With the session specific key and context, the AESU performs the following operations:

1. Initialize the IV, and encrypt with the symmetric key. Simultaneously, the counter (Initial Counter Value) from Context Registers 5–6 is encrypted with the symmetric key. The result is hashed with the Encrypted MAC (from Context Register 3–4), and the resulting Original MAC is written to Context Reg 3–4, overwriting the Encrypted MAC.

NOTE

Strictly speaking, the Counter is encrypted with the symmetric key, however the AESU should be set for "decrypt" to perform the counter and CBC processes in the correct order.

2. The IEEE 802.11 frame header is hashed with the encrypted IV. (The AESU automatically determines the header length.) Simultaneously, the counter is incremented, and is then encrypted with the symmetric key. The result is then hashed with the first block of ciphertext to produce the first block of plaintext. The plaintext is placed in the AESU output FIFO, while simultaneously, in CBC fashion, a copy of the first block of plaintext is hashed with the output of encryption of the IEEE 802.11 frame header. The output is encrypted with the symmetric key.
3. As each ciphertext block is converted to plaintext, the plaintext is CBC encrypted. When the final plaintext block has been processed, the CBC MAC (MAC Tag) is written to Context Registers 1–2. The first 64 bits of the MAC Tag are compared to the MAC Tag recovered in step 1.

NOTE

For both encrypt and decrypt operations, if the 802.11 frame is being processed as a whole (not split across multiple descriptors), the "Initialize" (AUX1) and "Final MAC" (AUX2) bits should be set in the AESU Mode Register.

11.7.1.11.7 Context and Operation for GCM Cipher Mode

Galois Counter Mode (GCM) uses AES Counter mode to achieve data confidentiality. Authentication is achieved by computing a GHASH Message Authentication Code (GMAC) through performing repetitive multiplication-accumulate functions in a Galois Field.

Normally, the IV (which is provided through the input FIFO) is 96 bits. If it is 96 bits, then the IV is padded with the value $(\{0\}^{31}1)$[1]. Else, the IV is hashed using the GHASH (H, {}, IV) function, where H is defined as $E(\{0\}^{128}, K)$, E stands for encryption operation, and K represents the key used. The resulting value Y_0 (the padded IV or the GHASHed IV) is provided as the counter input to Counter Mode AES. The result of encrypting Y_0 is called $E(Y_0, K)$, and is used to generate the final MAC tag.

Data is encrypted or decrypted by XORing input data with the Pseudorandom Key Stream generated by Counter Mode AES, starting with the *second* PRK block. Initial counter value Y_0 is incremented modulo 2^{32}.

GCM cipher mode can be used to perform only the authentication part (GHASH (H, AAD, ciphertext)). To do this, set AUX0 and specify encryption operation. This special sub-mode is called GCM-GHASH in

1.Notation: $\{0\}^{31}1$ is defined to mean a string of thirty-one bits of 0 followed by a single bit of 1.

this document. The format of the context registers for GCM-GHASH mode is shown in Table 11-36 on page 11-91. GCM cipher mode also has option of automatically verifying that the received and computed MAC tags are identical. This cipher mode is called GCM w/ICV and can be specified by setting Mode Register bits 56, 57 and 62 to 1, and bit 61 to 0. GCM w/ICV context format is shown in Table 11-36.

Messages (IV+AAD+textdata) are fed in through the input FIFO, and are always processed in the following order: IV, AAD, textdata, followed by the final MAC computation (where “textdata” refers to plaintext or ciphertext to be operated on). The whole message, however, does not have to be processed in one GCM execution. It can be split and processed in multiple GCM runs separated by resets of the AESU block such as in multiple descriptors. The boundaries can be set at the end of any full block (16 bytes) of the stream IV+AAD+textdata. Hence, any of the individual components (IV, AAD, or textdata) can be split into multiple descriptors. Refer to Table 11-32 for proper AUX mode specification in this case and to Table 11-33 through Table 11-36 for proper context formatting. It should be noted that in case of a late arrival of the MAC tag on the receiving side, the final MAC can be computed and verified against the received MAC in a separate descriptor after the rest of the message (IV+AAD+textdata) has already been processed.

For host-controlled operation of the AESU in GCM cipher mode, perform the following steps:

1. Reset.
2. Set Cipher Mode to GCM or GCM w/ICV and specify encrypt, decrypt in the Mode Register. To perform GCM-GHASH (only GHASH (H, AAD, ciphertext) is computed) set AUX0 and specify encrypt. Set AUX2 and AUX1 bits according to Table 11-32.
3. Load Key.
4. Load (Restore) Context as needed (see Table 11-33 through Table 11-36).
5. Set Key size.
6. Set the size of the computed/received MAC (8, 12 or 16 bytes; default is 16).
7. Set Data size.
8. While available:
   a) Load IV into the input FIFO (1 or multiple blocks up to 2^{64} bits in total).
   b) Load AAD into the input FIFO (0 or multiple blocks up to 2^{64} bits in total).
   c) Load plaintext (for encryption) or ciphertext (for decryption) blocks into the input FIFO.
   d) Unload ciphertext (for encryption) or plaintext (for decryption) blocks from the output FIFO.
9. Write to the End of Message register.
10. Unload final ciphertext (for encryption) or plaintext (for decryption) blocks.
11. Read (Save) context registers if another segment of the message is processed later.
12. Read final GCM MAC from Context Registers 1–2, if AUX2 bit was set in Mode Register.
13. For GCM w/ICV, check ICCR bits in the AESU Status Register.

Table 11-32. GCM Cipher Mode Auxiliary Bit Definitions

Auxiliary Bit	Definitions	
	0	1
AUX2 (bit 58)	Do not compute MAC	Compute MAC
AUX1 (bit 59)	One of the following cases: Descriptor contains the whole message (IV+AAD+textdata) Descriptor contains the whole IV and no or part of AAD or textdata Descriptor contains a non-final part of IV, AAD, textdata (IV, AAD or textdata split between descriptors) Descriptor contains the final part of AAD or textdata but no MAC is computed	One of the following cases: Descriptor contains the final part of IV (IV split between descriptors) - len(IV)^T needed Descriptor contains the final part of textdata and the final MAC is computed, such as AUX2=1 (textdata split between descriptors) - len(AAD)^T, len(textdata)^T needed Descriptor contains the whole textdata but no or part of AAD and the final MAC is computed - len(AAD)^T, len(textdata)^T needed Descriptor contains the final part of AAD and the final MAC is computed - len(AAD)^T, len(textdata)^T needed Descriptor computes only MAC (based on restored context) but does not contain either IV, AAD or textdata - len(AAD)^T, len(textdata)^T needed
AUX0 (bit 60) and Encrypt	—	GHASH-only mode
AUX0 (bit 60) and Decrypt	The key is to be unrolled	The key is already unrolled

AUX0 has different use depending on whether encryption or decryption is specified. For decryption, it determines whether the provided key should be first unrolled before processing starts, while in case of encryption it should generally be set to 0 unless GCM-GHASH cipher mode is desired. AUX2 bit determines whether the final MAC tag is to be computed or not. If AUX2 is set to 1, E(K, Y_0) and the last iteration of the GHASH(H, AAD, ciphertext) is going to be performed and then XORed to give the MAC tag. Hence, if the message is split into multiple descriptors, only the last one should have AUX2=1 for proper MAC tag computation. AUX1 is used to resolve the issues related to the splitting of messages into multiple descriptors. Table 11-32 shows the proper settings of AUX1 for several scenarios of message splitting. In general, whenever the final GHASH iteration needs to be computed (either for GHASH(H, {}, IV) or GHASH(H, AAD, ciphertext)), and the current length is not equal to total length for either IV, AAD, or textdata, AUX1 should be set to 1. Consequently, an AUX1 value of 1 also indicates that the context registers 9-10 need to provide the total length of IV, AAD, or textdata for this to be accomplished.

Table 11-33 to Table 11-36 describe the proper usage of context registers in case of encryption, decryption, GCM w/ICV and GCM-GHASH cipher mode settings. The context is in each case described in terms of the input context required for starting new GCM processing or continuation of processing after context

switch, and in terms of the results (output) stored in context registers after GCM execution run is completed.

Description of context registers are as follows:

Reg 1–2

Intermediate MAC value. This needs to be provided only when switching context during AAD and/or textdata processing (AAD+textdata stream split into multiple descriptors).

On the output side, these registers contain either intermediate MAC tag in case of context switching (requires AUX2=0) or the final MAC tag at the end of processing if AUX2=1. If AUX2=0 on the last descriptor processing a particular message, these registers contain partially computed GHASH(H, AAD, ciphertext) where the last GHASH iteration is not computed.

In case of GCM w/ICV, the final MAC tag written here as the result of GCM processing is truncated to 8, 12, or 16 (no truncation) bytes as defined in ICV Size register. Note that any size from 1 to 16 bytes can be specified in ICV Size register but any value other than 8 or 12 is defaulted to 16 bytes automatically.

Reg 3–4

Received MAC tag used only in case of inbound processing with GCM w/ICV. This can be a 8-, 12-, or 16-byte block as specified by writing to the ICV Size register.

Reg 5–6

Counter value Y_i is required only if restoring context to continue processing a message. Note that the same value read when saving context should be written to these registers when restoring context since it is automatically incremented after every processed block.

In case of GCM-GHASH, these registers are not used.

Reg 7

Total length of AAD in bits. This is the total AAD length irrespective of whether AAD is split in multiple descriptors. It is required when AUX1=1 and the current descriptor processes the last segment of AAD or textdata. It is also required if the whole message is already processed and the current descriptor only computes the final MAC tag.

Reg 8

Total length of the plaintext/ciphertext or IV in bits. Required only when AUX1=1 (see Table 11-32). If the current descriptor processes the last segment of the IV then total IV length should be provided, otherwise total length of textdata should be provided.

Reg 9–10

Initial counter value Y_0. Normally, this value is a result of the IV stream processing and needs to be provided only if the message is split into multiple descriptors and for those descriptors that come after IV processing is complete. Otherwise, value provided here is ignored and overwritten with computed Y_0.

In case of GCM-GHASH cipher mode setting, constant H from GHASH(H, AAD, ciphertext) should be provided in these registers. Note that this, in the general case, does not have to be equal to E(K, $\{0\}^{128}$) where K is a key as defined for GCM.

Reg 11

Length of AAD in bits. This pertains to the length of the AAD part processed in the current descriptor. If the current descriptor does not process AAD, 0 should be written. If AAD is not split into multiple descriptors, this field should contain the total AAD length. The value written here should be divisible by 128 for all AAD segments except for the last one that can be any number of bits. Note, however, that the actual AAD stream supplied to the AES engine through FIFOs has to be zero-padded to an integral number of 16-byte blocks.

Reg 12

Length of IV in bits or a part of it processed in the current descriptor. Similar remarks apply like in the case of AAD.

In case of GCM-GHASH, this register is not used.

Table 11-33. GCM Encryption Context

Context Register	GCM Encrypt (outbound)				
	Mode Register [7:0]				
	10_?_AUX1_0_01_1			10_AUX2_?_0_01_1	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed)
2					
3	—			—	
4					
5	Y_i (Counter)			—	
6					
7	—	len(AAD)^{T*}	—	—	
8	—	len(textdata)^{T*}	len(IV)^{T}	—	
9	Y_0 (Initial Counter)			—	
10					
11	len(AAD)^{C*}			—	
12	len(IV)^{C*}			—	

***** Must be written at the start of a new message, except if zero
C length of data processed with current descriptor (in bits)
T length of total data (in bits)
-- don't care

Table 11-34. GCM Decryption Context

Context Register	GCM Decrypt (inbound)				
	Mode Register [7:0]				
	10_?_AUX1_?_01_0			10_AUX2_?_?_01_0	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	
2					
3	—			—	MAC (Computed)
4					
5	Yi (Counter)			—	
6					
7	—	len(AAD)^{T*}	—	—	
8	—	len(textdata)^{T*}	len(IV)^{T*}	—	
9	Y_0 (Initial Counter)			—	
10				—	
11	len(AAD)^{C*}			—	
12	len(IV)^{C*}				

* Must be written at the start of a new message, except if zero
C length of data processed with current descriptor (in bits)
T length of total data (in bits)
-- don't care

Table 11-35. GCM w/ICV Context

Context Register	GCM w/ICV (inbound)				
	Mode Register [7:0]				
	11_?_AUX1_?_01_0			11_AUX2_?_?_01_0	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed and truncated to icv_size most significant bytes)
2					
3	MAC (Received)			—	
4					
5	Y_i (Counter)			—	
6					
7	—	len(AAD)^{T*}	—	—	
8	—	len(textdata)^{T*}	len(IV)^{T}	—	
9	Y_0 (Initial Counter)			—	
10				—	
11	len(AAD)^{C*}			—	
12	len(IV)^{C*}				

***** Must be written at the start of a new message, except if zero
C length of data processed with current descriptor (in bits)
T length of total data (in bits)
-- don't care

Table 11-36. GCM-GHASH Context

Context Register	GCM-GHASH (only GHASH computed)				
	Mode Register [7:0]				
	10_?_AUX1_1_01_1			10_AUX2_?_1_01_1	
	Inputs			Outputs	
	AUX1 = 0	AUX1 = 1		AUX2 = 0	AUX2 = 1
		last AAD or textdata segment, or MAC only	last IV segment		
1	MAC (Computed)			—	MAC (Computed)
2					
3	—			—	
4					
5	—			—	
6					
7	—	len(AAD)^{T*}	—	—	
8	—	len(textdata)^{T*}	len(IV)^{T}	—	
9	H*			—	
10					
11	len(AAD)^{C}			—	
12	—			—	

***** Must be written at the start of a new message, except if zero
C length of data processed with current descriptor (in bits)
T length of total data (in bits)
-- don't care

As an example, lets consider the case of GCM encrypt operation that generates the final MAC tag and where the whole message is small enough that it can be processed with one descriptor. The mode bits 56–63 (ECM, AUX, CM, and ED) should be set to 10_100_01_1. Only context registers 11–12 need to be written with the bit lengths of AAD and IV, respectively. The bit length of textdata should be written to the data size register up to the size of 2^{19} bits. IV, AAD, and textdata in that order should be sent through the input FIFO and the result (textdata) read from the output FIFO as available. At the end, the final MAC tag is read from the context registers 1–2.

11.7.1.11.8 AESU Key Registers

The AESU key registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to key 1. Any key data written to bytes beyond the value written to the key size register is ignored. The key data registers are cleared when the AESU is reset or re-initialized. If these registers are modified during message processing, a context error is generated.

The key data registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the “restore decrypt key” bit must be set in the mode register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

11.7.1.11.9 AESU FIFOs

AESU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For host-controlled operation, a write to anywhere in the AESU FIFO address space enqueues data to the AESU input FIFO, and a read from anywhere in the AESU FIFO address space dequeues data from the AESU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, word (4 bytes), or dword (8 bytes). When all 8 bytes of the staging register have been written, the entire dword is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AESU End of Message Register is written.

The output FIFO is readable by byte, word, or dword. When all 8 bytes of the head dword have been read, that dword is automatically dequeued from the FIFO so that the next dword (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the AESU FIFOs are reflected in the AESU interrupt status register.

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted and the results are placed in the output FIFO. The output size is the same as the input size.

The input FIFO may be written any time the number of dwords currently in the input FIFO (as indicated by the IFL field of the AESU status register) is less than 32. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the data size register.

The output FIFO may be read any time the OFR signal is asserted (as indicated in the AESU status register). This indicates that the number of bytes in the output FIFO is at or above the threshold specified in the mode register.

11.7.2 ARC Four Execution Unit (AFEU)

This section contains details about the ARC four execution unit (AFEU), including modes of operation, status and control registers, S-box memory, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the AFEU is used through channel-controlled access, which means that most reads and writes of AFEU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.2.1 AFEU Mode Register

The AFEU mode register, shown in Figure 11-35, contains 3 bits that are used to program the AFEU. The mode register is cleared when the AFEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

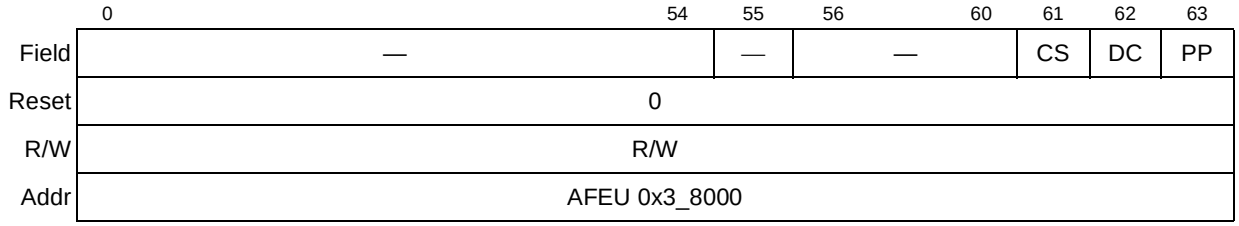


Figure 11-35. AFEU Mode Register

Table 11-37 describes AFEU mode register fields.

Table 11-37. AFEU Mode Register Field Descriptions

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–54	—	Reserved
55	—	Reserved
The following bits are controlled through the MODE0 field of the descriptor header.		
56–60	—	Reserved
61	CS	Context Source. If set, this causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the context registers or context should be generated automatically via key permutation. Context Source is only checked if the prevent permute bit is set. 0 Context not from FIFO 1 Context from input FIFO
62	DC	Dump Context. If set, this causes the context to be moved from the S-box to the output FIFO following assertion AFEU's done interrupt. 0 Do not dump context 1 After cipher, dump context
63	PP	Prevent Permute. Normally, AFEU receives a key and uses that information to randomize the S-box. If reusing a context from a previous descriptor, this bit should be set to prevent AFEU from re-performing this permutation step. 0 Perform S-box permutation 1 Do not permute

11.7.2.2 AFEU Key Size Register

This value, shown in Figure 11-36, indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in the key size register is ignored. This register is cleared when the AFEU is reset or re-initialized. If the key size specified is less than 1 or greater than 16, a key size error is generated. If the key size register is modified during processing, a context error is generated. Note: Although the AFEU supports key lengths as short as 1 byte, a 1-byte key offers little security. Most uses of ARC-4 specify keys of 5–16 bytes.

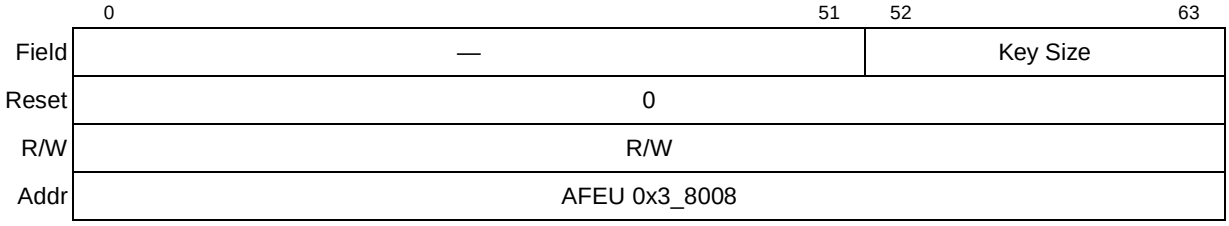


Figure 11-36. AFEU Key Size Register

NOTE

The device driver creates properly formatted descriptors for situations requiring a key permute prior to ciphering. When using host-controlled access (typically for debug), the user must set the AFEU mode register to perform “permute with key,” write the key data to AFEU Key Registers, then write the key size to the key size register. The AFEU starts permuting the memory with the contents of the key registers immediately after the key size is written.

11.7.2.3 AFEU Context/Data Size Register

The AFEU context/data size register, shown in Figure 11-37, specifies the number of bits in the final message, with an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 8. This value controls how much data is processed from the last block. The last message block must be a multiple of 8, from 8 to 64. If a data size that is not a multiple of 8 bits is written, a data size error is generated. Only bits 61:63 are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

The context/data size register is also used to specify the context size, when context is used. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

NOTE

When reloading an existing context using host-controlled access, the user must write the context to the input FIFO, then write the context size (always 2072 bits). The write of the context size indicates to the AFEU that all context has been loaded. The user then writes the message data size to the context/data size register. After this write, the user may begin writing message data to the FIFO.

Writing to this register signals the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

This register is cleared when the AFEU is reset or re-initialized.

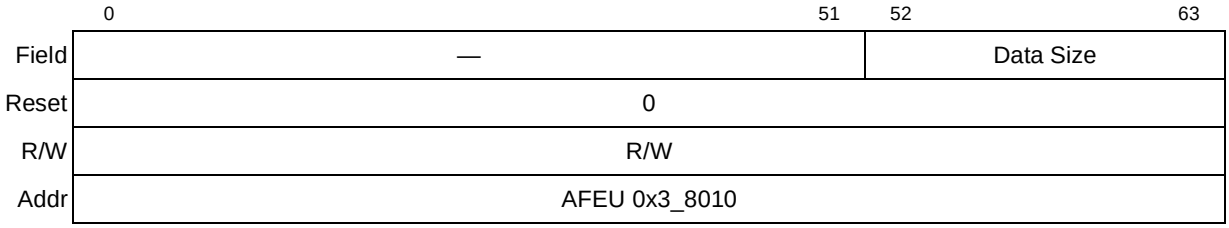


Figure 11-37. AFEU Context/Data Size Register

11.7.2.4 AFEU Reset Control Register

The AFEU reset control register, as shown in Figure 11-38, allows three levels reset that affect the AFEU only, as defined by 3 self-clearing bits. It should be noted that the AFEU executes an internal reset sequence for hardware reset, SW_RESET, or Module Initialization, which performs proper initialization of the S-box. To determine when this is complete, observe the RESET_DONE bit in the AFEU status register.

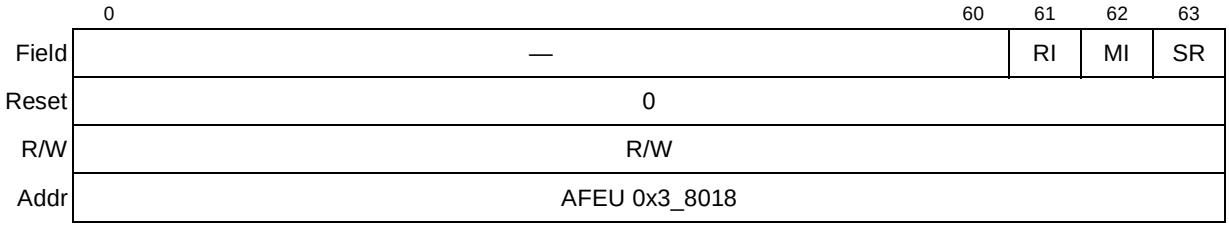


Figure 11-38. AFEU Reset Control Register

Table 11-38 describes AFEU reset control register fields.

Table 11-38. AFEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes AFEU interrupts signalling done and error to be reset. It further resets the state of the AFEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization resets everything reset by SR, with the exception of the AFEU Interrupt Mask Register. 0 Do not reset 1 Reset most of AFEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AFEU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the AFEU enters a routine to perform proper initialization of the S-box. 0 Do not reset 1 Full AFEU reset

11.7.2.5 AFEU Status Register

The AFEU status register, shown in Figure 11-39, reflects the state of AFEU internal signals.

The AFEU status register is read-only. Writing to this location results in address errors being reflected in the AFEU interrupt status register.

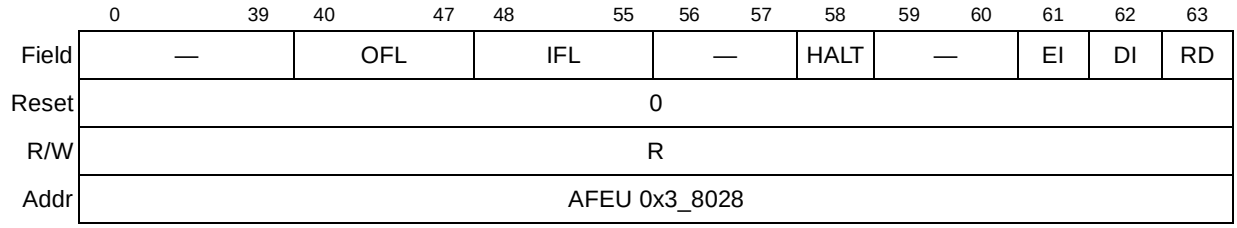


Figure 11-39. AFEU Status Register

Table 11-39 describes AFEU status register fields.

Table 11-39. AFEU Status Register Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the AFEU has halted due to an error. 0 AFEU not halted 1 AFEU halted **Note:** Because the error causing the AFEU to stop operating may be masked before reaching the interrupt status register, the AFEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling error 1 AFEU is signaling error
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 AFEU is not signaling done 1 AFEU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that AFEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.2.6 AFEU Interrupt Status Register

The AFEY interrupt status register, shown in Figure 11-40, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the AFEU interrupt mask register is zero (see Section 11.7.2.7, “AFEU Interrupt Mask Register”).

If the AFEU interrupt status register is non-zero, the AFEU halts and the AFEU error interrupt signal is asserted to the controller (see Section 11.6.3.3, “Interrupt Status Register (ISR)”). In addition, if the AFEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the channel status register (see Table 11-15) and generates a channel error interrupt to the controller.

If the interrupt status register is written from the host, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the interrupt mask register. All other bits are cleared. This register can also be cleared by setting the RI bit of the AFEU reset control register.

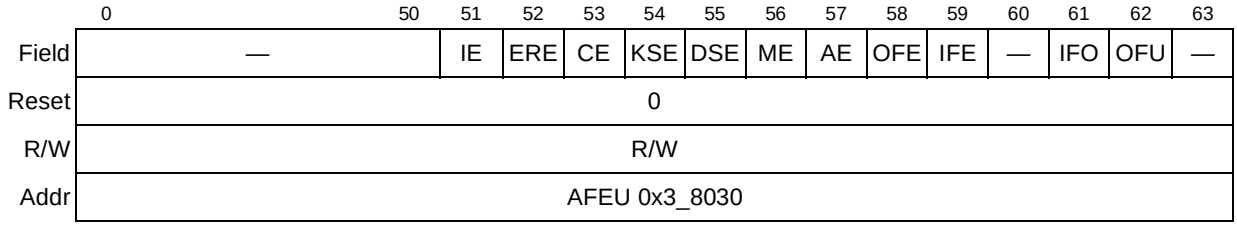


Figure 11-40. AFEU Interrupt Status Register

Table 11-40 describes AFEU interrupt status register fields.

Table 11-40. AFEU Interrupt Status Register

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error
52	ERE	Early Read Error. The AFEU Context Memory or Control was read while the AFEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context Error. The AFEU Mode Register, Key Register, Key Size Register, Data Size Register, or context memory is modified while AFEU processes data. 0 No error detected 1 Context error
54	KSE	Key Size Error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 No error detected 1 Key size error
55	DSE	Data Size Error. A value that is not a multiple of 8 bits was written to the AFEU Data Size Register: 0 No error detected 1 Data size error

Table 11-40. AFEU Interrupt Status Register (continued)

Bits	Names	Description
56	ME	Mode Error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address Error. An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO Error. The AFEU output FIFO was detected non-empty upon write of AFEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO Error. The AFEU Input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO Overflow. The AFEU input FIFO was pushed while full. 1 Input FIFO has overflowed 0 No error detected **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the AFEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO Underflow. The AFEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

11.7.2.7 AFEU Interrupt Mask Register

The interrupt mask register, shown in Figure 11-41, controls the result of detected errors. For a given error (as defined in Section 11.7.2.6, “AFEU Interrupt Status Register”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0–50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	—	IFO	OFU	—
Reset	1000													
R/W	R/W													
Addr	AFEU 0x3_8038													

Figure 11-41. AFEU Interrupt Mask Register

Table 11-41 describes AFEU interrupt mask register fields.

Table 11-41. AFEU Interrupt Mask Register

Bits	Names	Description
0–50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The AFEU Register was read while the AFEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. An AFEU Key Register, the Key Size Register, Data Size Register, Mode Register, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. A value outside the bounds 1–16 bytes was written to the AFEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. An inconsistent value was written to the AFEU Data Size Register: 0 Data Size error enabled 1 Data size error disabled
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO Error. The AFEU Output FIFO was detected non-empty upon write of AFEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO Error. The AFEU Input FIFO was detected non-empty upon generation of done interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO Overflow. The AFEU Input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO Underflow. The AFEU Output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

11.7.2.8 AFEU End of Message Register

The end of message register in the AFEU, shown in Figure 11-42, is used to signal the AFEU that all data to be processed has been written to the input FIFO. This allows the AFEU to do special processing when it reaches the last block of data. Before this register is written, the AFEU does not process the last block of data in its input FIFO. After this register is written, the AFEU continues to do normal processing on all but the last block of data, then goes on to processes the last block, using the value in the data size register to determine how much of the block to process. The data size register specifies the number of bits to process, which is a multiple of 8, from 8 to 64. Once processing of the last block is completed, the AFEU signals done interrupt. If the “dump context” bit in the AFEU Mode Register is set, the context is written to the output FIFO following the last message word. A read of the AFEU End of Message Register always returns a zero value.

	0	63
Field	AFEU End of Message	
Reset	0	
R/W	W	
Addr	AFEU 0x3_8050	

Figure 11-42. AFEU End of Message Register

11.7.2.9 AFEU Context

An ARC4 encryption session begins with an initial key permute to generate the initial S-box state. After that each subsequent message in the same session makes use of the S-box state and also modifies the S-box state.

To implement this behavior using the AFEU, the first descriptor of a session must do a key permute operation. If there are additional messages in the same session, then at the end of each descriptor execution the S-box state must be dumped out as context so that the next message of the session can re-load that context.

AFEU context consists of two parts, as follows:

- AFEU Context Memory—a 256-byte SRAM that holds the current S-box contents
- AFEU Context Memory Pointer Register—holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC-4 algorithm.

There is no standard data format for S-box state information. To ensure proper AFEU operation, the AFEU context should only be written with data that was read from the AFEU context during a previous operation.

11.7.2.9.1 Writing AFEU Context

In the default mode of operation, the key and key size are provided to the AFEU. The initial memory values in the S-box are permuted with the key to create new S-box values, which are used to encrypt the plaintext.

If the “Prevent Permute” (PP) mode bit is set in the AFEU Mode Register (see Section 11.7.2.1, “AFEU Mode Register”), then the AFEU does not require a key, but instead requires context to set the S-box state before processing data. This mode is used to resume processing in an ARC4 session using the previously computed S-box. The context may be written to the AFEU through the context registers or through the input FIFO, as selected by the CS mode bit.

After context data, the context length must be written to the context/data length register (see Section 11.7.2.3, “AFEU Context/Data Size Register”). After this, message data can be written. If the context registers are written during message processing or written when the “prevent permutation” bit is not set, a context error is generated.

For more information about writing AFEU context through the input FIFO, see Section 11.7.2.10.1, “AFEU FIFOs.”

11.7.2.9.2 Reading AFEU Context

Once message processing is complete and the output data has been read, the AFEU S-box state can be read out either through the context registers or through the output FIFO, as selected by the “Dump Context” (DC) mode bit (see Section 11.7.2.1, “AFEU Mode Register”).

Valid context data can only be read after AFEU has completed processing (as indicated by the *ipi_done_int* line and the “DI” bit of the AFEU Status Register, Section 11.7.2.5, “AFEU Status Register”). Reading context data before the module is done generates an error interrupt.

For more information about reading AFEU context through the output FIFO, see Section 11.7.2.10.1, “AFEU FIFOs.”

11.7.2.10 AFEU Key Registers

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-box, in conjunction with the AFEU key size register. AFEU performs permutation starting with the first byte of key register 0, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations generates an address error interrupt.

11.7.2.10.1 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For host-controlled operation, a write to anywhere in the AFEU FIFO address space enqueues data to the AFEU input FIFO, and a read from anywhere in the AFEU FIFO address space dequeues data from the AFEU output FIFO.

When context is written through the input FIFO (see Section 11.7.2.9.1, “Writing AFEU Context”), the first context write must be in the address range 3_8E00-3_8E07.

Similarly, when context is read through the input FIFO (see Section 11.7.2.9.1, “Writing AFEU Context”), the first context read must be in the address range 3_8E00-3_8E07. This causes any incomplete data word remaining in the output FIFO to be cleared out so that the context can be read.

Writes to the input FIFO go first to a staging register which can be written by byte, word (4 bytes), or dword (8 bytes). When all 8 bytes of the staging register have been written, the entire dword is

automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the AFEU End of Message Register is written.

The output FIFO is readable by byte, word, or dword. When all 8 bytes of the head dword have been read, that dword is automatically dequeued from the FIFO so that the next dword (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

11.7.3 Cyclical Redundancy Check Unit (CRCU)

This section contains details about the cyclical redundancy check accelerator (CRCU), including modes of operation, status and control registers, and FIFO.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the CRCU is used through channel-controlled access, which means that most reads and writes of CRCU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.3.1 ICV Checking

This EU includes an ICV checking feature, that is, it can verify a message/CRC pair by calculating a raw CRC and comparing it to the polynomial specific residue. The pass/fail result of this check can be returned to the host either via interrupt by a writeback of EU status fields into host memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see Section 11.5.4.1, “Channel Configuration Register (CCR)”), and mask the CICVF bit in the Interrupt Mask Register (see Section 11.7.3.9, “CRCU Interrupt Mask Register”). In this case the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the CICVF bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no CRC mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an CRC mismatch, there is an error interrupt to the host, but no done interrupt or writeback.

11.7.3.2 CRCU Mode Register

The CRCU mode register, shown in Figure 11-43, is used to program the function of the CRCU and is generally the first register written. A context error is generated if this register is written after processing has begun.

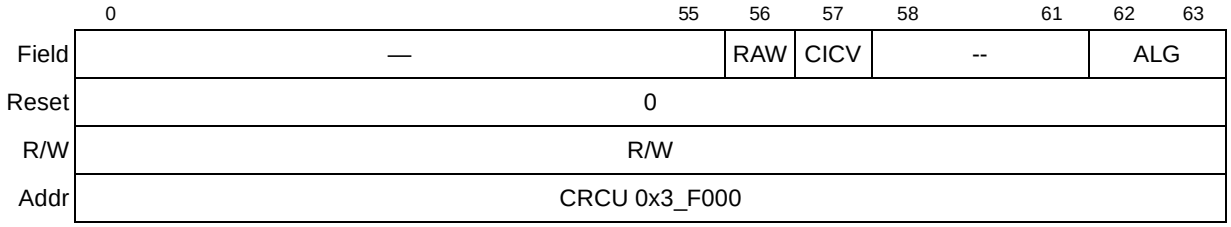


Figure 11-43. CRCU Mode Register

Table 11-42 describes CRCU mode register fields.

Table 11-42. CRCU Mode Register Bit Definitions

Bits	Name	Description
0–55	—	Reserved
56	RAW	RAW Raw output. Controls whether CRC result bit manipulations are performed. 0 Perform bit manipulations. The CRC result is bit-swapped, byte-swapped, and then complemented before it is available in the Context Register. This is compliance with the IEEE 802 and iSCSI CRC implementations. 1 Do not perform bit manipulations. Provide raw CRC result in the Context Register.
57	CICV	ICV (CRC) Check. Selects whether a comparison between the computed CRC and the residue should be performed. 0 Disabled. 1 Enable ICV check. Compares a CRC calculated across the message and received ICV against the stored residue. The comparison is performed prior to the bit manipulations controlled by the RAW bit.
58–61	—	Reserved
63	ALG	Algorithm. Selects the CRC algorithm mode for the CRCU. 00 IEEE 802 mode. The CRC32 algorithm is performed using the polynomial 0x04C11DB7 and the residue 0xC704DD7B is used for ICV checking. 01 iSCSI mode. The CRC32c algorithm is performed using the polynomial 0x1EDC6F41 and the residue 0x1C2D19ED is used for ICV checking. 10 Static custom mode. The CRC remainder is computed using the Control Register bits 32-63 as the polynomial and bits 0-31 as the residue. 11 Dynamic custom mode. The CRC remainder is computed using the Key Register bits 32-63 as the polynomial and bits 0-31 as the residue.

11.7.3.3 CRCU Key Size Register

The key size register, shown in Figure 11-44, stores the number of valid bytes in the dynamic polynomial written to the Key Register (Section 11.7.3.14, “CRCU Key Register”). This value is reset to zero, and if a value other that zero or four is written to it, an illegal key size error is generated. It is not necessary to write to this register, as the polynomial size is clearly fixed by the algorithm. A context error is generated if this register is written after processing has begun.

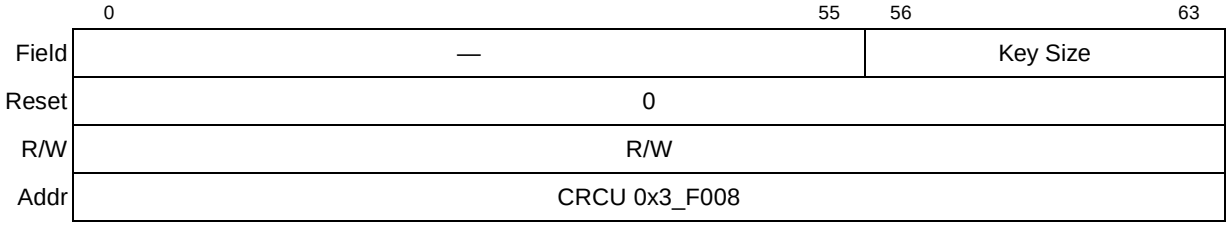


Figure 11-44. CRCU Key Size Register

11.7.3.4 CRCU Data Size Register

The CRCU data size register, shown in Figure 11-45, is written with the number of bits of data to be processed. Writing to this register puts the CRCU module into a busy state and starts data processing. This register may be written multiple times while data processing is in progress. The actual values written are ignored, although an error is generated if the value is not a multiple of 8 bits.

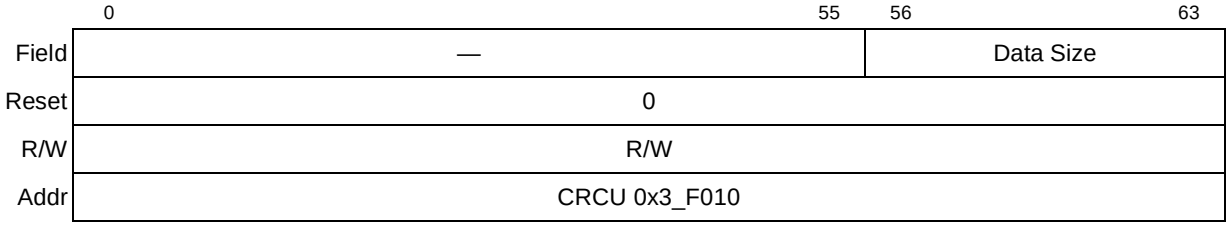


Figure 11-45. CRCU Data Size Register

11.7.3.5 CRCU Reset Control Register

The CRCU reset control register, shown in Figure 11-46, controls the reset/re-initialization of the block.

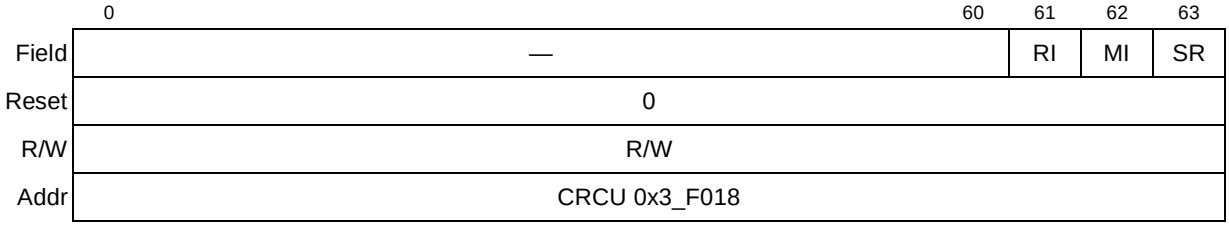


Figure 11-46. CRCU Reset Control Register

Table 11-43 describes CRCU reset control register fields.

Table 11-43. CRCU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes CRCU interrupts signalling done and error to be reset. It further resets the state of the CRCU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization resets everything reset by SR, with the exception of the CRCU Interrupt Mask Register. 0 Do not reset 1 Reset most of CRCU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for CRCU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the CRCU enters a routine to perform proper initialization of the S-box. 0 Do not reset 1 Full CRCU reset

11.7.3.6 CRCU Control Register

The CRCU control register, shown in Figure 11-47, stores the static polynomial used in custom CRC computations. Figure 11-47 shows the bit position of each coefficients in the polynomial.

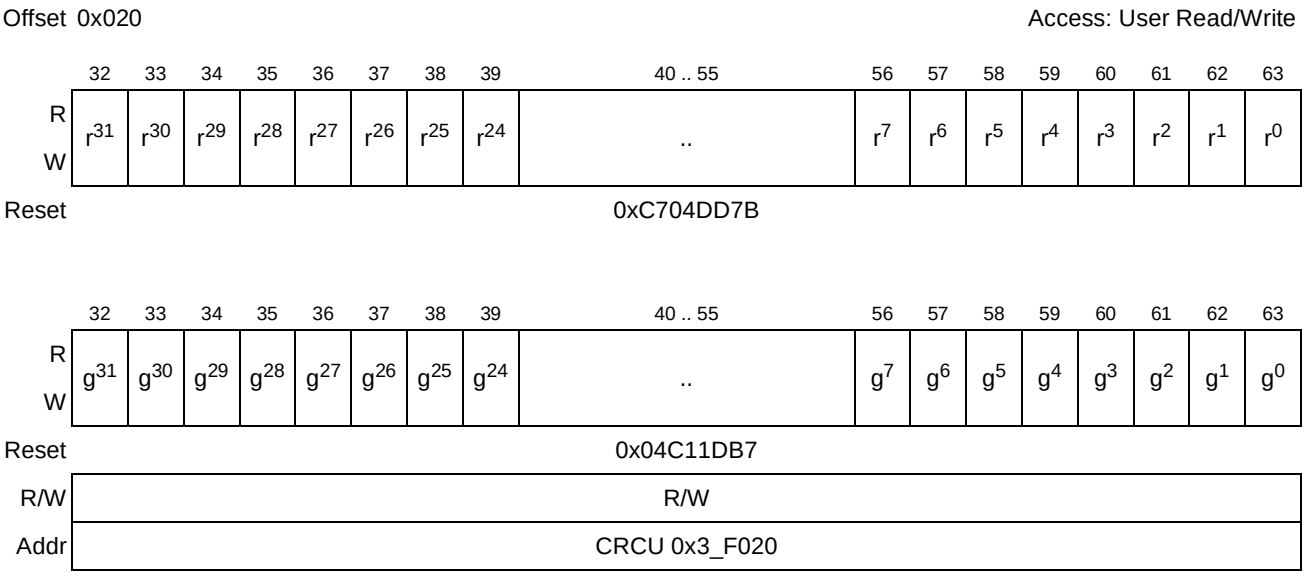


Figure 11-47. CRCU Control Register

The reset value of this register is the IEEE 802 CRC32 residue coefficient “r” and polynomial coefficient “g.” This register is static, in that it is only reset by performing a software reset, not by an EU re-initialize. This allows a platform specific custom polynomial to be written to the register once and used many times. A context error is generated if this register is written after processing has begun. A polynomial error is generated if a value is written to this register which does not have a one in g^0.

11.7.3.7 CRCU Status Register

The CRCU status register, shown in Figure 11-48, provides general information on the operational status of the CRCU. A read of the status register captures a snapshot of CRCU's operating state at a particular moment in time. Of notable interest to the user are the three interrupt flags *(irq_error, irq_done,* and *irq_reset*), providing visibility to EU *ports ipi_error_int, ipi_done_int,* and *ipi_reset_int,* respectively. Writes to this register are ignored.

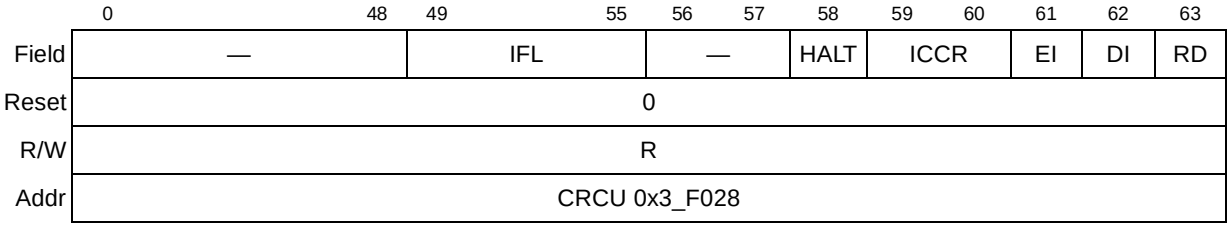


Figure 11-48. CRCU Status Register

Table 11-44 describes CRCU status register fields.

Table 11-44. CRCU Status Register Bit Definitions

Bits	Name	Description
0–48	—	Reserved
49–55	IFL	Input FIFO Level: The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Indicates when the CRCU core has halted due to an error. 0 CRCU not halted 1 CRCU core halted (Must be reset/re-initialized) **Note:** Because the error causing the CRCU to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity Check Comparison Result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved
61	EI	Error interrupt. Reflects the state of the error interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, "Interrupt Status Register (ISR)"). 0 CRCU is not signaling error 1 CRCU is signaling error
62	DI	Done Interrupt: Reflects the state of the done interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, "Interrupt Status Register (ISR)"). 0 Processing not done 1 All bytes processed
63	RD	Reset Done: Indicates when the CRCU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done

11.7.3.8 CRCU Interrupt Status Register

The CRCU interrupt status register, shown in Figure 11-49, latches the source of various error interrupts. This register is both readable and writable, thus allowing the setting and/or clearing of the individual status bits. All status bits are maskable by setting the corresponding bit in the Interrupt Mask Register. A status bit must be unmasked in order for it to be set. Individual status bits may be cleared by writing a zero to the respective bits. If these bits are masked, writing a one still clears them. The entire interrupt status register may be cleared by resetting/re-initializing CRCU, or by writing to the Clear IRQ bit in the reset control register. Exception: Bit 12, 'Halt on Error' may only be cleared by resetting/re-initializing CRCU.

If an error occurs and is not masked, the *ipi_error_int* signal is asserted and CRCU stops processing. If the error is masked the interrupt is not asserted and the CRCU attempts to continue.

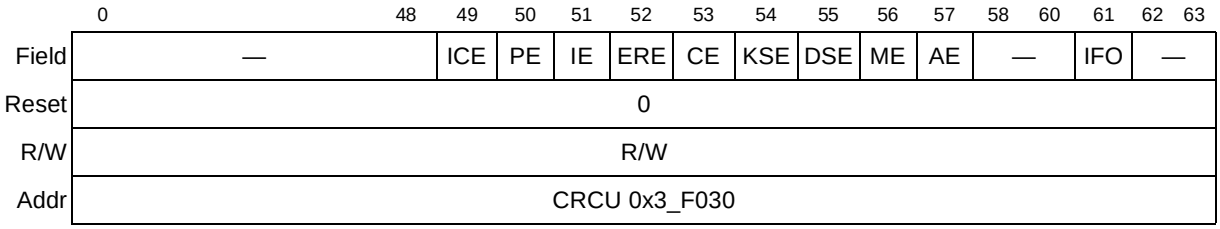


Figure 11-49. CRCU Interrupt Status Register

Table 11-45 describes CRCU interrupt status register fields.

Table 11-45. CRCU Interrupt Status Register Bit Definitions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check (CRC) Error: 0 No error detected 1 Integrity check error detected. An ICV (CRC) check was performed and the supplied ICV (CRC) did not match the one computed by the CRCU.
50	PE	Polynomial Error. 0 No error. 1 An invalid polynomial has been written to the Key or Control Register.
51	IE	Internal Error. Indicates the CRCU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected **Note:** This bit is asserted any time an enabled error condition occurs and can only be cleared by resetting the CRCU.
52	ERE	Early Read Error. The CRCU context was read before the CRCU completed the requested operation. 0 No error detected 1 Early read error
53	CE	Context Error. The CRCU Mode, Key Size, Control, Context, or Key Register was modified after processing had begun. 0 No error detected 1 Context error

Table 11-45. CRCU Interrupt Status Register Bit Definitions (continued)

Bits	Name	Description
54	KSE	Key Size Error. A value other than 0 or 4 has been written to the CRCU Key Size Register. 0 No error detected 1 Key size error
55	DSE	Data Size Error. An non-byte-multiple size has been written to the CRCU Data Size Register. 0 No error detected 1 Data size error
56	ME	Mode Error. Is set if any of the following error conditions is detected: • Any reserved bit of the Mode register is set • The ALG field of the Mode register contains an illegal value 0 No error detected 1 Mode error
57	AE	Address Error. An illegal read or write address was detected within the CRCU address space. 0 No error detected 1 Address Error
58–60	—	Reserved
61	IFO	Input FIFO Overflow. The CRCU Input FIFO was pushed while full. 0 No overflow detected 1 Input FIFO has overflowed **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input size. When operated through host-controlled access, the CRCU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62–63	—	Reserved

11.7.3.9 CRCU Interrupt Mask Register

The CRCU interrupt mask register, shown in Figure 11-50. allows for individual error sources to be masked, thus preventing the *ipi_error_int* signal from being asserted due to a masked condition. Error status is not captured for any error bits that are masked. Any host writes to a masked error bit results in clearing that error bit. Masking an error bit allows for a hardware error condition to go potentially undetected. Therefore, extreme care should be taken when masking errors as invalid results may be produced. It is recommended that errors only be masked during debug operation. This register may be reset by resetting the CRCU.

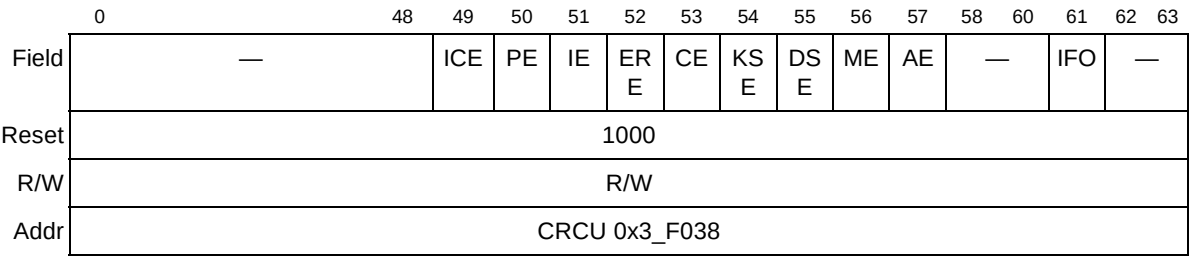


Figure 11-50. CRCU Interrupt Mask Register

Table 11-46 describes CRCU interrupt mask register fields.

Table 11-46. CRCU Interrupt Mask Register Bit Definitions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check Error. The supplied ICV(CRC) did not match the one computed by the CRCU. 0 Integrity check error enabled. WARNING: Do not enable this if using EU status writeback (see bits IWSE and AWSE in Section 11.5.4.1, "Channel Configuration Register (CCR)"). 1 Integrity check error disabled
50	PE	Polynomial Error. 0 Polynomial error enabled 1 Polynomial error disabled
51	IE	Internal Error. An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The CRCU register was read while the CRCU was performing hashing. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. The CRCU key register, the key size register, the data size register, or the mode register, was modified while the CRCU was performing hashing. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. A value outside the bounds was written to the CRCU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. An inconsistent value was written to the CRCU data size register: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the CRCU address space. 0 Address error enabled 1 Address error disabled
58–60	—	Reserved
61	IFO	Input FIFO Overflow. The CRCU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62–63	—	Reserved

11.7.3.10 CRCU ICV Size Register

This location is word readable/writable for compatibility with the MDEU. Values written to this location are always ignored and values read from this location always return zero. A context error is generated if this register is written after processing has begun.

11.7.3.11 CRCU End of Message Register

Used to indicate that all data has been written to the CRCU. A write to this register is required to complete a CRC32 operation. The CRCU starts processing message data as soon as the data size register is written and data becomes available in the FIFO, but it does not process a remaining partial word or perform an ICV check until a write to this register. The value written is not used. Reading this register returns a zero value.

11.7.3.12 CRCU Received ICV Register

This register is written with the ICV, a 32-bit CRC value computed over the original data. When the CICV feature is enabled in the mode register, the value written to this register is compared to the computed CRC result available in the data out register. As a direct comparison is done between the two registers, the bit position of each term is dependent on the mode in the same manner as the bits of the data out register. If the two values differ a CICV fail error is generated. This register can only be written before End of Message has been written.

11.7.3.13 CRCU Context Register

The CRCU context register, shown in Figure 11-51, can be written with an intermediate CRC result or desired initial state prior to processing any data. Once processing is complete, the CRC result is available from this register. The reset state of this register is all ones, as this allows the CRC32 algorithm to detect bit errors in the leading zeros of a message. Figure 11-51 shows the bit position of each term in the written context value. A context error is generated if this register is written after processing has begun. An early read error is generated if this register is read while the module is busy.

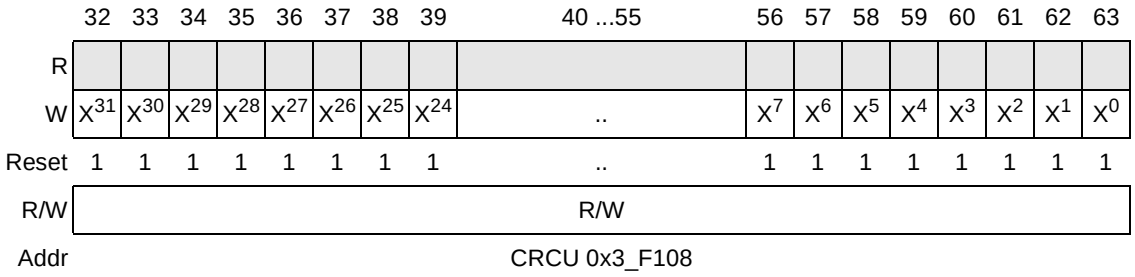


Figure 11-51. CRCU Context Register (Write)

If the CRCU is in the default output mode (RAW bit is 0), this register when read holds the CRC remainder after it has been bit swapped, byte swapped, and complemented. This sequence of operations is described in the protocol specifications and generates a result which can be written directly to the end of a frame or command. Figure 11-52 shows the value in each bit position.

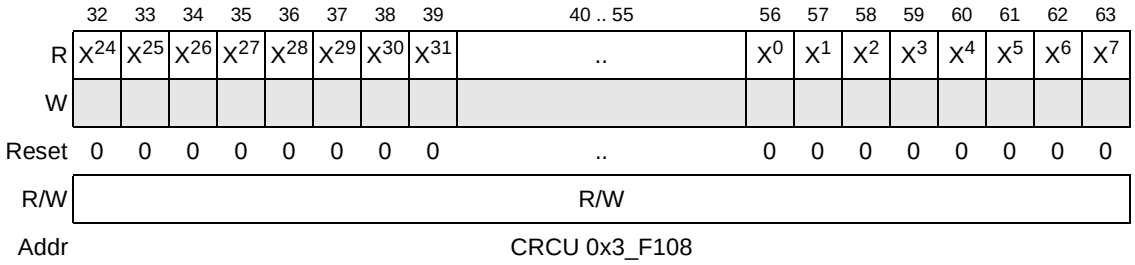


Figure 11-52. CRCU Context Register (Read-Default Mode)

If the CRCU is in the raw output mode (RAW bit is 1), this register when read holds the unmodified CRC remainder. This form is the one used internally to match against the polynomial specific residue when performing ICV checking. Figure 11-53 shows the value in each bit position.

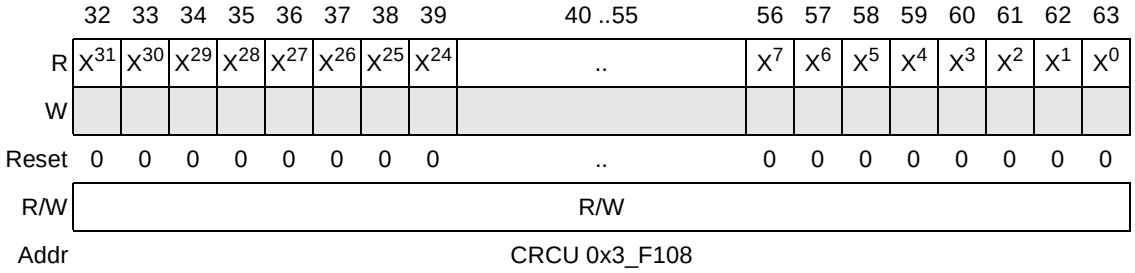


Figure 11-53. CRCU Context Register (Read-Raw Mode)

11.7.3.14 CRCU Key Register

The CRCU key register, shown in Figure 11-54, stores the polynomial and residue for the dynamic custom mode as set in the Mode Register (see Section 11.7.3.2, “CRCU Mode Register”). Figure 11-54 shows the bit position of each coefficients. The reset value of this register is all zeros with a one in bit position 0. This register is dynamic, in that it is reset by performing a re-initialize or a software reset. This allows a custom polynomial to be used for specific processing without changing the platform-specific static custom polynomial stored in the control register (see Section 11.7.3.6, “CRCU Control Register”). A residue does not need to be programmed unless ICV checking is being performed. A context error is generated if this register is written after processing has begun. A polynomial error is generated if a value is written to this register which does not have a one in bit position 0.

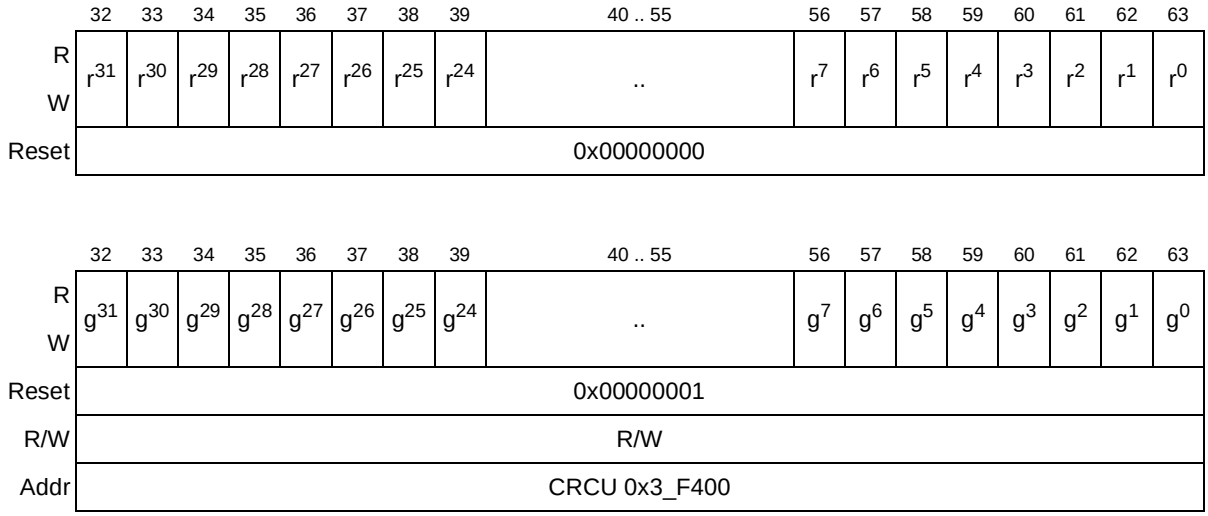


Figure 11-54. CRCU Key Register

11.7.3.15 CRCU FIFO

Words written to this address range are pushed onto the CRCU input FIFO, thereby buffering them for processing. Partial words and misaligned data can be written to this address and it is automatically realigned based on a big-endian byte order.

11.7.4 Data Encryption Standard Execution Unit (DEU)

This section contains details about the Data encryption standard execution unit (DEU), including modes of operation, status and control registers, and FIFOs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the DEU is used through channel-controlled access, which means that most reads and writes of DEU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.4.1 DEU Mode Register

The DEU mode register, shown in Figure 11-55, contains 3 bits that are used to program DEU operation.

The mode register is cleared when the DEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

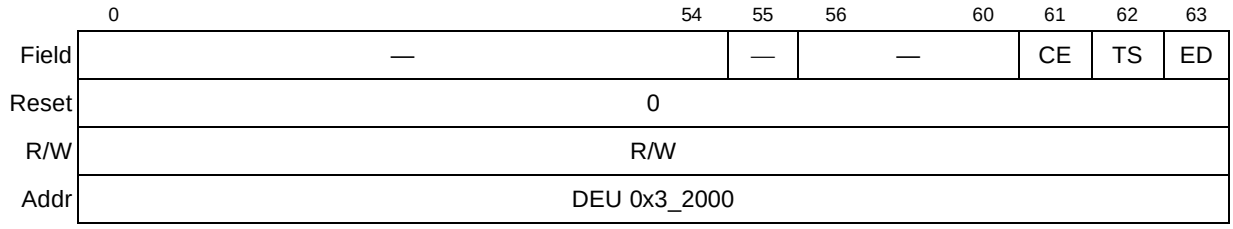


Figure 11-55. DEU Mode Register

Table 11-47 describes DEU mode register fields.

Table 11-47. DEU Mode Register Field Descriptions

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–54	—	Reserved
55	—	Reserved
The following bits are controlled through the MODE0 field of the descriptor header.		
56–59	—	Reserved
60-61	CM	Cipher Mode: Used to define the mode of DEU operation. See Table 11-48 on page 11-113 for mode bit combinations.
62	TS	Triple/Single DES. If set, DEU operates the Triple DES algorithm; if not set, DEU operates the single DES algorithm. 0 Single DES 1 Triple DES
63	ED	Encrypt/decrypt. If set, DEU operates the encryption algorithm; if not set, DEU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption

Table 11-48. DEU Cipher Modes

Mode	CM (60:61)
ECB	00
CBC	01
CFB-64	10
OFB-64	11

11.7.4.2 DEU Key Size Register

This value, shown in Figure 11-56**,** indicates the number of bytes of key memory that should be used in encrypting or decrypting. If the DEU mode register is set for single DES, any value other than 8 bytes automatically generates a key size error in the DEU interrupt status register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1=K3) or 24 bytes (168 bits for 3-key triple DES) generates an error. Triple DES always uses K1 to encrypt, K2 to decrypt, K3 to encrypt.

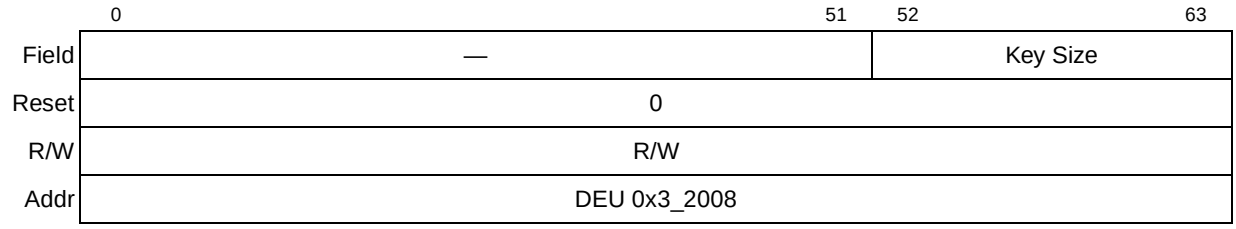


Figure 11-56. DEU Key Size Register

Table 11-49 shows the legal values for DEU key size.

Table 11-49. DEU Key Size Register Field Descriptions

Bits	Name	Description
0–51	—	Reserved
52–63	Key Size	8 bytes = 0x08 (only legal value if mode is single DES.) 16 bytes = 0x10 (for 2 key 3DES, K1 = K3) 24 bytes = 0x18 (for 3 key 3DES)

11.7.4.3 DEU Data Size Register

The DEU data size register, shown in Figure 11-57, specifies the number of bits in the final message, with an upper bound of 4096. Whatever number is written (and whatever truncated value is stored) must be a multiple of 64 except if OFB mode is selected. All data to be processed by the DEU must be a multiple of the DES algorithm block size of 64 bits if ECB, CBC, or CFB modes are selected; the DEU does not automatically pad messages out to 64-bit blocks. If a data size that is not a multiple of 64 bits is written, a data size error is generated, unless OFB mode is selected. Only bits 58:63 are checked to determine if there is a data size error. Since all upper bits are ignored, the entire message length (in bits) can be written to this register.

This register is cleared when the DEU is reset or re-initialized.

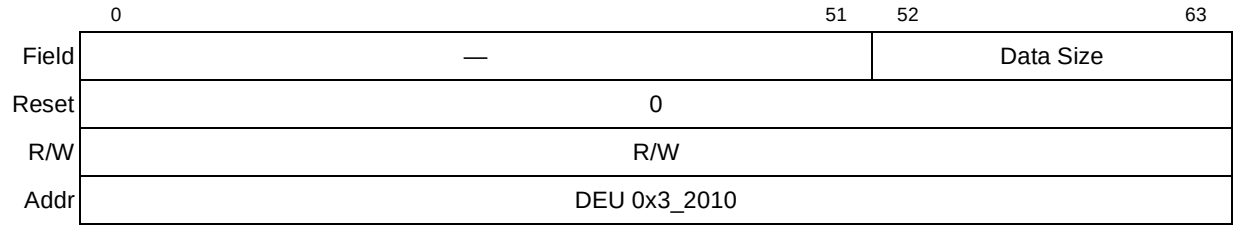


Figure 11-57. DEU Data Size Register

11.7.4.4 DEU Reset Control Register

The DUE reset control register, shown in Figure 11-58, allows three levels reset of just DEU, as defined by the three self-clearing bits:

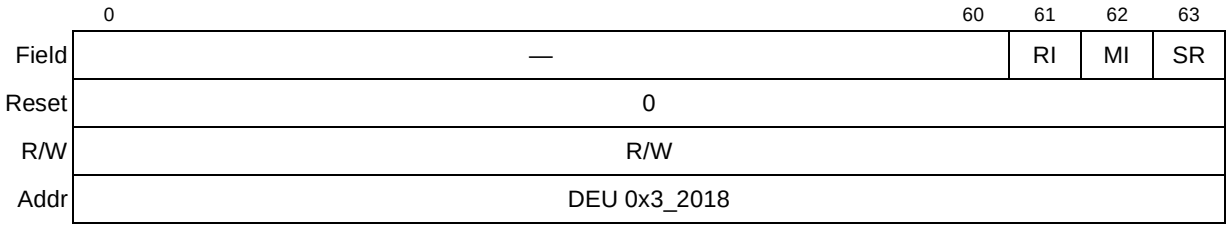


Figure 11-58. DEU Reset Control Register

Table 11-41 describes DEU reset control register fields.

Table 11-50. DEU Reset Control Register Field Descriptions

Bits	Names	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes DEU interrupts signalling done and error to be reset. It further resets the state of the DEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization is nearly the same as software reset, except that the interrupt mask register remains unchanged. this module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the DEU status register 0 Do not reset 1 Reset most of DEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU enters a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the DEU status register indicates when this initialization routine is complete 0 Do not reset 1 Full DEU reset

11.7.4.5 DEU Status Register

The DUE status register, shown in Figure 11-59, contains six fields that reflect the state of DEU internal signals. The DEU status register is read-only. Writing to this location results in address error being reflected in the DEU interrupt status register.

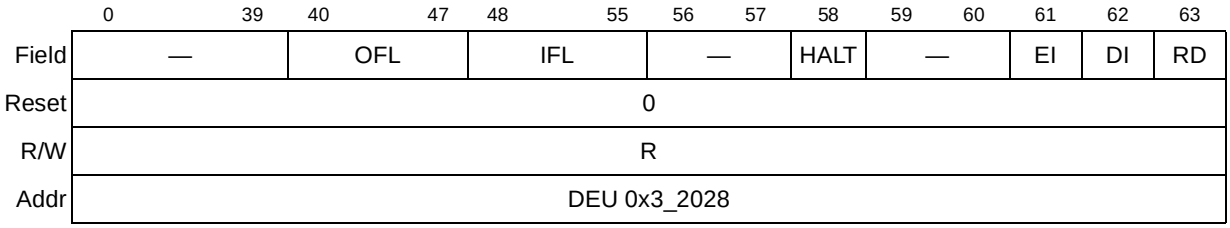


Figure 11-59. DEU Status Register

Table 11-51 describes the DEU status registers fields.

Table 11-51. DEU Status Register

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–55	IFL	The number of dwords currently in the input FIFO
56–57	—	Reserved
58	HALT	Halt. Indicates that the DEU has halted due to an error. 0 DEU not halted 1 DEU halted **Note:** Because the error causing the DEU to stop operating may be masked before reaching the interrupt status register, the DEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 DEU is not signaling error 1 DEU is signaling error
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 DEU is not signaling done 1 DEU is signaling done
63	RD	Reset done. This status bit, when high, indicates that DEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.4.6 DEU Interrupt Status Register

The DEU interrupt status register, shown in Figure 11-60, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the DEU interrupt mask register is zero (see Section 11.7.4.7, "DEU Interrupt Mask Register").

If the DEU interrupt status register is non-zero, the DEU halts and the DEU error interrupt signal is asserted to the controller (see Section 11.6.3.3, "Interrupt Status Register (ISR)"). In addition, if the DEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the Channel Status Register (see Table 11-15) and generates a channel error interrupt to the controller.

If the interrupt status register is written from the host, 1s in the value written are recorded in the interrupt status register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the DEU reset control register.

	0–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Reset	0														
R/W	R/W														
Addr	DEU 0x3_2030														

Figure 11-60. DEU Interrupt Status Register

Table 11-52 describes DEU interrupt status register fields.

Table 11-52. DEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key Parity Error. Defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are checked for parity only if the appropriate DEU mode register bit indicates triple DES. Also, key register 3 is checked only if key size reg = 24. Key register 2 is checked only if key size reg = 16 or 24.) 0 No error detected 1 Key parity error
51	IE	Internal Error. An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error **Note:** This bit is asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Mask Register or by resetting the DEU.
52	ERE	Early Read Error. The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
53	CE	Context Error. A DEU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while DEU was performing encryption. 0 No error detected 1 Context error

Table 11-52. DEU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
54	KSE	Key Size Error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU key size register 0 No error detected 1 Key size error
55	DSE	Data Size Error (DSE): A value was written to the DEU Data Size Register that is not a multiple of 64 bits if ECB, CBC, or CFB mode is selected. If OFB mode is selected, any data size value is permitted. 0 No error detected 1 Data size error
56	ME	Mode error. An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
57	AE	Address error. An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
59	IFE	Input FIFO error. The DEU input FIFO was detected non-empty upon generation of done interrupt. 0 No error detected 1 Input FIFO non-empty error
60	IFU	Input FIFO Underflow. The DEU input FIFO was read while empty. 0 No error detected 1 Input FIFO has had underflow error
61	IFO	Input FIFO Overflow. The DEU input FIFO was pushed while full. 0 No error detected 1 Input FIFO has overflowed **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO Underflow. The DEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	OFO	Output FIFO Overflow. The DEU output FIFO was pushed while full. 0 No error detected 1 Output FIFO has overflowed

11.7.4.7 DEU Interrupt Mask Register

The DEU interrupt mask register, shown in Figure 11-61, controls the result of detected errors. For a given error (as defined in Section 11.7.4.6, “DEU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is ignored; no bit is set in the DEU Interrupt Status Register, and no error interrupt occurs. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0–49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	KPE	IE	ERE	CE	KSE	DSE	ME	AE	OFE	IFE	IFU	IFO	OFU	OFO
Reset	3000														
R/W	R/W														
Addr	DEU 0x3_2038														

Figure 11-61. DEU Interrupt Mask Register

Table 11-53 describes DEU interrupt mask register fields.

Table 11-53. DEU Interrupt Mask Register Field Descriptions

Bits	Name	Description
0–49	—	Reserved
50	KPE	Key Parity Error. The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity error enabled 1 Key parity error disabled
51	IE	Internal Error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The DEU IV Register was read while the DEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context Error. A DEU Key Register, the Key Size Register, the Data Size Register, the Mode Register, or IV Register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for Triple DES) was written to the DEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error (DSE). A value that is not a multiple of 64 bits was written to the DEU Data Size Register if ECB, CBC, or CFB mode is selected. If OFB mode is selected, any data size value is permitted. 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled

Table 11-53. DEU Interrupt Mask Register Field Descriptions (continued)

Bits	Name	Description
58	OFE	Output FIFO Error. The DEU output FIFO was detected non-empty upon write of DEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO Error. The DEU input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	IFU	Input FIFO Underflow. The DEU input FIFO was read while empty. 0 Input FIFO Underflow error enabled 1 Input FIFO Underflow error disabled
61	IFO	Input FIFO Overflow. The DEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input. When operated through host-controlled access, the DEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62	OFU	Output FIFO Underflow. The DEU output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	OFO	Output FIFO Overflow. The DEU output FIFO was pushed while full. 0 Output FIFO Overflow error enabled 1 Output FIFO Overflow error disabled

11.7.4.8 DEU End_of_Message Register

Writing the DEU end-of-message register, shown in Figure 11-62, is a handshake mechanism signalling that no more data is written to the input FIFO. DESA signals done interrupt once the input FIFO is detected empty any time following the write to end-of-message. After the final message block is written to the input FIFO, the end_of_message register must be written. The value in the data size register is used to determine how many bits of the final message block (always 64) is processed. Note that the end_of_message register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned and no error is generated. Writing to this register is merely a trigger causing the DEU to process the final block of a message, allowing it to signal done interrupt.

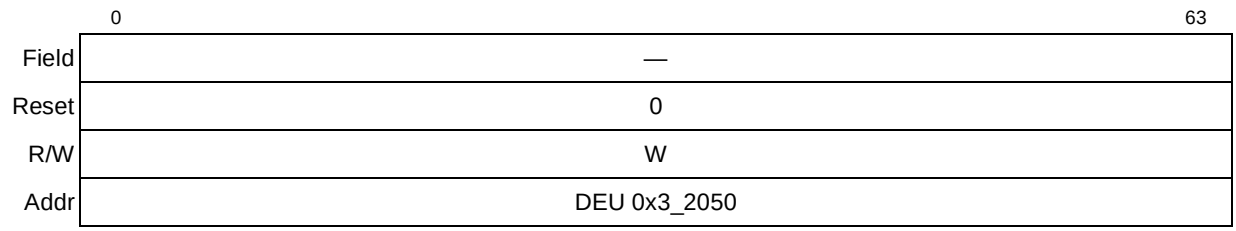


Figure 11-62. DEU End_of_Message Register

11.7.4.9 DEU IV Register

For CBC mode, the initialization vector is written to and read from the DEU IV register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

11.7.4.10 DEU Key Registers

The DEU uses three write-only key registers, K1, K2, and K3, to perform encryption and decryption. In Single DES mode, only K1 may be written. The value written to K1 is simultaneously written to K3, auto-enabling the DEU for 112-bit Triple DES if the key size register indicates 2 key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit Triple DES, K1 must be written first, followed by the write of K2, then K3.

Reading any of these memory locations generates an address error interrupt.

11.7.4.11 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For host-controlled operation, a write to anywhere in the DEU FIFO address space enqueues data to the DEU input FIFO, and a read from anywhere in the DEU FIFO address space dequeues data from the DEU output FIFO.

Writes to the input FIFO go first to a staging register which can be written by byte, word (4 bytes), or dword (8 bytes). When all 8 bytes of the staging register have been written, the entire dword is automatically enqueued into the FIFO. If any byte is written twice between enqueues, it causes an error interrupt of type AE from the EU. Since the DEU data length should always be a multiple of 8 bytes, the last write should complete a dword. However, if there is any partial dword in the staging register when the DEU End of Message Register is written, the partial dword is automatically padded with zeros to a full 8 bytes and enqueued to the input FIFO.

The output FIFO is readable by byte, word, or dword. When all 8 bytes of the head dword have been read, that dword is automatically dequeued from the FIFO so that the next dword (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Overflows and underflows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

11.7.5 Kasumi Execution Unit (KEU)

This section contains details about the Kasumi execution unit (KEU), including modes of operation, status and control registers, and FIFOs. The KEU has been designed to support the F8 confidentiality function of the 3GPP, GSM A5/3, EDGE A5/3, and GPRS GEA3 algorithms. The KEU also supports the 3GPP F9 integrity function.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purpose. In typical operation, the KEU is used through channel-controlled access, which means that most reads and writes of the KEU registers are directed by the SEC channels. Driver

software performs host-controlled register accesses only on a few registers for initial configuration and error handling.

This execution unit (EU) includes an ICV checking feature, which means it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the host either through interrupt or by using a writeback of EU status fields into the host memory, but not using both methods at the same time.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the channel configuration register (for more information, see Section 11.5.4.1, “Channel Configuration Register (CCR)”), and mask the ICE bit in the interrupt mask register (Section 11.7.5.7, “KEU Interrupt Mask Register (KEUIMR)”). In this case the normal DONE signal (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the interrupt mask register and turn off the IWSE and AWSE bits in the channel configuration register. If there is no ICV mismatch, the normal DONE signal (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an ERROR interrupt signal to the host, but no DONE interrupt signal or writeback.

11.7.5.1 KEU Mode Register (KEUMR)

The KEU mode register, shown in Figure 11-63, contains several bits which are used to program the KEU. The mode register is cleared when the KEU is reset or re-initialized. Setting a reserved mode bit generates a data error. Setting both the GSM and EDGE bits to one generates a data error. If the KEU mode register is modified during processing, a context error is generated.

	0 – 55	56	57	58	59	60	61	62 – 63
Field	—	GSM	CICV	EDGE	PE	INT	—	ALG
Reset	0							
R/W	R/W							
Addr	KEU 0x3_E000							

Figure 11-63. KEU Mode Register

Table 11-54 describes the KEU mode register fields.

Table 11-54. KEU Mode Register Field Descriptions

Bits	Name	Description
0–55	—	Reserved
56	GSM	Select GSM A5/3 blocks 0 GSM A5/3 blocks not selected 1 GSM A5/3 blocks selected **Note 1:** For GSM A5/3, Two 114-bit blocks are required to be produced each 4.615mS slot. If GSM = 1, the first read of the output FIFO retrieves the first 64 bits of block 1. The second read of the output FIFO retrieves the next 50 bits of block 1 (the remaining bits of this 64-bit word are cleared). The third read of the output FIFO retrieves the first 64 bits of block 2, while a fourth read of the output FIFO retrieves the next 50 bits of block 2 (the remaining bits of this 64-bit word are cleared). **Note 2:** If GSM = 0, 228 contiguous bits may be read with successive reads of the output FIFO. In this case the host (application) is responsible for handling the A5/3 block formatting. **Note 3:** If GSM is set to 1, while EDGE = 1, an interrupt/error is generated.
57	CICV	Compare integrity check values. 0 Normal operation; no ICV comparison. 1 After the ICV is computed, compare it to the data in the KEU's ICV_In register. If the ICVs do not match, send an error interrupt to the channel. Only applicable when the ALG field is set to a function that uses F9.
58	EDGE	Select EDGE A5/3 blocks 0 EDGE A5/3 blocks not selected 1 EDGE A5/3 blocks selected **Note 1:** For EDGE A5/3, two 348-bit blocks are required to be produced each 4.615mS slot. If EDGE = 1, the first five reads of the output FIFO retrieve the first 320 bits of block 1. The sixth read of the output FIFO retrieves the final 28 bits of block 1 (the remaining bits of the sixth 64-bit word are cleared). The next five reads of the output FIFO retrieve the first 320 bits of block 2. The following read of the output FIFO retrieves the final 28 bits of block 2 (the remaining bits of this 64-bit word are cleared). **Note 2:** If EDGE = 0, 696 contiguous bits may be read with successive reads of the output FIFO. In this case the host (application) is responsible for handling the A5/3 block formatting. **Note 3:** If EDGE is set to 1, whilst GSM = 1, an interrupt/error is generated.
59	PE	Process end of message. Enables final processing of last message block (F9 only). 0 Prevent final block processing (message incomplete) 1 Enable final block processing (message complete) **Note:** As the use of PE is tightly coupled with the use of the KEU data size register, see Section 11.7.5.3, "KEU Data Size Register (KEUDSR)" for more details.
60	INT	Initialization. Enables initialization for a new message. 0 Prevent initialization 1 Enable initialization **Note:** For F8 or F9 operations, if the 3G frame (or message) is being processed through a single descriptor, the Initialization bit should be set. If the frame is split across multiple descriptors, this bit should only be set in the descriptor that processes the first block of the message.
61	—	Reserved
62–63	ALG	Algorithm selection. Specifies the functions to perform. 00 Perform F8 function only 01 Reserved 10 Perform F9 function only 11 Reserved

11.7.5.2 KEU Key Size Register (KEUKSR)

The KEU key size register, shown in Figure 11-64, stores the number of bytes in the key. It should be set to 16 bytes. This register is cleared when the KEU is reset or re-initialized. If a key size is specified that does not match the selected algorithm(s), an illegal key size error is generated.

	0 — 51	52 — 63
Field	—	Key Size (Bytes)
Reset	0	
R/W	R/W	
Addr	KEU 0x3_E008	

Figure 11-64. KEU Key Size Register

11.7.5.3 KEU Data Size Register (KEUDSR)

The KEU data size register, shown in Figure 11-65, stores the number of bits to process in the final message word. As Kasumi allows for bit level granularity for encryption/decryption, there are no illegal data sizes. The proper bit length of the message must be written to notify the KEU of any padding performed by the host. This register is cleared when the KEU is reset or re-initialized.

Writing to this register signals the KEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error is generated.

Kasumi processing is determined by both the data size and the setting of the process end of message (PE) bit in the KEU mode register. The PE bit determines how the final block of message data is processed. In typical descriptor based operations, the data size register is loaded with values which are an integral number of bytes. For descriptor based F8 operations, the software is responsible for padding the data to the next byte boundary, and for removing this padding from the KEU's output. The output of the KEU is an integral number of bytes, as specified in the descriptor, automatically truncating any internal padding required to process the final 64 bits message block. As the KEU can infer when it has reached the final 64 bits message block from the length fields in the descriptor, setting the PE bit through the descriptor header is not required. While performing F8 operations, the KEU's output is the same irrespective of the setting of the PE bit.

For the descriptor based F9 operations, the PE bit must be set through the descriptor header whenever the descriptor is being used to process the final message block. This causes the KEU to automatically pad the final block before calculating the F9 MAC.

	0 — 51	52 — 63
Field	Reserved	Data Size (bits)
Reset	0	
R/W	R/W	
Addr	KEU 0xE010	

Figure 11-65. KEU Data Size Register

The details of data size register and the PE interaction are more relevant when operating the KEU in direct access (slave) mode, rather than using descriptors. Note that operating the KEU in direct access mode is not recommended other than for debug test cases, and the information provided here regarding the use of data size and PE in direct access mode is to explain the behaviors that might be encountered in direct access debug operations.

PE has the following effects for ***direct access mode F8 operations***, using the example of a 64-bit F8 keystream '0x1234567890abcdef' and the data size register containing '0x0a' (10 bits = 1 byte + 2 bits):

- PE = 0: The final ten message bits are eXclusive-ORed (XORed) with the entire last 64-bit block of keystream, which produces 54 additional non-zero bits after the end of the real message. These additional 54 bits must be removed by the software.
- PE = 1: The final ten message bits are eXclusive-ORed (XORed) with ten bits of keystream'0x120', and no additional bits of the false message are produced.

For ***direct access mode F9 operations***, assertion of PE enables F9 algorithm-specified padding per the value in the data size register. If this direct access mode operation includes the final message block, then PE must be set in the KEU mode register as follows:

- PE = 0: The KEU does not perform padding. For the KEU to perform the correct computation, the last block of the message provided must be a complete 64-bit block, and the value written into the KEU data size register must indicate a multiple of 64 bits.
- PE = 1: The KEU pads the final 64 bits of the message as specified in the 3GPP F9 algorithm. The communication direction (CD) bit (see Table 11-59 for KEU IV_1 Register Fields Description) and '1' is appended to the end of the message. The result is zero-padded to 64 bits.

  For example, if the last block is xF81A_0000_0000_0000 (big-endian) and data size contains 0x0F (15 bits = 1 byte + 7 bits), the word (0xF81A = 1111_1000_0001_1010), the most-significant 15 bits (underlined) are padded left to right as 1111_1000_0001_101$_1000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_000 0, where $ is the value of the CD bit in the mode register.

11.7.5.4 KEU Reset Control Register (KEURCR)

The KEU reset control register, shown in Figure 11-66, allows three levels of reset of the KEU, as defined by the three self-clearing bits.

	0 — 60	61	62	63
Field	—	CLI	RI	SR
Reset	0			
R/W	R/W			
Addr	KEU 0xE018			

Figure 11-66. KEU Reset Control Register

Table 11-55 describes the KEU reset control register fields.

Table 11-55. KEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	CLI	Clear Interrupts. Writing a 1 to this bit causes the KEU interrupt signals—DONE and ERROR—to be reset. It further resets the state of the KEU interrupt status register. 0 Normal operation 1 Clear interrupts and the KEU interrupt status register
62	RI	Re- Initialization. It is same as software reset (SR), except that the interrupt mask register remains unchanged. Completion of re-initialization is indicated by the RESET_DONE bit in the KEU status register. 0 Normal operation 1 Re-initialize the KEU
63	SR	Software reset. Functionally equivalent to hardware reset (the $\overline{\text{RESET}}$ signal), but only for the KEU. All registers and internal state are returned to their defined reset state. Upon negation of the SR bit, the KEU enters a routine to perform proper initialization of the parameter memories. The reset done (RD) bit in the KEU status register indicates when this initialization is complete 0 Normal operation 1 Full KEU reset

11.7.5.5 KEU Status Register (KEUSR)

The KEU status register, shown in Figure 11-67, is a read-only register that reflects the state of six status outputs. While writing to this location, an address error is reflected in the KEU interrupt status register.

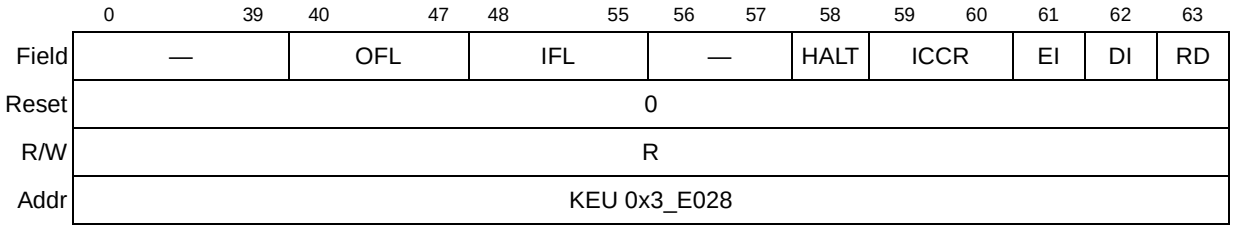


Figure 11-67. KEU Status Register

Table 11-56 describes the KEU status register fields.

Table 11-56. KEU Status Register Fields Description

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	Output FIFO level. The number of dwords currently in the output FIFO.
48–55	IFL	Input FIFO level. The number of dwords currently in the input FIFO.
56–57	—	Reserved

Table 11-56. KEU Status Register Fields Description (continued)

Bits	Name	Description
58	HALT	Indicates when the KEU core has halted due to an error. 0 KEU not halted 1 KEU core halted (must be reset/re-initialized) **Note:** As the error causing the KEU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	ICCR	Integrity check comparison result. 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved **Note:** A passed or failed result is generated only if ICV checking is enabled and the algorithm selected is F9.
61	EI	Error interrupt. Reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 KEU is not signaling error 1 KEU is signaling error
62	DI	Done interrupt. Reflects the state of the DONE interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 Processing not done 1 All bytes processed
63	RD	Reset done. Indicates when the KEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done

11.7.5.6 KEU Interrupt Status Register (KEUISR)

The KEU interrupt status register, shown in Figure 11-68, tracks the state of possible errors, provided those errors are not masked, through the KEU interrupt control register.

The KEU interrupt status register indicates the unmasked errors that have occurred and have generated the ERROR interrupt signals to the channel. Each bit in this register can only be set if the corresponding bit of the KEU interrupt mask register is zero (see Section 11.7.5.7, “KEU Interrupt Mask Register (KEUIMR)”).

If the KEU interrupt status register is non-zero, the KEU halts and the KEU ERROR interrupt signal is asserted to the controller (see Section 11.6.3.3, “Interrupt Status Register (ISR)”). In addition, if the KEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which the EU is assigned. The EU error then appears in the bit 55 of the channel pointer status register (for more information, see Table 11-15 on page 11-46) and generates a channel error interrupt to the controller.

This register can be cleared by setting the RI bit of the KEU reset control register. If a KEU error is reported by the channel while operating in descriptor mode, the user can rely on the channel to clear the KEU interrupt by writing the Continue bit in the channel configuration register (for more information, see Section 11.5.4.1, “Channel Configuration Register (CCR)”). Writing a 1 to any error bit in this register causes the KEU to signal the corresponding error, unless the associated error has been masked in the KEU interrupt mask register.

	0–48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	DE	AE	OFE	IFE	—	IFO	OFU	—
Reset	0															
R/W	R/W															
Addr	KEU 0x3_E030															

Figure 11-68. KEU Interrupt Status Register

Table 11-57 describes the KEU interrupt status register signals.

Table 11-57. KEU Interrupt Status Register Signals Description

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 No error detected 1 Integrity check error detected. An ICV check was performed on an F9 result and the supplied ICV did not match the one computed by the KEU.
50	—	Reserved
51	IE	Internal error. An internal processing error was detected while the KEU was processing. 0 No error detected 1 Internal error This bit is set any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the interrupt mask register or by resetting the KEU.
52	ERE	Early read error. A KEU context or IV register was read while the KEU was processing. 0 No error detected 1 Early read error
53	CE	Context error. A KEU key register, the key size register, the data size register, the mode register, or IV register was modified while the KEU was processing. 0 No error detected 1 Context error
54	KSE	Key size error. An inappropriate value (not 16 or 32 bytes) was written to the KEU key size register. 0 No error detected 1 Key size error
55	DSE	Data size error. A value was written to the KEU data size register that is greater than 64 bits. 0 No error detected 1 Data size error
56	DE	Data error. Invalid data was written to a register or a reserved mode bit was set. 0 Valid data 1 Reserved or invalid mode selected
57	AE	Address error. An illegal read or write address was detected within the KEU address space. 0 No error detected 1 Address error
58	OFE	Output FIFO error. The KEU output FIFO was non-empty upon write of the KEU data size register. 0 No error detected 1 Output FIFO non-empty error

Table 11-57. KEU Interrupt Status Register Signals Description (continued)

Bits	Name	Description
59	IFE	Input FIFO error. The KEU input FIFO was non-empty upon generation of the done interrupt. 0 No error detected 1 Input FIFO non-empty error
60	—	Reserved
61	IFO	Input FIFO overflow. The KEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed
62	OFU	Output FIFO underflow. The KEU output FIFO was read while empty. 0 No error detected 1 Output FIFO has underflow error
63	—	Reserved

11.7.5.7 KEU Interrupt Mask Register (KEUIMR)

The KEU interrupt mask register, shown in Figure 11-69, controls the result of detected errors. For a given error (as defined in Section 11.7.5.6, “KEU Interrupt Status Register (KEUISR)”), if the corresponding bit in this register is set, the error is ignored; no error interrupt occurs and the KEU interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the KEU interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0 48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	DE	AE	OFE	IFE	—	IFO	OFU	—
Reset	0	0	0	1	0											
R/W	R/W															
Addr	KEU 0x3_E038															

Figure 11-69. KEU Interrupt Mask Register

Table 11-58 describes the KEU interrupt mask register fields.

Table 11-58. KEU Interrupt Mask Register Fields Description

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity check error. 0 ICV check error enabled. WARNING: Do not enable this EU status writeback (see bits IWSE and AWSE in Section 11.5.4.1, “Channel Configuration Register (CCR)” is used. 1 ICV check error disabled
50	—	Reserved
51	IE	Internal error. An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled

Table 11-58. KEU Interrupt Mask Register Fields Description (continued)

Bits	Name	Description
52	ERE	Early read error. A KEU context or IV register was read while the KEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
53	CE	Context error. A KEU key register, the key size register, data size register, mode register, or IV register was modified while the KEU was performing encryption. 0 Context error enabled 1 Context error disabled
54	KSE	Key size error. An inappropriate value (not 16 or 32 bytes) was written to the KEU key size register. 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error. Indicates that the number of bits to process is out of range. 0 Data size error enabled 1 Data size error disabled
56	DE	Data error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 Data error enabled 1 Data error disabled
57	AE	Address error. An illegal read or write address was detected within the KEU address space. 0 Address error enabled 1 Address error disabled
58	OFE	Output FIFO error. The KEU output FIFO was detected non-empty upon write of the KEU data size register. 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
59	IFE	Input FIFO error. The KEU input FIFO was detected non-empty upon generation of done interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
60	—	Reserved
61	IFO	Input FIFO overflow. The KEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62	OFU	Output FIFO underflow. The KEU output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

11.7.5.8 KEU Data Out Register (F9 MAC) (KEUDOR)

Following a done interrupt, the read-only KEU data out register, shown in Figure 11-70, holds the F9 message authentication code. A 64-bit value is returned. This value may be truncated to 32 bits for some applications. While writing to this location, an address error is reflected in the KEU interrupt status register.

	0	63
Field	KEU Data Out Register (F9 MAC)	
Reset	0	
R/W	R	
Addr	KEU 0x3_E048	

Figure 11-70. KEU Data Out Register (F9 MAC)

NOTE

According to the ETSI/SAGE 3GPP specification for F9 (version 1.2), only 32 bits of the final MAC are used. This corresponds to the lower 4 bytes of the KEU data out register.

11.7.5.9 KEU End of Message Register (KEUEMR)

The KEU end of message register, shown in Figure 11-71, is used to signal to the KEU that the final message block has been written to the input FIFO. Writing to this register causes the KEU to process the final block of a message, allowing it to the interrupt signal, DONE.

When processing the last block, the value in the data size register determines how many bits of the final message word (1–64) are processed. The value written to this register does not matter. A read of this register always returns a zero value.

	0	63
Field	—	
Reset	0	
R/W	R/W	
Addr	KEU 0x3_E050	

Figure 11-71. KEU End of Message Register

11.7.5.10 KEU IV_1 Register (KEUIV1)

The KEU IV_1 register, shown in Figure 11-72, is a general purpose IV register and is used during the initialization phase of the F8 algorithms for 3GPP, GSM A5/3, EDGE A5/3, GPRS GEA3, and F9 algorithm for 3GPP. The appropriate value as defined by the standards for each algorithm must be written before a new message is started.

After the initialization phase has been completed, the KEU IV_1 register is no longer used for the remainder of F8 processing. However, if 3GPP F9 is selected because the KEU IV_1 register contains the direction bit as defined by the 3GPP standard, the KEU IV_1 register must be written back during context switches to complete the generation of the 3GPP MAC.

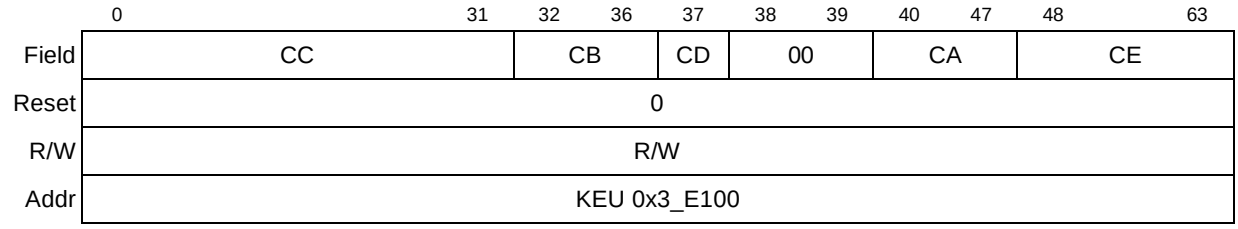


Figure 11-72. KEU IV_1 Register

Table 11-59 describes the KEU IV_1 register fields.

Table 11-59. KEU IV_1 Register Fields Description

Bits	Field	3GPP Definition	GSM—A5/3 Definition	EDGE—A5/3 Definition	GPRS - GEA3 Definition
0–31	CC	Count	Count	0000000000 | | Count	Frame dependent input value (32-bits)
32–36	CB	Bearer	00000	00000	00000
37	CD	Direction bit	0	0	0
38–39	0	00	00	00	00
40–47	CA	00000000	00001111	11110000	11111111
48–63	CE	0000000000000000	0000000000000000	0000000000000000	0000000000000000

The following figure shows how the KEU IV_1 register can be differentiated for different applications.

	0–31	32–36	37	38–39	40–47	48–63
3GPP (F8)	Count	Bearer	Dir.	00	00000000	0000000000000000
GSM (A5/3)	Count	00000	0	00	00001111	0000000000000000
EDGE (A5/3)	0000000000 || Count	00000	0	00	11110000	0000000000000000
GPRS (GEA3)	32 bit Frame Dependent Input Value	00000	0	00	11111111	0000000000000000

NOTE

It is the responsibility of the user to ensure that fields of the KEU IV_1 register are programmed correctly in accordance with the algorithm selected.

11.7.5.11 KEU ICV_In Register (KEUICV)

If ICV checking is required, then the value to be compared with the computed F9 MAC value must be written to the KEU ICV_In register before data size is written. As the KEU ICV_In register is in between

IV_1 and IV_2, any descriptor operation that loads IV_2 must also load ICV_In. If CICV = 0, the ICV_In register should be loaded with 0x0000_0000_0000_0000.

11.7.5.12 KEU IV_2 Register (Fresh) (KEUIV2)

The KEU IV_2 register, shown in Figure 11-73, holds the F9 value, Fresh, which is used during the initialization phase of the 3GPP F9 algorithm. This value is ignored when the F8 algorithm is selected. The Fresh value must be written to bits 0:31 of the KEU IV_2 register before a new message to be processed with 3GPP F9 is started. After the initialization phase has been completed, the KEU IV_2 register is no longer used during message processing. The KEU IV_2 register need not be written during context switches.

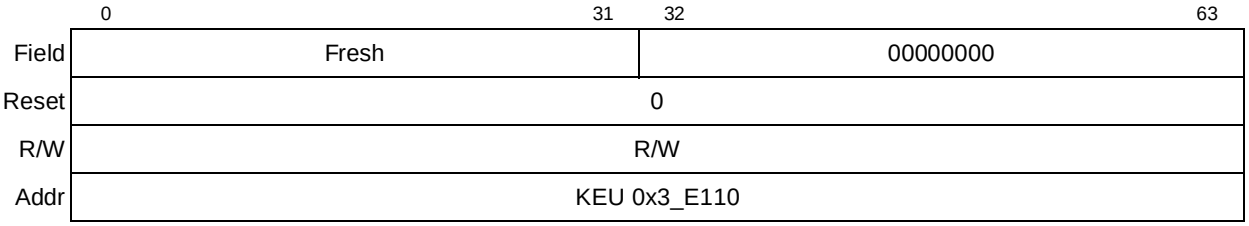


Figure 11-73. KEU IV_2 Register (Fresh)

11.7.5.13 KEU Context Data Registers (KEUC*n*)

The KEU includes six 64-bit KEU context data registers that store the running context used to process a message. The KEU context data registers must be read when changing context and are restored to their original values to resume processing of a partial message. For F8 and 3GPP F9 modes, all 64-bit KEU context data registers must be read to retrieve context, and all six registers must be written back to restore context. The context must be written prior to the key data. If the any of the KEU context data registers are written during message processing, a context error is generated. All KEU context data registers are cleared when a hard/soft reset or initialization is performed.

NOTE

For descriptor operation, if the entire context is unloaded for later reuse, the context data size must be 72 bytes, and the output consists of KEU IV_1, KEU ICV_In, KEU IV_2, and six KEU context data registers. For operations performing processing of partial messages, if the context is unloaded, the PE bit in the KEU mode register must not be set. Also, for partial message processing, if the context is reloaded, the INT bit in the KEU mode register must not be set.

11.7.5.14 KEU Key Data Registers_1 and _2 (Confidentiality Key) (KEUKD*n*)

The first two KEU key data registers, shown in Figure 11-74 and Figure 11-75, together hold one 128-bit key that is used for F8 encryption/decryption. The KEU key data register_1, (CK-high), holds the first 8 bytes (1–8). The KEU key data register_2, (CK-low), holds the second 8 bytes (9–16). The KEU key data registers must be written before message processing begins and cannot be written while the block is

processing data, or else, a context error occurs. While reading from either of these registers, an address error is reflected in the KEU interrupt status register.

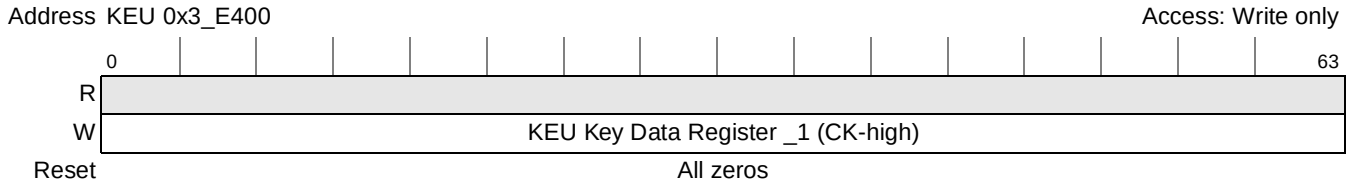


Figure 11-74. KEU Key Data Register_1 (CK-High)

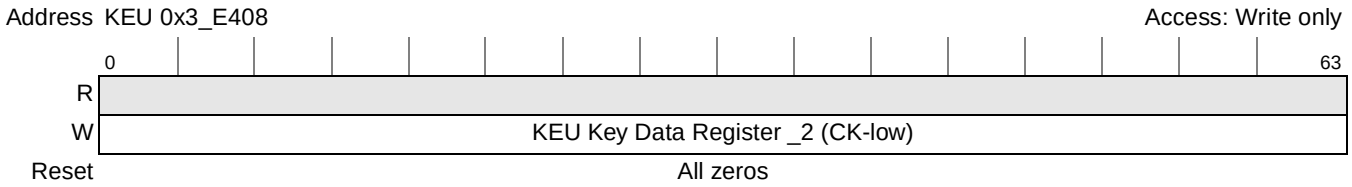


Figure 11-75. KEU Key Data Register_2 (CK-Low)

11.7.5.15 KEU Key Data Registers _3 and _4 (Integrity Key) (KEUKD*n*)

The third and fourth KEU key data registers, shown in Figure 11-76 and Figure 11-77, together hold one 128-bit key that is used for F9 message authentication. The KEU key data register_3, (IK-high), holds the first 8 bytes (1–8). The KEU key data register_4, (IK-low), holds the second 8 bytes (9–16). The KEU key data registers must be written before message processing begins and cannot be written while the block is processing data, or else, a context error occurs.

If the mode, F9 only, is set, the integrity key data may be optionally written to the KEU key data registers_1 and KEU key data registers_2. This eliminates the need for the host to offset from the base key address to write to the KEU key data registers_3 and KEU key data registers_4 while using the KEU exclusively for the F9 integrity function.

While reading from either of these registers, an address error is reflected in the KEU interrupt status register.

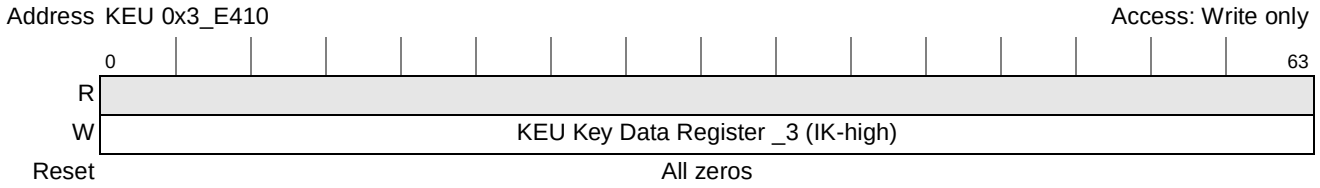


Figure 11-76. KEU Key Data Register_3 (IK-High)

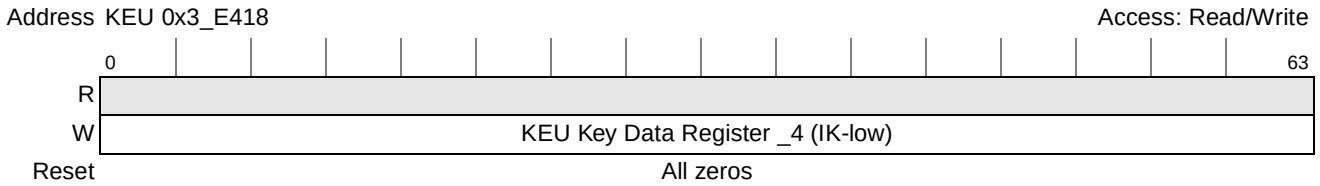


Figure 11-77. KEU Key Data Register_4 (IK-Low)

11.7.5.16 KEU FIFOs

The KEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. Normally, the channels control all access to these FIFOs. For host-controlled operation, a write to anywhere in the KEU FIFO address space en-queues data to the KEU input FIFO, and a read from anywhere in the KEU FIFO address space de-queues data from the KEU output FIFO.

A write to the input FIFO go first to a staging register, which can be written by byte, word (4 bytes), or dword (8 bytes). When all 8 bytes of the staging register have been written, the entire dword is automatically en-queued into the FIFO. If any byte is written twice between en-queues, it causes an error interrupt of type AE from the EU. When writing the last portion of data, it is not necessary to write all 8 bytes. The last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the KEU end of message register is written.

The output FIFO is readable by byte, word, or dword. When all 8 bytes of the head dword are read, the dword is automatically de-queued from the FIFO so that the next dword (if any) becomes available for reading. If any byte is read twice between de-queues, it causes an error interrupt of type AE from the EU.

The overflows and underflows caused by reading or writing the KEU FIFOs are reflected in the KEU interrupt status register.

The KEU fetches data 64 bits at a time from the KEU Input FIFO. During F8 processing, the input data is XORed with the generated keystream and the results are placed in the KEU output FIFO. During F9 processing, the input data is hashed with the integrity key and the resulting MAC is placed in the KEU data out register. The output size is the same as the input size.

11.7.6 Message Digest Execution Unit (MDEU)

This section contains details about the Message Digest Execution Unit (MDEU), including modes of operation, status and control registers, and FIFO.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the MDEU is used through channel-controlled access, which means that most reads and writes of MDEU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.6.1 ICV Checking

This EU includes an ICV checking feature, that is, it can generate an ICV and compare it to another supplied ICV. The pass/fail result of this ICV check can be returned to the host either via interrupt by a writeback of EU status fields into host memory, but not by both methods at once.

To signal the ICV checking result by status writeback, turn on either the IWSE bit or AWSE bit in the Channel Configuration Register (see Section 11.5.4.1, “Channel Configuration Register (CCR)”), and mask the ICE bit in the Interrupt Mask Register (Section 11.7.6.9, “MDEU Interrupt Mask Register”). In this case the normal done signalling (by interrupt or writeback) is undisturbed.

To signal the ICV checking result by interrupt, unmask the ICE bit in the Interrupt Mask Register and turn off the IWSE and AWSE bits in the Channel Configuration Register. If there is no ICV mismatch, then the normal done signalling (by interrupt or writeback) occurs. When there is an ICV mismatch, there is an error interrupt to the host, but no channel done interrupt or writeback.

11.7.6.2 MDEU Mode Register

The MDEU mode register, shown in Figure 11-78 and Figure 11-79, is used to program the function of the MDEU. For normal operation through a channel, bits 56–63 of this register are specified by the user through the MODE0 or MODE1 field of the descriptor header. The remaining two bits are supplied by the channel and thus are not under direct user control.

The two bits supplied by the channel are bits that control the meanings of other mode register fields. They are the MDEU_B bit, and the NEW bit.

The MDEU_B bit determines which of two sets of algorithms are available through the ALG bits. The two sets of algorithms are referred to as the MDEU-A set (MD5, SHA-1, SHA-224, and SHA-256) and the MDEU-B set (SHA-224, SHA-256, SHA-384, and SHA-512). MDEU_B = 0 selects the MDEU-A set, and MDEU_B = 1 selects the MDEU-B set. In channel-controlled operation, the MDEU_B mode bit is supplied by the channel, based on the EU_SEL field of the descriptor header, where the user can choose MDEU-A or MDEU-B (see Table 11-6).

The NEW bit determines the configuration of other mode register fields as shown in Figure 11-78 and Figure 11-79. The “new” configuration (NEW=1) is used only by TLS/SSL descriptor types (1000_1, 1001_1). The old configuration (NEW=0) is used by all other descriptor types. The old configuration is the same as the one used in SEC 2.0, except for the CICV and SMAC bits. When MDEU is configured by the Polychannel, the value of NEW is determined by the descriptor type field of the descriptor header.

The mode register is cleared when the MDEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated.

	0 – 50	51	52 – 53	54	55	56	57	58	59	60	61	62 – 63
Field	—	MDEU_B	—	NEW=0	—	CONT	CICV	SMAC	INIT	HMAC	PD	ALG
Reset	0											
R/W	R/W											
Addr	MDEU 0x3_6000											

Figure 11-78. MDEU Mode Register in Old Configuration

Table 11-60 describes MDEU mode register fields in old configuration.

Table 11-60. MDEU Mode Register in Old Configuration

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–50	—	Reserved

Table 11-60. MDEU Mode Register in Old Configuration (continued)

Bits	Name	Description
51	MDEU_B	Selects which algorithms are enabled by the ALG bits. 0 MDEU-A enables selection between SHA-1, SHA-256, MD5, and SHA-224 1 MDEU-B enables selection between SHA-384, SHA-256, SHA-512, and SHA-224.
52–53	—	Reserved, must be cleared.
54	NEW (=0)	Determines the configuration of the MDEU Mode Register. This table shows the configuration for NEW=0.
55	—	Reserved, must be cleared.
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue: Most operations require this bit to be cleared. It is set only when the data to be hashed is spread across multiple descriptors. The value programmed in PD must be opposite to the value in this bit. 0 Do autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash is continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare Integrity Check Values: 0 Normal operation; no ICV checking. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV Size Register. Only applicable to descriptor types that provide for reading an ICV in value.
58	SMAC	Specifies whether to perform an SSL-MAC operation: 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.
59	INIT	Initialization Bit: Most operations require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit must be 0 on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation: 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	PD	This bit must be programmed opposite to the CONT bit.
62–63	ALG	Message Digest algorithm selection 00 if MDEU-B, then SHA-384. If MDEU-A, then SHA-160 algorithm (full name for SHA-1) 01 SHA-256 algorithm 10 if MDEU-B, then SHA-512. If MDEU-A, then MD5 algorithm 11 SHA-224 algorithm

	0–50	51	52	53	54	55	56	57	58	59	60	61	62–63
Field	—	MDEU_B	—	STIB	NEW=1	—	CONT	CICV	SMAC	INIT	HMAC	EALG	ALG
Reset	0												
R/W	R/W												
Addr	MDEU 0x3_6000												

Figure 11-79. MDEU Mode Register in New Configuration

Table 11-61 describes MDEU Mode Register fields in new configuration.

Table 11-61. MDEU Mode Register in New Configuration

Bits	Name	Description
The following bits are described for information only. They are not under direct user control.		
0–50	—	Reserved
51	MDEU_B	Selects which algorithms are enabled by the ALG bits. 0 MDEU-A enables selection between SHA-1, SHA-256, MD5, and SHA-224 1 MDEU-B enables selection between SHA-384, SHA-256, SHA-512, and SHA-224.
52	—	Reserved, must be cleared.
53	STIB	SSL/TLS inbound, block cipher: 0 Normal operation. 1 Special operation only for SSL/TLS inbound, block cipher. Upon receiving End_of_message, the MDEU performs a calculation involving the last valid byte of data written into its input FIFO (which is Pad Length) to compute a final data size. The MDEU then processes the amount of data specified by this data size, and completes the message digest.
54	NEW (=1)	Determines the configuration of the MDEU Mode Register. This table shows the configuration for NEW=1.
55	—	Reserved, must be cleared.
The following bits are controlled through the MODE0 or MODE1 fields of the descriptor header.		
56	CONT	Continue: Most operations require this bit to be cleared. Set only when the data to be hashed is spread across multiple descriptors. 0 Do autopadding and complete the message digest. Used when the entire hash is performed with one descriptor, or on the last of a sequence of descriptors. 1 This hash is continued in a subsequent descriptor. Do not autopad and do not complete the message digest.
57	CICV	Compare Integrity Check Values: 0 Normal operation; no ICV checking. 1 After the message digest (ICV) is computed, compare it to the data in the MDEU's input FIFO. If the ICVs do not match, send an error interrupt to the channel. The number of bytes to be compared is given by the ICV Size Register.
58	SMAC	Specifies whether to perform an SSL-MAC operation: 0 Normal operation 1 Perform an SSL3.0 MAC operation. This requires a key and key length. If this is set then the HMAC bit should be 0.

Table 11-61. MDEU Mode Register in New Configuration (continued)

Bits	Name	Description
59	INIT	Initialization Bit: Most operations require this bit to be set. Cleared only for operations that load context from a known intermediate hash value. 0 Do not initialize digest registers. In this case the registers must be loaded from a hash context pointer in the descriptor. When the data to be hashed is spread across multiple descriptors, this bit is set on all but the first descriptor. 1 Do an algorithm-specific initialization of the digest registers.
60	HMAC	Specifies whether to perform an HMAC operation: 0 Normal operation 1 Perform an HMAC operation. This requires a key and key length. If this is set then the SMAC bit should be 0.
61	EALG	The EALG (Extended Algorithm bit) and ALG (Algorithm) bits together specify the message digest algorithm, as follows: 000 if MDEU-B, then SHA-384. If MDEU-A, then SHA-160 algorithm (full name for SHA-1) 001 SHA-256 algorithm 010 if MDEU-B, then SHA-512. If MDEU-A, then MD5 algorithm 011 SHA-224 algorithm others: Reserved
62–63	ALG	

11.7.6.3 Recommended Settings for MDEU Mode Register

The most common task likely to be executed via the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPsec, and TLS. The SSL 3.0 protocol uses a slightly different SSL-MAC. If an HMAC or SSL-MAC is to be performed using a single descriptor (with the MDEU acting as sole or secondary EU), the following mode register bit settings should be used:

Table 11-62. Mode Register—HMAC or SSL-MAC Generated by Single Descriptor

Bits	Field	Value	
		for HMAC	for SSL-MAC
56	CONT	0 (off)	0 (off)
58	SMAC	0(on)	1(on)
59	INIT	1(on)	1(on)
60	HMAC	1(on)	0(on)

To generate an HMAC for a message that is spread across a sequence of descriptors, the following mode register bit settings should be used:

Table 11-63. Mode Register—HMAC Generated Across a Sequence of Descriptors

Bits	Field	Value		
		First Descriptor	Middle Descriptor(s)	Final Descriptor
56	CONT	1 (on)	1 (on)	0 (off)
59	INIT	1 (on)	0 (off)	0 (off)
60	HMAC	1 (on)	0 (off)	1 (on)

All descriptors other than the final descriptor must output the intermediate message digest for the next descriptor to reload as MDEU context.

SSL-MAC operations cannot be spread across a sequence of descriptors.

Additional information on descriptors can be found in Section 11.3, “Descriptor Overview.”

11.7.6.4 MDEU Key Size Register

The MDEU key size register, shown in Figure 11-80, indicates the number of bytes of key memory that should be used in HMAC generation. MDEU supports at most one block of key. MDEU generates a key size error if the value written to this register exceeds 64 bytes for MD5, SHA-1, SHA-224, or SHA-256. If algorithms SHA-384 or SHA-512 are selected, then MDEU generates a key size error if the value written to this register exceed 128 bytes.

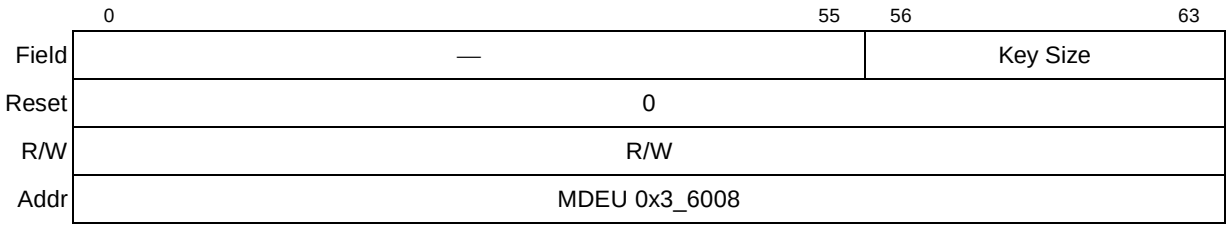


Figure 11-80. MDEU Key Size Register

11.7.6.5 MDEU Data Size Register

The MDEU data size register, shown in Figure 11-81, specifies the number of bits of data to be processed.

The data size field is a 21-bit signed number. Values written to this register are added to the current register value. Multiple writes are allowed. The MDEU processes data when there is a positive value in this register and there is data available in the MDEU input FIFO. (Negative values can arise in inbound processing, when it is necessary to hold back data from the MDEU until the pad length has been decrypted.)

Since the MDEU does not support bit offsets, bits 61–63 must be written as 0 and are always read as zero. Furthermore, when the CONT bit of the MDEU Mode Register is high, the data size must be a multiple of the block size (512 bits for MD5, SHA-1, SHA-224 and SHA-256; 1024 bits for SHA-384 and SHA-512). Violating either of these conditions causes a data size error (DSE in the MDEU Interrupt Status Register).

This register is cleared when the MDEU is reset or re-initialized. At the end of processing, its contents has been decremented down to zero (unless there is an error interrupt).

NOTE

Writing to the data size register allows the MDEU to enter auto-start mode. Therefore, the required context registers must be written prior to writing the data size.

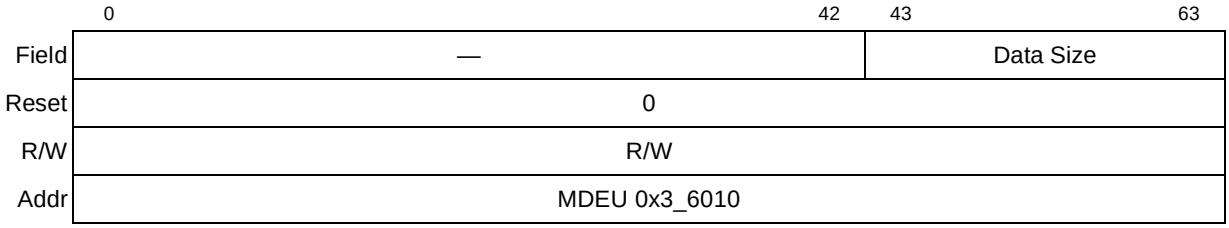


Figure 11-81. MDEU Data Size Register

11.7.6.6 MDEU Reset Control Register

The MDEU reset control register, shown in Figure 11-82, allows three levels reset of just the MDEU, as defined by the three self-clearing bits.

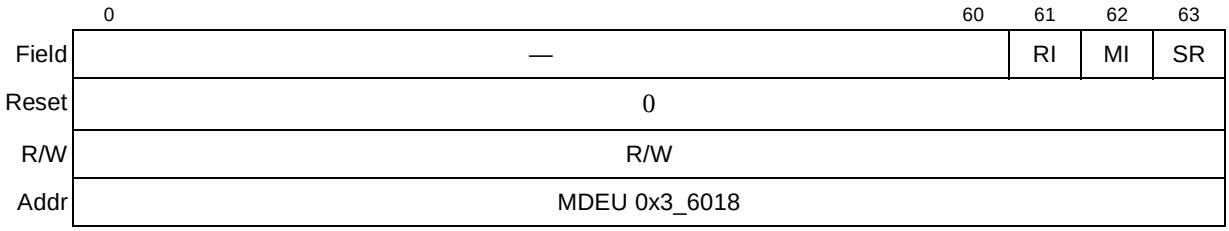


Figure 11-82. MDEU Reset Control Register

Table 11-64 describes MDEU reset control register fields.

Table 11-64. MDEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes MDEU interrupts signalling done and error to be reset. It further resets the state of the MDEU interrupt status register. 0 No reset 1 Reset interrupt logic

Table 11-64. MDEU Reset Control Register Field Descriptions (continued)

Bits	Name	Description
62	MI	Module initialization is nearly the same as software reset, except that the MDEU Interrupt mask register remains unchanged. 0 No reset 1 Reset most of MDEU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset

11.7.6.7 MDEU Status Register

The MDEU status register, as seen in Figure 11-83, reflects the state of the MDEU internal signals. The majority of these internal signals reflect the state of low-level MDEU functions, such as data padding, key padding, etc., and are not important to the user, however the user should be aware that reads of this register especially during processing are likely to return non-zero values for many bits between 0:57. The 4 signals shown are those which are most likely to be of interest to the user.

The MDEU status register is read-only. Writing to this location results in address error being reflected in the MDEU interrupt status register.

	0 — 57	58	59 60	61	62	63
Field	—	HALT	ICCR	EI	DI	RD
Reset	0					
R/W	R					
Addr	MDEU 0x3_6028					

Figure 11-83. MDEU Status Register

Table 11-65 describes MDEU status register fields.

Table 11-65. MDEU Status Register Field Descriptions

Bits	Name	Description
0–57	—	Reserved
58	HALT	Halt. Indicates that the MDEU has halted due to an error. 0 MDEU not halted 1 MDEU halted **Note:** Because the error causing the MDEU to stop operating may be masked before reaching the interrupt status register, the MDEU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.

Table 11-65. MDEU Status Register Field Descriptions

Bits	Name	Description
59–60	ICCR	Integrity Check Comparison Result 00 No integrity check comparison was performed. 01 The integrity check comparison passed. 10 The integrity check comparison failed. 11 Reserved **Note:** A “passed” or “failed” result is generated only if ICV checking is enabled.
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 MDEU is not signaling error 1 MDEU is signaling error
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 MDEU is not signaling done 1 MDEU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that MDEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.6.8 MDEU Interrupt Status Register

The MDEU interrupt status register, shown in Figure 11-84, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the MDEU interrupt mask register is zero (see Section 11.7.6.9, “MDEU Interrupt Mask Register”).

If the MDEU interrupt status register is non-zero, the MDEU halts and the MDEU error interrupt signal is asserted to the controller (see Section 11.6.3.3, “Interrupt Status Register (ISR)”). In addition, if the MDEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the Channel Status Register (see Table 11-15) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the host, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the MDEU Reset Control Register.

	0	49	50	51	52	53	54	55	56	57	58 60	61	62 63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
Reset	0												
R/W	R/W												
Addr	MDEU 0x3_6030												

Figure 11-84. MDEU Interrupt Status Register

Table 11-66 describes MDEU interrupt status register fields.

Table 11-66. MDEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check Error: 0 No error detected 1 Integrity check error detected. An ICV check was performed and the supplied ICV did not match the one computed by the MDEU.
50	—	Reserved
51	IE	Internal Error. Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected **Note:** This bit is asserted any time an enabled error condition occurs and can only be cleared by resetting the MDEU.
52	ERE	Early Read Error. The MDEU context was read before the MDEU completed the hashing operation. 0 No error detected 1 Early read error
53	CE	Context Error. The MDEU Key Register, Key Size Register, or Data Size Register was modified while MDEU was hashing. 0 No error detected 1 Context error
54	KSE	Key Size Error. Two possible causes: • A value greater than permitted was written to the MDEU key size register (128 bytes for SHA-384 and SHA-512; 64 bytes otherwise) • In either a HMAC or SMAC operation, key size was not written prior to writing data size or receiving an End of Message command. 0 No error detected 1 Key size error
55	DSE	Data Size Error. A value not a multiple of 512 bits (1024 bits for SHA-384 and SHA-512) while the MDEU mode register CONT bit is high. 0 No error detected 1 Data size error
56	ME	Mode Error. Is set if any of these error conditions is detected: • any reserved bit of the Mode register is set • the ALG field of the Mode register contains an illegal value 0 No error detected 1 Mode error
57	AE	Address Error. An illegal read or write address was detected within the MDEU address space. 0 No error detected 1 Address Error
58–60	—	Reserved

Table 11-66. MDEU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
61	IFO	Input FIFO Overflow. The MDEU Input FIFO was pushed while full. 0 No overflow detected 1 Input FIFO has overflowed **Note:** When operated through channel-controlled access, the SEC implements flow control, and FIFO size is not a limit to data input size. When operated through host-controlled access, the MDEU cannot accept FIFO inputs larger than 256 bytes without overflowing.
62–63	—	Reserved

11.7.6.9 MDEU Interrupt Mask Register

The MDEU interrupt mask register, shown in Figure 11-85, controls the result of detected errors. For a given error (as defined in Section 11.7.6.8, "MDEU Interrupt Status Register"), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0 – 48	49	50	51	52	53	54	55	56	57	58 – 60	61	62 – 63
Field	—	ICE	—	IE	ERE	CE	KSE	DSE	ME	AE	—	IFO	—
Reset	0x3000												
R/W	R/W												
Addr	MDEU 0x3_6038												

Figure 11-85. MDEU Interrupt Mask Register

Table 11-66 describes MDEU interrupt status register fields.

Table 11-67. MDEU Interrupt Mask Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	ICE	Integrity Check Error. The supplied ICV did not match the one computed by the MDEU. 0 Integrity check error enabled. WARNING: Do not enable this if using EU status writeback (see bits IWSE and AWSE in Section 11.5.4.1, "Channel Configuration Register (CCR)"). 1 Integrity check error disabled
50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
52	ERE	Early Read Error. The MDEU register was read while the MDEU was performing hashing. 0 Early read error enabled 1 Early read error disabled

Table 11-67. MDEU Interrupt Mask Register Field Descriptions (continued)

Bits	Name	Description
53	CE	Context Error. The MDEU key register, the key size register, the data size register, or the mode register, was modified while the MDEU was performing hashing. 0 Context error enabled 1 Context error disabled
54	KSE	Key Size Error. A value outside the bounds was written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled
55	DSE	Data Size Error. An inconsistent value was written to the MDEU data size register: 0 Data size error enabled 1 Data size error disabled
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
58–60	—	Reserved
61	IFO	Input FIFO Overflow. The MDEU input FIFO was pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
62–63	—	Reserved

11.7.6.10 MDEU ICV Size Register

The MDEU ICV size register, shown in Figure 11-86, specifies the number of bytes of the ICV result to be compared if the MDEU performs ICV checking (see Section 11.7.6.2, “MDEU Mode Register”). The data to be compared to the MDEU result is supplied to the MDEU through its input FIFO.

This register is cleared when the MDEU is reset or re-initialized.

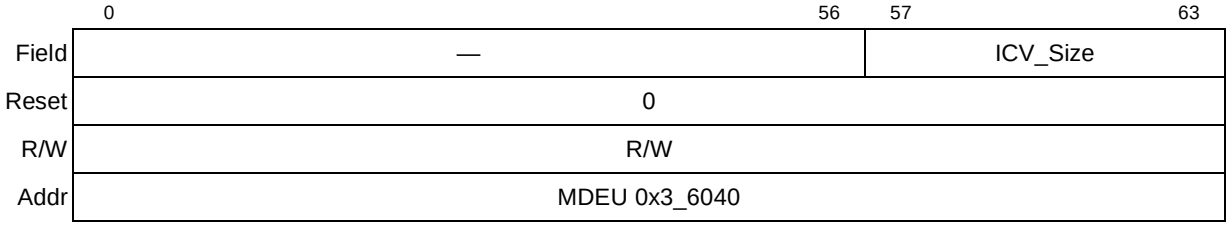


Figure 11-86. MDEU ICV Size Register

11.7.6.11 MDEU End_of_Message Register

The MDEU end_of_message register in the MDEU, shown in Figure 11-87, is used to indicate an authentication operation may be completed. After the final message block is written to the input FIFO, the end_of_message register must be written. The value in the data size register is used to determine how many

bits of the final message block (always 512) is processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal done interrupt.

	0	63
Field	—	
Reset	0	
R/W	W	
Addr	MDEU 0x3_6050	

Figure 11-87. MDEU End_of_Message Register

11.7.6.12 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bit count.

NOTE

All SHA algorithms are big-endian. MD5 is little-endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8-byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done generates an error interrupt.

After a Power On Reset, all the MDEU Context Register values are cleared to 0. Figure 11-88 shows how the MDEU Context Registers are initialized if the INIT bit is set in the MDEU Mode Register. All registers are initialized, regardless of mode selected, however only the appropriate context register values are used in hash generation per the mode selected. The user typically doesn't care about the MDEU Context Register initialization values; they are documented for completeness in the event the user reads these registers using host-controlled access. MDEU reset via the MDEU Reset Control Register (Figure 11-82) or SEC global software reset (Figure 11-24) does not clear these registers.

	0 — 31	32 — 63	Register
Algorithm			Context offset 0x3_6100
MD-5	A = 0x01234567	B = 0x89ABCDEF	
SHA-1	A = 0x67452301	B = 0xEFCDAB89	
SHA-224	A = 0xC1059ED8	B = 0x367CD507	
SHA-256	A = 0x6A09E667	B = 0xBB67AE85	
SHA-384	A = 0xcbbb9d5dc1059ed8		
SHA-512	A = 0x6a09e667f3bcc908		
Algorithm			Context offset 0x3_6108
MD-5	C = 0xFEDCBA98	D = 0x76543210	
SHA-1	C = 0x98BADCFE	D = 0x10325476	
SHA-224	C = 0x3070DD17	D = 0xF70E5939	
SHA-256	C = 0x3C6EF372	D = 0xA54FF53A	
SHA-384	B = 0x629a292a367cd507		
SHA-512	B = 0xbb67ae8584caa73b		
Algorithm			Context offset 0x3_6110
MD-5	E = 0xF0E1D2C3	F = 0x8C68059B	
SHA-1	E = 0xC3D2E1F0	F = 0x9B05688C	
SHA-224	E = 0xFFC00B31	F = 0x68581511	
SHA-256	E = 0x510E527F	F = 0x9B05688C	
SHA-384	C = 0x9159015a3070dd17		
SHA-512	C = 0x3c6ef372fe94f82b		

Figure 11-88. MDEU Context Register

Algorithm			Context offset 0x3_6118
MD-5	G = 0xABD9831F	H = 0x19CDE05B	
SHA-1	G = 0x1F83D9AB	H = 0x5BE0CD19	
SHA-224	G = 0x64F98FA7	H = 0xBEFA4FA4	
SHA-256	G = 0x1F83D9AB	H = 0x5BE0CD19	
SHA-384	D = 0xh152fecd8hf70e5939		
SHA-512	D = 0xha54ff53ah5f1d36f1		
Algorithm			Context offset 0x3_6120
MD5, SHA1, SHA-224, SHA-256	Message Length Count = 0		
SHA-384	E = 0x67332667ffc00b31		
SHA-512	E = 0x510e527fade682d1		
Algorithm			Context offset 0x3_6128
MD5, SHA1, SHA-224, SHA-256	reserved		
SHA-384	F = 0x8eb44a8768581511		
SHA-512	F = 0x9b05688c2b3e6c1f		
Algorithm			Context offset 0x3_6130
MD5, SHA1, SHA-224, SHA-256	reserved		
SHA-384	G = 0xdb0c2e0d64f98fa7		
SHA-512	G = 0x1f83d9abfb41bd6b		
Algorithm			Context offset 0x3_6138
MD5, SHA1, SHA-224, SHA-256	reserved		
SHA-384	H = 0x47b5481dbefa4fa4		
SHA-512	H = 0x5be0cd19137e2179		
Algorithm			Context offset 0x3_6140
MD5, SHA1, SHA-224, SHA-256	reserved		
SHA-384, SHA-512	Message Length Count = 0		

Figure 11-88. MDEU Context Register (continued)

If SHA-384 or SHA-512 are selected, then each of the registers A, B, C, D, E, F, G, H are 64-bits (instead of 32 bits for other hash algorithms). As a result, the base address for each context register is shifted to adjust.

11.7.6.13 MDEU Key Registers

The MDEU maintains sixteen 64-bit registers for writing an HMAC key; only the first eight are used for MD5, SHA-1, SHA-224, or SHA-256. The IPAD and OPAD operations are performed automatically on the key data when required.

NOTE

All SHA algorithms are big-endian. MD5 is little-endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

11.7.6.14 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed (followed in some case by an ICV value for ICV checking). Normally, the channels control all access to this FIFO. For host-controlled operation, a write to anywhere in the MDEU FIFO address space enqueues data to the MDEU input FIFO, and a read from anywhere in this address space returns all zeros.

When the host writes to the MDEU FIFO (using host-controlled access), it can write to any FIFO address by byte, word (4 bytes), or dword (8 bytes). The MDEU assembles these bytes from left to right, for example, the first bytes written are placed in the most significant bit-positions. Whenever the MDEU accumulates 8 bytes, this dword is automatically enqueued into the FIFO, and any remaining bytes are left-justified in preparation for assembling the next dword. It is not necessary to fill all bytes of the final dword. Any last bytes remaining in the staging register are automatically padded with zeros and forced into the input FIFO when the MDEU End of Message Register is written.

Overflows caused by writing the MDEU FIFO are reflected in the MDEU interrupt status register.

11.7.7 Public Key Execution Units (PKEU)

This section contains details about the public key execution unit (PKEU), including modes of operation, status and control registers, and parameter RAMs.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the PKEU is used through channel-controlled access, which means that most reads and writes of PKEU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.7.1 PKEU Mode Register

The PKEU mode register, shown in Figure 11-89, specifies the internal PKEU routine to be executed. The mode register is cleared when the PKEU is reset or re-initialized. Setting a reserved mode bit generates a data error. If the mode register is modified during processing, a context error is generated. Table 11-68 lists the possible values for the ROUTINE field.

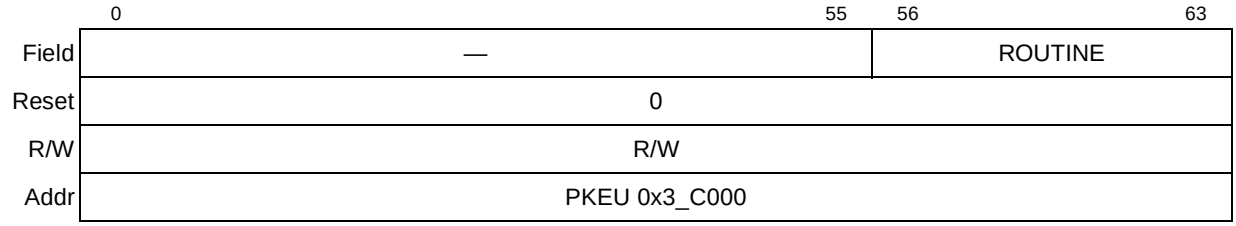


Figure 11-89. PKEU Mode Register

For channel-controlled access to the PKEU, the descriptor type is determined by the ROUTINE to be used. The descriptor type used with each ROUTINE is listed in Table 11-68.

11.7.7.2 PKEU Key Size Register

The PKEU key size register, shown in Figure 11-90, specifies the number of significant bytes to be used from PKEU Parameter Memory E in performing modular exponentiation or elliptic curve point multiplication. The range of values for this register, when performing either modular exponentiation or elliptic curve point multiplication, is from 1 to 512. Specifying a key size outside of this range causes a key size error (KSE) in the PKEU Interrupt Status Register.

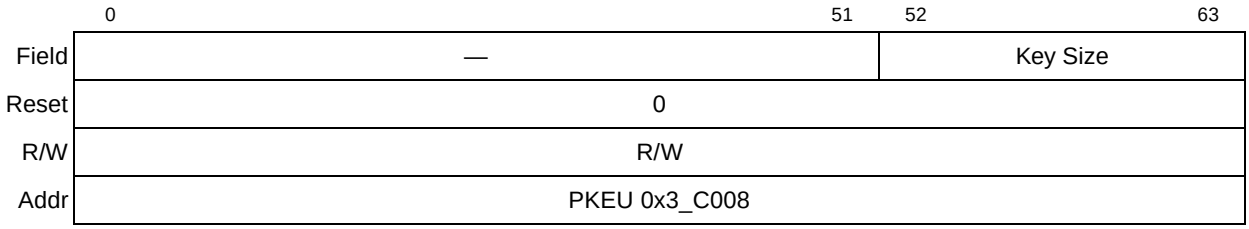


Figure 11-90. PKEU Key Size Register

Table 11-68. ROUTINE Field Description

Mode [56-63]	Routine Name	Routine Description	Descriptor Type	Section
0x00	RESERVED	Reserved	—	—
0x01	CLEARMEMORY	Clear memory	pkeu_mm	11.7.7.11.1
0x02	MOD_EXP	FP: Exponentiate mod N and deconvert from Montgomery format	pkeu_mm	11.7.7.11.2
0x03	MOD_R2MODN	FP: Compute Montgomery converter (R^2 mod N)	pkeu_mm	11.7.7.11.4

Table 11-68. ROUTINE Field Description (continued)

Mode [56-63]	Routine Name	Routine Description	Descriptor Type	Section
0x04	MOD_RRMODP	FP: Compute Montgomery converter for Chinese Remainder Theorem (R_nR_p mod N)	pkeu_mm	11.7.7.11.5
0x05	EC_FP_AFF_PTMULT	FP EC: Multiply scalar times point in affine coordinates	pkeu_ptmul	11.7.7.11.6
0x06	EC_F2M_AFF_PTMULT	F2m EC: Multiply scalar times point in affine coordinates	pkeu_ptmul	11.7.7.11.7
0x07	EC_FP_PROJ_PTMULT	FP EC: Multiply scalar times point in projective coordinates	pkeu_ptmul	11.7.7.11.8
0x08	EC_F2M_PROJ_PTMULT	F2m EC: Multiply scalar times point in projective coordinates	pkeu_ptmul	11.7.7.11.9
0x09	EC_FP_ADD	FP EC: Add two points in projective coordinates	pkeu_ptadd_dbl	11.7.7.11.10
0x0A	EC_FP_DOUBLE	FP EC: Double a point in projective coordinates	pkeu_ptadd_dbl	11.7.7.11.11
0x0B	EC_F2M_ADD	F2m EC: Add two points in projective coordinates	pkeu_ptadd_dbl	11.7.7.11.12
0x0C	EC_F2M_DOUBLE	F2m EC: Double a point in projective coordinates	pkeu_ptadd_dbl	11.7.7.11.13
0x0D	F2M_R2	F2m: Compute Montgomery converter (R^2 mod N)	pkeu_mm	11.7.7.11.14
0x0E	F2M_INV	F2m: Invert mod N	pkeu_mm	11.7.7.11.15
0x0F	MOD_INV	FP: Invert mod N	pkeu_mm	11.7.7.11.16
0x10	MOD_ADD	FP: Add mod N	pkeu_mm	11.7.7.11.17
0x20	MOD_SUB	FP: Subtract mod N	pkeu_mm	11.7.7.11.18
0x30	MOD_MULT1_MONT	FP: Multiply mod N in Montgomery format	pkeu_mm	11.7.7.11.19
0x40	MOD_MULT2_DECONV	FP: Multiply mod N and deconvert from Montgomery format	pkeu_mm	11.7.7.11.20
0x50	F2M_ADD	F2m: Add mod N	pkeu_mm	11.7.7.11.21
0x60	F2M_MULT1_MONT	F2m: Multiply mod N in Montgomery format	pkeu_mm	11.7.7.11.22
0x70	F2M_MULT2_DECONV	F2m: Multiply mod N and deconvert from Montgomery format	pkeu_mm	11.7.7.11.23
0x80	RSA_SSTEP	FP: Exponentiate mod N (combines MOD_R2MODN, POLY_F2M_MULT1_MONT, and MOD_EXP)	pkeu_mm	11.7.7.11.24
0x1d	MOD_EXP_TEQ	FP: Exponentiate mod N and deconvert from Montgomery format with timing equalization	pkeu_mm	11.7.7.11.3
0x1e	RSA_SSTEP_TEQ	FP: Exponentiate mod N with timing equalization (combines MOD_R2MODN, EC_F2M_MULT1_MONT, and MOD_EXP_TEQ)	pkeu_mm	11.7.7.11.25
0xFF	SPK_BUILD	Build PK data structure (data structure used by all elliptic curve routines)	pkeu_build	11.7.7.11.26

11.7.7.3 PKEU AB Size Register

The PKEU AB size register, shown in Figure 11-91, represents the size of each operand written into parameter memory A and parameter memory B in bits. An exact size in bits must be provided since a Big to little-endian re-alignment is performed based on this value. No error checking is performed as to whether the operand sizes are greater than the prime modulus or the field size written in N-ram. In other words, it is assumed that operands are modular reduced before being written into the PKEU module. This register must be written to before each write to parameter memory A or parameter memory B and must be written before each read of parameter memory A and parameter memory B if the amount of data being taken out is different than the amount of data put in A or B. The value written to the AB_Size register must also adhere to the constraints on parameters A and B, set by the chosen routine (see Table 11-68). The AB_Size register should not be read or written to while the PKEU is processing as this causes a ‘data modify during processing error.’ This register is cleared when the PKEU is reset. The maximum size acceptable is 4096 bits.

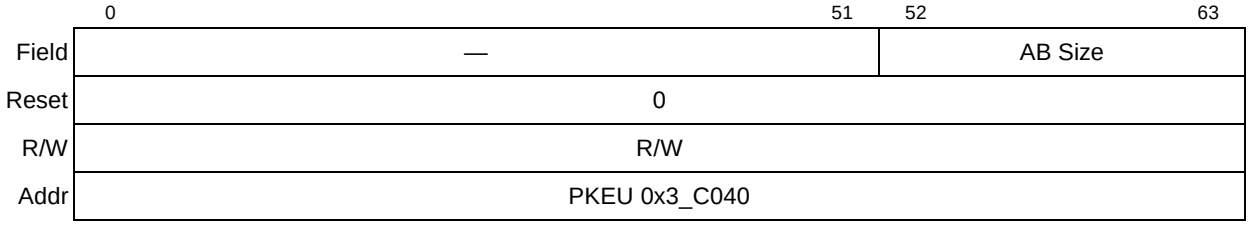


Figure 11-91. PKEU AB Size Register

11.7.7.4 PKEU Data Size Register

The PKEU data size register, shown in Figure 11-92, specifies, in bits, the size of the significant portion of the modulus or irreducible polynomial. The minimum size valid for all routines to operate properly is 33 bits. The maximum size to operate properly is 4096 bits. A value in bits larger than 4096 results in a data size error (see the DSE bit in Table 11-71).

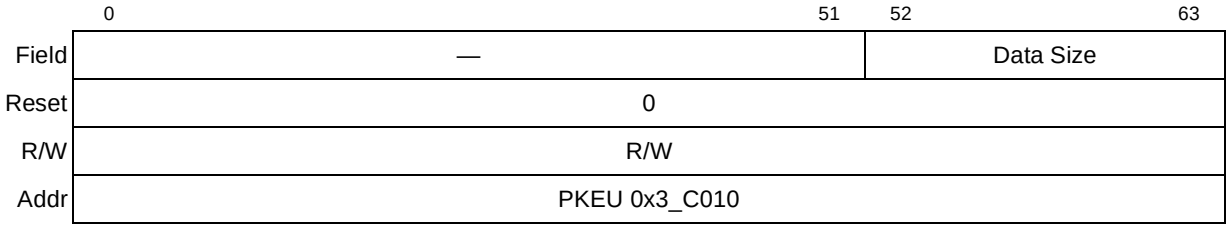


Figure 11-92. PKEU Data Size Register

11.7.7.5 PKEU Reset Control Register

The PKEU reset control register, shown in Figure 11-93, contains three reset options specific to the PKEU.

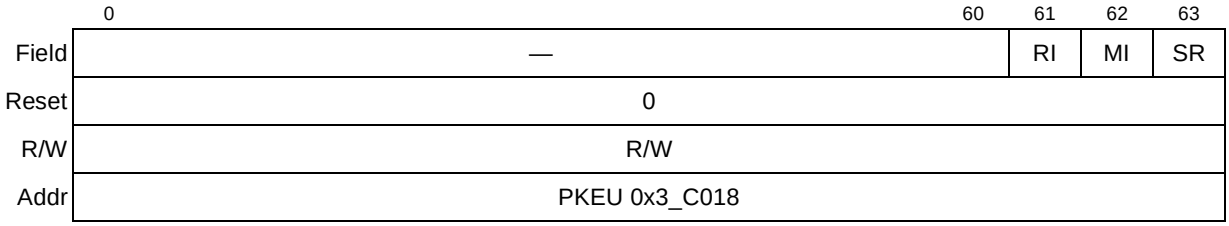


Figure 11-93. PKEU Reset Control Register

Table 11-69 describes the PKEU reset control register fields.

Table 11-69. PKEU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset interrupt. Writing this bit active high causes PKEU interrupts signalling done and error to be reset. It further resets the state of the PKEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
62	MI	Module initialization. Module initialization is nearly the same as Software Reset, except that the interrupt mask register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the PKEU status register (Section 11.7.7.6, “PKEU Status Register”). 0 Do not reset 1 Reset most of PKEU
63	SR	SW reset. Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the PKEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the PKEU enters a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the PKEU status register indicates when this initialization routine is complete (Section 11.7.7.6, “PKEU Status Register”). 0 Do not reset 1 Full PKEU reset

11.7.7.6 PKEU Status Register

The PKEU status register, shown in Figure 11-94, contains six fields that reflect the state of PKEU internal fields.

The PKEU status register is read-only. Writing to this location results in address error being reflected in the PKEU interrupt status register.

	0–55	56	57	58	59–60	61	62	63
Field	—	I	Z	HALT	—	EI	DI	RD
Reset	0							
R/W	R							
Addr	PKEU 0x3_C028							

Figure 11-94. PKEU Status Register

Table 11-70 describes the PKEU status register's fields.

Table 11-70. PKEU Status Register Field Descriptions

Bits	Name	Description
0–55	—	Reserved
56	I	Infinity. This bit reflects the state of the PKEU infinity detect bit when last sampled. Only particular instructions within routines cause infinity to be modified, so this bit should be used with great care.
57	Z	Zero. This bit reflects the state of the PKEU zero detect bit when last sampled. Only particular instructions within routines cause zero to be modified, so this bit should be used with great care.
58	HALT	Halt indicates that the PKEU has halted due to an error. 0 PKEU not halted 1 PKEU halted **Note:** Because the error causing the PKEU to stop operating may be masked before reaching the Interrupt Status Register, the PKEU Interrupt Status Register is used to provide a second source of information regarding errors preventing normal operation.
59-60	—	Reserved
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the controller interrupt status register (Section 11.6.3.3, "Interrupt Status Register (ISR)"). 0 PKEU is not signaling error 1 PKEU is signaling error
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, "Interrupt Status Register (ISR)"). 0 PKEU is not signaling done 1 PKEU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that PKEU has completed its reset sequence, as reflected in the signal sampled by the appropriate channel. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.7.7 PKEU Interrupt Status Register

The PKEU interrupt status register, shown in Figure 11-95, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the PKEU interrupt mask register is zero (see Section 11.7.7.8, "PKEU Interrupt Mask Register").

If the PKEU interrupt status register is non-zero, the PKEU halts and the PKEU error interrupt signal is asserted to the controller (see Section 11.6.3.3, "Interrupt Status Register (ISR)"). In addition, if the PKEU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the Channel Status Register (see Table 11-15) and generates a channel error interrupt to the controller.

If the interrupt status register is written from the host, 1s in the value written are recorded in the interrupt status register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the PKEU Reset Control Register.

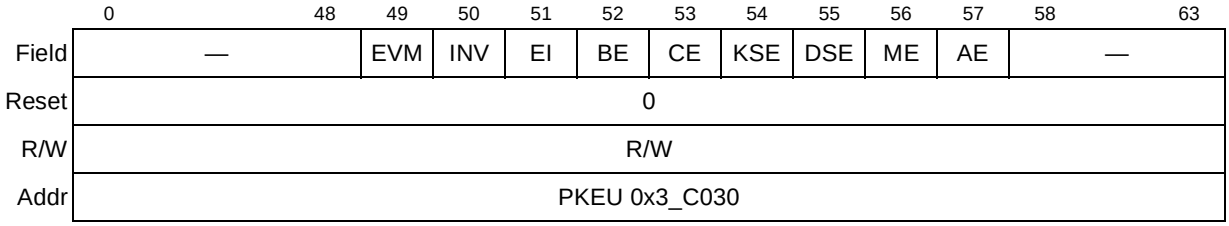


Figure 11-95. PKEU Interrupt Status Register

Table 11-71 describes PKEU interrupt status register signals.

Table 11-71. PKEU Interrupt Status Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	EVM	Even modulus error. Indicates that an even modulus was supplied for a PK operation that requires an odd modulus. 0 No even modulus error detected 1 Even modulus error detected
50	INV	Inversion error. Indicates that the inversion routine has a zero operand. 0 No inversion error detected 1 Inversion error detected
51	IE	Internal error. An internal processing error was detected while the PKEU was operating. 0 No error detected 1 Internal error **Note:** This bit is asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Mask Register or by resetting the PKEU.
52	BE	Boot Error. Indicates that either at boot (reset) sequence, RAM locations are not reset correctly. 0 No boot error detected 1 Boot error

Table 11-71. PKEU Interrupt Status Register Field Descriptions (continued)

Bits	Name	Description
53	CE	Context error. A PKEU key register, the key size register, the data size register, or mode register was modified while the PKEU was operating. 0 No error detected 1 Context error
54	KSE	Key size error. Value outside the bounds of 1–256 bytes was written to the PKEU key size register 0 No error detected 1 Key size error detected
55	DSE	Data size error. Value outside the bounds 97– 4096 bits was written to the PKEU data size register 0 No error detected 1 Data size error detected
56	ME	Mode error. An illegal value was detected in the mode register. 0 No error detected 1 Mode error **Note:** Writing to reserved bits in a mode register is a likely source of error.
57	AE	Address error. Illegal read or write address was detected within the PKEU address space. 0 No error detected 1 Address error
58–63	—	Reserved

11.7.7.8 PKEU Interrupt Mask Register

The PKEU interrupt mask register, shown in Figure 11-96, controls the result of detected errors. For a given error (as defined in Section 11.7.7.7, “PKEU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the PKEU interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

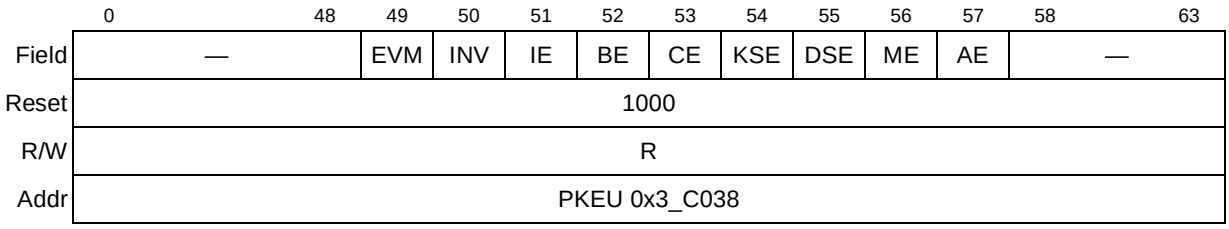


Figure 11-96. PKEU Interrupt Mask Register

Table 11-72 describes PKEU interrupt mask register fields.

Table 11-72. PKEU Interrupt Mask Register Field Descriptions

Bits	Name	Description
0–48	—	Reserved
49	EVM	Even modulus error 0 Even modulus error enabled 1 Even modulus error disabled
50	INV	Inversion error 0 Inversion error enabled 1 Inversion error disabled
51	IE	Internal error 0 Internal error enabled 1 Internal error disabled
52	BE	Boot error 0 Boot error enabled 1 Boot error disabled
53	CE	Context error 0 Context error enabled 1 Context error disabled
54	KSE	Key size error 0 Key size error enabled 1 Key size error disabled
55	DSE	Data size error 0 Data size error enabled 1 Data size error disabled
56	ME	Mode error 0 Mode error enabled 1 Mode error disabled
57	AE	Address error 0 Address error enabled 1 Address error disabled
58–63	—	Reserved

11.7.7.9 PKEU End_of_Message Register

The PKEU end_of_message register, shown in Figure 11-97, is used to indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the ROUTINE field, per the contents of the parameter memories listed below. This register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Reading from this register is not meaningful, but a zero value is always returned, and no error is generated.

	0	63
Field	—	
Reset	0	
R/W	W	
Addr	PKEU 0x3_C050	

Figure 11-97. PKEU End_of_Message Register

11.7.7.10 PKEU Parameter Memories

The PKEU uses four 4096-bit memories to receive and store operands for the arithmetic operations the PKEU is asked to perform. In addition, results are stored in one particular parameter memory.

Data addressing within these memories is big-endian, that is, the most significant byte is stored in the lowest address.

11.7.7.10.1 PKEU Parameter Memory A

This 4096 bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 1024 bit memories, and is used to specify particular curve parameters and input values.

11.7.7.10.2 PKEU Parameter Memory B

This 4096-bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented in to four 1024 bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.

11.7.7.10.3 PKEU Parameter Memory E

This 4096-bit memory is non-segmentable, and specifies the exponent for modular exponentiation, or the multiplier k for elliptic curve point multiplication. This memory space is write only; a read of this memory space causes address error to be reflected in the PKEU interrupt status register.

11.7.7.10.4 PKEU Parameter Memory N

This 4096-bit memory is non-segmentable, and specifies the modulus for modular arithmetic and F_p elliptic curve routines. For F_2m elliptic curve routines, this memory specifies the irreducible polynomial.

11.7.8 Random Number Generator (RNGU)

This section contains details about the random number generator, including modes of operation, status and control registers, and FIFO.

The RNGU is an execution unit capable of generating 64-bit random numbers. It combines a True Random Number Generator (TRNG) and an implementation of a NIST-approved Pseudo-Random Number Generator (PRNG) (as described in Annex C of FIPS 140-2 and ANSI X9.62). The RNGU is designed to comply with the FIPS-140 standard for randomness and non-determinism.

The RNGU consists of five major functional blocks:

- 64-bit internal bus interface, registers, and FIFO
- True Random Number Generator (ring oscillator, LFSRs, Statistical Checker)
- Xseed Generator
- Pseudo-Random Number Generator (XKEY, SHA-1, FSM)
- Simultaneous Reseed LFSR

The states of the LFSRs in the TRNG are advanced at an unknown frequency determined by the ring oscillator clock. The entropy generated by this structure is then added into the XKEY structure of the PRNG during seed generation. Seed generation takes approximately 2,000,000 cycles as 20,000 bits of entropy are sampled from the output of the LFSRs of the TRNG.

After the initial seeding, the RNGU turns off the TRNG and uses solely the PRNG to generate random data. After 1,000,000 times through the algorithm the RNGU is once again seeded. This second seed occurs the next time through the algorithm by using data from the Simultaneous Reseed LFSR to modify the algorithm. The data in the simultaneous reseed LFSR comes directly from the TRNG as well and was being generated during the first 20,000 times through the PRNG algorithm after the initial seed was completed.

Most of the registers described here would not normally be accessed by the host. They are documented here mainly for debug purposes. In typical operation, the RNGU is used through channel-controlled access, which means that most reads and writes of RNGU registers are directed by the SEC channels. Driver software would perform host-controlled register accesses only on a few registers for initial configuration and error handling.

11.7.8.1 RNGU Mode Register

The RNGU mode register, shown in Figure 11-98, is a writable location but all mode bits are currently reserved. It is documented for the sake of consistency with the other EUs.

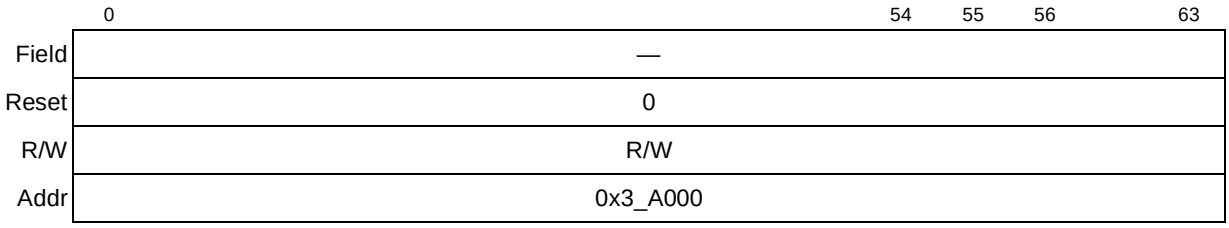


Figure 11-98. RNGU Mode Register

11.7.8.2 RNGU Data Size Register

The RNGU data size register, shown in Figure 11-99, is used to tell the RNGU to begin generating random data. The actual contents of the data size register does not affect the operation of the RNGU. After a reset and prior to the first write of data size, the RNGU builds entropy without pushing data onto the FIFO. Once the data size register is written, the RNGU begins pushing data onto the FIFO. One dword (64 bits) of data is pushed onto the FIFO every 112 cycles until the FIFO is full. The RNGU then attempts to keep the FIFO full.

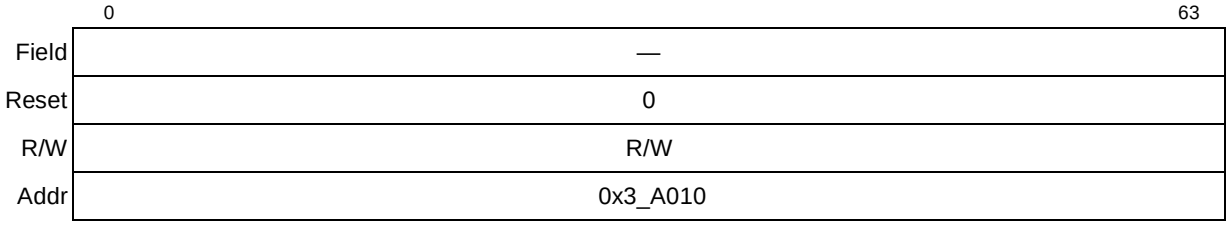


Figure 11-99. RNGU Data Size Register

11.7.8.3 RNGU Reset Control Register

The RNGU reset control register, shown in Figure 11-100, contains three reset options specific to the RNGU.

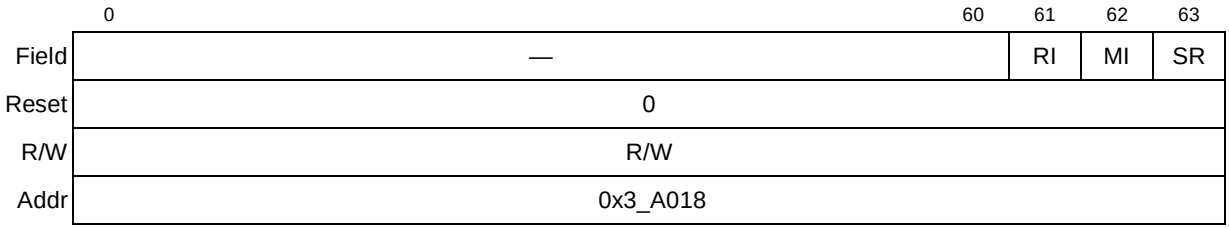


Figure 11-100. RNGU Reset Control Register

Table 11-73 describes RNGU reset control register fields.

Table 11-73. RNGU Reset Control Register Field Descriptions

Bits	Name	Description
0–60	—	Reserved
61	RI	Reset Interrupt. Writing this bit active high causes RNGU interrupts signalling done and error to be reset. It further resets the state of the RNGU interrupt status register. 0 No reset 1 Reset interrupt logic

Table 11-73. RNGU Reset Control Register Field Descriptions (continued)

Bits	Name	Description
62	MI	Module Initialization. This reset value performs enough of a reset to prepare the RNGU for another request, without forcing the internal control machines and the output FIFO to be reset, thereby invalidating stored random numbers or requiring reinvocation of a warm-up period. Module initialization is nearly the same as software reset, except that the interrupt mask register remains unchanged. 0 No reset 1 Reset most of RNGU
63	SR	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the RNGU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNGU reset

11.7.8.4 RNGU Status Register

This RNGU status register, shown in Figure 11-101, contains six fields that reflect the state of the RNGU internal signals.

The RNGU status register is read-only. Writing to this location results in an address error being reflected in the RNGU interrupt status register.

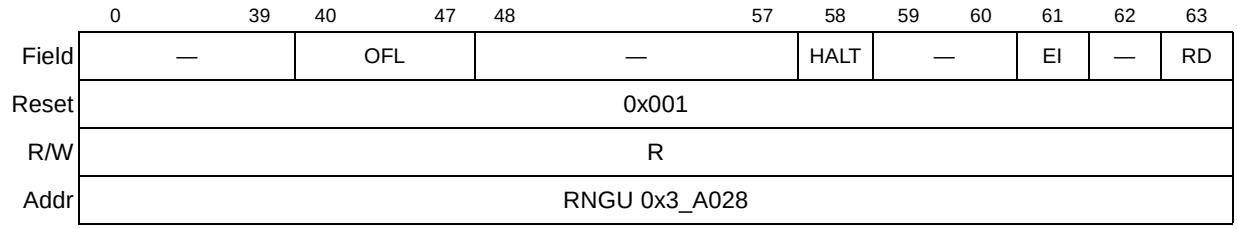


Figure 11-101. RNGU Status Register

Table 11-74 describes RNGU status register fields.

Table 11-74. RNGU Status Register Field Descriptions

Bits	Name	Description
0–39	—	Reserved
40–47	OFL	The number of dwords currently in the output FIFO
48–57	—	Reserved
58	HALT	Halt. Indicates that the RNGU has halted due to an error. 0 RNGU not halted 1 RNGU halted Note: Because the error causing the RNGU to stop operating may be masked before reaching the interrupt status register, the RNGU interrupt status register is used to provide a second source of information regarding errors preventing normal operation.
59–60	—	Reserved

Table 11-74. RNGU Status Register Field Descriptions (continued)

Bits	Name	Description
61	EI	Error interrupt: This status bit reflects the state of the error interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 RNGU is not signaling error 1 RNGU is signaling error
62	DI	Done interrupt: This status bit reflects the state of the done interrupt signal, as sampled by the Controller Interrupt Status Register (Section 11.6.3.3, “Interrupt Status Register (ISR)”). 0 RNGU is not signaling done 1 RNGU is signaling done
63	RD	Reset Done. This status bit, when high, indicates that the RNGU has completed its reset sequence. 0 Reset in progress 1 Reset done **Note:** Reset Done resets to 0, but has typically switched to 1 by the time a user checks the register, indicating the EU is ready for operation.

11.7.8.5 RNGU Interrupt Status Register

The RNGU interrupt status register, shown in Figure 11-102, indicates which unmasked errors have occurred and have generated error interrupts to the channel. Each bit in this register can only be set if the corresponding bit of the RNGU interrupt mask register is zero (see Section 11.7.8.6, “RNGU Interrupt Mask Register”).

If the RNGU interrupt status register is non-zero, the RNGU halts and the RNGU error interrupt signal is asserted to the controller (see Section 11.6.3.3, “Interrupt Status Register (ISR)”). In addition, if the RNGU is being operated through channel-controlled access, then an interrupt signal is generated to the channel to which this EU is assigned. The EU error then appears in bit 55 of the Channel Pointer Status Register (see Table 11-15) and generates a channel error interrupt to the controller.

If the Interrupt Status Register is written from the host, 1s in the value written are recorded in the Interrupt Status Register if the corresponding bit is unmasked in the Interrupt Mask Register. All other bits are cleared. This register can also be cleared by setting the RI bit of the RNGU Reset Control Register.

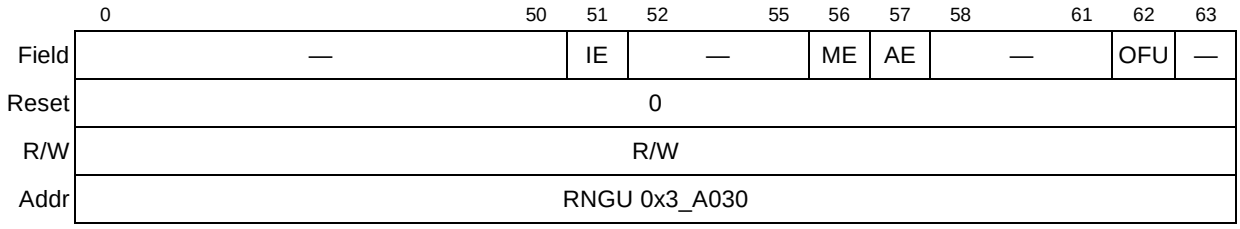


Figure 11-102. RNGU Interrupt Status Register

Table 11-75 describes RNGU interrupt status register fields.

Table 11-75. RNGU Interrupt Status Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal Error 0 No internal error detected 1 Internal error
52–55	—	Reserved
56	ME	Mode Error. Indicates that the host has attempted to write an illegal value to the mode register 0 Valid data 1 Invalid data error
57	AE	Address Error. An illegal read or write address was detected within the RNGU address space. 0 No error detected 1 Address error
58–61	—	Reserved
62	OFU	Output FIFO Underflow. The RNGU Output FIFO was read while empty. 0 No overflow detected 1 Output FIFO has underflowed
63	—	Reserved

11.7.8.6 RNGU Interrupt Mask Register

The RNGU interrupt mask register, shown in Figure 11-103, controls the result of detected errors. For a given error (as defined in Section 11.7.8.5, “RNGU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

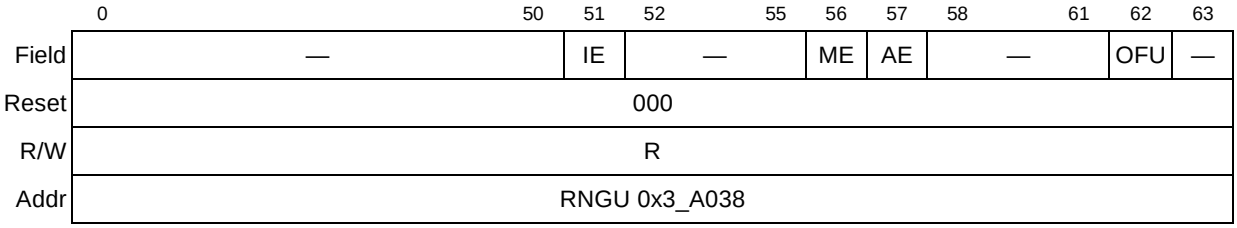


Figure 11-103. RNGU Interrupt Mask Register

Table 11-76 describes RNGU interrupt status register fields.

Table 11-76. RNGU Interrupt Mask Register Field Descriptions

Bits	Name	Description
0–50	—	Reserved
51	IE	Internal Error. An internal processing error was detected while generating random numbers. This error is no longer maskable and can only be cleared by setting one of the reset bits in the Reset Control Register 0 Internal error enabled 1 Internal error disabled
52–55	—	Reserved
56	ME	Mode Error. An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
57	AE	Address Error. An illegal read or write address was detected within the RNGU address space. 0 Address error enabled 1 Address error disabled
58–61	—	Reserved
62	OFU	Output FIFO Underflow. RNGU Output FIFO was read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
63	—	Reserved

11.7.8.7 RNGU End_of_Message Register

The RNGU end_of_message register, shown in Figure 11-104, is a write only register that can be used to start the RNGU to produce random numbers. A write of any value to this register causes the RNGU to begin to produce random numbers in the FIFO.

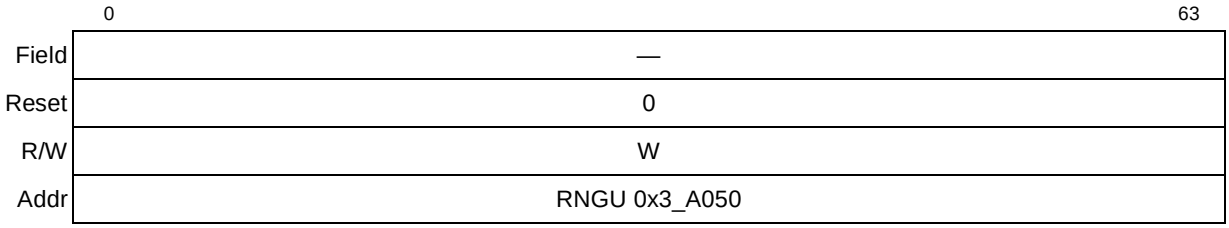


Figure 11-104. RNGU End_of_Message Register

11.7.8.8 RNGU ENTROPY

RNGU allows the user to input “Entropy” into the PRNG algorithm to modify the randomness of the RNGU. This group of registers are write only and all writes to these registers are ignored when the RNGU is busy. However when the RNGU is IDLE (FIFO is full or RNGU has not yet been started), all data written to these registers are used to modify the internal XKEY structure. These registers cannot be written back to back, there must be a clock cycle in between writes, so that the RNGU can process all 64-bits of data, as the RNGU processes only 32-bits per cycle.

11.7.8.9 RNGU FIFO

RNGU uses an output FIFO to collect periodically sampled random 64-bit-words, with the intent that random data always be available for reading. Normally, the channels control all access to this FIFO. For host-controlled operation, a read from anywhere in the RNGU FIFO address space dequeues data from the RNGU output FIFO.

The output FIFO is readable by byte, word, or dword. When all 8 bytes of the head dword have been read, that dword is automatically dequeued from the FIFO so that the next dword (if any) becomes available for reading. If any byte is read twice between dequeues, it causes an error interrupt of type AE from the EU.

Underflows caused by reading or writing the RNGU output FIFO are reflected in the RNGU interrupt status register. Also, a write to the RNGU output FIFO space is reflected as an addressing error in the RNGU interrupt status register.

NOTE

Host reads of the RNGB FIFO should be performed on an 8-byte basis, regardless of how many bits of random number is actually required. Partial host reads can leave the RNGB FIFO in a state that results in a channel error.

11.8 Power Saving Mode

The SEC may be disabled by setting DEVDISR[SEC]. The clocks to the SEC are active by default. The SEC should not be enabled/disabled during normal operation.

SEC may take a significant time to be disabled. SEC does not permit disablement while descriptors are being processed. Once warned a disablement of SEC is pending, the SEC channels complete current tasks, and then be forced to idle. They are not permitted to pop a new descriptor address off the Fetch Descriptor FIFO. Once all channels are idle, SEC then permits disablement.

Chapter 12
I2C Interfaces

This chapter describes the two inter-IC (IIC or I2C) bus interfaces implemented on this device.

12.1 Introduction

The I2C bus is 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. Figure 12-1 shows a block diagram of the two I2C interfaces.

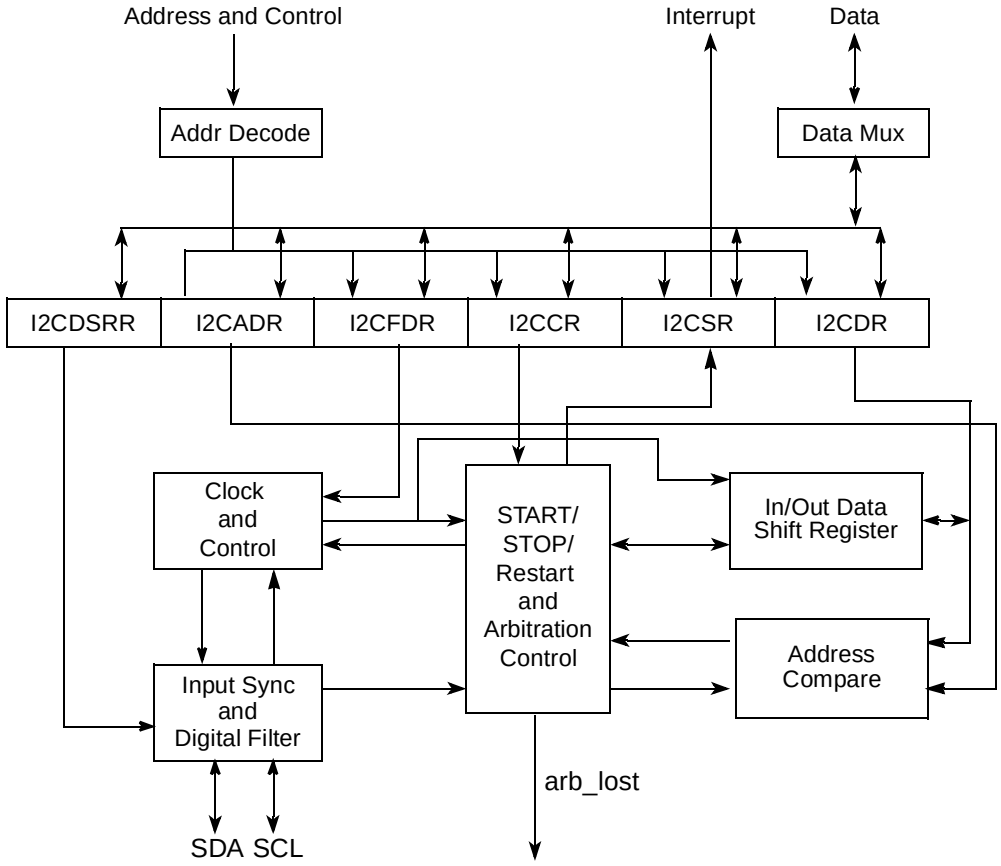


Figure 12-1. I2C Block Diagram

12.1.1 Overview

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.

12.1.2 Features

The I²C interface includes the following features:

- Two-wire interface
- Multiple-master operation
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

12.1.3 Modes of Operation

The I²C units on this device can operate in one of the following modes:

- Master mode—The I²C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode—The I²C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I²C master is detected.
- Interrupt-driven byte-to-byte data transfer—When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode—This mode can be used to initialize the configuration registers in the device after the I²C1 module is initialized. Note that the device powers up with boot sequencer mode disabled as a default, but this mode can be selected with the cfg_boot_seq[0:1] power-on reset (POR) configuration signals that are located on the LGPL3 and LGPL5 signals.

Additionally, the following three I²C-specific states are defined for the I²C interface:

- START condition—This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.
- Repeated START condition—A START condition that is generated without a STOP condition to terminate the previous transfer.
- STOP condition—The master can terminate the transfer by generating a STOP condition to free the bus.

12.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

12.2.1 Signal Overview

The I²C interface uses the SDA and SCL signals, described in Table 12-1, for data transfer. Note that the signal patterns driven on SDA represent address, data, or read/write information at different stages of the protocol.

Table 12-1. I²C Interface Signal Descriptions

Signal Name	Idle State	I/O	State Meaning
Serial Clock (IIC*n*_SCL)	HIGH	I	When the I²C module is idle or acts as a slave, SCL defaults as an input. The unit uses SCL to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.
		O	As a master, the I²C module drives SCL along with SDA when transmitting. As a slave, the I²C module drives SCL low for data pacing.
Serial Data (IIC*n*_SDA)	HIGH	I	When the I²C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I²C devices on SDA. The bus is assumed to be busy when SDA is detected low.
		O	When writing as a master or slave, the I²C module drives data on SDA synchronous to SCL.

12.2.2 Detailed Signal Descriptions

SDA and SCL, described in Table 12-2, serve as a communication interconnect with other devices. All devices connected to these two signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the device hardware specifications for the electrical characteristics of these signals.

Table 12-2. I²C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description	
IIC*n*_SCL	I/O	Serial clock. Performs as an input when the device is programmed as an I²C slave. SCL also performs as an output when the device is programmed as an I²C master.	
	O	As outputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—Driven along with SDA as the clock for the data.
	I	As inputs for the bidirectional serial clock, these signals operate as described below.	
		State Meaning	Asserted/Negated—The I²C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.
IIC*n*_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 (as an I²C master or a slave).	
	O	As outputs for the bidirectional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated— Data is driven.
	I	As inputs for the bidirectional serial data, these signals operate as described below.	
		State Meaning	Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

12.3 Memory Map/Register Definition

Table 12-3 lists the I²C-specific registers 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 block base address and offset listed in Table 12-3. The offsets to the memory map table are defined for both I²C interfaces. That is, I²C1 starts at address offset 0x000, and I²C2 starts at address offset 0x100. The registers for I²C1 are listed in Table 12-3, but the registers for I²C2 are not. Note that the registers are the same for I²C2 except that the offsets change from 0x0*nn* to 0x1*nn*.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

All I²C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

Table 12-3. I²C Memory Map

Offset	I²C Register	Access	Reset	Section/Page
I²C1 Registers				
Block Base Address: 0x0_3000				
0x000	I2CADR—I²C address register	R/W	0x00	12.3.1.1/12-6
0x004	I2CFDR—I²C frequency divider register	R/W	0x2C	12.3.1.2/12-6
0x008	I2CCR—I²C control register	Mixed	0x00	12.3.1.3/12-7
0x00C	I2CSR—I²C status register	Mixed	0x81	12.3.1.4/12-9
0x010	I2CDR—I²C data register	R/W	0x00	12.3.1.5/12-10
0x014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	12.3.1.6/12-11
I²C2 Registers				
Block Base Address: 0x0_3000				
0x100–0x114	I²C2 Registers[1]			

1 I²C2 has the same memory-mapped registers that are described for I²C1 from 0x000 to 0x014, except the offsets range from 0x100 to 0x114.

12.3.1 Register Descriptions

This section describes the I²C registers in detail.

NOTE

Reserved bits should always be written with the value they returned when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero.

This note does not apply to the I²C data register (I2CDR).

12.3.1.1 I²C Address Register (I2CADR)

Figure 12-2 shows the I2CADR register, which contains the address to which the I²C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I²C module is in master mode.

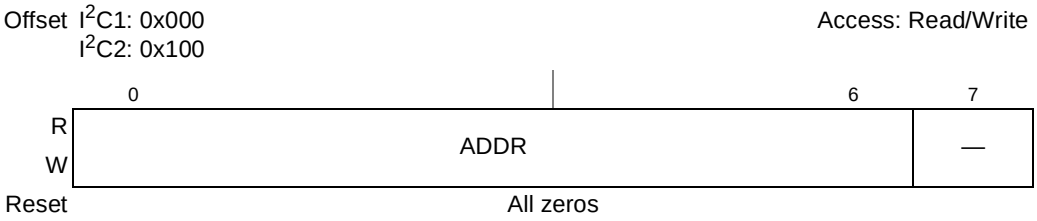


Figure 12-2. I²C Address Register (I2CADR)

Table 12-4 describes the fields of I2CADR.

Table 12-4. I2CADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I²C interface. Note that the default mode of the I²C interface is slave mode for an address match. Note that an address match is one of the conditions that can cause I2CSR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved

12.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 12-3 shows the bits of the I²C frequency divider register. Refer to application note AN2919, *Determining the I²C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

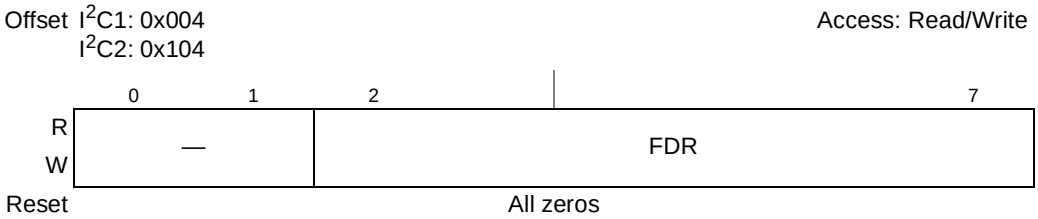


Figure 12-3. I²C Frequency Divider Register (I2CFDR)

Table 12-5 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 12-5. I2CFDR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	FDR	Frequency divider ratio. Used to prescale the clock for bit rate selection. The serial bit clock frequency of SCL is equal to one half the platform (CCB) clock divided by the designated divider. Note that the frequency divider value can be changed at any point in a program. The serial bit clock frequency divider selections are described as follows:

FDR	Divider (Decimal)	FDR	Divider (Decimal)	FDR	Divider (Decimal)
0x00	384	0x16	12288	0x2B	1024
0x01	416	0x17	15360	0x2C	1280
0x02	480	0x18	18432	0x2D	1536
0x03	576	0x19	20480	0x2E	1792
0x04	640	0x1A	24576	0x2F	2048
0x05	704	0x1B	30720	0x30	2560
0x06	832	0x1C	36864	0x31	3072
0x07	1024	0x1D	40960	0x32	3584
0x08	1152	0x1E	49152	0x33	4096
0x09	1280	0x1F	61440	0x34	5120
0x0A	1536	0x20	256	0x35	6144
0x0B	1920	0x21	288	0x36	7168
0x0C	2304	0x22	320	0x37	8192
0x0D	2560	0x23	352	0x38	10240
0x0E	3072	0x24	384	0x39	12288
0x0F	3840	0x25	448	0x3A	14336
0x10	4608	0x26	512	0x3B	16384
0x11	5120	0x27	576	0x3C	20480
0x12	6144	0x28	640	0x3D	24576
0x13	7680	0x29	768	0x3E	28672
0x14	9216	0x2A	896	0x3F	32768
0x15	10240				

12.3.1.3 I²C Control Register (I2CCR)

Figure 12-4 shows the I²C control register, I2CCR.

Offset I²C1: 0x008
I²C2: 0x108

Access: Mixed

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK		—	BCST
W						RSTA		
Reset	All zeros							

Figure 12-4. I²C Control Register (I2CCR)

Table 12-6 describes the bit settings of the I2CCR.

Table 12-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I²C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other control register bits have any effect. All I²C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I²C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I²C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from one to zero, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from zero to one, a START condition is generated on the bus, and master mode is selected.
3	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit is always high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I²C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeated START. Setting this bit always generates a repeated START condition on the bus, provides the device with the current bus master. Attempting a repeated START at the wrong time (or if the bus is owned by another master), results in loss of arbitration. Note that this bit is not readable, which means if a read is performed to I2CCR[RSTA], a zero value is returned. 0 No START condition is generated 1 Generates repeated START condition
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I²C to accept broadcast messages at address zero

12.3.1.4 I²C Status Register (I2CSR)

The I²C status register, shown in Figure 12-5, is read only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

Offset I²C1: 0x00C
I²C2: 0x10C

Access: Mixed

	0	1	2	3	4	5	6	7
R	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
W								
Reset	1	0	0	0	0	0	0	1

Figure 12-5. I²C Status Register (I2CSR)

Table 12-7 describes the bit settings of the I2CSR.

Table 12-7. I2CSR Field Descriptions

Bits	Name	Description
0	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under the following conditions: •When I2CDR is read in receive mode or •When I2CDR is written in transmit mode 1 Byte transfer is completed
1	MAAS	Addressed as a slave. When the value in I2CDR matches with the calling address, this bit is set. The processor is interrupted, if I2CCR[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 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 I²C bus is idle 1 I²C bus is busy
3	MAL	Arbitration lost. Automatically set when the arbitration procedure 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 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. This is also set if this I²C drives an address of all 0s and broadcast mode is enabled.
5	SRW	Slave read/write. When MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave. This bit is valid only when both of the following conditions are true: •A complete transfer occurred and no other transfers have been initiated. •The I²C interface is configured as a slave and has an address match. By checking this bit, the processor can select slave transmit/receive mode according to the command of the master.

Table 12-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] is set). The interrupts for I²C1 and I²C2 are combined into one interrupt, which is sourced by the dual I²C controller. 0 No interrupt is pending. Can be cleared only by software. 1 Interrupt is pending. MIF is set when one of the following events occurs: •One byte of data is transferred (set at the falling edge of the 9th clock). •The value in I2CADR matches with the calling address in slave-receive mode. •Arbitration is lost.
7	RXAK	Received acknowledge. The value of SDA during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

12.3.1.5 I²C Data Register (I2CDR)

The I2C data register is shown in Figure 12-6.

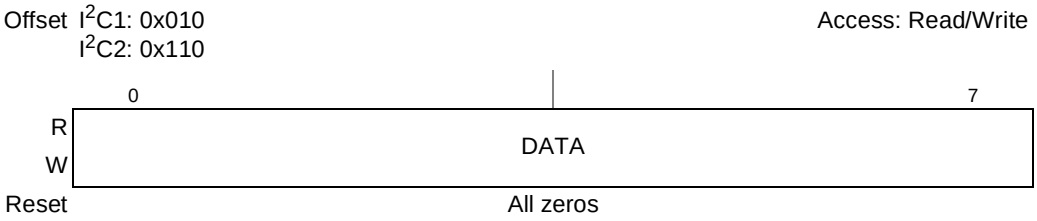


Figure 12-6. I²C Data Register (I2CDR)

Table 12-8 shows the bit descriptions for I2CDR.

Table 12-8. I2CDR Field Descriptions

Bits	Name	Description
0–7	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I²C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In master receive mode, reading the data register allows the read to occur, but also allows the I²C module to receive the next byte of data on the I2C interface. In slave mode, the same function is available after it is addressed. Note that in both master receive and slave receive modes, the very first read is always a dummy read.

12.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 12-7. Refer to application note AN2919, *Determining the I2C Frequency Divider Ratio for SCL*, for additional guidance regarding the proper use of I2CFDR and I2CDFSRR.

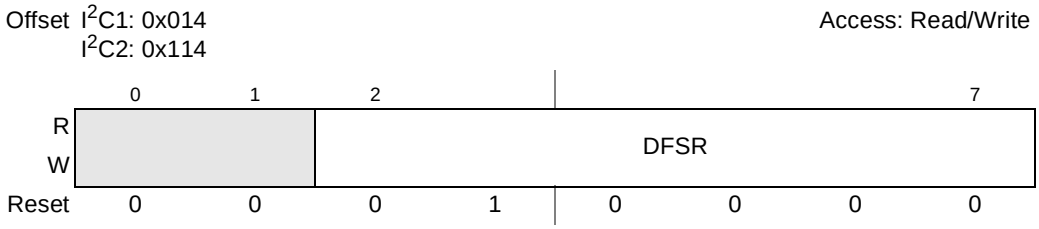


Figure 12-7. I2C Digital Filter Sampling Rate Register (I2CDFSRR)

Table 12-9 shows the field descriptions for I2CDFSRR.

Table 12-9. I2CDFSRR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	DFSR	Digital filter sampling rate. To assist in filtering out signal noise, the sample rate is programmed. This field is used to prescale the frequency at which the digital filter takes samples from the I2C bus. The resulting sampling rate is calculated by dividing one half the platform (CCB clock) frequency by the non-zero value of DFSR.

12.4 Functional Description

The I2C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. After the boot sequencer has completed (when powered up in boot sequencer mode), the I2C interface performs as a slave receiver.

Note that the boot sequencer only functions from the I2C1 interface; the I2C2 interface cannot be used for this purpose.

12.4.1 Transaction Protocol

A standard I2C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 12-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following sections.

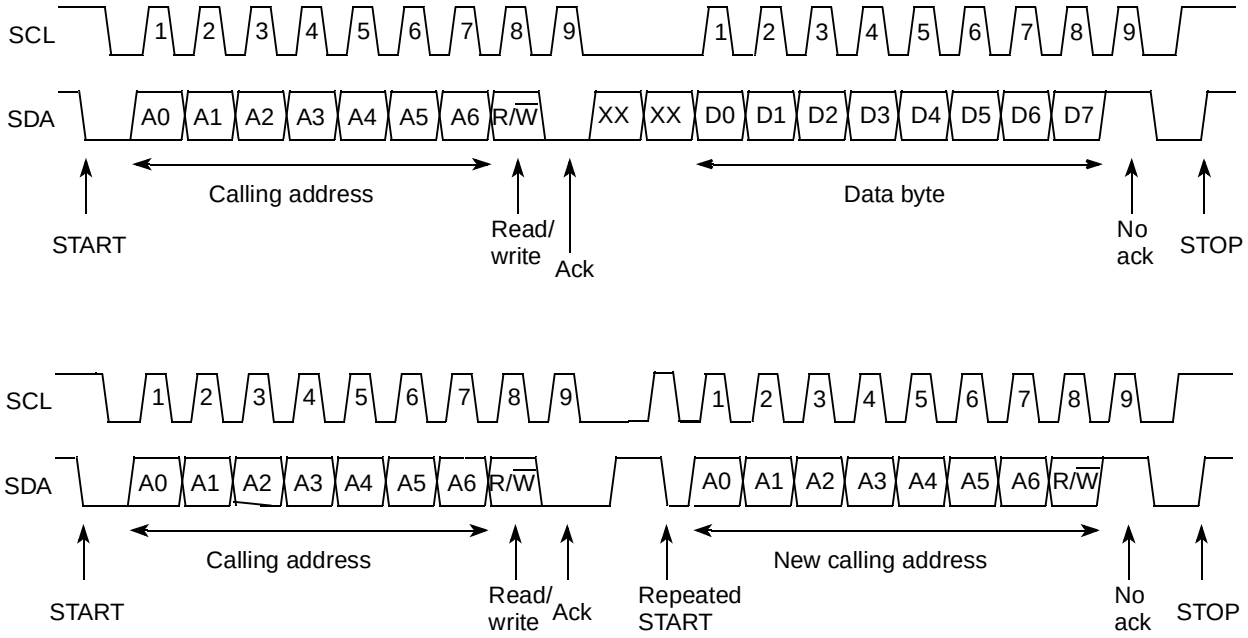


Figure 12-8. I²C Interface Transaction Protocol

12.4.1.1 START Condition

When the I²C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 12-8, a START condition is defined as a high-to-low transition of SDA while SCL is high. This condition denotes the beginning of a new data transfer. Each data transfer can contain several bytes and awakens all slaves. The START condition is initiated by a software write that sets I2CCR[MSTA].

12.4.1.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, when the I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time; if this is attempted, the results are boundedly undefined.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 12-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

The I²C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero; however the I²C module does not check the R/$\overline{W}$ bit. The second byte of the broadcast message is the master 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.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 12-8. There is one clock pulse on SCL for each data bit, and the most significant bit (msb) is transmitted first. Each byte of data must be followed by an acknowledge bit, which is signaled from the receiving device by pulling the SDA line low at the 9th clock. Therefore, one complete data byte transfer takes 9 clock pulses. Several bytes can be transferred during a data transfer session.

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.

12.4.1.3 Repeated START Condition

Figure 12-8 shows a repeated START condition, which is generated without a STOP condition that can terminate the previous transfer. The master uses this method to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

12.4.1.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA signal while SCL is high. For more information, see Figure 12-8. Note that a master can generate a STOP even if the slave has transmitted an acknowledge bit, at which point the slave must release the bus. The STOP condition is initiated by a software write that clears I2CCR[MSTA].

As described in Section 12.4.1.3, “Repeated START Condition,” the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

12.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in this I²C module.

12.4.1.5.1 Transaction Monitoring—Implementation Details

The different conditions of the I²C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when an SDA rise occurs 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.

12.4.1.5.2 Control Transfer—Implementation Details

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 pulled 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 restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Data bit (transmit)
  - Ack bit (receive)
  - START condition
  - STOP condition
  - Restart condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - Ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end

- Slave mode
  - Address cycle
  - Transmit cycle
  - Ack cycle

12.4.1.6 Address Compare—Implementation Details

Address compare block determines if a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The three performed address comparisons are described as follows:

- Whether a broadcast message has been received, to update the I2CSR
- Whether the module has been addressed as a slave, to update the I2CSR and to generate an interrupt
- If the address transmitted by the current master matches the general broadcast address

12.4.2 Arbitration Procedure

The I^2C interface is a true multiple-master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I^2C module) determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I^2C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

If the I^2C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—The I^2C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—The I^2C 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^2C interface losing arbitration, after which bus operations return to normal.

12.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2CSR[MAL] is set) under the following conditions:

- SDA samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA samples low when the master drives high during a data-receive cycle of the acknowledge (Ack) bit (receive).
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.

- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

Note that the I²C module does not automatically retry a failed transfer attempt.

12.4.3 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, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

12.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
  - Transmit slave address after START condition
  - Transmit slave address after restart condition
  - Transmit data
  - Receive data
- Slave mode
  - Transmit data
  - Receive data
  - Receive slave address after START or restart condition

12.4.4.1 Clock Synchronization

Due to the wire 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.

12.4.4.2 Input Synchronization and Digital Filter

The following sections describes the synchronizing of the input signals, and the filtering of the SCL and SDA lines in detail.

12.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

12.4.4.2.2 Filtering of SCL and SDA Lines

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.

The sampling rate is equal to a binary value stored in the frequency register I2CDFSRR. The duration of the sampling cycle is controlled by a down counter. This allows a software write to the frequency register to control the filtered sampling rate.

12.4.4.3 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.

12.4.5 Boot Sequencer Mode

If boot sequencer mode is selected on POR (by the settings on the cfg_boot_seq[0:1] reset configuration signals, as described in Section 4.4.3.8, “Boot Sequencer Configuration”), the I²C1 module communicates with one or more EEPROMs through the I²C interface on IIC1_SCL and IIC1_SDA. The boot sequencer accesses the I²C1 serial ROM device at a serial bit clock frequency equal to the platform (CCB) clock frequency divided by 2560. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device.

If the boot sequencer is enabled for normal I²C addressing mode, the I²C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the 8-bit starting address
5. Generate a repeated START
6. Transmit 0xA1 which is the 7-bit calling address (0b101_0000) with a read command appended (1 as the least significant bit).
7. Receive 256 bytes of data from the EEPROM (unless the CONT bit is cleared in the data structure).
8. Generate a repeated START

9. Transmit 0xA2 which is the 7-bit calling address of the second target (0b101_0001) with a write command appended (0 as the least significant bit).
10. Transmit 0x00 which is the 8-bit starting address for the second target.
11. Generate a repeated START
12. Transmit 0xA3 which is the 7-bit calling address (0b101_0001) with a read command appended (1 as the least significant bit).
13. Receive another 256 bytes of data from the second EEPROM (unless the CONT bit is cleared in the data structure).

The sequence repeats with successive targets until the CONT bit in the data structure is cleared and the CRC check is executed. If the last register is not detected (that is, the CONT bit is never cleared) before wrapping back to the first address, an error condition is detected, causing the device to hang and the $\overline{\text{HRESET\_REQ}}$ signal to assert externally. The I²C module continues to read from the EEPROM(s) as long as the continue (CONT) bit is set in the EEPROM(s). The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in Section 12.4.5.1, "EEPROM Calling Address." There should be no other I²C traffic when the boot sequencer is active.

The boot sequencer mode also supports an extension of the standard I²C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes, and this extended addressing mode is selectable during POR with a different encoding on the cfg_boot_seq[0:1] reset configuration signals. In this mode, only one EEPROM device may be used, and the maximum number of registers is limited by the size of the EEPROM. If the boot sequencer is enabled for extended I²C addressing mode, the I²C interface initiates the following sequence during reset:

1. Generate RESET sequence (START then 9 SCL cycles) to the EEPROM twice. This clears any transactions that may have been in progress prior to the reset.
2. Generate START
3. Transmit 0xA0 which is the 7-bit calling address (0b101_0000) with a write command appended (0 as the least significant bit).
4. Transmit 0x00 which is the high-order starting address
5. Transmit 0x00 which is the low-order starting address
6. Generate a repeated START
7. Transmit 0xA1 which is the 7-bit calling address (0b101_0000) with a read command appended (1 as the least significant bit).
8. Receive data continuously from the EEPROM until the CONT bit is cleared and the CRC check is executed. See Section 12.4.5.2, "EEPROM Data Format," for more information.

Note that as described in Section 4.4.3.8, "Boot Sequencer Configuration," the default value for the cfg_boot_seq[0:1] reset configuration pins is 0b11, which corresponds to the I²C boot sequencer being disabled at power-up.

12.4.5.1 EEPROM Calling Address

The MPC8572E uses 0b101_0000 for the EEPROM calling address. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. If more EEPROMs are used, they are addressed in sequential order.

12.4.5.2 EEPROM Data Format

The I²C module expects that a particular data format be used for data in the EEPROM. A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C module checks to ensure that this preamble is correctly detected before proceeding further. Following the preamble, there should be a series of configuration registers (known as register preloads) programmed into the EEPROM. Each configuration register should be programmed according to a particular format, as shown in Figure 12-9. The first 3 bytes hold the attributes and address offset, as follows. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302.

After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of the transaction. Byte enables should be asserted for any byte that is written to the configuration register, and they should be asserted contiguously, creating a 1-, 2-, or 4-byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the LSB of data (data[24:31]).

By setting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If CONT is cleared, the first 3 bytes, including ACS, the byte enables, and the address, must also be cleared. Also, the data contains the final cyclic redundancy check (CRC). A CRC-32 algorithm is used to check the integrity of the data. The polynomial used is:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC should cover all bytes stored in the EEPROM prior to the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros). If a preamble or CRC fail is detected, the device hangs and the external $\overline{\text{HRESET\_REQ}}$ signal asserts. If there is a preamble fail, the boot sequencer may continue to pull I²C pins low until a hard reset occurs.

0	1	4	5	6	7
ACS	BYTE_EN		CONT	ADDR[0–1]	
ADDR[2–9]					
ADDR[10–17]					
DATA[0–7]					
DATA[8–15]					
DATA[16–23]					
DATA[24–31]					

Figure 12-9. EEPROM Data Format for One Register Preload Command

Figure 12-10 shows an example of the EEPROM contents, including the preamble, data format, and CRC.

0	1	2	3	4	5	6	7	
1	0	1	0	1	0	1	0	Preamble
0	1	0	1	0	1	0	1	
1	0	1	0	1	0	1	0	
ACS	BYTE_EN				1	ADDR[0–1]		First Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
ACS	BYTE_EN				1	ADDR[0–1]		Second Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
. . .								

Figure 12-10. EEPROM Contents

ACS	BYTE_EN				1	ADDR[0–1]		Last Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
CRC[0–7]								Cyclic Redundancy Check
CRC[8–15]								
CRC[16–23]								
CRC[24–31]								

Figure 12-10. EEPROM Contents (continued)

12.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. Figure 12-11 is a recommended flowchart for I²C interrupt service routines.

The I²C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. This appropriate byte swapping is needed as I²C registers are byte registers. Also, an **msync** assembly instruction must be executed after each I²C register read/write access to guarantee in-order execution.

The I²C controller does not guarantee its recovery from all illegal I²C bus activity. In addition, 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.

12.5.1 Initialization Sequence

A hard reset initializes all the I²C registers to their default states. The following initialization sequence initializes the I²C unit:

1. All I²C registers must be located in a cache-inhibited page.
2. Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CCB (platform) clock. Note that the platform frequency must first be divided by two; see Section 12.3.1.2, "I²C Frequency Divider Register (I2CFDR)," for more details.
3. Update I2CADR to define the slave address for this device.

4. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CCR[MEN] to enable the I²C interface.

12.5.2 Generation of 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 free (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.

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 (provided interrupt reporting is enabled with I2CCR[MIEN] =1) so that the I²C interrupt handler can handle the interrupt. Note that the interrupts for I²C1 and I²C2 are combined into one interrupt, which is sourced by the dual I²C controller.

12.5.3 Post-Transfer Software Response

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. The I²C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set). 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 causes I2CSR[MCF] to be cleared. See Section 12.5.8, “Interrupt Service Routine Flowchart.”

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 Section 12.5.8, “Interrupt Service Routine Flowchart.”

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.

During slave-mode address cycles (I2CSR[MAAS] is set), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CSR[SRW] is not valid and I2CCR[MTX] must be read to determine the direction of the current transfer. See Section 12.5.8, “Interrupt Service Routine Flowchart,” for more details.

12.5.4 Generation of STOP

A data transfer ends with a STOP condition generated by the master device. A master transmitter can generate a STOP condition after all the data has been transmitted.

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data (by setting the transmit acknowledge bit (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I²C interface, so the last byte does not receive the data acknowledge (because I2CCR[TXAK] is set). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I²C controller automatically generates a STOP if I2CCR[TXAK] is set. Therefore, I2CCR[TXAK] must be set before allowing the I²C module to receive the last data byte on the I²C bus. Eventually, I2CCR[TXAK] needs to be cleared again for subsequent I²C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

12.5.5 Generation of Repeated START

At the end of a data transfer, if the master still wants to communicate 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].

12.5.6 Generation of SCL When SDA Low

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 that the bus is busy). This can occur when a system reset does not cause all I²C devices to be reset. Thus, SDA can be driven low by another I²C device while this I²C module is coming out of reset and stays 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 and set the master bit by setting I2CCR to 0x20
2. Enable the I²C module by setting I2CCR to 0xA0
3. Read the I2CDR
4. Return the I²C module to slave mode by setting I2CCR to 0x80

12.5.7 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the module addressed as a slave should be tested to check if 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/$\overline{W}$ 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 where 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.

12.5.7.1 Slave Transmitter and Received Acknowledge

In the slave transmitter routine, the received acknowledge bit (I2CSR[RXAK]) must be tested before sending the next byte of data. The master signals an end-of-data by not acknowledging the data transfer from the slave. When no acknowledge is received (I2CSR[RXAK] is set), the slave transmitter interrupt routine must clear I2CCR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2CDR then releases SCL so that the master can generate a STOP condition. See Section 12.5.8, "Interrupt Service Routine Flowchart."

12.5.7.2 Loss of Arbitration and Forcing of 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 9th clock of this transfer

Thus, the slave interrupt service routine should first test I2CSR[MAL] and software should clear it if it is set. See Section 12.4.2.1, "Arbitration Control," for more information.

12.5.8 Interrupt Service Routine Flowchart

Figure 12-11 shows an example algorithm for an I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior. However, in the slave receive mode the interrupt service routine may need to set I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that an **msync** instruction follow each I^2C register read or write to guarantee in-order instruction execution.

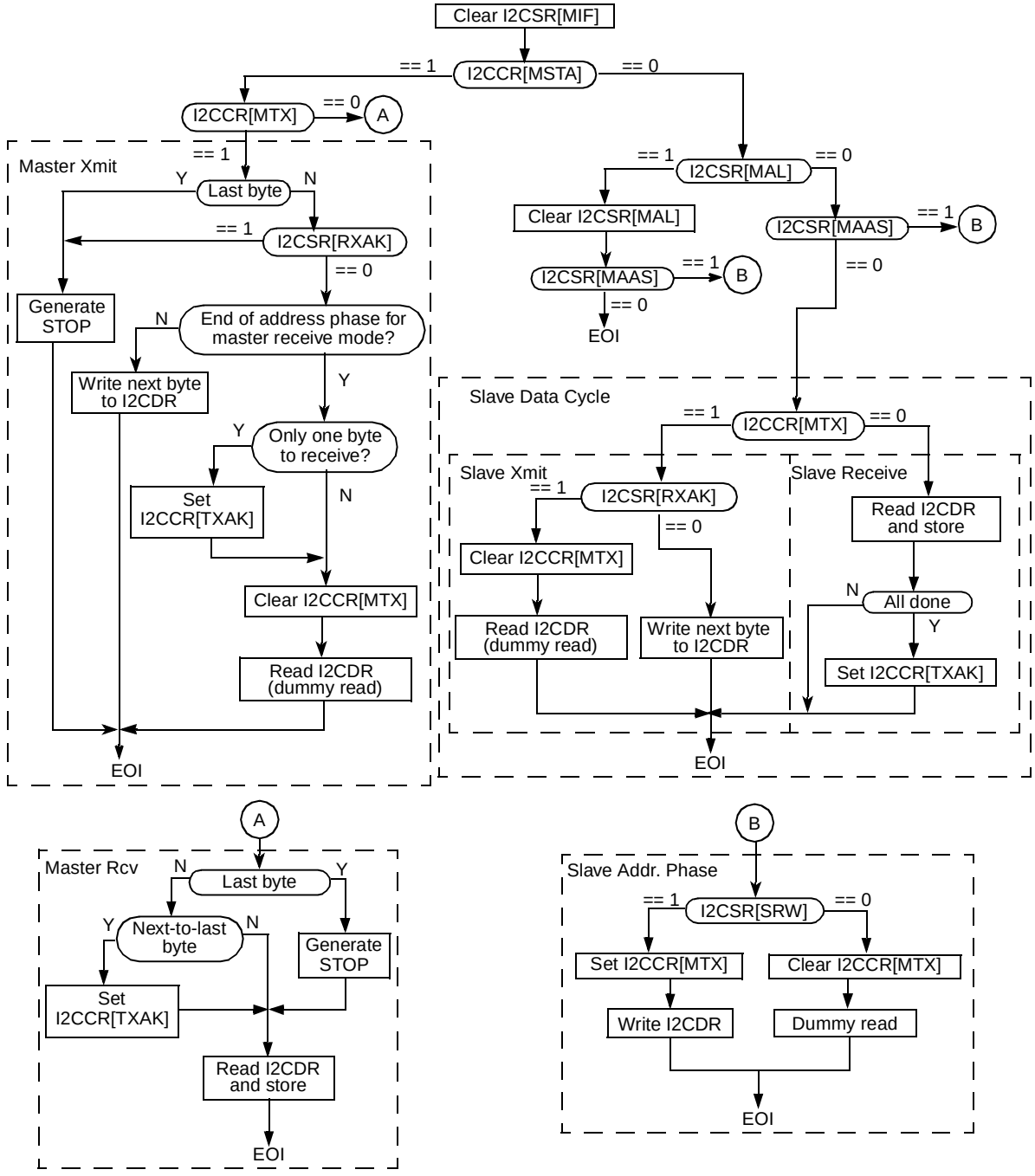


Figure 12-11. Example I²C Interrupt Service Routine Flowchart

Chapter 13
DUART

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.

13.1 Overview

The DUART consists of two universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the platform (CCB) clock. 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 13-1, each UART module consists of the following:

- Receive and transmit buffers
- Clear to send ($\overline{\text{CTS}}$) input port and request to send ($\overline{\text{RTS}}$) output port for data flow control
- 16-bit counter for baud rate generation
- Interrupt control logic

13.1.1 Features

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the 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 platform clock by 1 to ($2^{16} - 1$) and generate a 16x clock for the transmitter and receiver engines

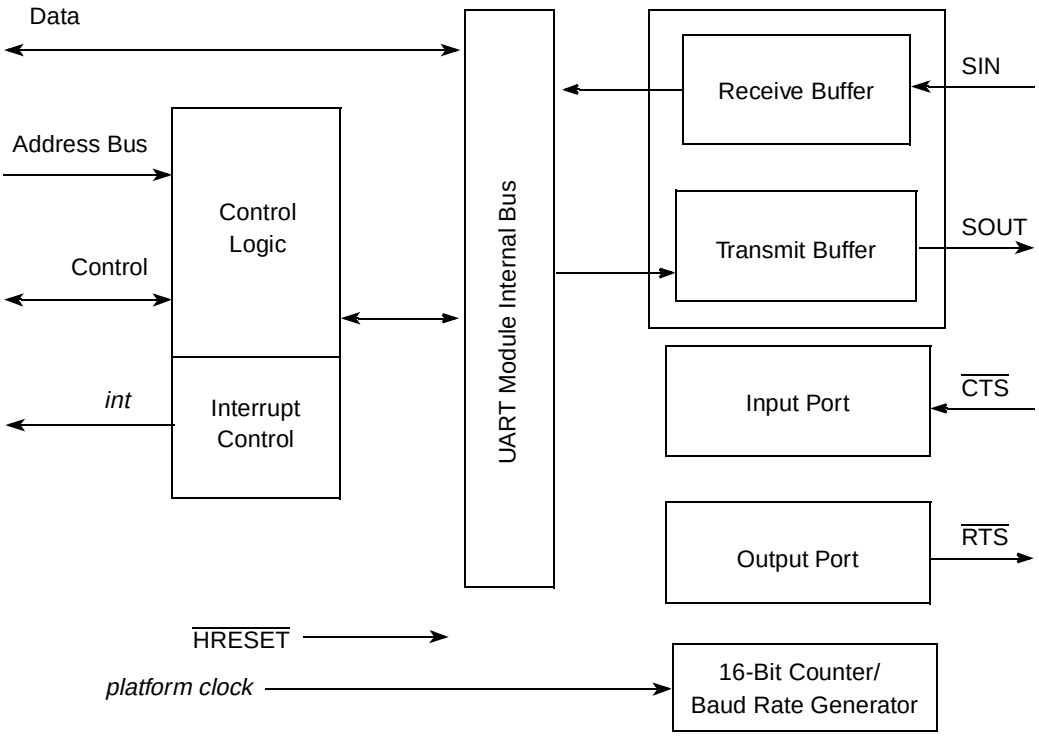


Figure 13-1. UART Block Diagram

- Clear to send ($\overline{\text{CTS}}$) and ready to send ($\overline{\text{RTS}}$) 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

13.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the platform clock.

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.

13.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

13.2.1 Signal Overview

The DUART signals are described in Table 13-1. 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.

13.2.2 Detailed Signal Descriptions

Table 13-1 provided detailed descriptions of the external DUART signals.

Table 13-1. DUART Signals—Detailed Signal Descriptions

Signal	I/O	Description	
UART_SIN[0:1]	I	Serial data in. Data is received on the receivers of UART0 and UART1 through the respective serial data input signal, with the least-significant bit received first.	
		State Meaning	Asserted/Negated—Represents the data being received on the UART interface.
		Timing	Assertion/Negation—An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.
UART_SOUT[0:1]	O	Serial data out. The serial data output signals for the UART0 and UART1 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.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation— An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.
$\overline{\text{UART\_CTS}}$[0:1]	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective $\overline{\text{RTS}}$ outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal.	
		State Meaning	Asserted/Negated—Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation—Sampled at the rising edge of every platform clock.
$\overline{\text{UART\_RTS}}$[0:1]	O	Request to send. $\overline{\text{UART_RTS}}$x are active-low output signals that can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send ($\overline{\text{CTS}}$) input of a transmitter, this signal can be used to control serial data flow.	
		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 platform clock.

13.3 Memory Map/Register Definition

Table 13-2 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 listed in Table 13-2.

There are two complete sets of DUART registers (one for each UART). The two 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 UART0 or UART1.

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 Section 13.3.1.7, "Line Control Registers (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. Table 13-2 provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 13-2. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
Block Base Address: 0x0_4000				
UART0 Registers				
0x500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	13.3.1.1/13-5
0x500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	13.3.1.2/13-5
0x500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	13.3.1.3/13-6
0x501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	13.3.1.4/13-8
0x501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	13.3.1.3/13-6
0x502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	13.3.1.5/13-8
0x502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	13.3.1.6/13-10
0x502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	13.3.1.12/13-16
0x503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	13.3.1.7/13-11
0x504	UMCR—ULCR[DLAB] = x UART0 modem control register	R/W	0x00	13.3.1.8/13-13
0x505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	13.3.1.9/13-14
0x506	UMSR—ULCR[DLAB] = x UART0 modem status register	R	0x00	13.3.1.10/13-15
0x507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	13.3.1.11/13-16
0x510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	13.3.1.13/13-17

Table 13-2. DUART Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
UART1 Registers				
0x600–0x610	UART1 Registers[1]			

[1] UART1 has the same memory-mapped registers that are described for UART0 from 0x500 to 0x510, except the offsets range from 0x600 to 0x610.

13.3.1 Register Descriptions

The following sections describe the UART*n* registers.

13.3.1.1 Receiver Buffer Registers (URBR*n*) (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, Section 13.3.1.9, "Line Status Registers (ULSRn)." Figure 13-3 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

Figure 13-2 shows the bits in the URBRs.

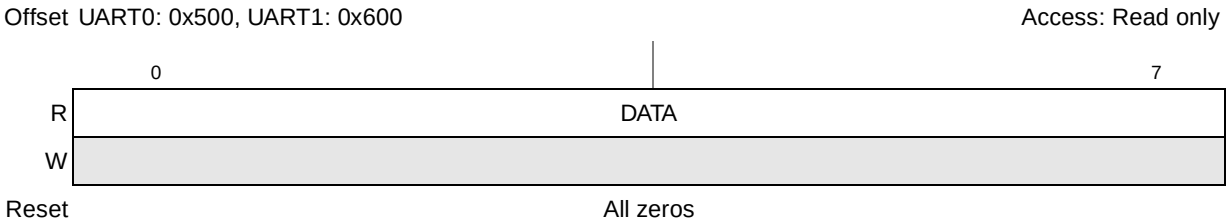


Figure 13-2. Receiver Buffer Registers (URBR*n*)

Table 13-3 describes the fields of URBR.

Table 13-3. URBR Field Descriptions

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus (read only)

13.3.1.2 Transmitter Holding Registers (UTHR*n*) (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[$\overline{\text{TXRDY}}$] indicates when the FIFO is full. Refer to the Table 13-20 and the Table 13-21 for more details.

Figure 13-3 shows the bits in the UTHRs.

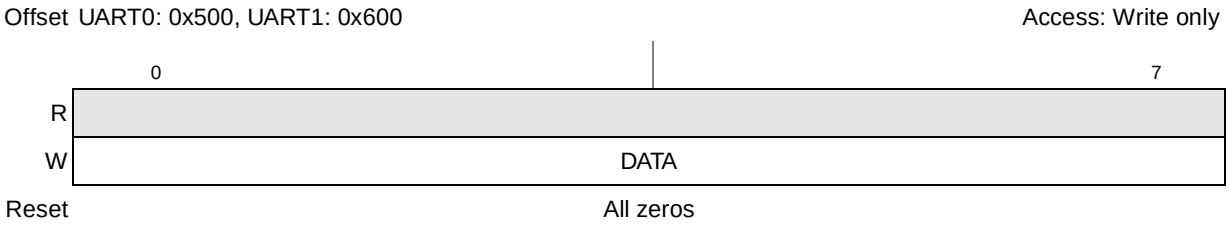


Figure 13-3. Transmitter Holding Registers (UTHR*n*)

Table 13-4 describes the fields of UTHR.

Table 13-4. UTHR Field Descriptions

Bits	Name	Description
0–7	DATA	Data that is written to UTHR (write only)

13.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB) (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 = platform clock frequency / (16 × [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency / desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 13-7.

Figure 13-4 shows the bits in the UDMBs.

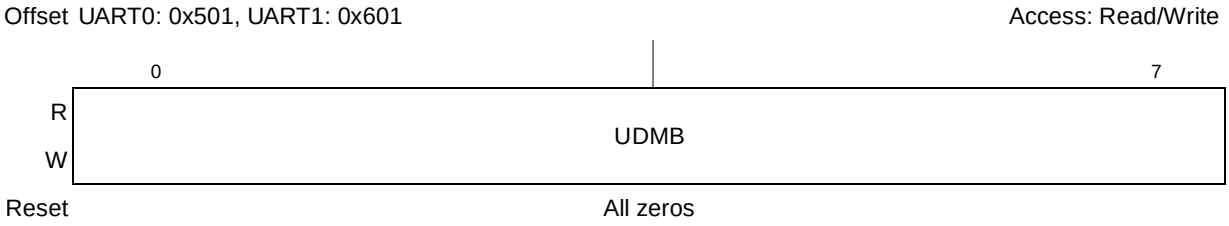


Figure 13-4. Divisor Most Significant Byte Registers (UDMB0, UDMB1)

Table 13-5 describes the fields of UDMB registers.

Table 13-5. UDMB Field Descriptions

Bits	Name	Description
0–7	UDMB	Divisor most-significant byte

Figure 13-5 shows the bits in the UDLBs.

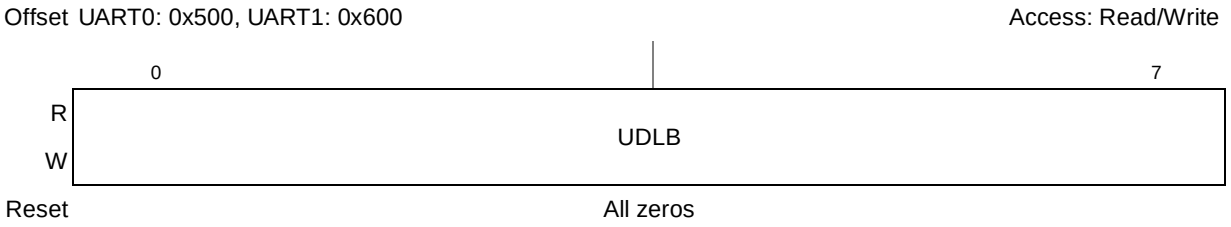


Figure 13-5. Divisor Least Significant Byte Registers (UDLB*n*)

Table 13-6 describes the fields of UDLB registers.

Table 13-6. UDLB Field Descriptions

Bits	Name	Description
0–7	UDLB	Divisor least significant byte. This is concatenated with UDMB.

Table 13-7 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 13-7. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		Platform Clock (CCB) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	2170	87A	333	9600.61444	0.0064
19,200	1085	43D	333	19,201.22888	0.0064
38,400	543	21F	333	38,367.09638	0.0858
57,600	362	16A	333	57,550.64457	0.0857
115,200	181	B5	333	115,101.28913	0.0857
230,400	90	5A	333	231,481.48148	0.4694
9,600	3472	D90	533	9600.61444	0.0064
19,200	1736	6C8	533	19,201.22888	0.0064
38,400	868	364	533	38,402.45776	0.0064
57,600	579	243	533	57,570.52389	0.0512
115,200	289	121	533	115,340.25375	0.1217
230,400	145	91	533	229,885.05747	0.2235

13.3.1.4 Interrupt Enable Register (UIER) (ULCR[DLAB] = 0)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Figure 13-6 shows the bits in the UIER.

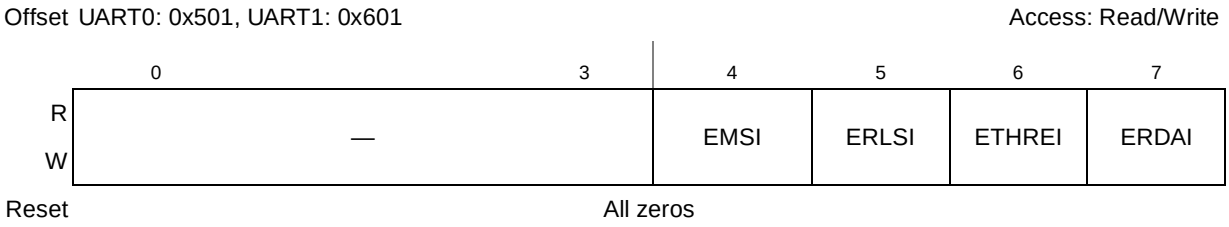


Figure 13-6. Interrupt Enable Register (UIER)

Table 13-8 describes the fields of UIER.

Table 13-8. UIER Field Descriptions

Bits	Name	Description
0–3	—	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

13.3.1.5 Interrupt ID Registers (UIIR*n*) (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 Table 13-10 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.

Figure 13-7 shows the bits in the UIIR.

Offset UART0: 0x502, UART1: 0x602 Access: Read only

	0	1	2	3	4	5	6	7
R	FE		—		IID3	IID2	IID1	IID0
W								
Reset	0	0	0	0	0	0	0	1

Figure 13-7. Interrupt ID Registers (UIIR)

Table 13-9 describes the fields of the UIIR.

Table 13-9. UIIR Field Descriptions

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 13-10. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5–6	IID2–1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 13-10.
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.

The bits contained in the UIIR registers are described in Table 13-10.

Table 13-10. 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.

Table 13-10. UIIR IID Bits Summary (continued)

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
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	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Read the UMSR.

13.3.1.6 FIFO Control Registers (UFCR*n*) (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.

Figure 13-8 shows the bits in the UFCRs.

Offset UART0: 0x502, UART1: 0x602 Access: Write only

	0	1	2	3	4	5	6	7
R			—					
W	RTL				DMS	TFR	RFR	FEN
Reset	All zeros							

Figure 13-8. FIFO Control Registers (UFCR*n*)

Table 13-11 describes the fields of the UFCRs.

Table 13-11. UFCR Field Descriptions

Bits	Name	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	—	Reserved

Table 13-11. UFCR Field Descriptions (continued)

Bits	Name	Description
4	DMS	DMA mode select. See Section 13.4.5.2, "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
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

13.3.1.7 Line Control Registers (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 Table 13-13 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.

Figure 13-9 shows the bits in the ULCRs.

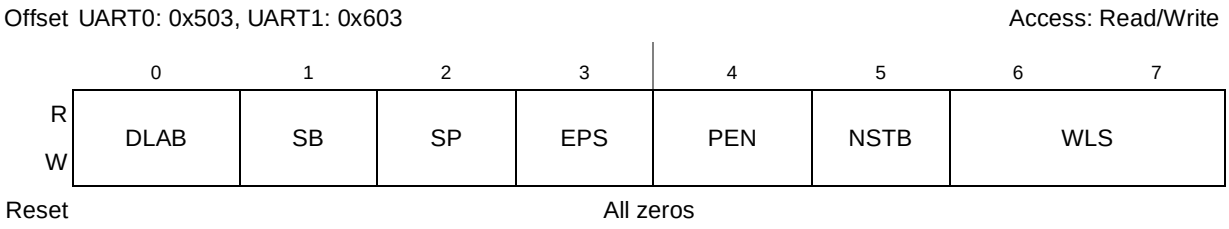


Figure 13-9. Line Control Register (ULCR)

Table 13-12 describes the fields of the ULCRs.

Table 13-12. ULCR Field Descriptions

Bits	Name	Description
0	DLAB	Divisor latch access bit. 0 Access to all registers except UDLB, UAFR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
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. 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 Table 13-13 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	NTSB	Number of STOP bits. 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1∫ 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

Table 13-13. 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

13.3.1.8 Modem Control Registers (UMCR*n*)

The UMCRs control the interface with the external peripheral device on the UART bus.

Figure 13-10 shows the bits in the UMCRs

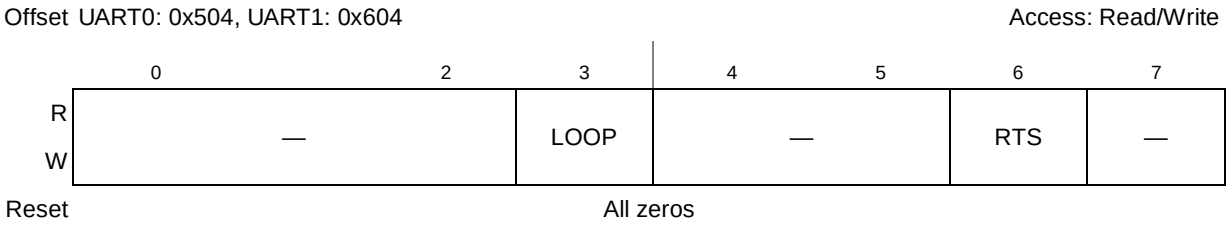


Figure 13-10. Modem Control Register (UMCR)

Table 13-14 describes the fields of UMCRs.

Table 13-14. UMCR Field Descriptions

Bits	Name	Description
0–2	—	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].
4–5	—	Reserved
6	RTS	Ready to send. 0 Negates corresponding $\overline{\text{UART\_RTS}}$ output 1 Assert corresponding $\overline{\text{UART\_RTS}}$ output. Informs external modem or peripheral that the UART is ready for sending/receiving data
7	—	Reserved

13.3.1.9 Line Status Registers (ULSR*n*)

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.

Figure 13-11 shows the bits in the ULSRs.

Offset UART0: 0x505, UART1: 0x605 Access: Read only

	0	1	2	3	4	5	6	7
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0	1	1	0	0	0	0	0

Figure 13-11. Line Status Register (ULSR)

Table 13-15 describes the fields of the ULSRs.

Table 13-15. ULSR Field Descriptions

Bits	Name	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.
3	BI	Break interrupt. 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	Framing error. 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.

Table 13-15. ULSR Field Descriptions (continued)

Bits	Name	Description
5	PE	Parity error. 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	Overrun error. 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	Data ready. 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.

13.3.1.10 Modem Status Registers (UMSR*n*)

The UMSRs track the status of the modem (or external peripheral device) clear to send ($\overline{\text{CTS}}$) signal for the corresponding UART.

Figure 13-12 shows the bits in the UMSRs.

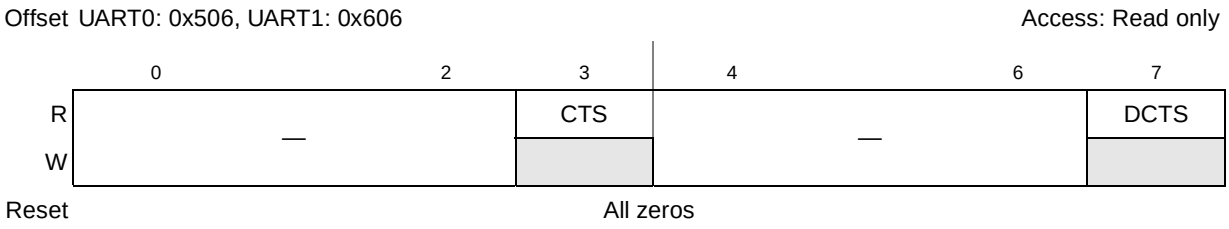


Figure 13-12. Modem Status Register (UMSR)

Table 13-16 describes the fields of the UMSRs.

Table 13-16. UMSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	CTS	Clear to send. Represents the inverted value of the $\overline{\text{CTS}}$ input pin from the external peripheral device 0 Corresponding $\overline{\text{CTS}}n$ is negated 1 Corresponding $\overline{\text{CTS}}n$ is asserted. The modem or peripheral device is ready for data transfers.
4–6	—	Reserved
7	DCTS	Clear to send 0 No change on the corresponding $\overline{\text{CTS}}n$ signal since the last read of UMSR[CTS] 1 The $\overline{\text{CTS}}n$ value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition

13.3.1.11 Scratch Registers (USCR*n*)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Figure 13-13 shows the bits in USCRs.

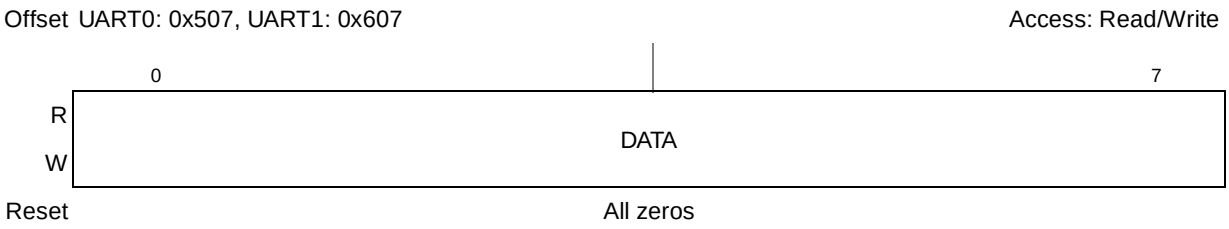


Figure 13-13. Scratch Register (USCR)

Table 13-17 describes the fields of the USCRs.

Table 13-17. USCR Field Descriptions

Bits	Name	Description
0–7	DATA	Data

13.3.1.12 Alternate Function Registers (UAFR*n*) (ULCR[DLAB] = 1)

The UAFRs give software the ability to gate off the baud clock and write to both UART0/UART1 registers simultaneously with the same write operation.

Figure 13-14 shows the bits in the UAFRs.

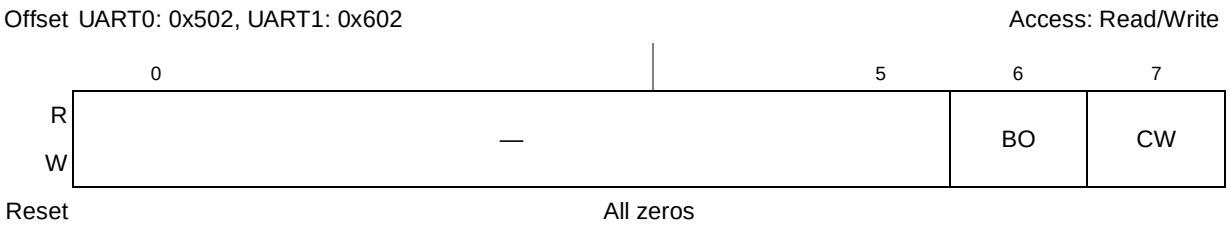


Figure 13-14. Alternate Function Register (UAFR)

Table 13-18 describes the fields of the UAFRs.

Table 13-18. UAFR Field Descriptions

Bits	Name	Description
0–5	—	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 writing to both UART0 and UART1 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART0 is also a write to the corresponding register in UART1 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 0x*n*00, 0x*n*01 and 0x*n*02, where *n* is the offset of the corresponding UART.

13.3.1.13 DMA Status Registers (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.

Figure 13-15 shows the bits in UDSRs.

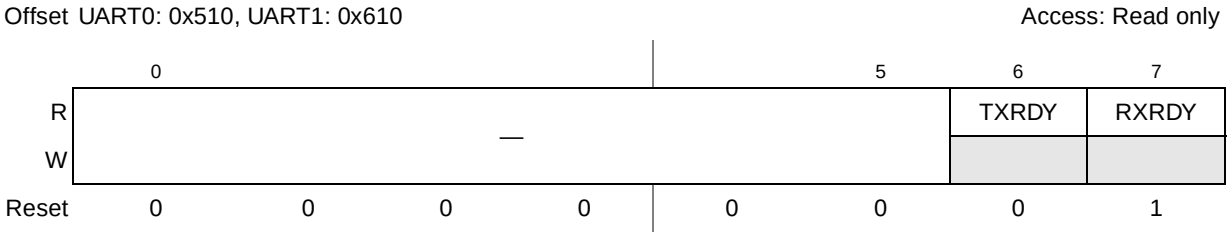


Figure 13-15. DMA Status Register (UDSR)

Table 13-19 describes the fields of the UDSRs.

Table 13-19. UDSR Field Descriptions

Bits	Name	Description
0–5	—	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 in Table 13-21. 1 This bit is set, as shown in Table 13-20.
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 in Table 13-23. 1 This bit is set, as shown in Table 13-22.

Table 13-20. 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 13-21. 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 13-22. 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 13-23. 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	
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.

13.4 Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the platform clock signal.

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 Section 13.4.5, “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.

13.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 13-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

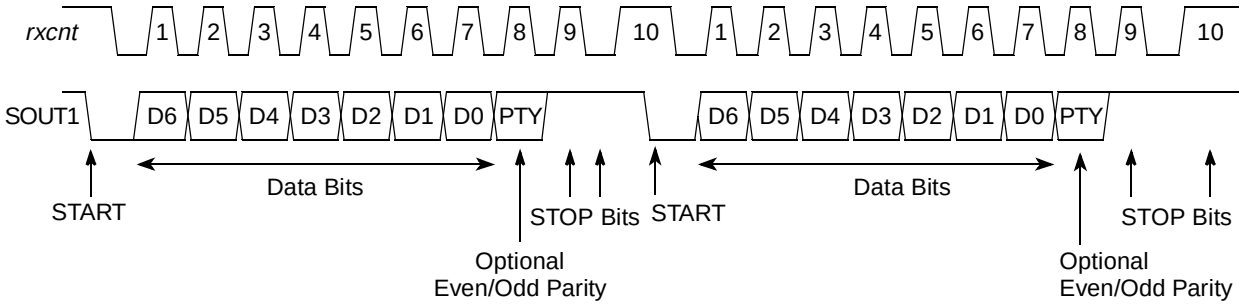


Figure 13-16. UART Bus Interface Transaction Protocol Example

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.

13.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 13-16 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.

13.4.1.2 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.

13.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 13.3.1.7, "Line Control Registers (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 Section 13.3.1.9, "Line Status Registers (ULSRn).")

13.4.1.4 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.

13.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the platform clock input and dividing the input 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:

$$\text{Baud rate} = (1/16) \times (\text{platform clock frequency/divisor value})$$

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.

13.4.3 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 $\overline{\text{CTS}}$ (input signal) is disconnected, $\overline{\text{RTS}}$ is internally connected to $\overline{\text{CTS}}$, and the $\overline{\text{RTS}}$ (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).

13.4.4 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.

13.4.4.1 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.

13.4.4.2 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.

13.4.4.3 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.

13.4.5 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.

13.4.5.1 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 Section 13.3.1.4, “Interrupt Enable Register (UIER) (ULCR[DLAB] = 0),” 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[IID]) is cleared when the URBR is read. See Section 13.3.1.5, “Interrupt ID Registers (UIIRn) (ULCR[DLAB] = 0),” for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

13.4.5.2 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 Section 13.3.1.13, “DMA Status Registers (UDSRn),” for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

13.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 0 (UIIR[0]), is cleared. The interrupt enable register (UIER) is used to mask specific interrupt types. For more details refer to the description of UIER in Section 13.3.1.4, “Interrupt Enable Register (UIER) (ULCR[DLAB] = 0).”

When the interrupts are disabled in UIER, polling software can not use UIIR[0] 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[0] can be used for polling if the interrupts are enabled in UIER.

13.5 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 0b01X1.)
- 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. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external modem or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR if the interrupts generated by the DUART are masked.

Chapter 14
Enhanced Local Bus Controller

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.

14.1 Introduction

Figure 14-1 is a functional block diagram of the eLBC, which supports three interfaces: GPCM, FCM, and UPM controllers.

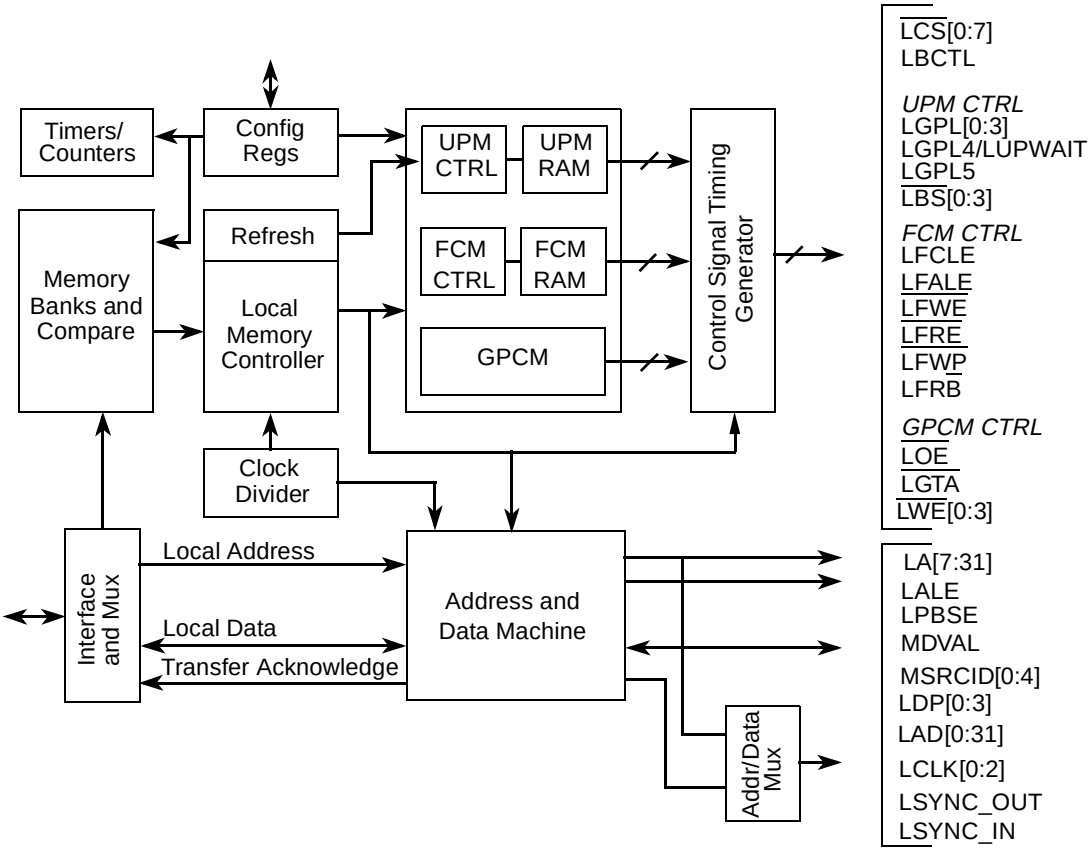


Figure 14-1. Enhanced Local Bus Controller Block Diagram

14.1.1 Overview

The main component of the eLBC is its memory controller, which provides a seamless 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.

14.1.2 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 including read-modify-write (RMW) parity for single accesses
  - Write-protection capability
  - Atomic operation
  - 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-, 16-, and 32-bit devices
  - Minimum three-clock access to external devices
  - Four byte-write-enable signals ($\overline{\text{LWE}}$[0:3])
  - Output enable signal ($\overline{\text{LOE}}$)
  - External access termination signal ($\overline{\text{LGTA}}$)
- 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
  - Boot chip-select support for 8-bit devices

  - Dual 2-Kbyte/eight 512-byte buffers allow simultaneous data transfer during flash reads and programming
  - 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-, 16-, and 32-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)
- Support for phase-locked loop (PLL) with software-configurable bypass for low frequency bus clocks

14.1.3 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.

14.1.3.1 eLBC Bus Clock and Clock Ratios

The eLBC supports ratios of 4, 8, and 16 between the faster internal (system) clock and slower external bus clock (LCLK[0:2]). 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[0:2], to allow the clock load to be shared equally across a set of signal nets, thereby enhancing the edge rates of the bus clock.

14.1.3.2 Source ID Debug Mode

The eLBC provides the ID of a transaction source on external device pins. When those pins are selected, the 5-bit internal ID of the current transaction source appears on MSRCID[0:4] whenever valid address or data is available on the eLBC external pins. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID pins at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (MDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on MSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:26] and combined with LA[27:31].
- If a valid source ID is detected on MSRCID[0:4] and MDVAL is asserted, valid data may be latched from LAD.

The MSRCID[0:4] and MDVAL signals are multiplexed with other functions sharing the same external pins. Refer to Chapter 3, “Signal Descriptions,” and Chapter 4, “Reset, Clocking, and Initialization,” to learn how to enable the MSRCID/MDVAL pins.

14.2 External Signal Descriptions

Table 14-1 contains a list of external signals related to the eLBC and summarizes their function. Note that during assertion of $\overline{\text{HRESET}}$, the PLL is initially unlocked, so the LCLK and LSYNC_OUT values are likely to be unstable/jittery for several microseconds; after the PLL locks, stable clock signals are driven on these signals.

Table 14-1. Signal Properties—Summary

Name	Alternate Function(s)	Mode	Descriptions	No. of Signals	I/O
LALE	—	—	External address latch enable	1	O
$\overline{\text{LCS}}$[0]	—	—	Chip select 0	1	O
$\overline{\text{LCS}}$[1:7]	—	—	Chip selects [1–7]	7	O
$\overline{\text{LWE}}$/ $\overline{\text{LFWE}}$/ $\overline{\text{LBS}}$	$\overline{\text{LWE}}$	GPCM	Write enable	4	O
	$\overline{\text{LFWE}}$[0]	FCM	Write enable	1	
	$\overline{\text{LBS}}$	UPM	Byte (lane) select	4	
LGPL0/ LFCLE	LGPL0	UPM	General purpose line 0	1	O
	LFCLE	FCM	Flash command latch enable	1	
LGPL1/ LFALE	LGPL1	UPM	General purpose line 1	1	O
	LFALE	FCM	Flash address latch enable	1	

Table 14-1. Signal Properties—Summary (continued)

Name	Alternate Function(s)	Mode	Descriptions	No. of Signals	I/O
$\overline{\text{LOE}}$/ LGPL2/ $\overline{\text{LFRE}}$	$\overline{\text{LOE}}$	GPCM	Output enable	1	O
	$\overline{\text{LFRE}}$	FCM	Flash read enable	1	
	LGPL2	UPM	General purpose line 2	1	
LGPL3/ $\overline{\text{LFWP}}$	LGPL3	UPM	General purpose line 3	1	O
	$\overline{\text{LFWP}}$	FCM	Flash write protect	1	
$\overline{\text{LGTA}}$/ LFR$\overline{\text{B}}$/ LGPL4/ LUPWAIT/ LPBSE	$\overline{\text{LGTA}}$	GPCM	Transaction termination	1	I
	LFR$\overline{\text{B}}$	FCM	Flash ready/$\overline{\text{busy}}$, open-drain shared pin	1	I
	LGPL4	UPM	General purpose line 4	1	O
	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[7:31]	—	—	Non-multiplexed address bus	25	O
LAD[0:31]	—	—	Multiplexed address/data bus	32	I/O
LDP[0:3]	—	—	Local bus data parity	4	I/O
LCLK[0:2]	—	—	Local bus clocks	3	O
LSYNC_IN	—	—	PLL synchronize input	1	I
LSYNC_OUT	—	—	PLL synchronize output	1	O
MDVAL	—	eLBC debug	Local bus data valid	1	O
MSRCID[0:4]	—	eLBC debug	Local bus source ID	5	O

Table 14-2 shows the detailed external signal descriptions for the eLBC.

Table 14-2. 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.
$\overline{\text{LCS}}$[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. $\overline{\text{LCS}}$[0:7] are provided on a per-bank basis with $\overline{\text{LCS0}}$ corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.

Table 14-2. Enhanced Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LWE[0:3]/ LFWE/ LBS[0:3]	O	GPCM write enable/FCM write enable/UPM byte select. These signals select or validate each byte lane of the data bus. For banks with port sizes of 32 bits (as set by BR*n*[PS]), all four signals are defined. For a 16-bit port size, only bits 0–1 are defined; and 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[0:3] assert for each byte lane enabled for writing. LFWE (bit 0) enables command, address, and data writes to NAND Flash EEPROMs controlled by FCM. LBS[0:3] are programmable byte-select signals in UPM mode. See Section 14.4.4.4, "RAM Array," for programming details about LBS[0:3].
		Timing	Assertion/Negation—See Section 14.4.2, "General-Purpose Chip-Select Machine (GPCM)," for details regarding the timing of LWE[0:3].
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.
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 EPROMs.
LOE/LGPL2/ LFRE	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 enables data read cycles from NAND Flash EEPROMs controlled by FCM.
LGPL3/ LFWP	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 protects NAND Flash EEPROMs from accidental erasure and programming when LFWP is asserted low—see Section 14.3.1.16, "Flash Mode Register (FMR)," for programming of FCM operations to control LFWP.
LGTA/LGPL4/ LFRB/ 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 to stall during long-latency read and programming operations, continuing once LFRB 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 LBS[0:3] 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.

Table 14-2. Enhanced Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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 OR*n*[BCTLD].	
		State Meaning	Asserted/Negated—The LBCTL pin normally functions as a write/$\overline{\text{read}}$ 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[7:31]	O	Nonmultiplexed address bus. All bits driven on LA[7:31] are defined for 8-bit port sizes. For 32-bit port sizes, LA[30:31] are don't cares; 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 Section 14.5, "Initialization/Application Information," for address signal multiplexing.
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in BR*n*[PS] as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). 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); LAD[16:31] are unused for 16-bit port sizes. 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 32-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.
LDP[0:3]	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[0:31], such that LDP0 is even/odd parity for LAD[0:7], while LDP[3] is even/odd parity for LAD[24:31]. Unused byte lanes for port sizes less than 32 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[0:3] with the same timing that read data is sampled on LAD. LDP[0:3] change impedance in concert with LAD.
LCLK[0:2]	O	Local bus clocks	
		State Meaning	Asserted/Negated—LCLK[0:2] drive an identical bus clock signal for distributed loads. If the eLBC PLL is enabled (see LCRR[PBYP], Figure 14-19), the bus clock phase is shifted earlier than transitions on other eLBC signals (such as LAD*n* and $\overline{\text{LCS}}$*n*) by a time delay matching the delay of the PLL timing loop set up between LSYNC_OUT and LSYNC_IN.

Table 14-2. Enhanced Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LSYNC_OUT	O	PLL synchronization out.	
		State Meaning	Asserted/Negated—A replica of the bus clock, appearing on LSYNC_OUT, should be propagated through a passive timing loop and returned to LSYNC_IN for achieving correct PLL lock.
		Timing	Assertion/Negation—The time delay of the timing loop should be such that it compensates for the round-trip flight time of LCLK[0:2] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	PLL synchronization in.	
		State Meaning	Asserted/Negated—See description of LSYNC_OUT.
MDVAL	O	Local bus data valid (eLBC debug mode only)	
		State Meaning	Asserted/Negated—For a read, MDVAL asserts for one bus cycle in the cycle immediately preceding the sampling of read data on LAD. For a write, MDVAL asserts for one bus cycle during the final cycle for which the current write data on LAD is valid. During burst transfers, MDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the eLBC is in system debug mode. In debug mode, MDVAL asserts when the eLBC generates a data transfer acknowledge.
MSRCID[0:4]	O	Local bus source ID (eLBC debug mode only). In debug mode, all MSRCID[0:4] pins are driven high unless MSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the eLBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until MDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, MSRCID[0:4] is valid only when the address on LAD consists of all physical address bits—with optional padding—for reconstructing the system address presented to the eLBC.

14.3 Memory Map/Register Definition

Table 14-3 shows the memory mapped registers of the eLBC. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 14-3. Enhanced Local Bus Controller Registers

Enhanced Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000	BR0—Base register 0	R/W	0x0000_*nnnn*	14.3.1.1/14-10
0x008	BR1—Base register 1	R/W	0x0000_0000	14.3.1.1/14-10
0x010	BR2—Base register 2	R/W	0x0000_0000	14.3.1.1/14-10
0x018	BR3—Base register 3	R/W	0x0000_0000	14.3.1.1/14-10
0x020	BR4—Base register 4	R/W	0x0000_0000	14.3.1.1/14-10

Table 14-3. Enhanced Local Bus Controller Registers (continued)

Enhanced Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x028	BR5—Base register 5	R/W	0x0000_0000	14.3.1.1/14-10
0x030	BR6—Base register 6	R/W	0x0000_0000	14.3.1.1/14-10
0x038	BR7—Base register 7	R/W	0x0000_0000	14.3.1.1/14-10
0x004	OR0—Options register 0	R/W	0x0000_0FF7	14.3.1.2/14-12
0x00C	OR1—Options register 1	R/W	0x0000_0000	14.3.1.2/14-12
0x014	OR2—Options register 2	R/W	0x0000_0000	14.3.1.2/14-12
0x01C	OR3—Options register 3	R/W	0x0000_0000	14.3.1.2/14-12
0x024	OR4—Options register 4	R/W	0x0000_0000	14.3.1.2/14-12
0x02C	OR5—Options register 5	R/W	0x0000_0000	14.3.1.2/14-12
0x034	OR6—Options register 6	R/W	0x0000_0000	14.3.1.2/14-12
0x03C	OR7—Options register 7	R/W	0x0000_0000	14.3.1.2/14-12
0x040–0x064	Reserved	—	—	—
0x068	MAR—UPM address register	R/W	0x0000_0000	14.3.1.3/14-20
0x06C	Reserved	—	—	—
0x070	MAMR—UPMA mode register	R/W	0x0000_0000	14.3.1.4/14-21
0x074	MBMR—UPMB mode register	R/W	0x0000_0000	14.3.1.4/14-21
0x078	MCMR—UPMC mode register	R/W	0x0000_0000	14.3.1.4/14-21
0x07C–0x080	Reserved	—	—	—
0x084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	14.3.1.5/14-23
0x088	MDR—UPM/FCM data register	R/W	0x0000_0000	14.3.1.6/14-23
0x08C	Reserved	—	—	—
0x090	LSOR—Special operation initiation register	R/W	0x0000_0000	14.3.1.7/14-24
0x094–0x09C	Reserved	—	—	—
0x0A0	LURT—UPM refresh timer	R/W	0x0000_0000	14.3.1.4/14-21
0x0A4–0x0AC	Reserved	—	—	—
0x0B0	LTESR—Transfer error status register	w1c	0x0000_0000	14.3.1.9/14-26
0x0B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	14.3.1.10/14-28
0x0B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	14.3.1.11/14-29
0x0BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	14.3.1.12/14-30

Table 14-3. Enhanced Local Bus Controller Registers (continued)

Enhanced Local Bus Controller—Block Base Address 0x0_5000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x0C0	LTEAR—Transfer error address register	R/W	0x0000_0000	14.3.1.13/14-31
0x0C4–0x0CC	Reserved	—	—	—
0x0D0	LBCR—Configuration register	R/W	0x0000_0000	14.3.1.14/14-32
0x0D4	LCRR—Clock ratio register	R/W	0x8000_0008	14.3.1.15/14-33
0x0D8–0x0DC	Reserved	—	—	—
0x0E0	FMR—Flash mode register	R/W	0x0000_0*n*00	14.3.1.16/14-34
0x0E4	FIR—Flash instruction register	R/W	0x0000_0000	14.3.1.17/14-36
0x0E8	FCR—Flash command register	R/W	0x0000_0000	14.3.1.18/14-37
0x0EC	FBAR—Flash block address register	R/W	0x0000_0000	14.3.1.19/14-38
0x0F0	FPAR—Flash page address register	R/W	0x0000_0000	14.3.1.20/14-38
0x0F4	FBCR—Flash byte count register	R/W	0x0000_0000	14.3.1.21/14-40
0x0F8–0x0FC	Reserved	—	—	—
0x110–0xFFC	Reserved	—	—	—

14.3.1 Register Descriptions

This section provides a detailed description of the eLBC configuration, status, and control registers with detailed bit and field descriptions.

Address offsets in the eLBC address range that are not defined in Table 14-3 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.

14.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*n*), shown in Figure 14-2, 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 power-on configuration settings.

Offset BR0: 0x0_5000
BR1: 0x0_5008
BR2: 0x0_5010
BR3: 0x0_5018
BR4: 0x0_5020
BR5: 0x0_5028
BR6: 0x0_5030
BR7: 0x0_5038

Access: Read/Write

	0 – 16	17	18	19	20	21	22	23	24 – 26	27	28	29	30	31
R W	BA	—		PS	DECC			WP	MSEL	—	ATOM		—	V
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	0	P	S	0	0	0	0 0 M	0	0	0	1	V[1]

[1] BR0 has its valid bit (V) set at reset. Thus bank 0 is valid with the port size (PS) configured from cfg_rom_loc[0:3] at power-on reset. M = 0 for MSEL of GPCM, 1 for MSEL of FCM at boot. All other base registers have all bits cleared to zero during reset.

Figure 14-2. Base Registers (BR*n*)

Table 14-4 describes BR*n* fields.

Table 14-4. BR*n* Field Descriptions

Bits	Name	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	—	Reserved
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the power-on reset configuration setting. For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8-bit 10 16-bit (not supported for FCM) 11 32-bit (not supported for FCM)
21–22	DECC	Specifies the method for data error checking. The reset value of this field is configured during power-on reset by the signal, cfg_fcm_ecc. If the signal is pulled high during POR then DECC = 01. If the signal is pulled low during POR then DECC = 00. 00 Data error checking disabled, but normal parity generation for GPCM and UPM. No ECC generation for FCM. 01 Normal parity generation and checking for GPCM and UPM. ECC checking is enabled, but ECC generation is disabled, for FCM on full-page transfers. 10 Read-modify-write parity generation and normal parity checking for GPCM and UPM. 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 $\overline{\text{LCS}n}$ 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.

Table 14-4. BR*n* Field Descriptions (continued)

Bits	Name	Description
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	—	Reserved
28–29	ATOM	Atomic operation. Writes (reads) to the address space handled by the memory controller bank reserve the selected memory bank for the exclusive use of the accessing device. The reservation is released when the device performs a read (write) operation to this memory controller bank. If a subsequent read (write) request to this memory controller bank is not detected within 256 bus clock cycles of the last write (read), the reservation is released and an atomic error is reported (if enabled). 00 The address space controlled by this bank is not used for atomic operations. 01 Read-after-write-atomic (RAWA). 10 Write-after-read-atomic (WARA). 11 Reserved
30	—	Reserved
31	V	Valid bit. Indicates that the contents of the BR*n* and OR*n* pair are valid. $\overline{\text{LCS}n}$ 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.

14.3.1.2 Option Registers (OR0–OR7)

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. Because bank 0 can be used to boot, the reset value of OR0 may be different depending on power-on configuration options. Table 14-5 shows the reset values for OR0.

Table 14-5. 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

14.3.1.2.1 Address Mask

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. Table 14-6 shows memory bank sizes from 32 Kbytes to 4 Gbytes.

Table 14-6. 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

14.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 14-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

Offset OR0: 0x0_5004 Access: Read/Write
OR1: 0x0_500c
OR2: 0x0_5014
OR3: 0x0_501c
OR4: 0x0_5024
OR5: 0x0_502c
OR6: 0x0_5034
OR7: 0x0_503c

	0															15
R W	AM															
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	27	28	29	30	31
R W	AM	—		BCTLD	CSNT	ACS		XACS	SCY		SETA	TRLX	EHTR	EAD
Reset	All zeros[1]													

1 Refer to Table 14-5 for the OR0 reset value. All other option registers have all bits cleared.

Figure 14-3. Option Registers (OR*n*) in GPCM Mode

Table 14-7 describes OR*n* fields for GPCM mode.

Table 14-7. OR*n*—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	GPCM 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 and therefore don't care for address checking. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18	—	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	CSNT	Chip select negation time. Determines when $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{\text{LWE}}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated one quarter of a bus clock cycle earlier.

Table 14-7. OR*n*—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>21–22</td><td>ACS</td><td>Address to chip-select setup. Determines the delay of the $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>00</td><td>$\overline{\text{LCS}n}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT = 0.</td></tr>
<tr><td>01</td><td>Reserved.</td></tr>
<tr><td>10</td><td>$\overline{\text{LCS}n}$ is output one quarter bus clock cycle after the address lines.</td></tr>
<tr><td>11</td><td>$\overline{\text{LCS}n}$ is output one half bus clock cycle after the address lines.</td></tr>
</table></td></tr>
<tr><td>23</td><td>XACS</td><td>Extra address to chip-select setup. Setting this bit increases the delay of the $\overline{\text{LCS}n}$ 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 14-30 and Table 14-31).</td></tr>
<tr><td>24–27</td><td>SCY</td><td>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</td></tr>
<tr><td>28</td><td>SETA</td><td>External address termination.
0 Access is terminated internally by the memory controller unless the external device asserts $\overline{\text{LGTA}}$ earlier to terminate the access.
1 Access is terminated externally by asserting the $\overline{\text{LGTA}}$ external pin. (Only $\overline{\text{LGTA}}$ can terminate the access).</td></tr>
<tr><td>29</td><td>TRLX</td><td>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:
• 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.
• $\overline{\text{LCS}n}$ (only if ACS is not equal to 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes.</td></tr>
</table>

Table 14-7. OR*n*—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>30</td><td>EHTR</td><td>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>
<tr><th>TRLX</th><th>EHTR</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>The memory controller generates normal timing. No additional cycles are inserted.</td></tr>
<tr><td>0</td><td>1</td><td>1 idle clock cycle is inserted.</td></tr>
<tr><td>1</td><td>0</td><td>4 idle clock cycles are inserted.</td></tr>
<tr><td>1</td><td>1</td><td>8 idle clock cycles are inserted.</td></tr>
</table></td></tr>
<tr><td>31</td><td>EAD</td><td>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]).</td></tr>
</table>

14.3.1.2.3 Option Registers (OR*n*)—FCM Mode

Figure 14-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the FCM machine.

Offset OR0: 0x0_5004
OR1: 0x0_500c
OR2: 0x0_5014
OR3: 0x0_501c
OR4: 0x0_5024
OR5: 0x0_502c
OR6: 0x0_5034
OR7: 0x0_503c

Access: Read/Write

	0	15
R W	AM	
Reset	All zeros	

	16	17	18	19	20	21	22	23	24	25	27	28	29	30	31
R W	AM	—		BCTLD	—	PGS	CSCT	CST	CHT	SCY		RST	TRLX	EHTR	—
Reset	All zeros[1]														

[1] Refer to Table 14-5 for the OR0 reset value. All other option registers have all bits cleared.

Figure 14-4. Option Registers (OR*n*) in FCM Mode

Table 14-8 describes OR*n* fields for FCM mode.

Table 14-8. OR*n*—FCM Field Descriptions

Bits	Name	Description
0–16	AM	FCM 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 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18	—	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	—	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 $\overline{\text{LCS}n}$ 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.

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 $\overline{\text{LFWE}}$ assertion relative to the command, address, or data change when the external memory access is handled by the FCM.

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.

Table 14-8. OR*n*—FCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>24</td><td>CHT</td><td>Command hold time. Determines the $\overline{\text{LFWE}}$ negation prior to the command, address, or data change when the external memory access is handled by the FCM.
<table>
<tr><th>TRLX</th><th>CHT</th><th>Meaning</th></tr>
<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>
</table></td></tr>
<tr><td>25–27</td><td>SCY</td><td>Cycle length in bus clocks. Determines:
• 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 4×(2+SCY) clock cycles (TRLX = 0) or 8×(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/$\overline{\text{BSY}}$ pin (connected to LFR$\overline{\text{B}}$). LFR$\overline{\text{B}}$ is not sampled until 8×(2+SCY) clock cycles (TRLX = 0) or 16×(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</td></tr>
<tr><td>28</td><td>RST</td><td>Read setup time. Determines the delay of $\overline{\text{LFRE}}$ assertion relative to sampling of read data when the external memory access is handled by the FCM.
<table>
<tr><th>TRLX</th><th>RST</th><th>Meaning</th></tr>
<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>
<tr><td>1</td><td>0</td><td>The read-enable is asserted 0.5 clock cycles prior to any wait states.</td></tr>
<tr><td>1</td><td>1</td><td>The read-enable is asserted 1 clock cycle prior to any wait states.</td></tr>
</table></td></tr>
<tr><td>29</td><td>TRLX</td><td>Timing relaxed. Modifies the settings of timing parameters for slow memories.
0 Normal timing is generated by the FCM.
1 Relaxed timing on the following parameters:
• Doubles the number of clock cycles between $\overline{\text{LCS}}$n 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.</td></tr>
</table>

Table 14-8. OR*n*—FCM Field Descriptions (continued)

Bits	Name	Description
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. **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	—	Reserved

14.3.1.2.4 Option Registers (OR*n*)—UPM Mode

Figure 14-5 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

Offset OR0: 0x0_5004
OR1: 0x0_500c
OR2: 0x0_5014
OR3: 0x0_501c
OR4: 0x0_5024
OR5: 0x0_502c
OR6: 0x0_5034
OR7: 0x0_503c

Access: Read/Write

	0			15
R W	AM			
Reset	All zeros			

	16	17	18	19	20	22	23	24	28	29	30	31
R W	AM	—		BCTLD	—		BI	—		TRLX	EHTR	EAD
Reset	All zeros[1]											

1 Refer to Table 14-5 for the OR0 reset value. All other option registers have all bits cleared.

Figure 14-5. Option Registers (OR*n*) in UPM Mode

Table 14-9 describes BR*n* fields for UPM mode.

Table 14-9. OR*n*—UPM Field Descriptions

Bits	Name	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	—	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.
20–22	—	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	—	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.
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]).

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.

14.3.1.3 UPM Memory Address Register (MAR)

Figure 14-6 shows the fields of the UPM memory address register (MAR).

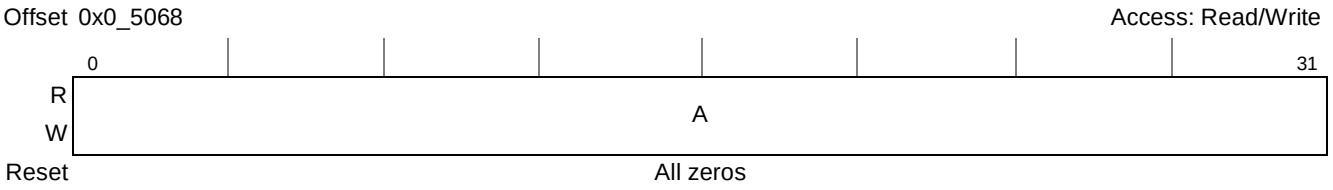


Figure 14-6. UPM Memory Address Register (MAR)

Table 14-10 describes the MAR fields.

Table 14-10. MAR Field Descriptions

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

14.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 14-7, contain the configuration for the three UPMs.

Offset MAMR: 0x0_5070
MBMR: 0x0_5074
MCMR: 0x0_5078

Access: Read/Write

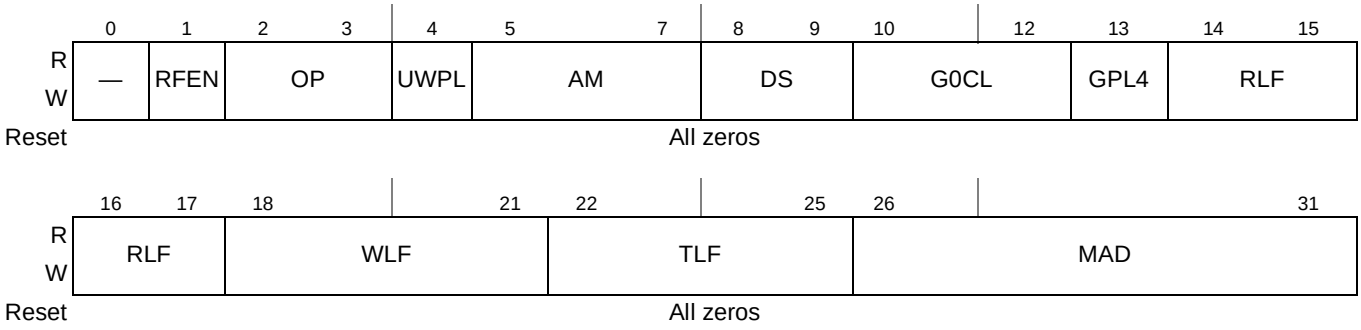


Figure 14-7. UPM Mode Registers (M*x*MR)

Table 14-11 describes UPM mode fields.

Table 14-11. M*x*MR Field Descriptions

Bits	Name	Description
0	—	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*n* 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. 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. 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.
4	UWPL	LUPWAIT polarity active low. Sets the polarity of the LUPWAIT pin when in UPM mode. 0 LUPWAIT is active high. 1 LUPWAIT is active low.

Table 14-11. M*x*MR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>5–7</td><td>AM</td><td>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 Section 14.4.4.4.7, “Address Multiplexing (AMX)” for more information.
000 Internal transaction address a[8:23] driven on LAD[16:31]; LAD[0:15] driven low.
001 Internal transaction address a[7:22] driven on LAD[16:31]; LAD[0:15] driven low.
010 Internal transaction address a[6:21] driven on LAD[16:31]; LAD[0:15] driven low.
011 Internal transaction address a[5:20] driven on LAD[16:31]; LAD[0:15] driven low.
100 Internal transaction address a[4:19] driven on LAD[16:31]; LAD[0:15] driven low.
101 Internal transaction address a[3:18] driven on LAD[16:31]; LAD[0:15] driven low.
110 Reserved
111 Reserved</td></tr>
<tr><td>8–9</td><td>DS</td><td>Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by UPMn. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPMn allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPMn 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.
00 1-bus clock cycle disable period
01 2-bus clock cycle disable period
10 3-bus clock cycle disable period
11 4-bus clock cycle disable period</td></tr>
<tr><td>10–12</td><td>G0CL</td><td>General line 0 control. Determines which logical address line can be output to the LGPL0 pin when the UPMn is selected to control the memory access.
000 A12
001 A11
010 A10
011 A9
100 A8
101 A7
110 A6
111 A5</td></tr>
<tr><td>13</td><td>GPL4</td><td>LGPL4 output line disable. Determines how the LGPL4/LUPWAIT pin is controlled by the corresponding bits in the UPMn array. See Table 14-39 on page 14-79.
<table>
<tr><th rowspan="2">Value</th><th rowspan="2">LGPL4/LUPWAIT Pin Function</th><th colspan="2">Interpretation of UPM Word Bits</th></tr>
<tr><th>G4T1/DLT3</th><th>G4T3/WAEN</th></tr>
<tr><td>0</td><td>LGPL4 (output)</td><td>G4T1</td><td>G4T3</td></tr>
<tr><td>1</td><td>LUPWAIT (input)</td><td>DLT3</td><td>WAEN</td></tr>
</table></td></tr>
<tr><td>14–17</td><td>RLF</td><td>Read loop field. Determines the number of times a loop defined in the UPMn will be executed for a burst- or single-beat read pattern or when MxMR[OP] = 11 (RUN command)
0000 16
0001 1
0010 2
0011 3
...
1110 14
1111 15</td></tr>
</table>

Table 14-11. M*x*MR Field Descriptions (continued)

Bits	Name	Description
18–21	WLF	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. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
22–25	TLF	Refresh loop field. Determines the number of times a loop defined in the UPM*n* will be executed for a refresh service pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
26–31	MAD	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*.

14.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in Figure 14-8, is used to divide the system clock to provide the UPM refresh timers clock.

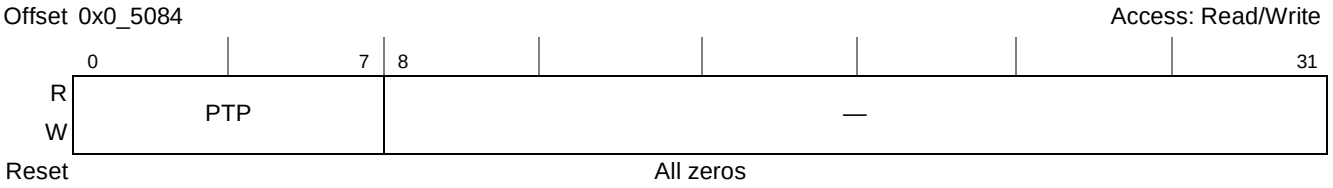


Figure 14-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 14-12 describes MRTPR fields.

Table 14-12. MRTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 00000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

14.3.1.6 UPM/FCM Data Register (MDR)

The memory data register (MDR), shown in Figure 14-9 and Figure 14-10, 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.

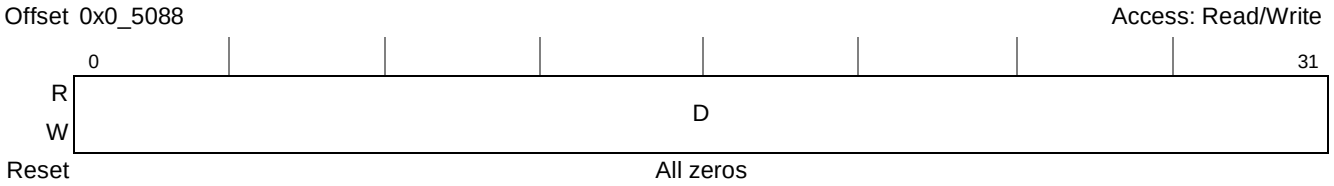


Figure 14-9. UPM Data Register in UPM Mode (MDR)

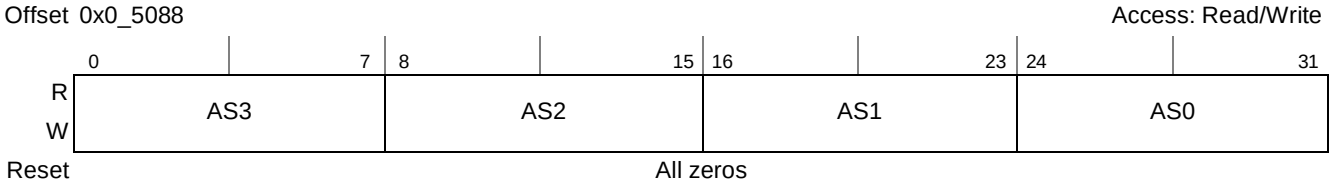


Figure 14-10. FCM Data Register in FCM Mode (MDR)

Table 14-13 describes MDR[D].

Table 14-13. MDR Field Description

Bits	Name	Description
0–31	D	In UPM mode, 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*x*MR[OP] = 01 or M*x*MR[OP] = 10).
0–7	AS3	In FCM mode, 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	In FCM mode, 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	In FCM mode, 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	In FCM mode, 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.

14.3.1.7 Special Operation Initiation Register (LSOR)

The special operation initiation register (LSOR), shown in Figure 14-11, 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 re-written 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*x*MR[OP], see Section 14.3.1.4, “UPM Mode Registers (MxMR).” FCM special operation modes are set in FMR[OP], see Section 14.3.1.16, “Flash Mode Register (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 Section 14.4.4.2.1, “UPM Programming Example (Two Sequential Writes to the RAM Array).”

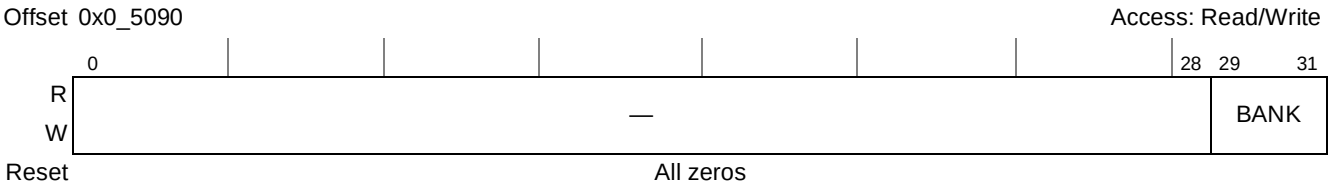


Figure 14-11. Special Operation Initiation Register (LSOR)

Table 14-14 describes LSOR.

Table 14-14. LSOR Field Description

Bits	Name	Description
0–28	—	Reserved
29–31	BANK	Bank on which a special operation is initiated. If the bank identified by BANK is marked valid (BR*n*[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

14.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 14-12, generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled (M*x*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.

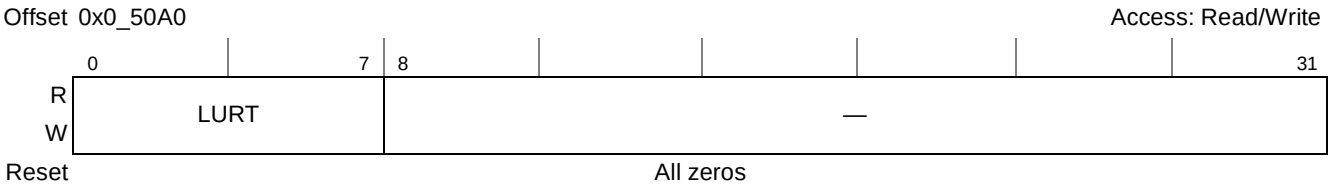


Figure 14-12. UPM Refresh Timer (LURT)

Table 14-15 describes LURT fields.

Table 14-15. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $TimerPeriod = \frac{LURT}{\left(\frac{Fsystemclock}{MRTPR[PTP]}\right)}$ Example: For a 266-MHz system clock and a required service rate of 15.6 µs, given MRTPR[PTP] = 32, the LURT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value (0x00) sets the maximum period to 256 x MRTPR[PTP] system clock cycles.
8–31	—	Reserved

14.3.1.9 Transfer Error Status Register (LTESR)

The transfer error status register (LTESR) indicates the cause of an error or event. LTESR, shown in Figure 14-13, 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 updated again.

Offset 0x0_50B0 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	15
R W	BM	FCT	PAR	—		WP	—		ATMW	ATMR	—		CS	—	
Reset	All zeros														

	16	30	31
R W	—		CC
Reset	All zeros		

Figure 14-13. Transfer Error Status Register (LTESR)

Table 14-16 describes LTESR fields.

Table 14-16. LTESR Field Descriptions

Bits	Name	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 uncorrectable 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	—	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).
6–7	—	Reserved
8	ATMW	Atomic error write 0 No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.
9	ATMR	Atomic error read 0 No atomic read error occurred. 1 The subsequent read (RAWA) to a memory bank did not occur within 256 bus clock cycles.
10–11	—	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.
13–30	—	Reserved
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.

14.3.1.10 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 14-14, is used to disable error/event checking. Note that control of error/event checking is independent of control of reporting of errors/events (LTEIR) through the interrupt mechanism.

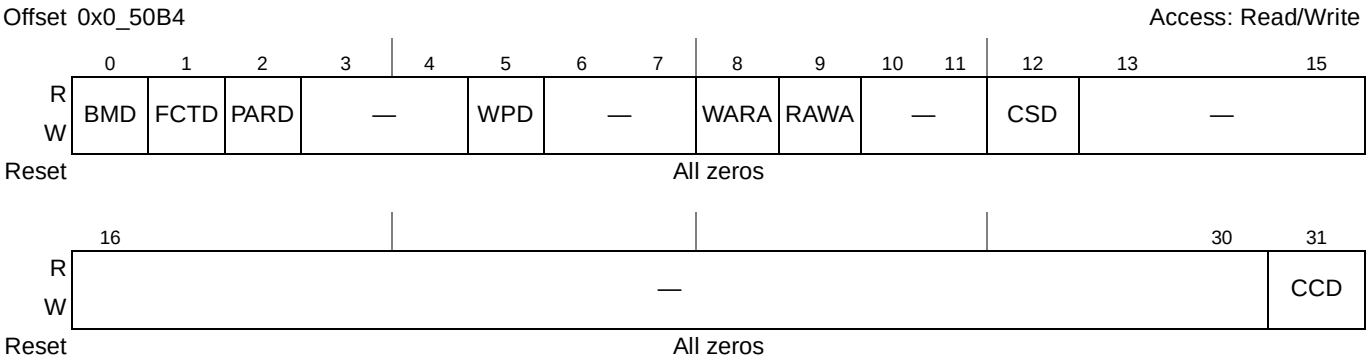


Figure 14-14. Transfer Error Check Disable Register (LTEDR)

Table 14-17 describes LTEDR fields.

Table 14-17. LTEDR Field Descriptions

Bits	Name	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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled. 0 Parity and ECC error checking is enabled. 1 Parity and ECC error checking is disabled.
3–4	—	Reserved
5	WPD	Write protect error checking disable. 0 Write protect error checking is enabled. 1 Write protect error checking is disabled.
6–7	—	Reserved
8	WARA	Write after read atomic (WARA) error checking disable. 0 WARA error checking is enabled. 1 WARA error checking is disabled.
9	RAWA	Read after write atomic (RAWA) error checking disable. 0 RAWA error checking is enabled. 1 RAWA error checking is disabled.
10–11	—	Reserved

Table 14-17. LTEDR Field Descriptions (continued)

Bits	Name	Description
12	CSD	Chip select error checking disable. 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–30	—	Reserved
31	CCD	FCM command completion checking disable. 0 Command completion checking is enabled. 1 Command completion checking is disabled.

14.3.1.11 Transfer Error Interrupt Enable Register (LTEIR)

The transfer error interrupt enable register (LTEIR), shown in Figure 14-15, 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.

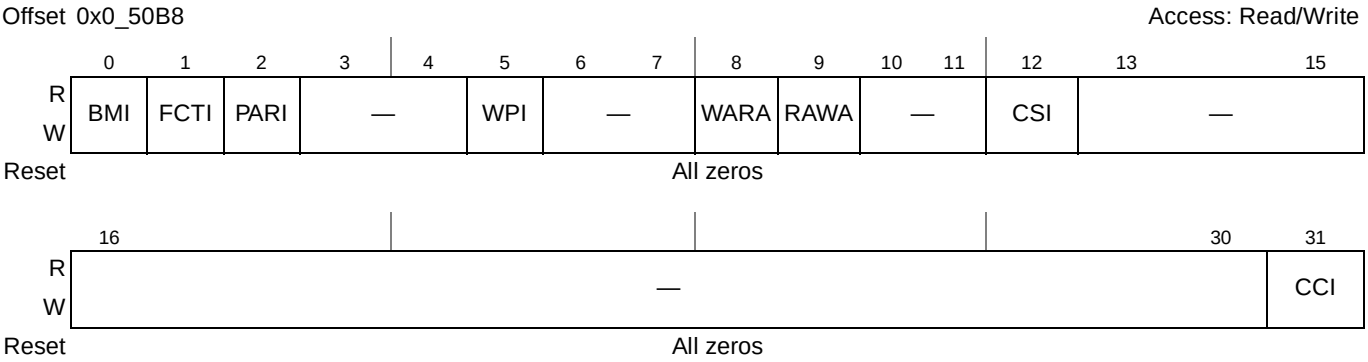


Figure 14-15. Transfer Error Interrupt Enable Register (LTEIR)

Table 14-18 describes LTEIR fields.

Table 14-18. LTEIR Field Descriptions

Bits	Name	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.
2	PARI	Parity and ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of core_fault_in, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, LTEDR[PARD] must be cleared and PARI must be set to ensure that an interrupt is generated. 0 Parity and ECC error reporting is disabled. 1 Parity and ECC error reporting is enabled.
3–4	—	Reserved

Table 14-18. LTEIR Field Descriptions (continued)

Bits	Name	Description
5	WPI	Write protect error interrupt enable. 0 Write protect error reporting is disabled. 1 Write protect error reporting is enabled.
6–7	—	Reserved
8	WARA	Write after read atomic (WARA) error interrupt enable. 0 WARA error reporting is disabled. 1 WARA error reporting is enabled.
9	RAWA	Read after write atomic (RAWA) error interrupt enable. 0 RAWA error reporting is disabled. 1 RAWA error reporting is enabled.
10–11	—	Reserved
12	CSI	Chip select error interrupt enable. 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–30	—	Reserved
31	CCI	FCM command completion interrupt enable. 0 Command completion reporting is disabled. 1 Command completion reporting is enabled.

14.3.1.12 Transfer Error Attributes Register (LTEATR)

The transfer error attributes register (LTEATR) captures source attributes of an error/event. Figure 14-16 shows the LTEATR. After LTEATR[V] has been set, software must clear this bit to allow LTESR, LTEATR, and LTEAR to update following any subsequent events/errors.

Offset 0x0_50BC Access: Read/Write

	0–2	3	4–10	11–15	16–19	20–27	28–30	31
R W	—	RWB	—	SRCID	PB	BNK	—	V
Reset	All zeros							

Figure 14-16. Transfer Error Attributes Register (LTEATR

Table 14-19 describes LTEATR fields.

Table 14-19. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	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.
4–10	—	Reserved
11–15	SRCID	Captures the source of the transaction when this information is provided on the internal interface to the eLBC.

Table 14-19. LTEATR Field Descriptions (continued)

Bits	Name	Description
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 uncorrectable 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	—	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.

14.3.1.13 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) captures the address of a transaction that caused an error/event. The transfer error address register (LTEAR) is shown in Figure 14-17.

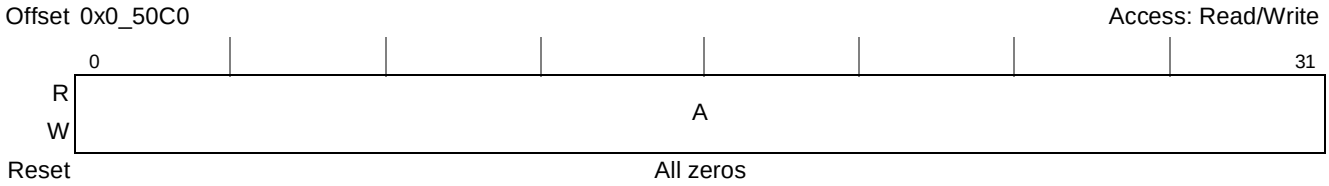


Figure 14-17. Transfer Error Address Register (LTEAR)

Table 14-20 describes LTEAR fields.

Table 14-20. LTEAR Field Descriptions

Bits	Name	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.

14.3.1.14 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in Figure 14-18.

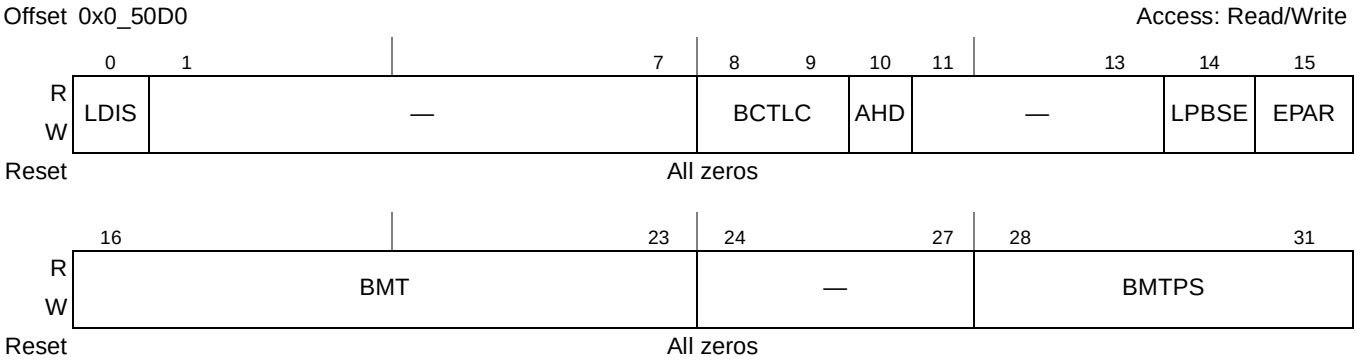


Figure 14-18. Local Bus Configuration Register

Table 14-21 describes LBCR fields.

Table 14-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled. 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/$\overline{R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ 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 two platform clock periods prior to the address being invalidated. At 66.6 MHz, this provides 3 ns of address hold time at the external address latch. 1 During address phases on the local bus, the LALE signal negates one 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	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LFR$\overline{B}$/LGPL4/LUPWAIT/LPBSE signal. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/LPBSE signal is available for memory control as LGPL4 (output) or $\overline{LGTA}$/LFR$\overline{B}$/LUPWAIT (input). 1 Parity byte select is enabled. LPBSE signal is dedicated as the parity byte select output, and $\overline{LGTA}$/LFR$\overline{B}$/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

Table 14-21. LBCR Field Descriptions (continued)

Bits	Name	Description
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of bus clock cycles. For non-zero values of BMT, the number of LCLK clock cycles to count down before a time-out error is generated is given by: bus cycles = BMT × PS, where PS is set according to LBCR[BMTPS]. The value of BMT × PS must not be less than 40 bus cycles for reliable operation.
24–27	—	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

14.3.1.15 Clock Ratio Register (LCRR)

The clock ratio register, shown in Figure 14-19, sets the system 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.

Offset 0x0_50D4 Access: Read/Write

	0	1				13	14	15	16						26	27		31
R W	PBYP	—					EADC		—								CLKDIV	
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 14-19. Clock Ratio Register (LCRR)

Table 14-22 describes LCRR fields.

Table 14-22. LCRR Field Descriptions

Bits	Name	Description
0	PBYP	PLL bypass. This bit should be set when using low bus clock frequencies (See device hardware specifications for applicable frequencies.). When in PLL bypass mode, incoming data is captured in the middle of the bus clock cycle. 0 The PLL is enabled. 1 The PLL is bypassed.
1–13	—	Reserved
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	—	Reserved
27–31	CLKDIV	System clock divider. Sets the frequency ratio between the system clock and the local bus clock. The system clock is equivalent to ccb_clk . Only the values shown below are allowed. **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 4 00011 Reserved 00100 8 00101–00111 Reserved 01000 16 01001–11111 Reserved

14.3.1.16 Flash Mode Register (FMR)

The local bus Flash mode register (FMR), shown in Figure 14-20, controls global operation of the FCM.

Offset 0x0_50E0 Access: Read/Write

	0–15	16–19	20	21 22	23	24 25 26 27	28 29	30 31
R W	—	CWTO	BOOT	—	ECCM	—	AL	—	OP
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0	R[1]	0 0	0	0 0 0 0	0 0	0 0

Figure 14-20. Flash Mode Register

1 Bit R (field BOOT) is set if power-on-reset configuration selects FCM as the boot ROM target.

Table 14-23 describes FMR fields.

Table 14-23. FMR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	CWTO	Command wait time-out. For FCM commands that wait on LFR$\overline{B}$ being sampled high (CW0, CW1, RBW and RSW), FCM pauses execution of the instruction sequence until either LFR$\overline{B}$ is sampled high, or a timer controlled by CTO expires, whichever occurs first. The time-out in the latter case is: 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 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 auto-load 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	—	Reserved
23	ECCM	ECC mode. When hardware checking and/or generation of error correcting codes (ECC) is enabled (that is, when BR*n*[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*×16)+6 through (*N*×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*×16)+8 through (*N*×16)+10 for spare region *N*, *N* = 0–3.
24–25	—	Reserved

Table 14-23. FMR Field Descriptions (continued)

Bits	Name	Description
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 OR*n*[PGS] = 0, 1 CA byte is issued; for OR*n*[PGS] = 1, 2 CA bytes are issued). 00 2 bytes are issued for page addresses, thus a total of 3 (OR*n*[PGS] = 0) or 4 (OR*n*[PGS] = 1) address bytes are issued for a {CA,PA} sequence 01 3 bytes are issued for page addresses, thus a total of 4 (OR*n*[PGS] = 0) or 5 (OR*n*[PGS] = 1) address bytes are issued for a {CA,PA} sequence 10 4 bytes are issued for page addresses, thus a total of 5 (OR*n*[PGS] = 0) or 6 (OR*n*[PGS] = 1) address bytes are issued for a {CA,PA} sequence 11 —
28–29	—	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. 10 Execute the command sequence contained in FIR, but with write protection enabled (pin $\overline{\text{LFWP}}$ 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 $\overline{\text{LFWP}}$ remaining high during the access.

14.3.1.17 Flash Instruction Register (FIR)

The local bus Flash instruction register (FIR), shown in Figure 14-21, 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.

Offset 0x0_50E4 Access: Read/Write

	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R W	OP0	OP1	OP2	OP3	OP4	OP5	OP6	OP7
Reset	All zeros							

Figure 14-21. Flash Instruction Register

Table 14-24 describes FIR fields.

Table 14-24. FIR Field Descriptions

Bits	Name	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 LFR$\overline{B}$ to return high or time-out, then issue command from FCR[CMD0] 1101 CW1—Wait for LFR$\overline{B}$ to return high or time-out, then issue command from FCR[CMD1] 1110 RBW—Wait for LFR$\overline{B}$ to return high or time-out, then read FBCR bytes of data from Flash device into current FCM RAM buffer 1111 RSW—Wait for LFR$\overline{B}$ 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	
8–11	OP2	
12–15	OP3	
16–19	OP4	
20–23	OP5	
24–27	OP6	
28–31	OP7	

14.3.1.18 Flash Command Register (FCR)

The local bus Flash command register (FCR), shown in Figure 14-22, 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.

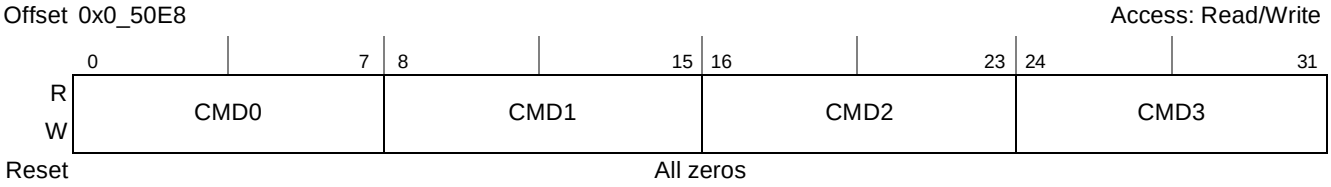


Figure 14-22. Flash Command Register

Table 14-25 describes FCR fields.

Table 14-25. FCR Field Descriptions

Bits	Name	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.

14.3.1.19 Flash Block Address Register (FBAR)

The local bus Flash block address register (FBAR), shown in Figure 14-23, locates the NAND Flash block index for the page currently accessed.

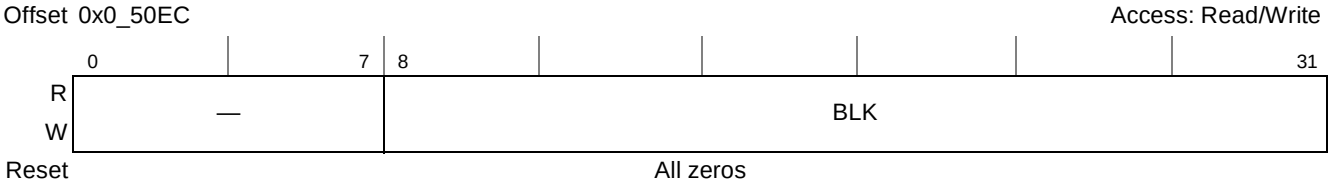


Figure 14-23. Flash Block Address Register

Table 14-26 describes FBAR fields.

Table 14-26. FBAR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–31	BLK	Flash block address. The size of the NAND Flash, as configured in OR*n*[PGS] and FMR[AL], determines the number of bits of BLK that are issued to the EEPROM during block address phases.

14.3.1.20 Flash Page Address Register (FPAR)

The local bus Flash page address register (FPAR), shown in Figure 14-24 and Figure 14-25, locates the current NAND Flash page in both the external NAND Flash device and FCM buffer RAM.

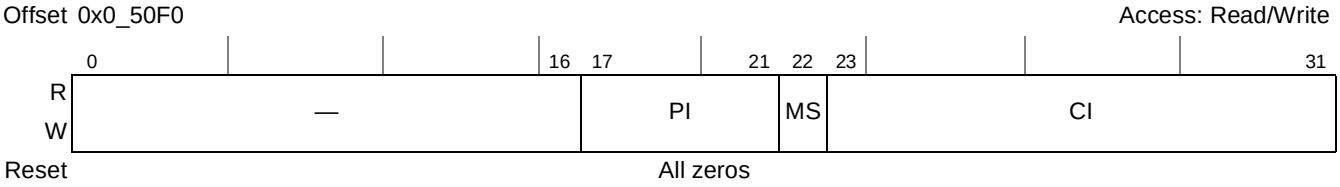


Figure 14-24. Flash Page Address Register, Small Page Device (ORx[PGS] = 0)

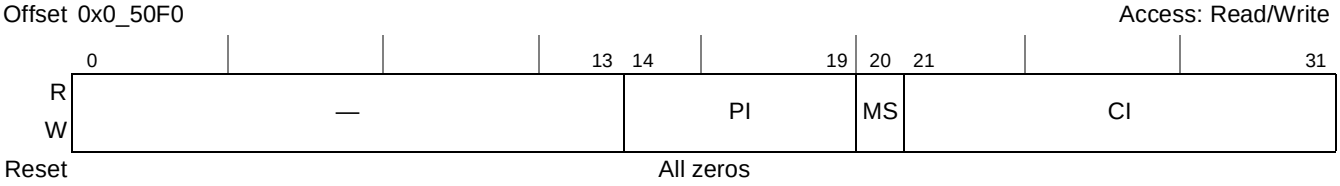


Figure 14-25. Flash Page Address Register, Large Page Device (ORx[PGS] = 1)

Table 14-27 describes FPAR fields for small page devices.

Table 14-27. FPAR Field Descriptions, Small Page Device (ORx[PGS] = 0)

Bits	Name	Description
0–16	—	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.

Table 14-28 describes FPAR fields for large page devices.

Table 14-28. FPAR Field Descriptions, Large Page Device (ORx[PGS] = 01)

Bits	Name	Description
0–13	—	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 0x0000–0x0FFF 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.

14.3.1.21 Flash Byte Count Register (FBCR)

The local bus Flash byte count register (FBCR), shown in Figure 14-26, defines the size of FCM block transfers for reads and writes to the NAND Flash EEPROM.

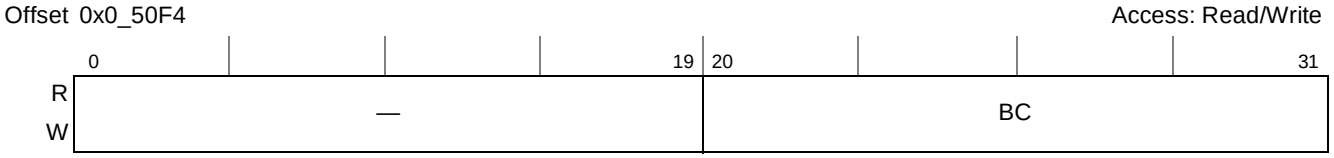


Figure 14-26. Flash Byte Count Register

Table 14-29 describes FBCR fields.

Table 14-29. FBCR Field Descriptions

Bits	Name	Description
0–19	—	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.

14.4 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.

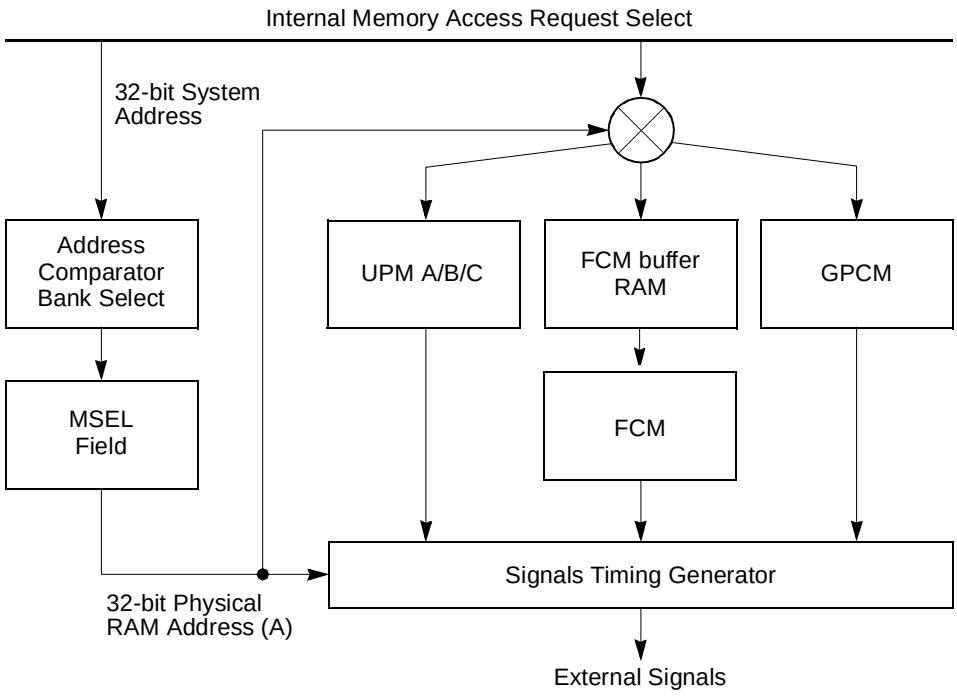


Figure 14-27. Basic Operation of Memory Controllers in the eLBC

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 Figure 14-27. 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.

14.4.1 Basic Architecture

The following subsections describe the basic architecture of the eLBC.

14.4.1.1 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).

14.4.1.2 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 two platform clock period (which, divided by LCRR[CLKDIV], yields the bus clock). For example, if the eLBC is operating at 66.6 MHz internally, then 3 ns of address hold time is introduced. However, when LCRR[CLKDIV] = 2 (clock ratio of 4) and the LCLK frequency exceeds 100 MHz, 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 platform clock cycle, and decreases the address hold time by the same amount. At 66.6 MHz and with LCRR[CLKDIV] = 2 (clock ratio of 4), the duration of LALE would then be 4.5 ns, with 1.5 ns of hold time. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the OR*n*[EAD] and LCRR[EADC] fields, and the LBCR[AHD] bit can be left at 0. However, this will add latency to all address tenures.

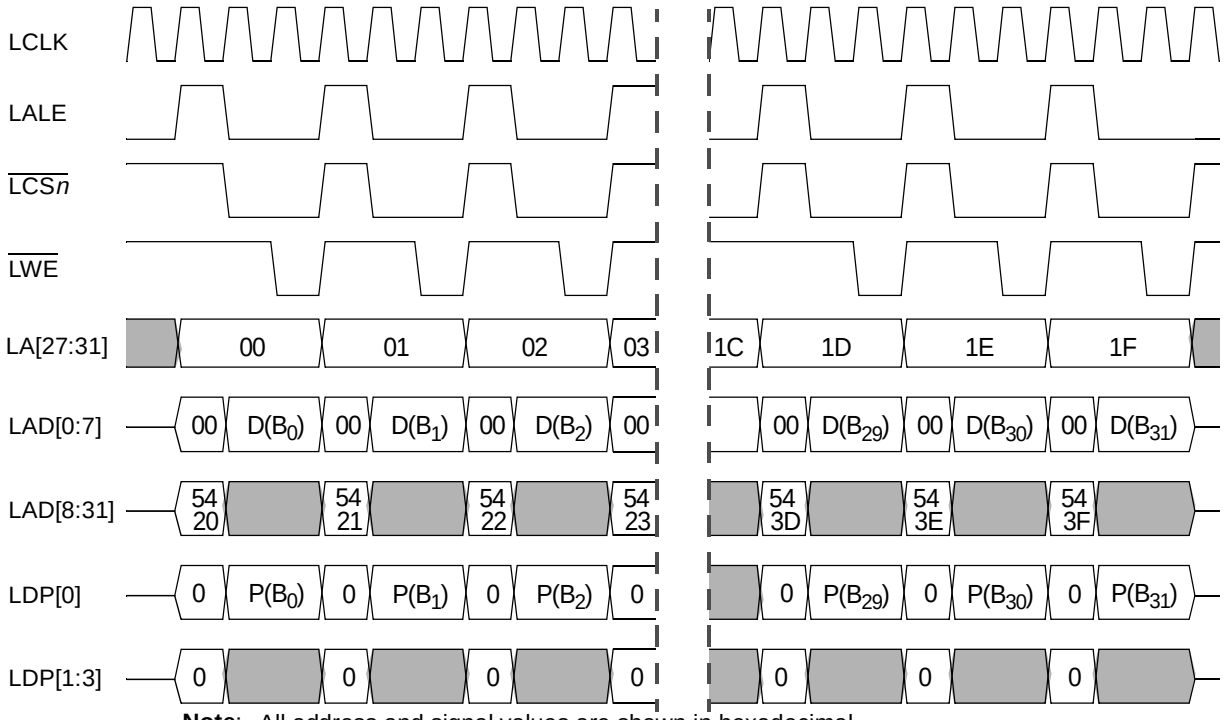
The frequency of LALE assertion varies across the three memory controllers:

- For GPCM, every assertion of $\overline{\text{LCS}n}$ 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 $\overline{\text{LCS}n}$ 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 not necessary to use LA if a sufficiently wide latch is used 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, Figure 14-28 shows eLBC signals for the GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP[0] are driven with valid data and parity, respectively.



Note: All address and signal values are shown in hexadecimal.
D(Bk) = kth of 32 data bytes, P(Bk) = parity bit of kth data byte.

Figure 14-28. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420 (LCRR[PBYP] = 0)

14.4.1.3 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. In eLBC debug mode, TA is also visible externally on the MDVAL pin. 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 14-29 shows LALE, TA (internal), and $\overline{\text{LCS}n}$. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCS}n}$ (or any other control signal) remains negated or frozen.

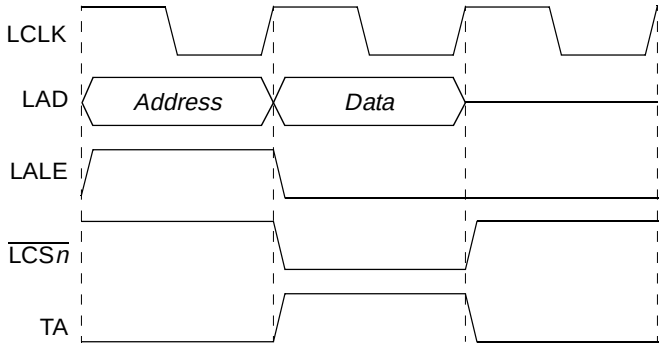


Figure 14-29. Basic eLBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

14.4.1.4 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 OR*n*[BCTLD]. LBCTL can be further configured by LBCR[BCTLC] to act as an extra $\overline{\text{LWE}}$ or an extra $\overline{\text{LOE}}$ 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.

14.4.1.5 Atomic Operation

The eLBC supports the following kinds of atomic bus operations (set by BR*n*[ATOM]):

- Read-after-write atomic (RAWA). When a write access hits a memory bank in which ATOM = 01, the eLBC reserves the selected memory bank for the exclusive use of the accessing master.

  While the bank is reserved, no other device can be granted access to this bank. The reservation is released when the master that created it accesses the same bank with a read transaction. Additional write transactions prior to the releasing read do not change reservation status, but are otherwise processed normally. If the master fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled); additional write transactions prior to the releasing read restart the reservation timer. This feature is intended for CAM operations.
- Write-after-read atomic (WARA). When a read access hits a memory bank in which ATOM = 10, the eLBC reserves the bus for the exclusive use of the accessing master.

  During the reservation period, no other device can be granted access to the atomic bank. The reservation is released when the device that created it accesses the same bank with a write transaction. Additional read transactions prior to the releasing write are otherwise processed normally and do not change the reservation status. If the device fails to release the reservation

within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled); additional read transactions prior to the releasing write restart the reservation timer.

14.4.1.6 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[0:3] for the bank if BR*n*[DECC] = 01 (normal parity) or BR*n*[DECC] = 10 for read-modify-write (RMW) parity. Byte lane parity on LDP[0:3] is generated regardless of the BR*n*[DECC] setting. Note that RMW parity can be used only for 32-bit port size banks. LBCR[EPAR] determines the global type of parity (odd or even).

FCM calculates an ECC over 512-byte blocks, and hence does not use the LDP[0:3] 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).

14.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle 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] × 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 (i.e. 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 Section 14.4.4.1.4, “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.

14.4.1.8 PLL Bypass Mode

At LCLK frequencies in excess of 66 MHz the local bus PLL is used to provide improved hold times at external receivers, and ease set-up margins for read data captured by eLBC. A wire loop between pins LSYNC_OUT and LSYNC_IN establishes the amount of LCLK skewing achieved by the PLL, which locks so as to produce edges on LCLK before the transition of other eLBC control and data signals.

At lower frequencies, the PLL may be unable to lock or provide sufficient hold time improvement for particularly slow devices. Accordingly, LCRR[PBYP] should be set to 1 to bypass the PLL at low frequencies, with the eLBC generating LCLK directly, while skewing it by half a bus clock cycle. An illustration of GPCM or UPM timing both with and without the PLL activated is shown in Figure 14-30. When LCRR[PBYP] = 0, the skew, t_{LSKEW}, matches the round-trip propagation delay of the timing loop between LSYNC_OUT and LSYNC_IN, and data is generated or sampled on the next rising edge of LCLK. The timing diagrams shown normally in this chapter assume that LCRR[PBYP] = 0. When LCRR[PBYP] = 1, the skew equals half the period of LCLK to maximize hold time at the external receiver; in this 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

Since LCLK is not used for NAND Flash EEPROMs controlled by FCM, the eLBC drives and samples data on the same edge (rising edge when LCRR[PBYP] = 0 and falling edge when LCRR[PBYP] = 1) on FCM controlled banks.

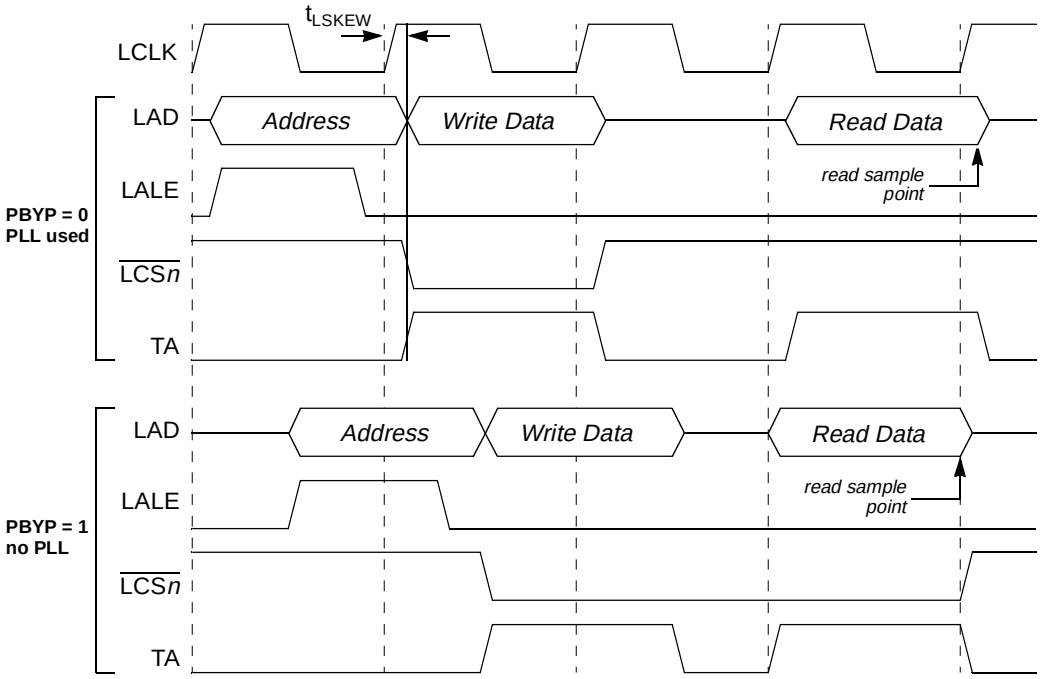


Figure 14-30. eLBC Bus Cycles in PLL and PLL-bypassed Modes (GPCM and UPM only)

14.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BR*n* and OR*n*.

Figure 14-31 shows a simple connection between an 8-bit port size SRAM device and the eLBC in GPCM mode. Byte-write enable signals ($\overline{\text{LWE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{LOE}}$) is provided to minimize external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ($\overline{\text{LCS0}}$) prior to the system being fully configured.

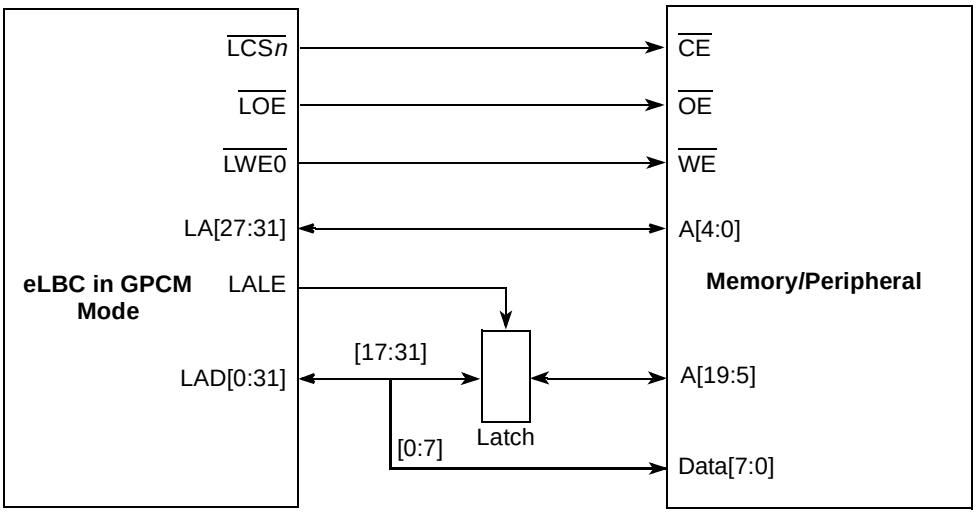


Figure 14-31. Enhanced Local Bus to GPCM Device Interface

Figure 14-32 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure OR*n*[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement. Generally, the attributes for the memory cycle are taken from OR*n*. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR and SETA fields.

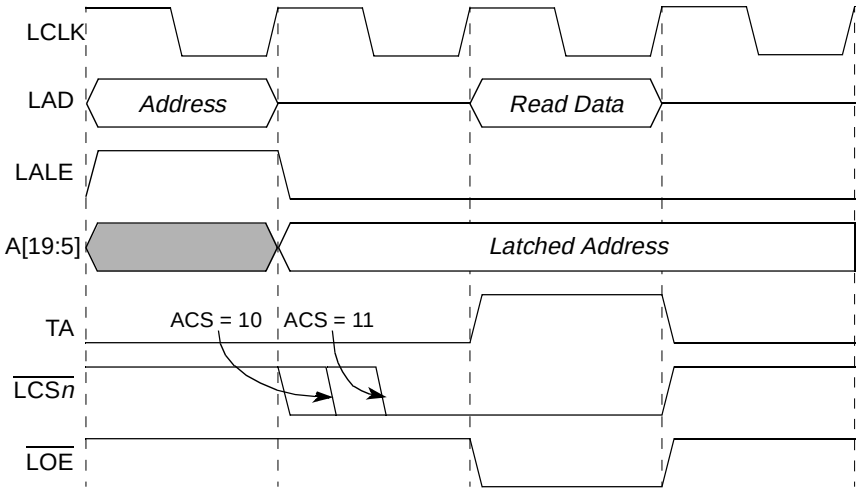
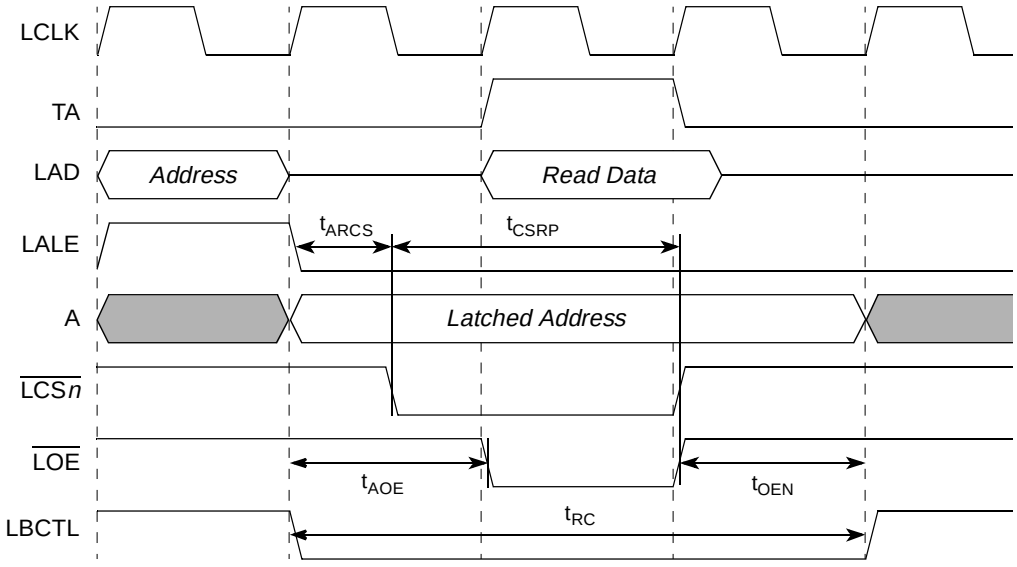


Figure 14-32. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0)

14.4.2.1 GPCM Read Signal Timing

The basic GPCM read timing parameters that may be set by the OR*n* attributes are shown in Figure 14-33. 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 by eLBC on the falling edge of TA. $\overline{\text{LOE}}$ and $\overline{\text{LCS}}n$ 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_{ARCS} = Address valid to read chip-select time.
t_{AOE} = Address valid to output enable time.
t_{CSRP} = Read chip-select assertion period.
t_{OEN} = Output enable negated time.

Figure 14-33. GPCM General Read Timing Parameters

Table 14-30 lists the signal timing parameters for a GPCM read access as the option register attributes are varied.

Table 14-30. GPCM Read Control Signal Timing

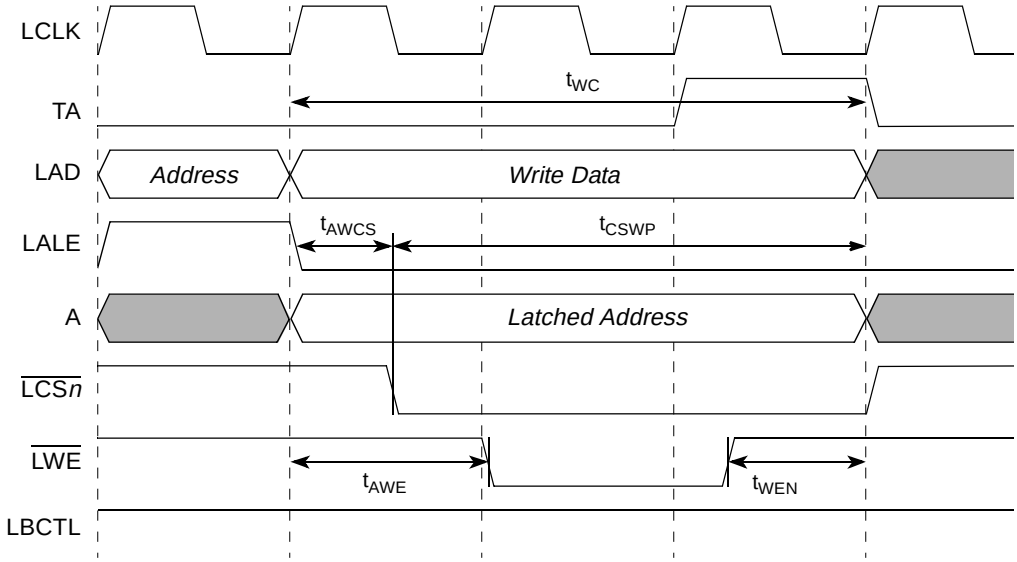
Option Register Attributes				Signal Timing (LCLK clock cycles)				
TRLX	EHTR	XACS	ACS	t_{ARCS}	t_{CSRP}	t_{AOE}	t_{OEN}	t_{RC}
0	0	0	0X	0	2+SCY	1	0	2+SCY
0	0	0	10	¼	1¾+SCY	1	0	2+SCY
0	0	0	11	½	1½+SCY	1	0	2+SCY
0	0	1	0X	0	2+SCY	1	0	2+SCY
0	0	1	10	1	1+SCY	1	0	2+SCY
0	0	1	11	2	1+SCY	2	0	3+SCY
0	1	0	0X	0	2+SCY	1	1	3+SCY
0	1	0	10	¼	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×SCY	1	4	6+2×SCY
1	0	0	10	1¼	1¾+2×SCY	2	4	7+2×SCY

Table 14-30. GPCM Read Control Signal Timing (continued)

Option Register Attributes				Signal Timing (LCLK clock cycles)				
TRLX	EHTR	XACS	ACS	t_{ARCS}	t_{CSRP}	t_{AOE}	t_{OEN}	t_{RC}
1	0	0	11	1½	1½+2×SCY	2	4	7+2×SCY
1	0	1	0X	0	2+2×SCY	1	4	6+2×SCY
1	0	1	10	2	1+2×SCY	2	4	7+2×SCY
1	0	1	11	3	1+2×SCY	3	4	8+2×SCY
1	1	0	0X	0	2+2×SCY	1	8	10+2×SCY
1	1	0	10	1¼	1¾+2×SCY	2	8	11+2×SCY
1	1	0	11	1½	1½+2×SCY	2	8	11+2×SCY
1	1	1	0X	0	2+2×SCY	1	8	10+2×SCY
1	1	1	10	2	1+2×SCY	2	8	11+2×SCY
1	1	1	11	3	1+2×SCY	3	8	12+2×SCY

14.4.2.2 GPCM Write Signal Timing

The basic GPCM write timing parameters that may be set by the OR*n* attributes are shown in Figure 14-34. The write access cycle commences upon latching of the memory address (LALE negated), and concludes when $\overline{\text{LCS}n}$ 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. $\overline{\text{LWE}}$ 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_{AWCS} = Address valid to write chip-select time.
t_{AWE} = Address valid to write enable time.
t_{CSWP} = Write chip-select assertion period.
t_{WEN} = Write enable negated time wrt chip-sele

Figure 14-34. GPCM General Write Timing Parameters

Table 14-31 lists the signal timing parameters for a GPCM write access as the option register attributes are varied.

Table 14-31. 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	¼	1¾+SCY	1	0	2+SCY
0	0	11	0	½	1½+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	¼	2+SCY
0	0	10	1	¼	1½+SCY	1	0	1¾+SCY
0	0	11	1	½	1¼+SCY	1	0	1¾+SCY
0	1	00	1	0	2+SCY	1	¼	2+SCY
0	1	10	1	1	¾+SCY	1	0	1¾+SCY
0	1	11	1	2	¾+SCY	2	0	2¾+SCY
1	0	00	0	0	2+2×SCY	1	0	2+2×SCY
1	0	10	0	1¼	1¾+2×SCY	2	0	3+2×SCY
1	0	11	0	1½	1½+2×SCY	2	0	3+2×SCY
1	1	00	0	0	2+2×SCY	1	0	2+2×SCY
1	1	10	0	2	1+2×SCY	2	0	3+2×SCY
1	1	11	0	3	1+2×SCY	3	0	4+2×SCY
1	0	00	1	0	3+2×SCY	1	1¼	3+2×SCY
1	0	10	1	1¼	1½+2×SCY	2	0	2¾+2×SCY
1	0	11	1	1½	1¼+2×SCY	2	0	2¾+2×SCY
1	1	00	1	0	3+2×SCY	1	1¼	3+2×SCY
1	1	10	1	2	¾+2×SCY	2	0	2¾+2×SCY
1	1	11	1	3	¾+2×SCY	3	0	3¾+2×SCY

14.4.2.3 Chip-Select Assertion Timing

The banks selected to work with the GPCM support an option to drive the $\overline{\text{LCS}n}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCS}n}$ 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 (for LCRR[CLKDIV] = 2 (clock ratio of 4)), when OR*n*[XACS] = 1.
- Two clock cycles later, when OR*n*[XACS] = 1.
- Three clock cycles later, when OR*n*[XACS] = 1 and OR*n*[TRLX] = 1.

The timing diagram in Figure 14-32 shows two chip-select assertion timings.

14.4.2.3.1 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 OR*n*[SCY] and OR*n*[TRLX]. Internal generation of transfer acknowledge is enabled if OR*n*[SETA] = 0. If $\overline{\text{LGTA}}$ 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 $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR*n*[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}}$*n* in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR*n*[SCY] cycles to 2×OR*n*[SCY] cycles, allowing a maximum of 30 wait states.

14.4.2.3.2 Chip-Select and Write Enable Negation Timing

Figure 14-31 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}}$*n* is connected directly to $\overline{\text{CE}}$ of the memory device. The $\overline{\text{LWE}}$[0:3] signals are connected to the respective $\overline{\text{WE}}$[3:0] signals on the memory device where each $\overline{\text{LWE}}$[0:3] signal corresponds to a different data byte.

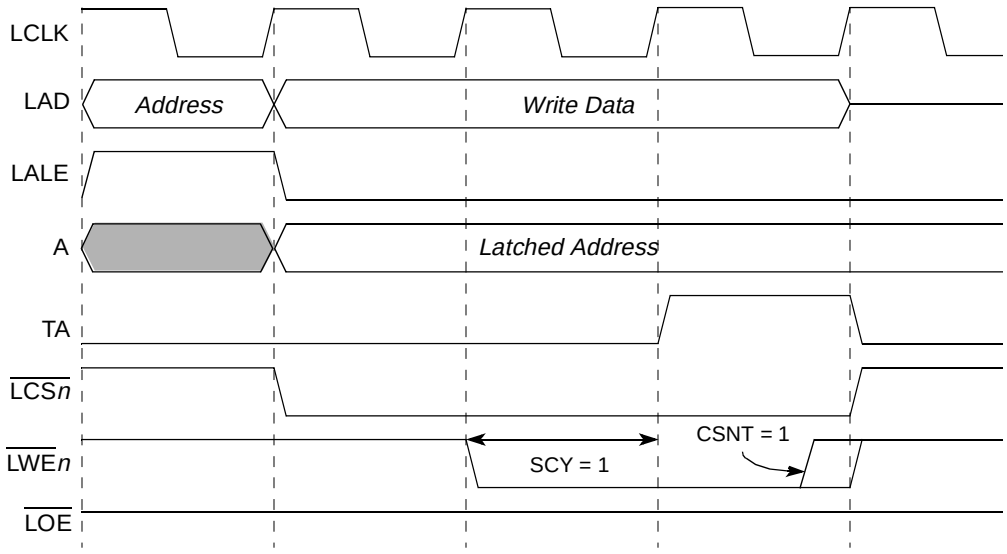


Figure 14-35. GPCM Basic Write Timing
(XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0)

1. As Figure 14-35 shows, the timing for $\overline{\text{LCS}}$*n* is the same as for the latched address. The strobes for the transaction are supplied by $\overline{\text{LOE}}$ or $\overline{\text{LWE}}$*n*, depending on the transaction direction—read or write (write case shown in the figure). OR*n*[CSNT], along with OR*n*[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, $\overline{\text{LWE}}n$ is negated one quarter of a clock earlier, as shown in Figure 14-35. $\overline{\text{LCS}}n$ is affected by CSNT and TRLX only if ACS[0] is non zero. However, $\overline{\text{LWE}}n$ 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 $\overline{\text{LCS}}n$ and $\overline{\text{LWE}}n$ 1+1/4 cycle earlier .

For example, when ACS = 00, CSNT = 1 and TRLX = 0, $\overline{\text{LWE}}n$ is negated one quarter of a clock earlier and $\overline{\text{LCS}}n$ is negated normally as shown in Figure 14-35.

14.4.2.3.3 Relaxed Timing

OR*x*[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.
- $\overline{\text{LCS}}n$ signals are negated one cycle earlier during writes (but only if ACS is not equal to 00).
- $\overline{\text{LWE}}$[0:3] signals are negated one cycle earlier during writes.

Figure 14-36 and Figure 14-37 show relaxed timing read and write transactions. The example in Figure 14-37 also shows address and data multiplexing on LAD for a pair of writes issued consecutively.

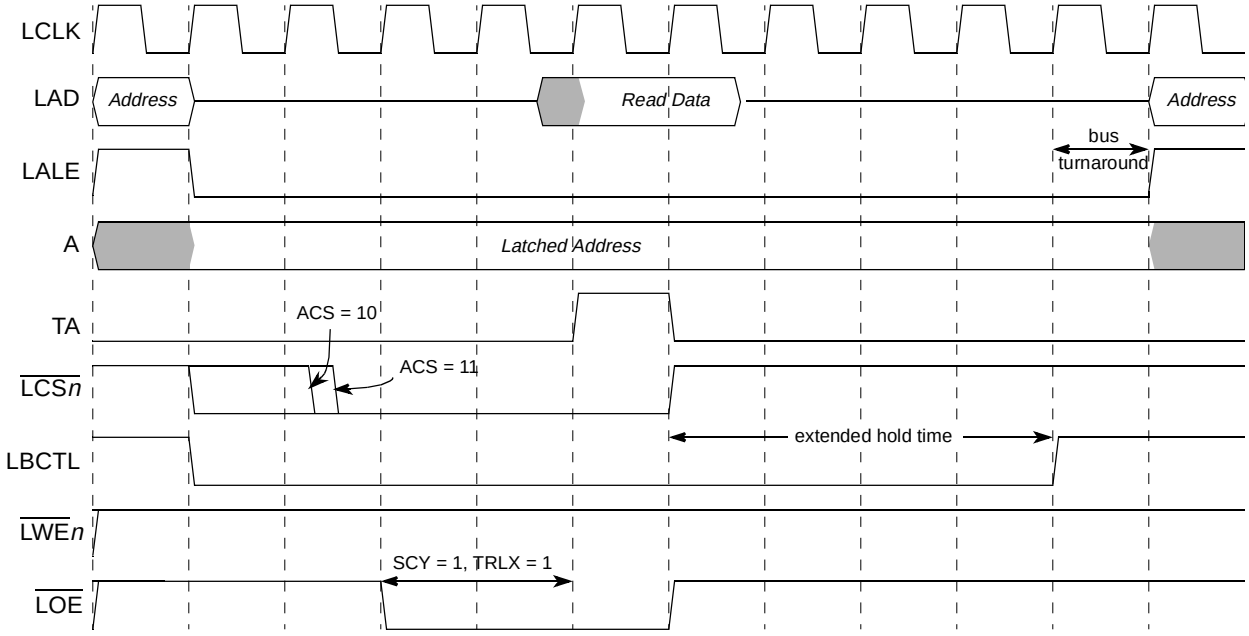


Figure 14-36. GPCM Relaxed Timing Back-to-Back Reads (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, EHTR = 0)

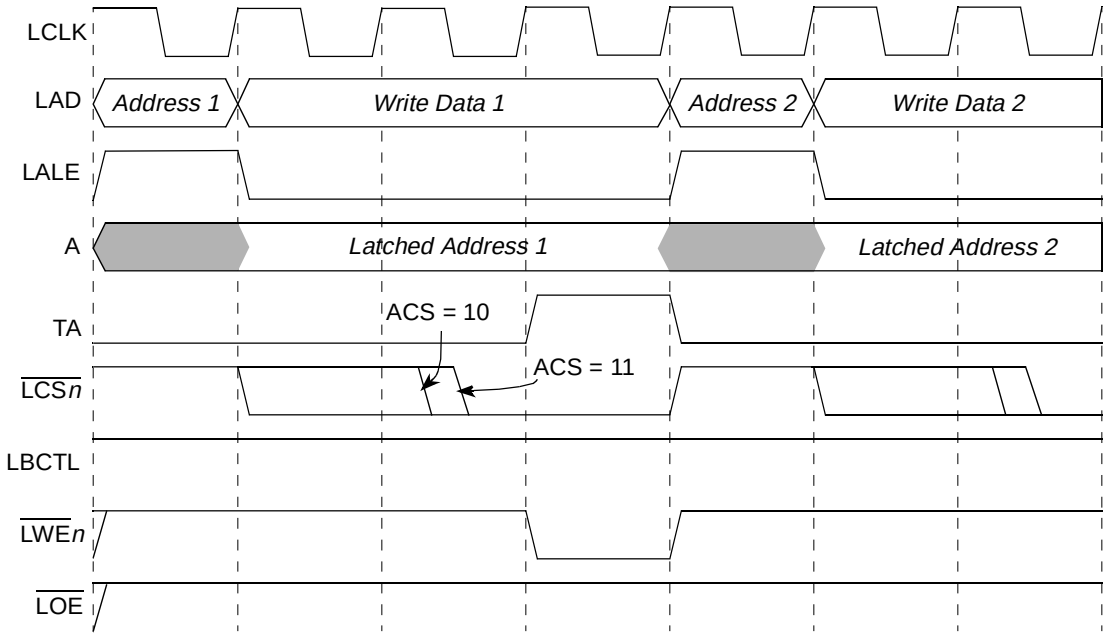


Figure 14-37. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1)

When TRLX and CSNT are set in a write access, the $\overline{\text{LWE}}$[0:3] strobe signals are negated one clock earlier than in the normal case, as shown in Figure 14-38 and Figure 14-39. If ACS ≠ 00, $\overline{\text{LCS}}n$ is also negated one clock earlier.

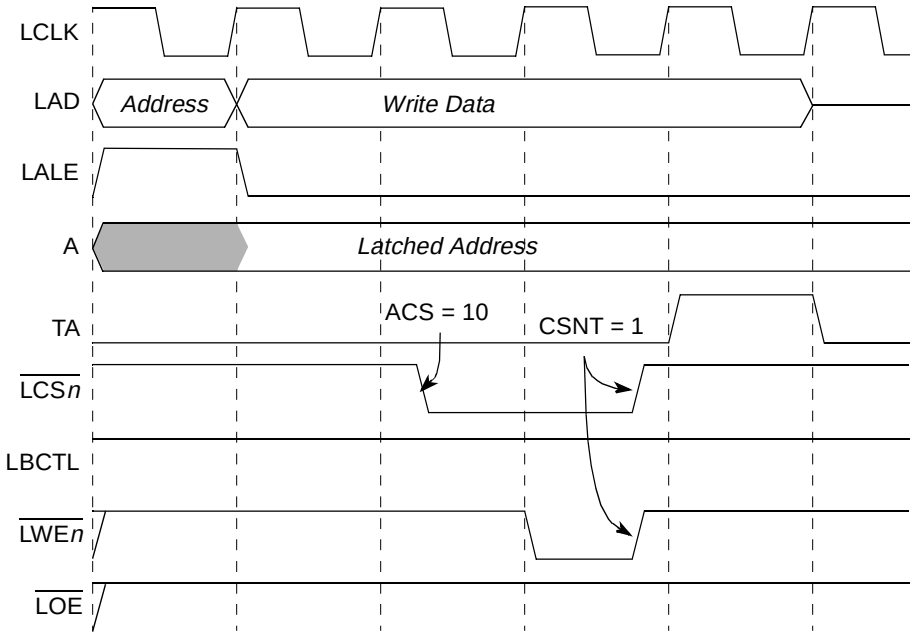


Figure 14-38. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1)

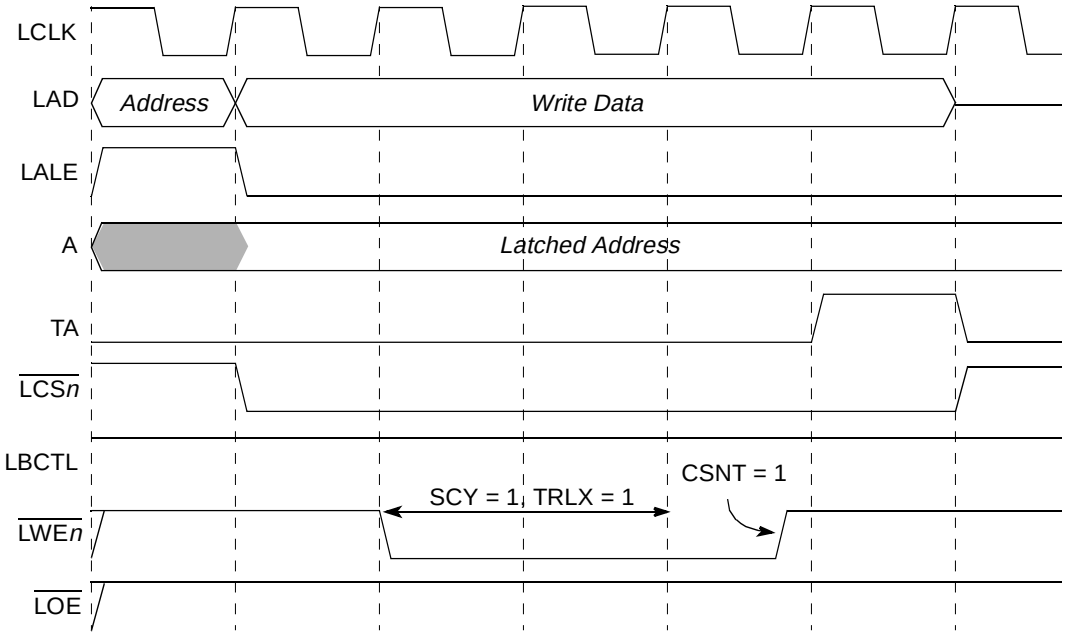


Figure 14-39. GPCM Relaxed Timing Write
(XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1)

14.4.2.3.4 Output Enable ($\overline{\text{LOE}}$) Timing

The timing of the $\overline{\text{LOE}}$ is affected only by TRLX. It always asserts and negates on the rising edge of the bus clock. $\overline{\text{LOE}}$ asserts either on the rising edge of the bus clock after $\overline{\text{LCS}n}$ is asserted or coinciding with $\overline{\text{LCS}n}$ (if XACS = 1 and ACS = 10 or ACS = 11). Accordingly, assertion of $\overline{\text{LOE}}$ can be delayed (along with the assertion of $\overline{\text{LCS}n}$) by programming TRLX = 1. $\overline{\text{LOE}}$ negates on the rising clock edge coinciding with $\overline{\text{LCS}n}$ negation

14.4.2.3.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of OR*n*[TRLX,EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in Table 14-7 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 OR*n*[EHTR].

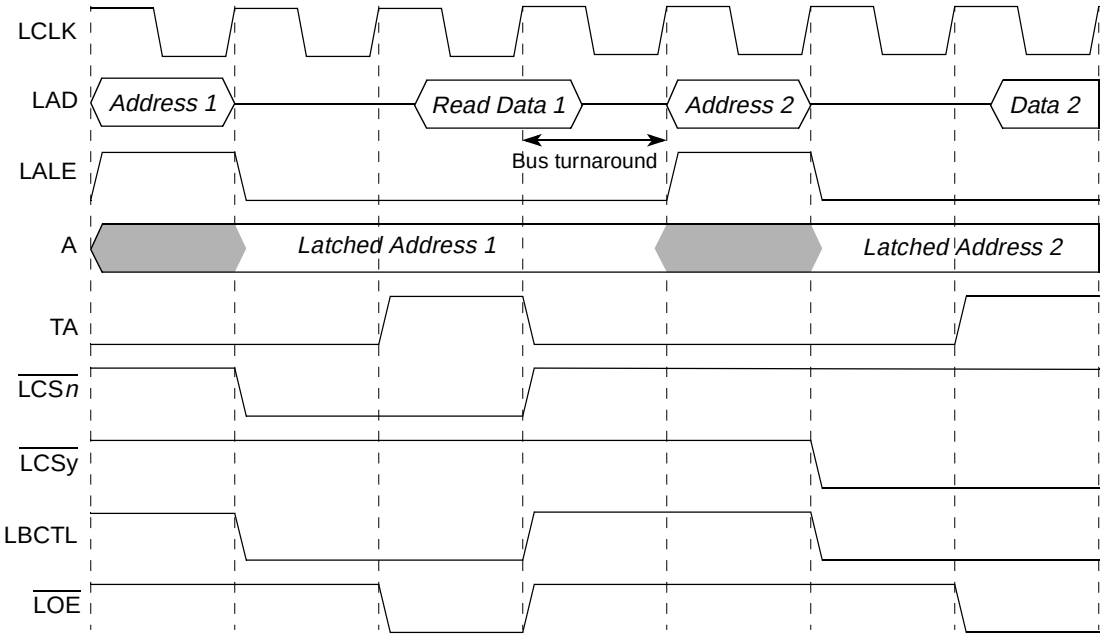


Figure 14-40. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

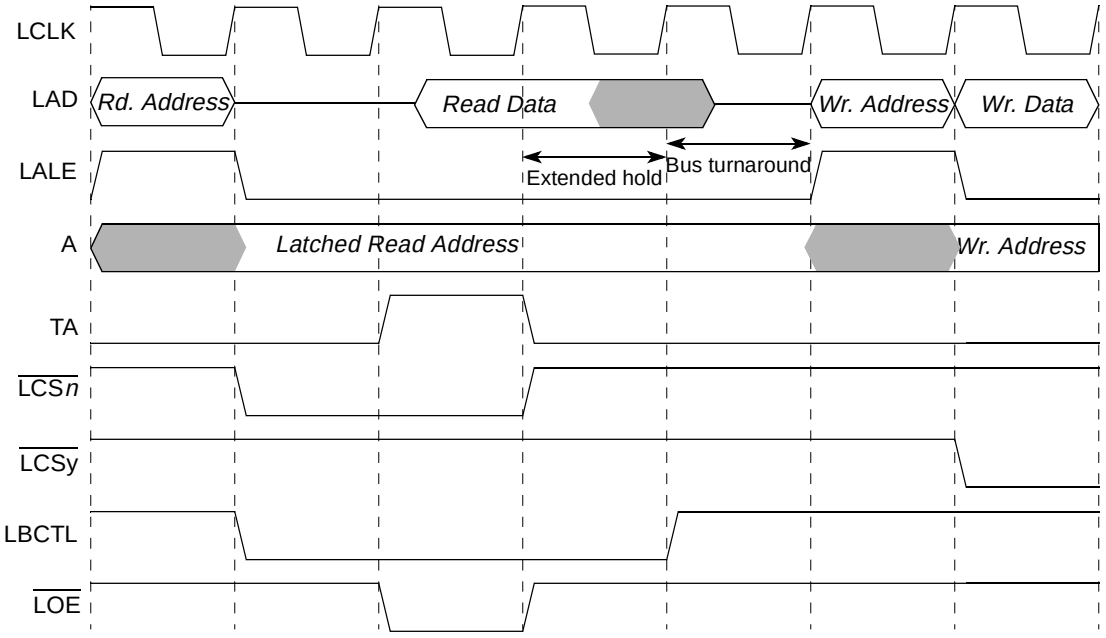


Figure 14-41. GPCM Read Followed by Write
(TRLX = 0, EHTR = 1, One-Cycle Extended Hold Time on Reads)

14.4.2.4 External Access Termination ($\overline{\text{LGTA}}$)

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCS}}n$, the sampled $\overline{\text{LGTA}}$ signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of OR*n*[SETA]). $\overline{\text{LGTA}}$ should be asserted for at least one

bus cycle to be effective. Note that because $\overline{\text{LGTA}}$ is synchronized, bus termination occurs two cycles after $\overline{\text{LGTA}}$ assertion, so in case of read cycle, the device still must drive data as long as $\overline{\text{LOE}}$ is asserted.

The user selects whether transfer acknowledge is generated internally or externally ($\overline{\text{LGTA}}$) by programming OR*n*[SETA]. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if OR*n*[SETA] = 0 (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if OR*n*[SETA] = 1. The timing of $\overline{\text{LGTA}}$ is illustrated by the example in Figure 14-42.

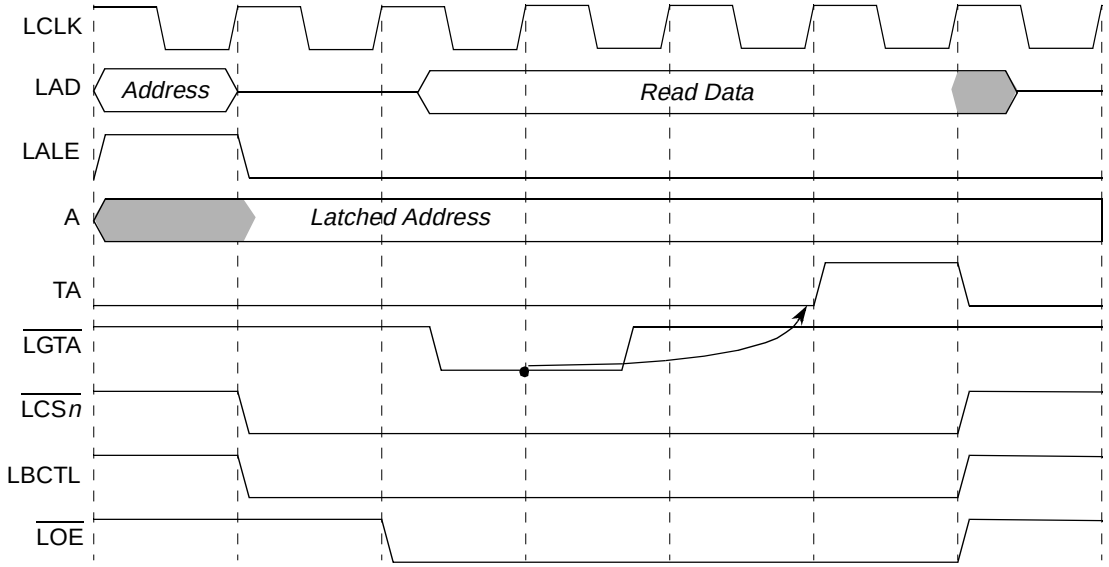


Figure 14-42. External Termination of GPCM Access

14.4.2.5 GPCM Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{\text{LCS0}}$ 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, $\overline{\text{LCS0}}$ 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. $\overline{\text{LCS0}}$ 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 14-32 describes the initial values of the boot bank in the memory controller.

Table 14-32. Boot Bank Field Values after Reset for GPCM as Boot Controller

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	PS	*From cfg_rom_loc*
	DECC	00
	WP	0
	MSEL	000
	ATOM	00
	V	1

Table 14-32. Boot Bank Field Values after Reset for GPCM as Boot Controller (continued)

Register	Field	Setting
OR0	AM	0000_0000_0000_0000_0
	BCTLD	0
	CSNT	1
	ACS	11
	XACS	1
	SCY	1111
	SETA	0
	TRLX	1
	EHTR	1
	EAD	1

14.4.3 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—BR*n*, OR*n*, and FMR.

Figure 14-43 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][1], with $\overline{\text{LFWE}}$ asserted for transfers written to the device, or $\overline{\text{LFRE}}$ 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/$\overline{\text{BSY}}$ 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 ($\overline{\text{LCS0}}$) prior to the system being fully configured.

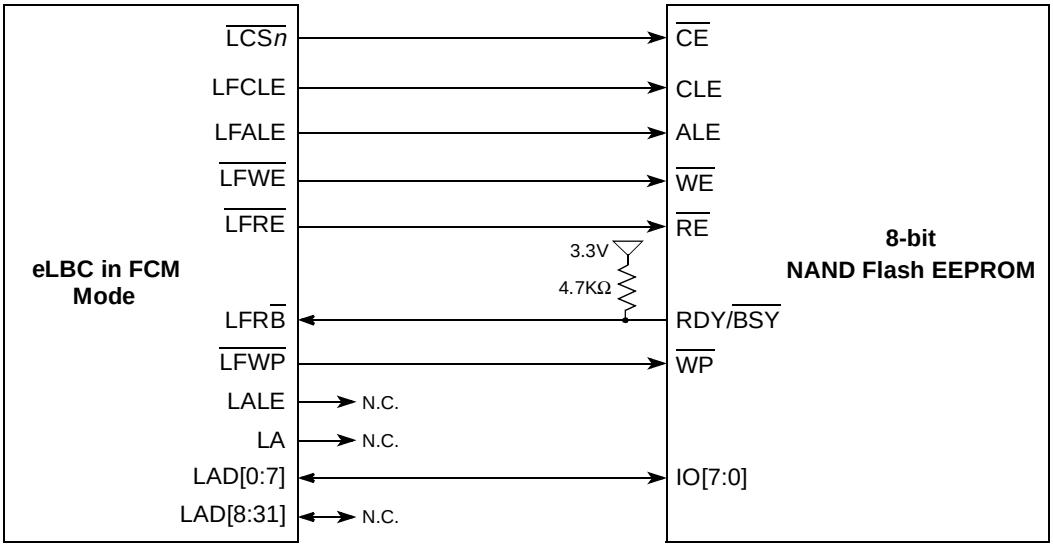


Figure 14-43. Local Bus to 8-bit FCM Device Interface

Basic read access timing for FCM is shown in Figure 14-44. Although LCLK is shown for reference, NAND Flash EEPROMs do not make use of the clock.

1. Note bit numbering reversal: LAD[0] (msb) connects to Flash IO[7], while LAD[7] (lsb) connects to IO[0].

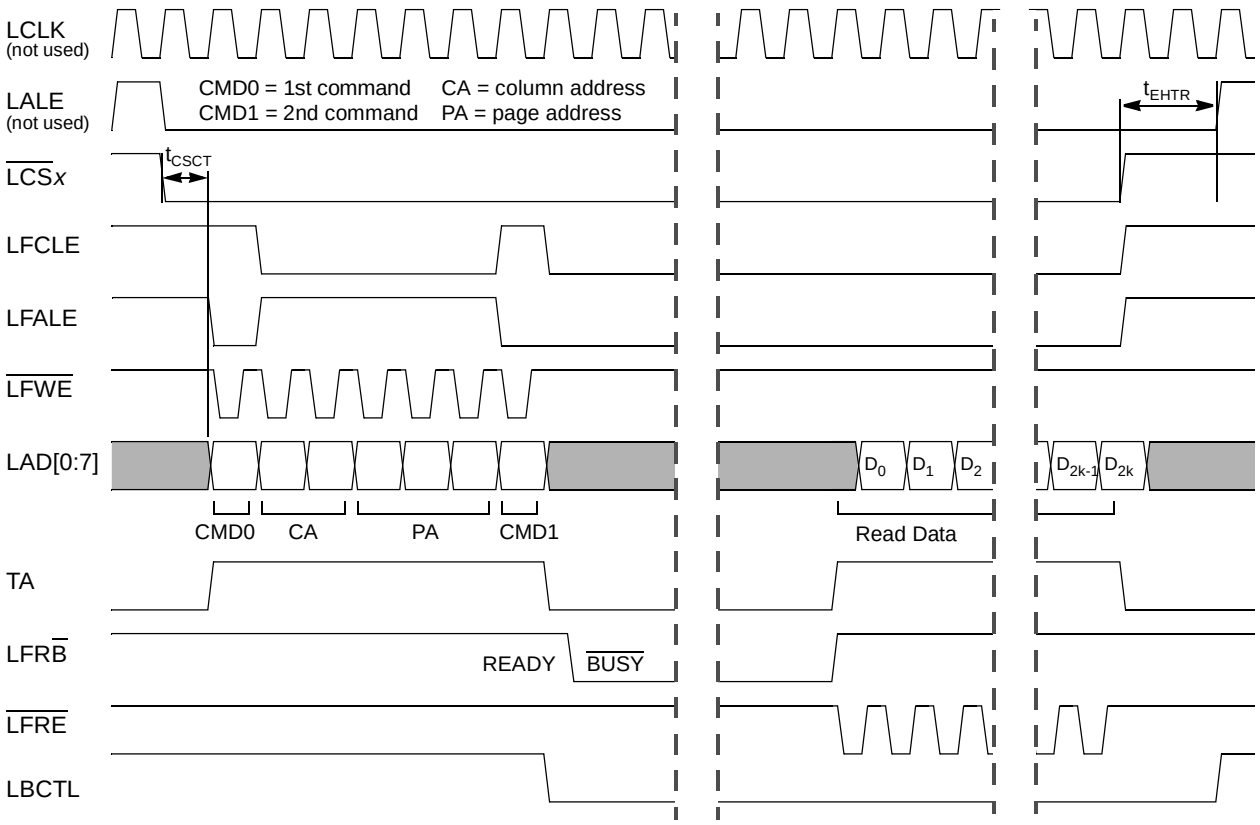


Figure 14-44. FCM Basic Page Read Timing
(PGS = 1, CSCT = 0, CST = 0, CHT = 1, RST = 1, SCY = 0, TRLX = 0, EHTR = 1)

Following the assertion of LALE, FCM asserts $\overline{\text{LCS}n}$ 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 $\overline{\text{LFWE}}$, 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 LFR$\overline{\text{B}}$ low to indicate that the device is busy. Since in Figure 14-44 FCM is now expecting a read response, it takes LBCTL low to turnaround any bus transceivers that are present. Upon LFR$\overline{\text{B}}$ indicating ready status, FCM asserts $\overline{\text{LFRE}}$ 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 $\overline{\text{LCS}n}$ and delays eLBC by t_{EHTR} before any subsequent memory access occurs.

14.4.3.1 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 $\overline{\text{LCS}n}$, 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 OR*n*[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 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 $\overline{\text{LFWP}}$ 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 Section 14.4.3.4.2, “Boot Block Loading into the FCM Buffer RAM,” for a description of the shared buffer RAM layout during boot.

14.4.3.1.1 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 (OR*n*[PGS] = 0), mapped as shown in Figure 14-45. 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* mod 8), where *P* = FPAR[PI]. Since the bank size set by OR*n*[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 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.

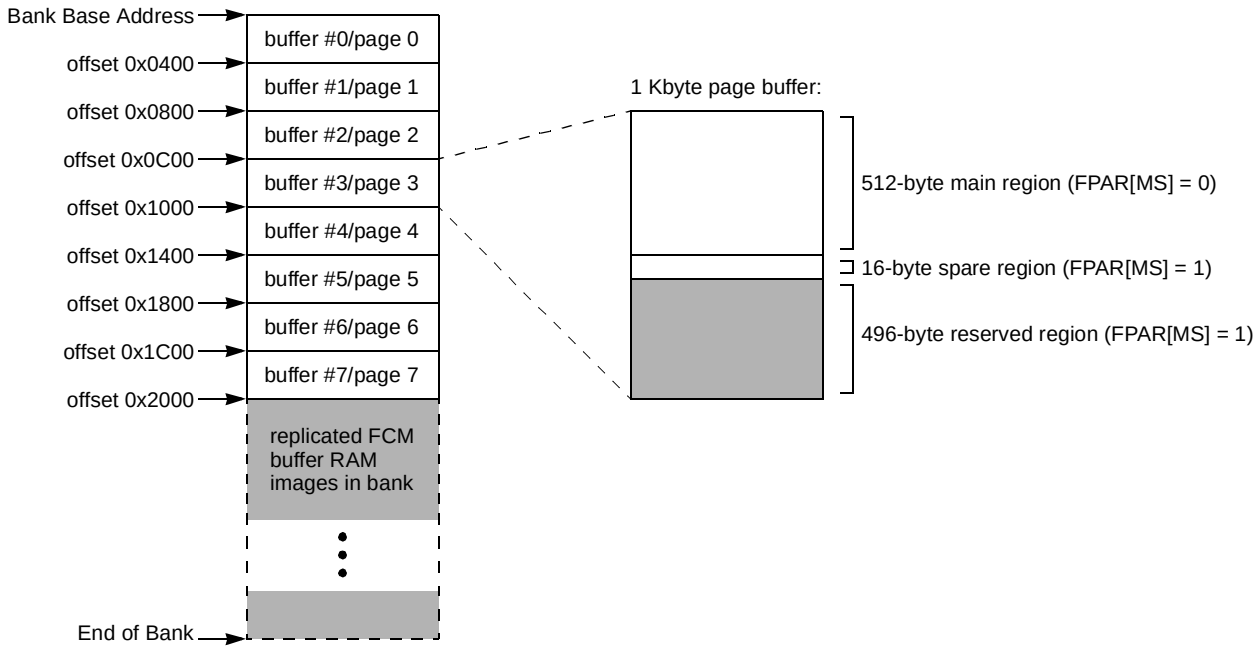


Figure 14-45. FCM Buffer RAM Memory Map for Small-Page (512-byte page) NAND Flash Devices

14.4.3.1.2 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 (OR*n*[PGS] = 1), mapped as shown in Figure 14-46. 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* mod 2), where *P* = FPAR[PI]. Since the bank size set by OR*n*[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.

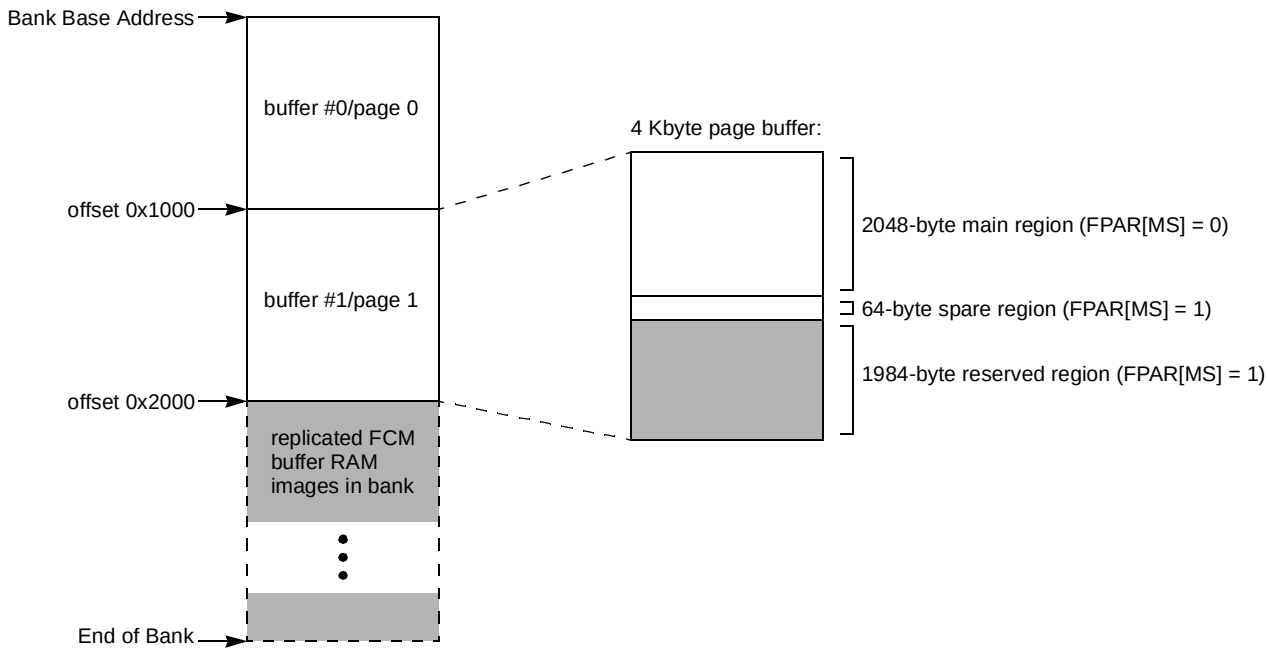


Figure 14-46. FCM Buffer RAM Memory Map for Large-Page (2-Kbyte page) NAND Flash Devices

14.4.3.1.3 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 14-47. 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}', ..., 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, etc.), 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, etc.).

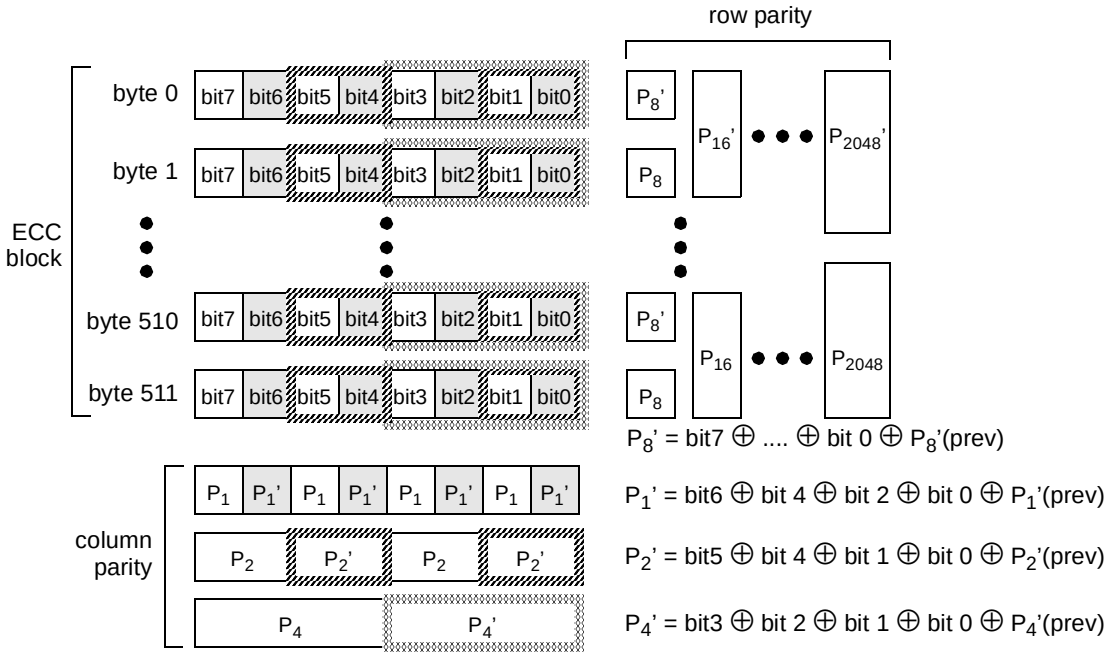


Figure 14-47. FCM ECC Calculation

The 24-bit ECC code word format is shown in Figure 14-48 for normal ECC polarity. Setting LBCR[EPAR] = 1 changes ECC polarity, and thus omits negation of each P_N and P_N' bit.

	0 (MSB)	1	2	3	4	5	6	7 (LSB)
EC0	~P_{64}	~P_{64}'	~P_{32}	~P_{32}'	~P_{16}	~P_{16}'	~P_8	~P_8'
EC1	~P_{1024}	~P_{1024}'	~P_{512}	~P_{512}'	~P_{256}	~P_{256}'	~P_{128}	~P_{128}'
EC2	~P_4	~P_4'	~P_2	~P_2'	~P_1	~P_1'	~P_{2048}	~P_{2048}'

Figure 14-48. ECC Layout for LBCR[EPAR] = 0 (~ represents logical negation)

The placement of ECC code words in relation to FMR[ECCM] is shown in Figure 14-49. 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 (BR*n*[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 Figure 14-49. 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 BR*n*[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 uncorrectable.

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	—				

Figure 14-49. ECC Placement in NAND Flash Spare Regions in Relation to FMR[ECCM]

14.4.3.2 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 14-50. 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 $\overline{\text{LCS}n}$, otherwise, $\overline{\text{LCS}n}$ 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 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 Section 14.3.1.17, "Flash Instruction Register (FIR)."

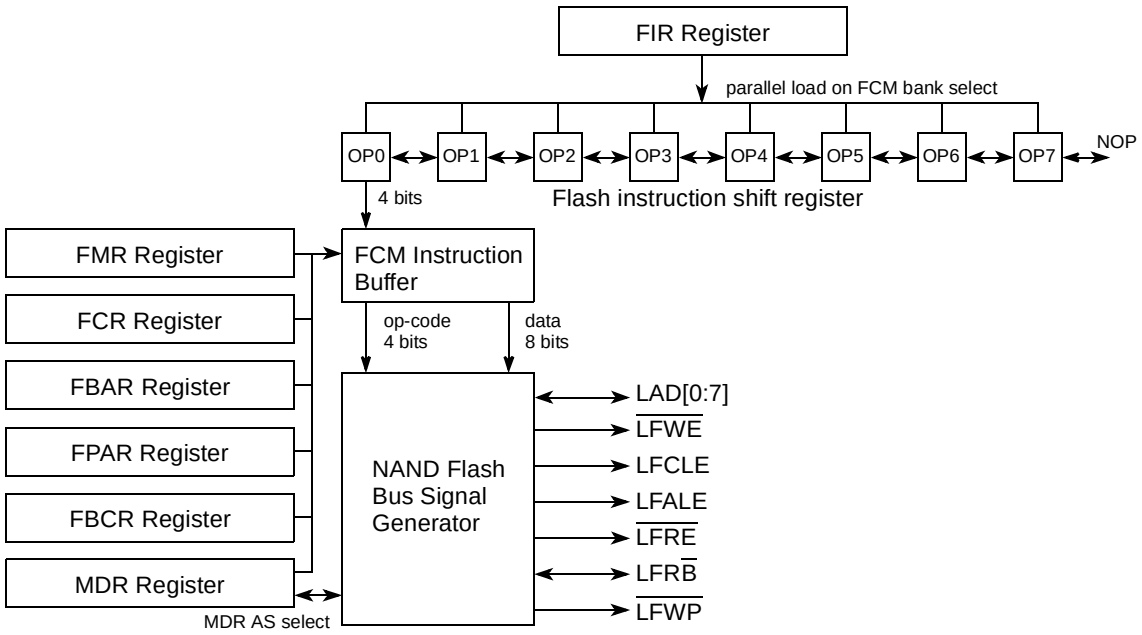


Figure 14-50. FCM Instruction Sequencer Mechanism

14.4.3.2.1 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 $\overline{\text{LFWE}}$ 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 LFR$\overline{\text{B}}$ to be sampled high (EEPROM in ready state) before issuing—CW0, and CW1. These commands first poll the LFR$\overline{\text{B}}$ 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 LFR$\overline{\text{B}}$. Rather, FCM waits for 8×(2+OR*n*[SCY]) clock cycles (when OR*n*[TRLX] = 0) or 16×(2+OR*n*[SCY]) clock cycles (when OR*n*[TRLX] = 1) before sampling the level of LFR$\overline{\text{B}}$. If the level of LFR$\overline{\text{B}}$ 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.

14.4.3.2.2 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 LFCLE nor $\overline{\text{LFWE}}$ are asserted. Thus a NOP instruction may be used to insert a pause matching the time taken for a regular command write.

14.4.3.2.3 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 $\overline{\text{LFWE}}$ asserted together. There are three kinds of address generation provided:

- Column address—CA. A column address comprises one byte (OR*n*[PGS] = 0) or two bytes (OR*n*[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.

14.4.3.2.4 FCM Data Read Instructions

Data read instructions assert $\overline{\text{LFRE}}$ 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 $\overline{\text{LFRE}}$ 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 LFR$\overline{\text{B}}$ 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 LFR$\overline{\text{B}}$ pin follow the behavior of CW*n* instructions.
- Read data/status to MDR once waited on ready—RSW. This instruction first polls the LFR$\overline{\text{B}}$ 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 LFR$\overline{\text{B}}$ pin follow the behavior of CW*n* instructions.

14.4.3.2.5 FCM Data Write Instructions

Data write instructions assert $\overline{\text{LFWE}}$ 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 $\overline{\text{LFWE}}$ 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.

14.4.3.3 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.

14.4.3.3.1 FCM Chip-Select Timing

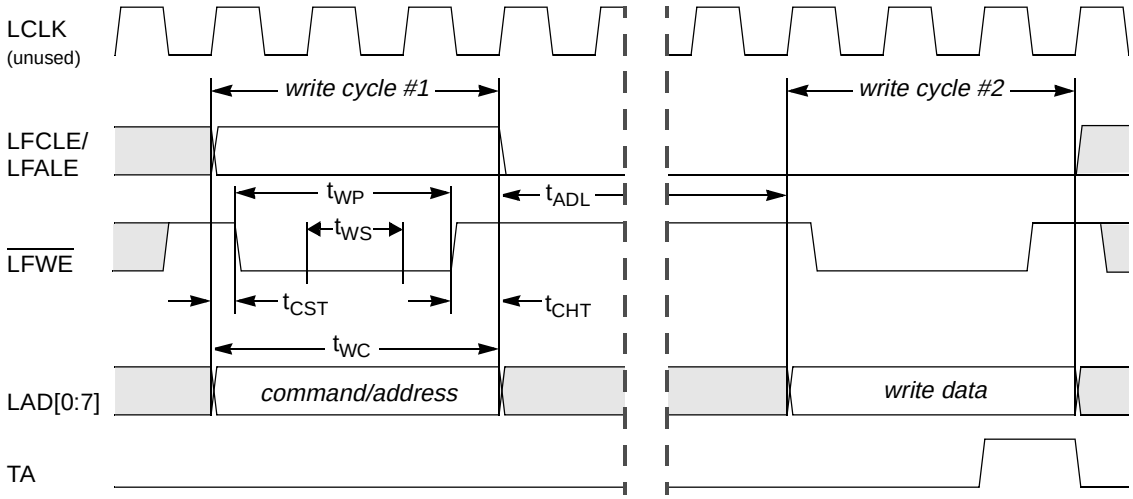
The timing of $\overline{\text{LCS}n}$ assertion in FCM mode is illustrated by the timing diagram in Figure 14-44. $\overline{\text{LCS}n}$ is asserted immediately following LALE negation, and remains asserted until the last instruction in FIR has completed. The delay, t_{CSCT}, between $\overline{\text{LCS}n}$ assertion and commencement of the first NAND Flash instruction is controlled by OR*n*[CSCT] and OR*n*[TRLX], as shown in Table 14-33. OR*n*[CSCT] should be set in accordance with the NAND Flash EEPROM chip-select to $\overline{\text{WE}}$ set-up time specification.

Table 14-33. FCM Chip-Select to First Command Timing

ORn[TRLX]	ORn[CSCT]	$\overline{\text{LCS}n}$ 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

14.4.3.3.2 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 $\overline{\text{LFWE}}$ 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 $\overline{\text{LFWE}}$, as shown in Figure 14-51.



Notes: t_{CST} = Command to $\overline{\text{LFWE}}$ set-up time. t_{WP} = $\overline{\text{LFWE}}$ pulse time, driven low.
t_{CHT} = Command to $\overline{\text{LFWE}}$ hold time. t_{WS} = Command wait state time.
t_{ADL} = Command/address to write data delay. t_{WC} = Command cycle time.

Figure 14-51. Timing of FCM Command/Address and Write Data Cycles (for TRLX = 0, CHT = 0, CST = 1, SCY = 1)

The timing parameters are summarized in Table 14-34.

Table 14-34. 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	½	SCY	1½+SCY	2+SCY	4×(2+SCY)
0	0	1	¼	½	SCY	1¼+SCY	2+SCY	4×(2+SCY)
0	1	0	0	1	SCY	1+SCY	2+SCY	4×(2+SCY)
0	1	1	¼	1	SCY	¾+SCY	2+SCY	4×(2+SCY)
1	0	0	½	1½	2×SCY	1+2×SCY	3+2×SCY	8×(2+SCY)
1	0	1	1	1½	2×SCY	½+2×SCY	3+2×SCY	8×(2+SCY)
1	1	0	½	2	2×SCY	½+2×SCY	3+2×SCY	8×(2+SCY)
1	1	1	1	2	2×SCY	2×SCY	3+2×SCY	8×(2+SCY)

1 In the parameters, SCY refers to a delay of OR*n*[SCY] clock cycles.

An example of minimum delay command timing appears in Figure 14-52. Note that the set-up, wait-state, and hold timing of command, address, and write data cycles with respect to $\overline{\text{LFWE}}$ assertion are all identical, and that the minimum cycle extends for two LCLK clock cycles.

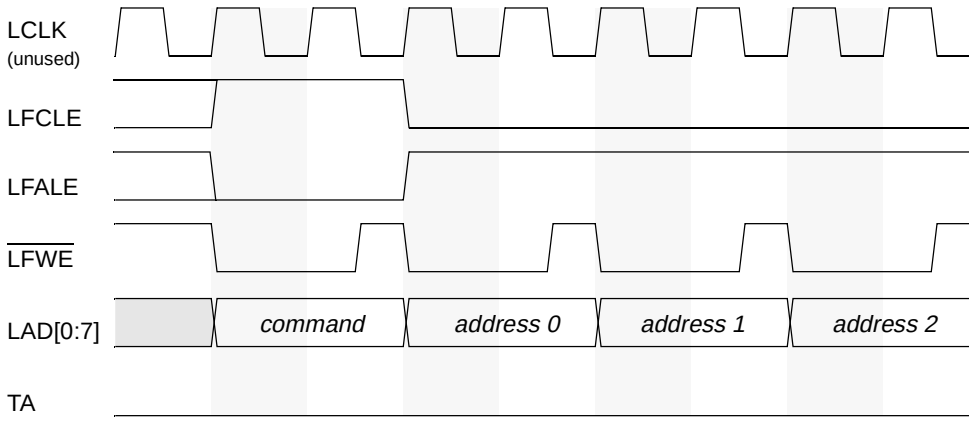


Figure 14-52. Example of FCM Command and Address Timing with Minimum Delay Parameters (for TRLX = 0, CHT = 0, CST = 0, SCY = 0)

An example of relaxed command timing is shown in Figure 14-53.

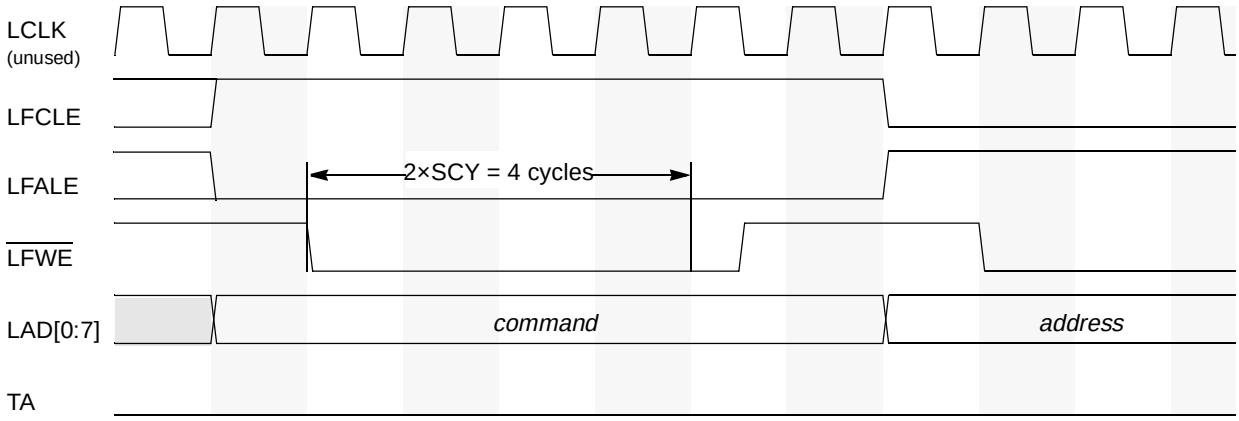


Figure 14-53. Example of FCM Command and Address Timing with Relaxed Parameters (for TRLX = 1, CHT = 0, CST = 1, SCY = 2)

14.4.3.3.3 FCM Ready/Busy Timing

Instructions CW0, CW1, RBW, and RSW force FCM to observe the state of the LFR$\overline{B}$ 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 14-54 before sampling the state of LFR$\overline{B}$. This guards against observing LFR$\overline{B}$ before it has been properly driven low by the device, but does not preclude LFR$\overline{B}$ from remaining high after a command. In addition, FCM samples and compares the state of LFR$\overline{B}$ on two consecutive cycles of LCLK to filter out noise on this signal as it rises to the ready state (LFR$\overline{B}$ = 1).

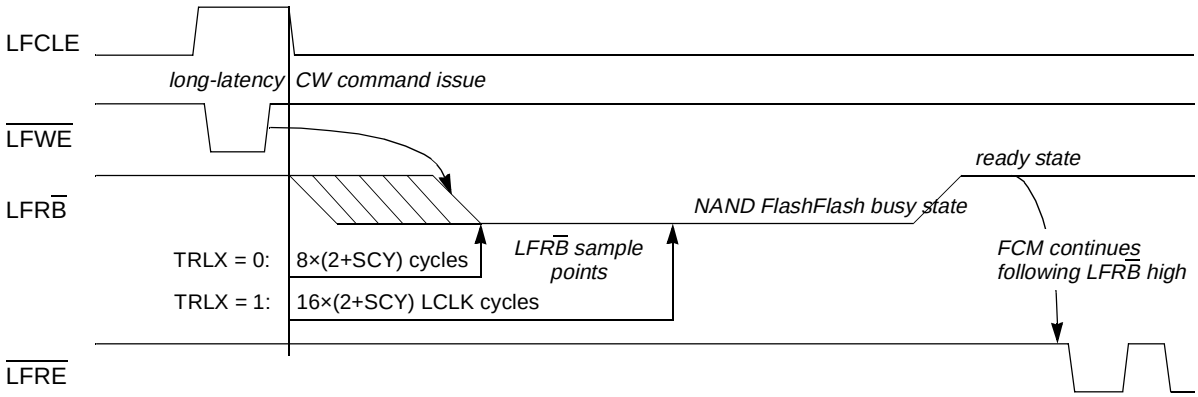
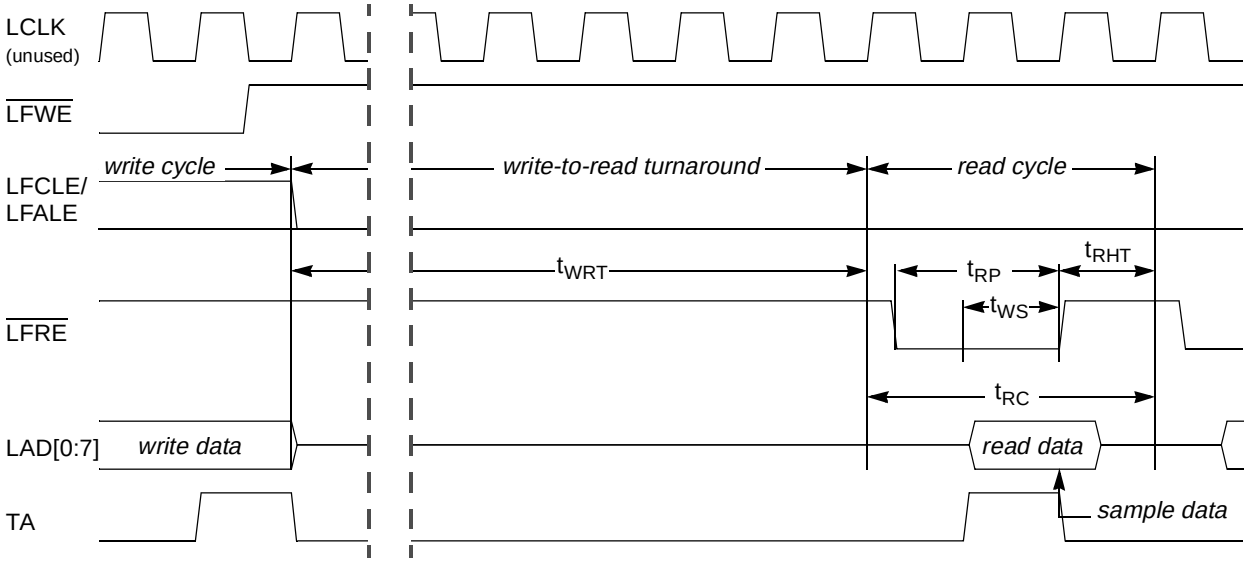


Figure 14-54. FCM Delay Prior to Sampling LFR$\overline{B}$ State

14.4.3.3.4 FCM Read Data Timing

The timing for read data transfers is shown in Figure 14-55. Upon assertion of $\overline{\text{LFRE}}$, the Flash device will enable its output drivers and drive valid read data while $\overline{\text{LFRE}}$ is held low. FCM samples read data on the rising edge of $\overline{\text{LFRE}}$, 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 LFR$\overline{B}$ sample timing takes effect (see Section 14.4.3.3.3, “FCM Ready/Busy Timing”).



Notes: t_{RP} = $\overline{\text{LFRE}}$ pulse time, read period. t_{WS} = Read wait state time.
t_{RHT} = $\overline{\text{LFRE}}$ hold time. t_{RC} = Read data cycle time.
t_{WRT} = Write to read turnaround time.

Figure 14-55. FCM Read Data Timing (for TRLX = 0, RST = 0, SCY = 1)

The timing parameters are summarized in Table 14-35.

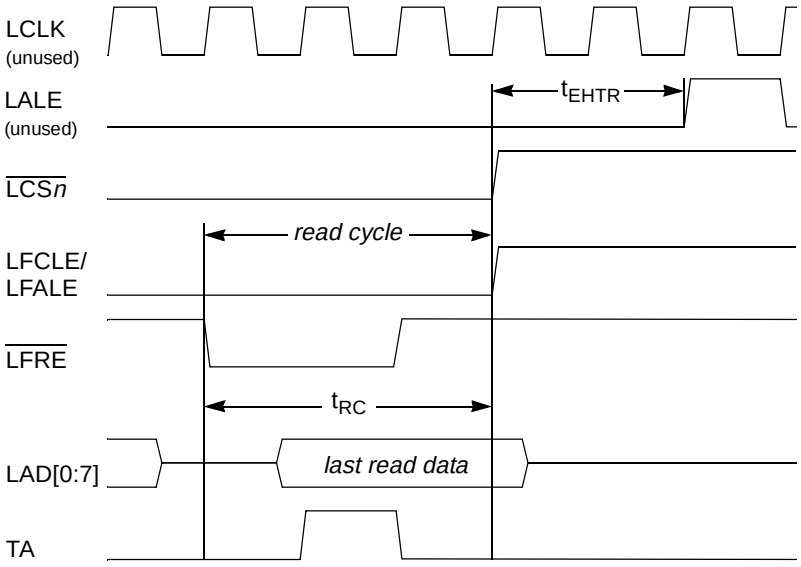
Table 14-35. FCM Read Data Timing Parameters

Option Register Attributes		Timing Parameter (LCLK Clock Cycles)[1]				
TRLX	RST	t_{RP}	t_{RHT}	t_{WS}	t_{RC}	t_{WRT}
0	0	¾+SCY	1	SCY	2+SCY	4×(2+SCY)
0	1	1+SCY	1	SCY	2+SCY	4×(2+SCY)
1	0	½+2×SCY	2	2×SCY	3+2×SCY	8×(2+SCY)
1	1	1+2×SCY	2	2×SCY	3+2×SCY	8×(2+SCY)

1 In the parameters, SCY refers to a delay of OR*n*[SCY] clock cycles.

14.4.3.3.5 FCM Extended Read Hold Timing

Allowance for slow output driver turn-off when reading NAND Flash EEPROMs is made via setting of OR*n*[EHTR] and OR*n*[TRLX]. The extended read data hold time, shown at t_{EHTR} in Figure 14-44 and Figure 14-56, is a delay inserted by FCM between the last data read and another eLBC memory access (requiring LALE assertion). $\overline{\text{LCS}}$*n* is negated during t_{EHTR} to allow external devices and bus transceivers time to disable their drivers.



Notes: t_{RC} = Read data cycle time.
t_{EHTR} = Extended read data hold time.

Figure 14-56. FCM Read Data Timing with Extended Hold Time (for TRLX = 0, EHTR = 1, RST = 1, SCY = 1)

14.4.3.4 FCM Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{\text{LCS0}}$ 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, $\overline{\text{LCS0}}$ is asserted initially to load a 4-Kbyte boot block into the FCM buffer RAM, but core instruction fetches occur from the buffer RAM.

14.4.3.4.1 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. $\overline{\text{LCS0}}$ 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 14-36 describes the initial values of the boot bank in the memory controller.

Table 14-36. Boot Bank Field Values after Reset for FCM as Boot Controller

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	PS	*From cfg_rom_locz*
	DECC	00
	WP	0
	MSEL	001
	ATOM	00
	V	0

Table 14-36. Boot Bank Field Values after Reset for FCM as Boot Controller (continued)

Register	Field	Setting
OR0	AM	0000_0000_0000_0000_0
	BCTLD	0
	PGS	*From cfg_rom_loc*
	CSCT	1
	CST	1
	CHT	1
	RST	1
	SCY	010
	TRLX	1
	EHTR	1

14.4.3.4.2 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 $\overline{\text{HRESET}}$ (See Section 4.4.3.4, "Boot ROM Location."). 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, as indicated in Figure 14-57.

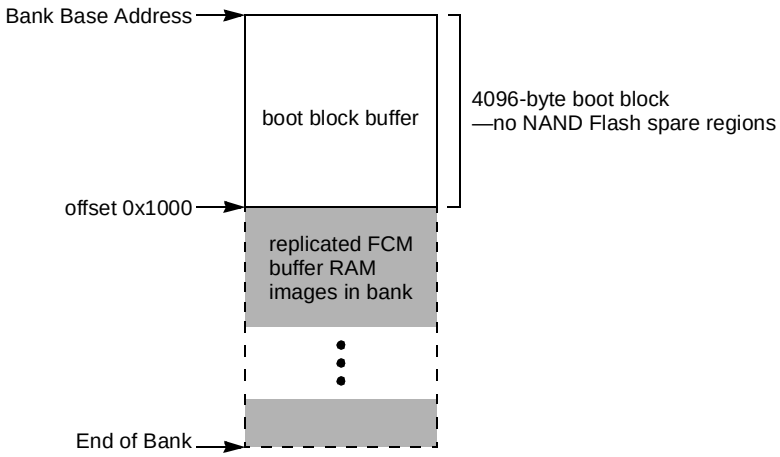


Figure 14-57. FCM Buffer RAM Memory Map During Boot Loading

The process for booting is as follows:

1. Following negation of $\overline{\text{HRESET}}$, 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 $\overline{\text{LCS0}}$. eLBC drives $\overline{\text{LFWP}}$ 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. 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 $\overline{hreset\_req}$ signal and indicates as much in the RSTRSCR register of the global utilities block.
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.

14.4.4 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 ($\overline{LCS}n$, $\overline{LBS}$[0:3] 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.

NOTE

If the LGPL4/$\overline{LGTA}$/LFR$\overline{B}$/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pull-up is required. Refer to the hardware specification for details regarding termination options.

Figure 14-58 shows the basic operation of each UPM.

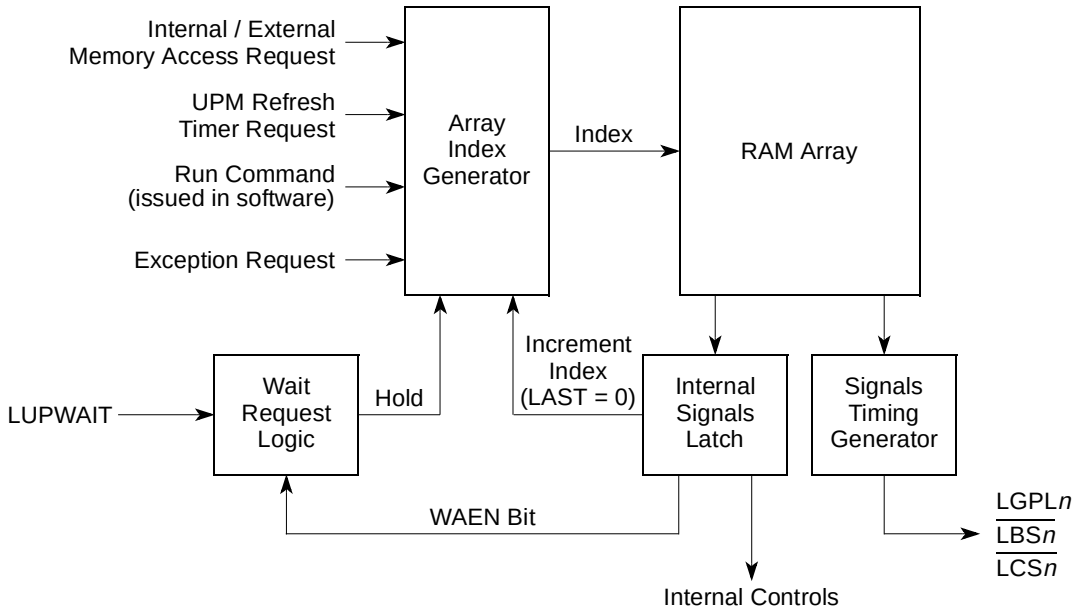


Figure 14-58. User-Programmable Machine Functional Block Diagram

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.

14.4.4.1 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 (M*x*MR[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).

Figure 14-59 and Table 14-37 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*x*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

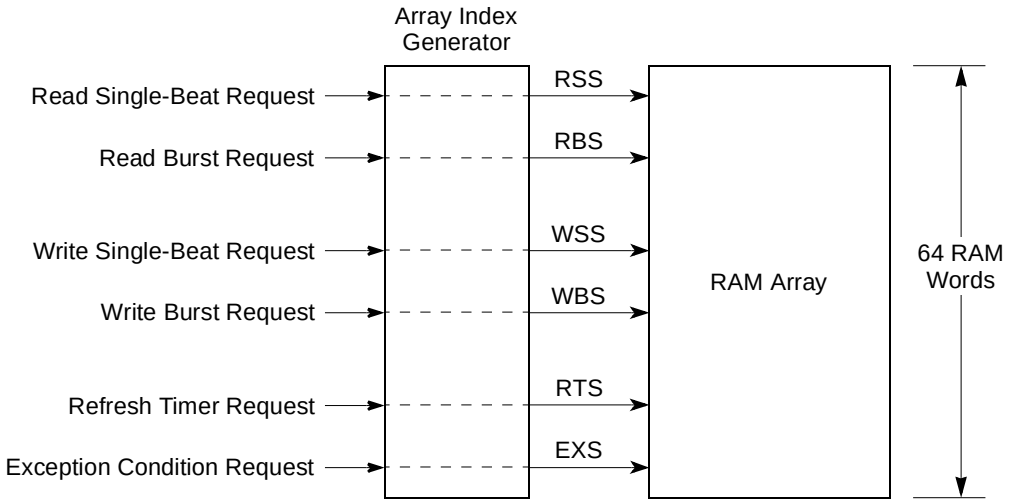


Figure 14-59. RAM Array Indexing

Table 14-37. 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

14.4.4.1.1 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 OR*n*[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.

14.4.4.1.2 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. Figure 14-60 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.

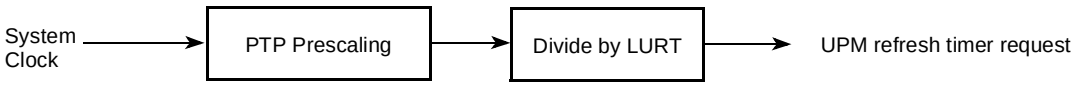


Figure 14-60. Memory Refresh Timer Request Block Diagram

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.

14.4.4.1.3 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 M*x*MR[OP] = 11 and accessing UPM*n* memory region with any write transaction that hits the corresponding UPM machine. M*x*MR[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.

14.4.4.1.4 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.

14.4.4.2 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 M*x*MR.

Patterns are written to the RAM array by setting M*x*MR[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 M*x*MR[OP] = 10).

NOTE

M*x*MR / 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 M*x*MR/MDR register and the dummy accesses to the UPM memory region, two rules must be followed:

1. Since the result of any update to the M*x*MR/MDR register must be in effect before the dummy read or write to the UPM region, a write to M*x*MR/MDR should be followed immediately by a read of M*x*MR/MDR.
2. The UPM memory region should have the same MMU settings as the memory region containing the M*x*MR 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 M*x*MR. 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.

14.4.4.2.1 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 M*x*MR 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 M*x*MR[MAD]. If incremented, the previous dummy write transaction is completed; proceed to step 6. Repeat step 5 until incremented.
6. Program M*x*MR for the second write with the desired RAM array address.
7. Write pattern/data to MDR to ensure that the M*x*MR 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 M*x*MR[MAD]. If incremented, the previous dummy write transaction is completed.

Note that if step 1 (or 6) and 2 (or 7) are reversed, step 3 (or 8) is replaced by the following:

- Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration.

14.4.4.2.2 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 (M*x*MR[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 M*x*MR for the first read with the desired RAM array address.
2. Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration, such as RAM array address.
3. Perform a dummy read transaction.
4. Read/check M*x*MR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 5. Repeat step 4 until incremented.
5. Read MDR.
6. Program M*x*MR for the second read with the desired RAM array address.
7. Read M*x*MR to ensure that the M*x*MR has already been updated with the desired configuration, such as RAM array address.
8. Perform a dummy read transaction.
9. Read/check M*x*MR[MAD]. If incremented, the previous dummy read transaction is completed; proceed to step 10. Repeat step 9 until incremented.
10. Read MDR.

14.4.4.3 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 $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$ timing specifies the value of the corresponding external signal at each quarter phase of the bus clock.

The division of UPM bus cycles into phases is shown in Figure 14-61.

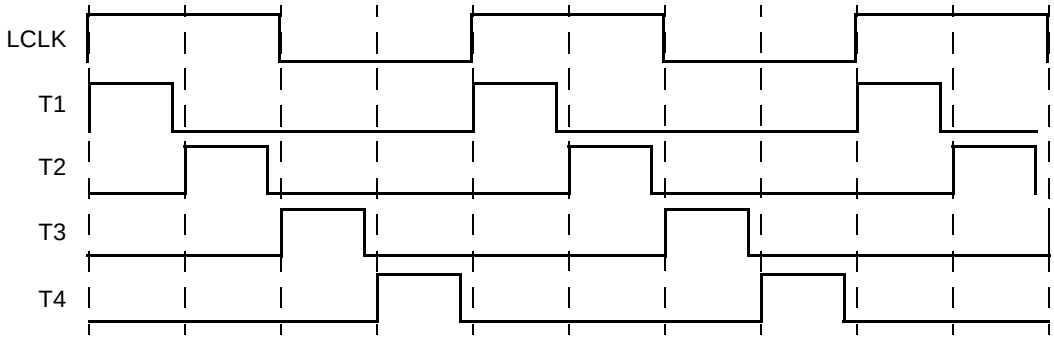


Figure 14-61. UPM Clock Scheme

14.4.4.4 RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 14-62. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}n}$ is for the bank that matches the current address. The selected $\overline{\text{LBS}}$ is for the byte lanes read or written by the access.

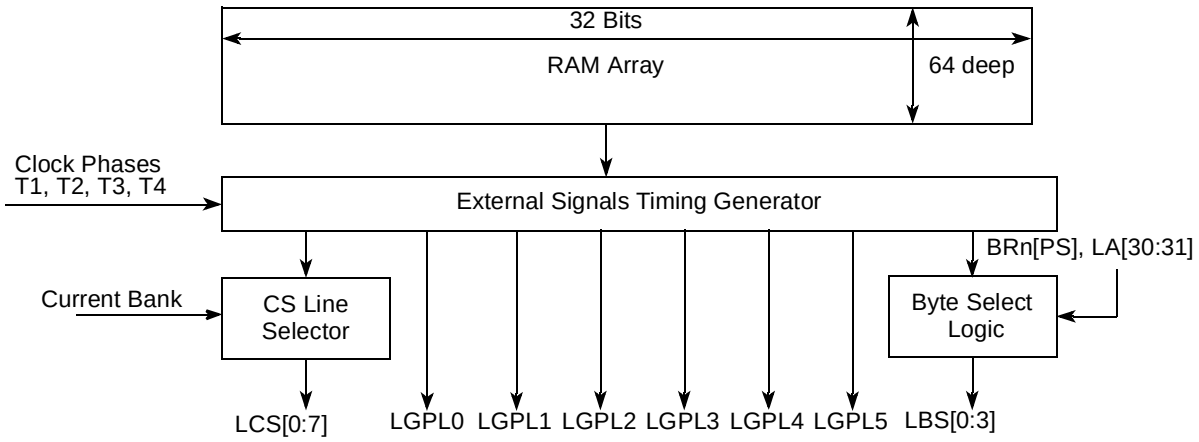


Figure 14-62. RAM Array and Signal Generation

14.4.4.4.1 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. Figure 14-37 shows the RAM word fields. The CST*n* and BST*n* bits determine the state of UPM signals $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$[0:3] at each quarter phase of the bus clock.

Offset Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H		G1T1	G1T3	G2T1	G2T3
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	G3T1	G3T3	G4T1/ DLT3	G4T3/ WAE N	G5T1	G5T3	REDO		LOOP	EXEN	AMX		NA	UTA	TODT	LAST
Reset	All zeros															

Table 14-38. RAM Word Field Descriptions

Table 14-39 describes RAM word fields.

Table 14-39. RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 1.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 2.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 3.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 4.
4	BST1	Byte select timing1. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 1.
5	BST2	Byte select timing 2. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3.
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{\text{LBS}}$ 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 M*x*MR[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 M*x*MR[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).

Table 14-39. RAM Word Field Descriptions (continued)

Bits	Name	Description
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/DLT3	General purpose line 4 timing 1/delay time 3. The function of this bit is determined by M*x*MR[GPL4]. If M*x*MR[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). If M*x*MR[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: 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	General purpose line 4 timing 3/wait enable. Bit function is determined by M*x*MR[GPL4]. If M*x*MR[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). If M*x*MR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), G4T3/WAEN is used to enable the wait mechanism: 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	Redo current RAM word. Defines the number of times to execute the current RAM word. 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	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 M*x*MR. 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. **Note:** AMX must not change values in any RAM word which begins a loop

Table 14-39. RAM Word Field Descriptions (continued)

Bits	Name	Description
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. 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. 0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.
26–27	AMX	Address multiplexing. Determines the source of LAD during an LALE phase. Any change in the AMX field initiates a new LALE (address) phase. 00 LAD (and/or in conjunction with LA) is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD (and/or in conjunction with LA) is driven with the multiplexed address according to M*x*MR[AM]. For example, row address. See Section 14.4.4.4.7, “Address Multiplexing (AMX)” for more information. 11 LAD (and/or in conjunction with LA) is driven with the contents of MAR. Used, for example, to initialize a mode. **Note:** AMX must not change values in any RAM word which begins a loop. **Note:** Source ID debug mode is only supported for the AMX = 00 setting.
28	NA	Next burst address. Determines when the address is incremented during a burst access. 0 The address increment function is disabled. 1 The address is incremented in the next cycle. In conjunction with the BR*n*[PS], the increment value of LA*n* is 1, 2, or 4 for port sizes of 8 bits, 16 bits, and 32 bits, respectively.
29	UTA	UPM transfer acknowledge. Indicates assertion of transfer acknowledge in the current cycle. 0 Transfer acknowledge is not asserted in the current cycle. 1 Transfer acknowledge is asserted in the current cycle.
30	TODT	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 M*x*MR[DS*n*]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored. 0 The disable timer is turned off. 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.
31	LAST	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 M*x*MR[DS*n*]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.

14.4.4.4.2 Chip-Select Signal Timing (CST*n*)

If BR*n*[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCS}n}$ for that bank with timing as specified in the UPM RAM word CST*n* fields. The selected UPM affects only the assertion and negation of the appropriate $\overline{\text{LCS}n}$ signal. The state of the selected $\overline{\text{LCS}n}$ signal of the corresponding bank depends on the value of each CST*n* bit. Figure 14-63 shows how UPMs control $\overline{\text{LCS}n}$ signals.

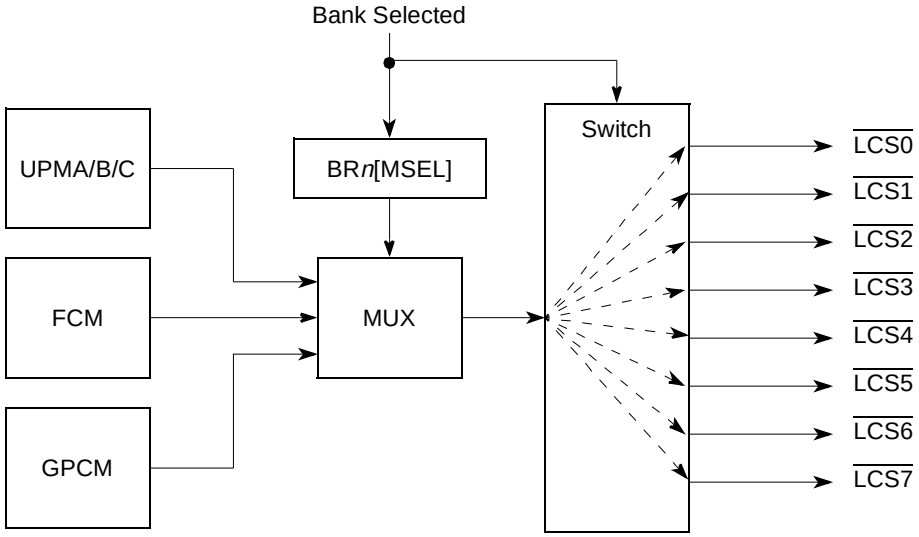


Figure 14-63. $\overline{\text{LCS}n}$ Signal Selection

14.4.4.4.3 Byte Select Signal Timing (BST*n*)

If BR*n*[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 $\overline{\text{LBS}}$[0:3] signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:3] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed. Figure 14-64 shows how UPMs control $\overline{\text{LBS}}$[0:3].

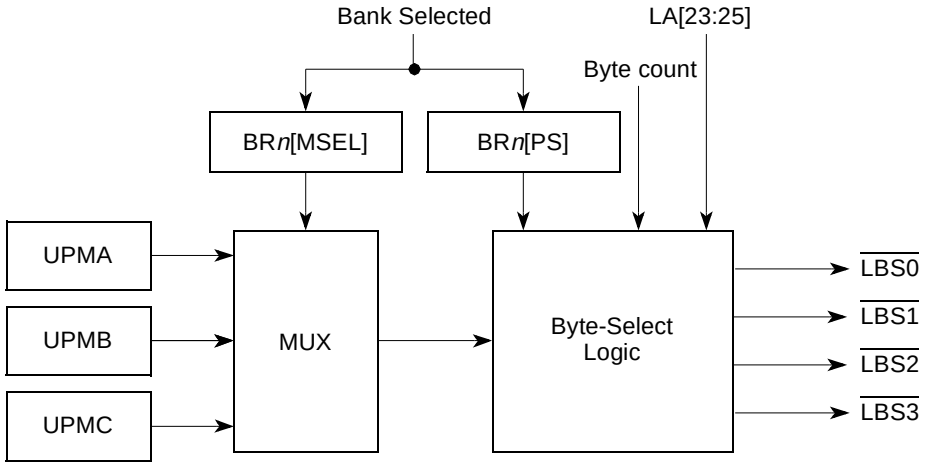


Figure 14-64. $\overline{\text{LBS}}$ Signal Selection

The uppermost byte select ($\overline{\text{LBS0}}$), when asserted, indicates that LAD[0:7] contains valid data during a cycle. Likewise, $\overline{\text{LBS1}}$ indicates that LAD[8:15] contain valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ indicates that LAD[24:31] contain valid data. For a UPM refresh timer request, all $\overline{\text{LBS}}$[0:3] signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, the $\overline{\text{LBS}}$[0:3] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

14.4.4.4.4 General-Purpose Signals (G*n*T*n*, GO*n*)

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. LGPL0 offers enhancements beyond the other LGPL*n* lines.

GPL0 can be controlled by an address line specified in M*x*MR[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.

14.4.4.4.5 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 Table 14-40. 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 14-40. M*x*MR 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

14.4.4.4.6 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.

14.4.4.4.7 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 M*x*MR[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 14-41 shows how the RAM word AMX bits and M*x*MR[AM] settings can be used to affect row × column address multiplexing on the LAD[16:31] signals. When AMX = 10, LAD[0:15] are driven low during an address phase.

Table 14-41. UPM Address Multiplexing

	msb	Internal Transaction Address																												lsb
	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
AMX = 10 MxMR[AM] = 000 (Row)									16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31								
AMX = 00 (Col)																									24	25	26	27	28	29	30	31
AMX = 10 MxMR[AM] = 001 (Row)								16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31									
AMX = 00 (Col)																								23	24	25	26	27	28	29	30	31
AMX = 10 MxMR[AM] = 010 (Row)							16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31										
AMX = 00 (Col)																							22	23	24	25	26	27	28	29	30	31
AMX = 10 MxMR[AM] = 011 (Row)						16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31											

Table 14-41. UPM Address Multiplexing (continued)

	msb	Internal Transaction Address																													lsb
	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
AMX = 00 (Col)																						21	22	23	24	25	26	27	28	29	30	31
AMX = 10 MxMR[AM] = 100 (Row)					16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31												
AMX = 00 (Col)																					20	21	22	23	24	25	26	27	28	29	30	31
AMX = 10 MxMR[AM] = 101 (Row)				16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31													
AMX = 00 (Col)																				19	20	21	22	23	24	25	26	27	28	29	30	31
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*, LA*n*} 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.

14.4.4.4.8 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 M*x*MR[GPL*n*4DIS], determines when the data input is sampled by the eLBC as follows:

- If M*x*MR[GPL*n*4DIS] = 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 GPL*n*4DIS = 0 (G4T4/DLT3 functions as G4T4), or if GPL*n*4DIS = 1 but DLT3 = 0, 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 14-65 shows how data sampling is controlled by the UPM.

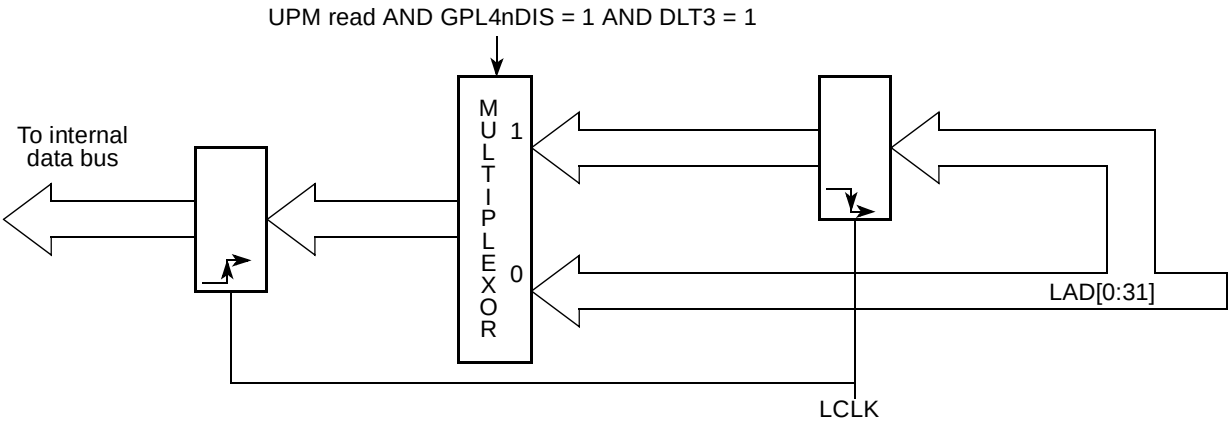


Figure 14-65. UPM Read Access Data Sampling

14.4.4.4.9 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.

14.4.4.4.10 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 rising edge of the bus clock and takes at least 1 bus cycle to complete. 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.

Figure 14-66 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 $\overline{\text{LCS}}n$ 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.

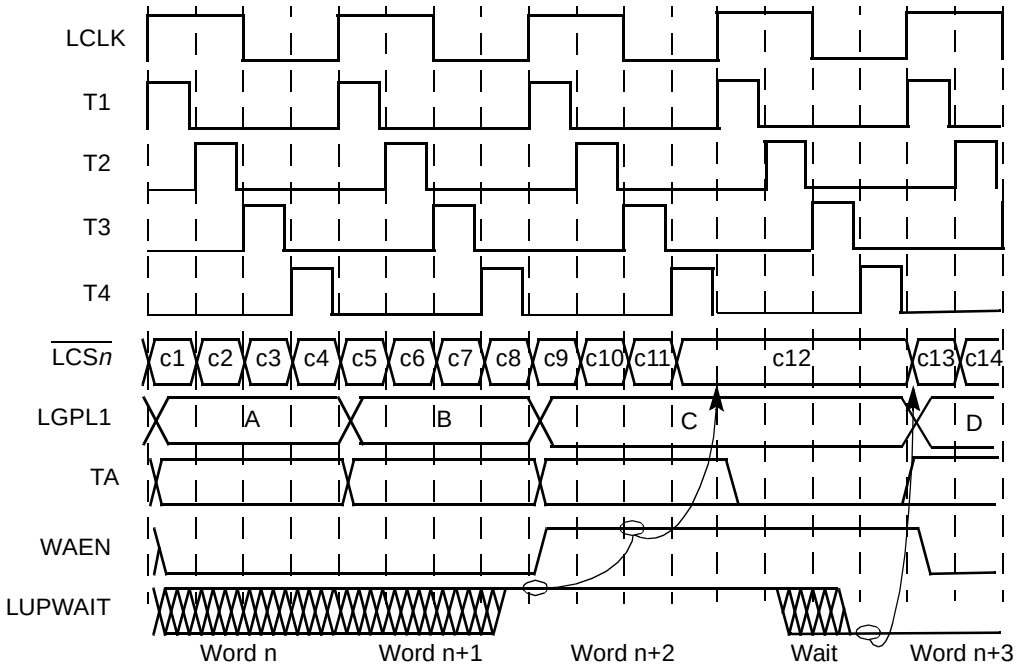


Figure 14-66. Effect of LUPWAIT Signal

14.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

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.

However, programming WAEN = 1 and UTA = 1 in the same RAM word, under certain conditions, allows the UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. The conditions are as follows:

- The PLL must be enabled, that is, LCRR[PBYP] = 0.
- DLT3 bit must be cleared in the same RAM word to avoid mid-sampling of read data.
- LBCR[LPBSE] = 0 and MXMR[GPL4] = 1
- The combination WAEN=1 and UTA=1 should be in the RAM word next to the word which gets frozen by LUPWAIT assertion. This condition limits the use of this mode to cases where the exact cycle of LUPWAIT assertion is predictable.

In this mode, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized. The acknowledge occurs early or normally depending on whether the UPM was already frozen inWAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to affect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

14.4.4.6 Extended Hold Time on Read Accesses

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.

14.5 Initialization/Application Information

14.5.1 Interfacing to Peripherals in Different Address Modes

This section provides guidelines for interfacing to peripherals.

14.5.1.1 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. Since 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[0:31] to appear on LAD[0:31]. The eLBC can then continue on into the data phase.

The eLBC supports port sizes of 8, 16, and 32 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 32-bit devices, LA[30:31] are irrelevant since these address bits are implicit in the byte lanes which carry data. Similarly, for 16-bit devices, LA[30] is used and LA[31] is irrelevant; however, for 8-bit devices, LA[30:31] are 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 separatelyand should be used whenever a device requires the five least-significant addresses. The five least-significant address bits should not be used from LAD[27:31]. All other address bits, A[0:26], must be reconstructed through the latch, as shown in Figure 14-67.

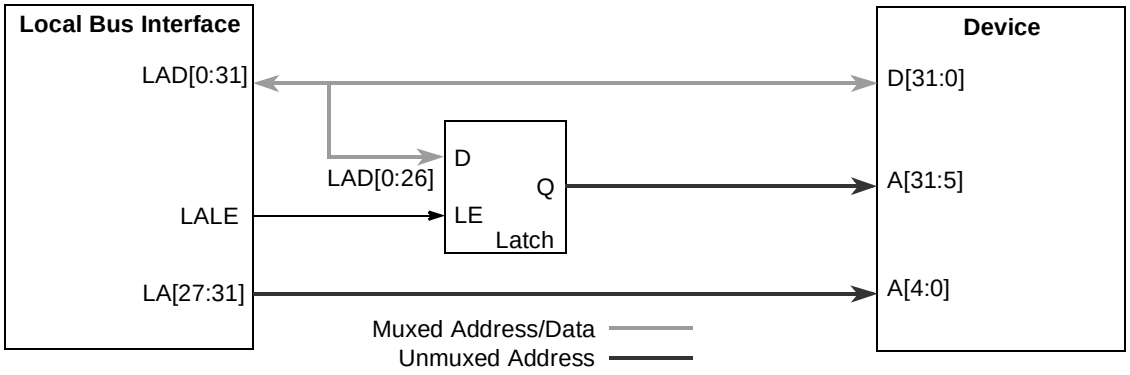


Figure 14-67. Multiplexed Address/Data Bus for 32-Bit Addressing

14.5.1.2 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. Figure 14-68 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.

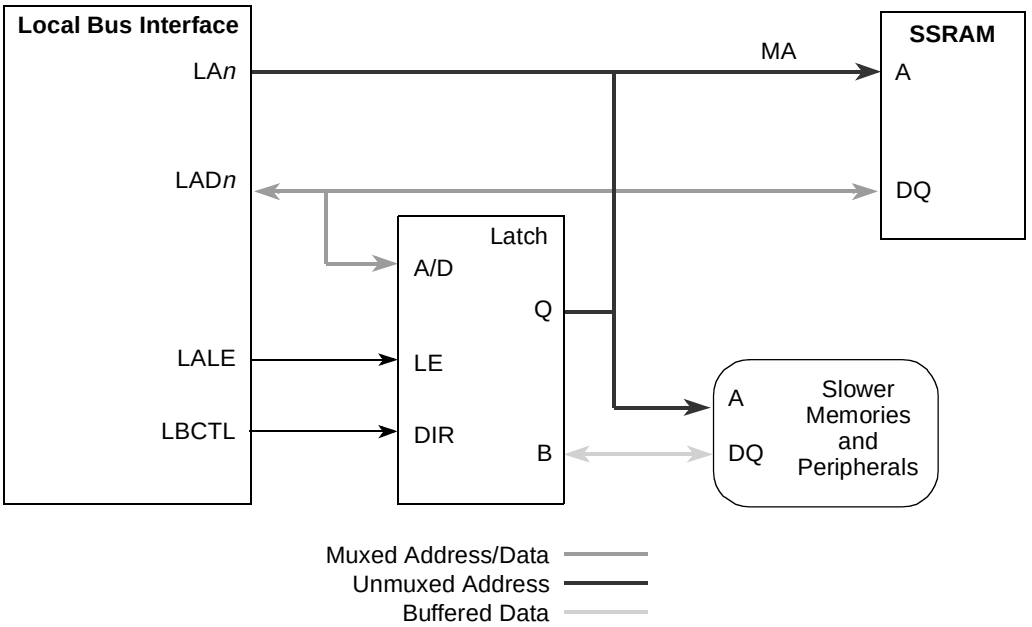


Figure 14-68. Local Bus Peripheral Hierarchy for High Bus Speeds

14.5.1.3 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.

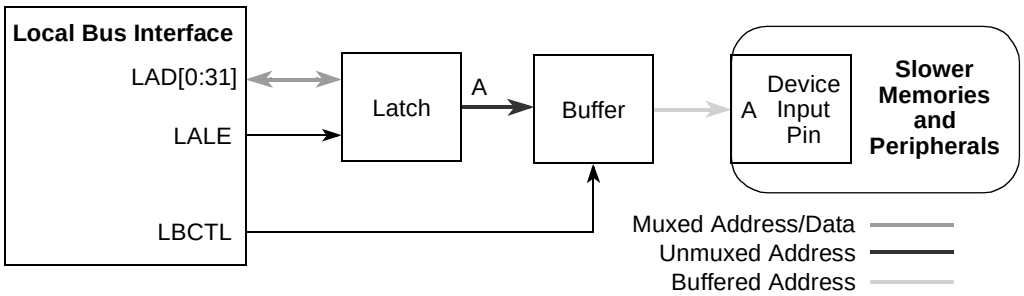


Figure 14-69. GPCM Address Timings

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 3–6 ns, so for a 133-MHz bus frequency, $\overline{\text{LCS}}$ should arrive on the order of 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered.

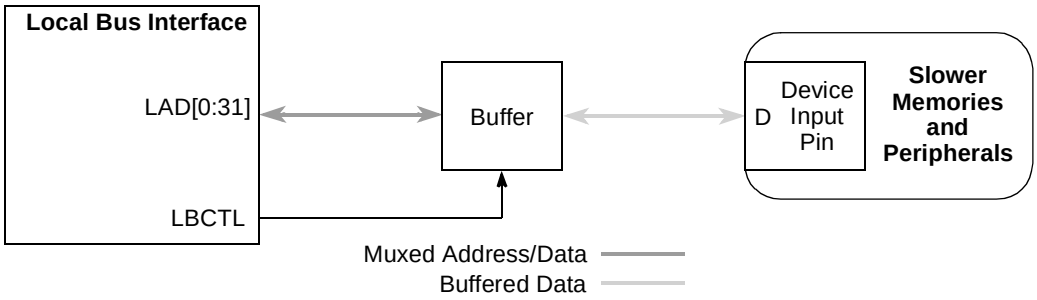


Figure 14-70. GPCM Data Timings

14.5.2 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

14.5.2.1 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.

14.5.2.2 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}(transceiver) time. The system designer has to ensure, that [t_{en}(LB) + t_{en}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

14.5.2.3 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.

14.5.2.4 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.

14.5.3 Interface to Different Port-Size Devices

The eLBC supports 8-, 16-, and 32-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 32-bit port must reside on LAD[0:31], 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. Figure 14-71 shows the device connections on the data bus.

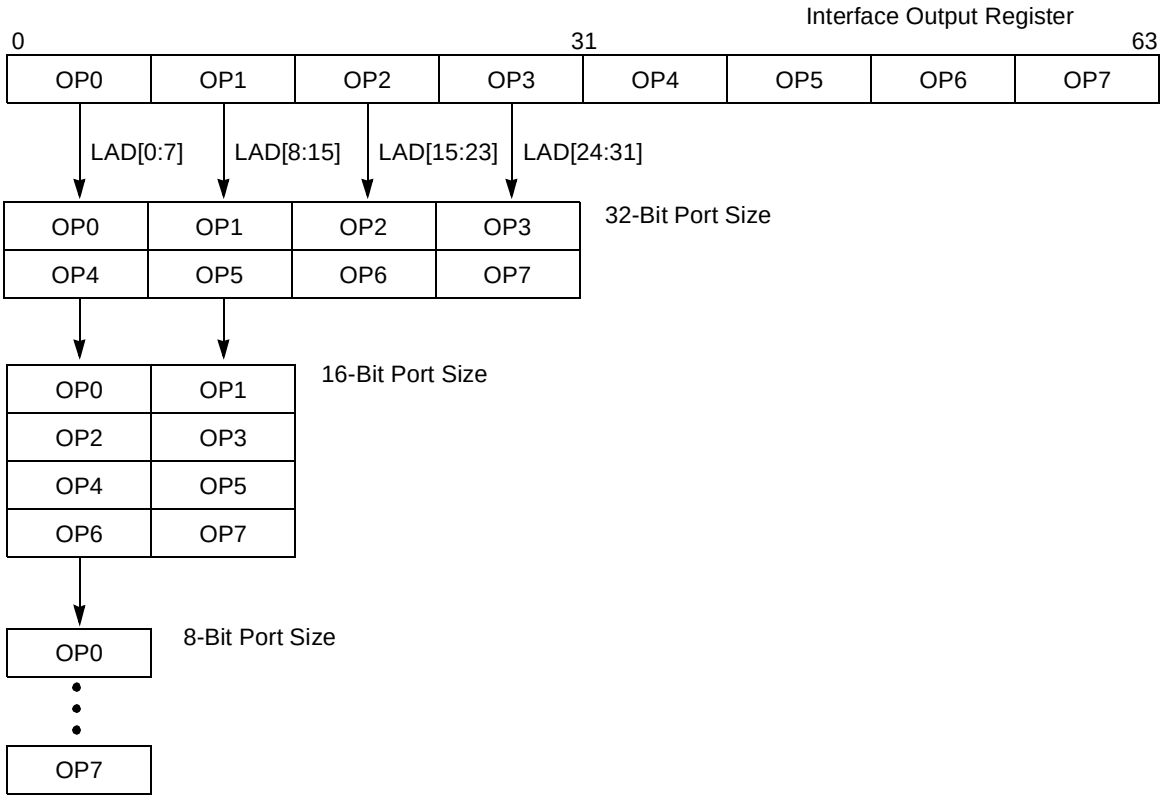


Figure 14-71. Interface to Different Port-Size Devices

Table 14-42 lists the bytes required on the data bus for read cycles.

Table 14-42. Data Bus Drive Requirements For Read Cycles

Transfer Size	Address State [1] 3 lsbs	Port Size/LAD Data Bus Assignments											
		32-Bit				16-Bit				8-Bit			
		0–7	8–15	16–23	24–31	0–7	8–15	16–23	24–31	0–7	8–15	16–23	24–31
Byte	000	OP0[2]	—[3]	—	—	OP0	—			OP0			
	001	—	OP1	—	—	—	OP1			OP1			
	010	—	—	OP2	—	OP2	—			OP2			
	011	—	—	—	OP3	—	OP3			OP3			
	100	OP4	—	—	—	OP4	—			OP4			
	101	—	OP5	—	—	—	OP5			OP5			
	110	—	—	OP6	—	OP6	—			OP6			
	111	—	—	—	OP7	—	OP7			OP7			

Table 14-42. Data Bus Drive Requirements For Read Cycles (continued)

Transfer Size	Address State [1] 3 lsbs	Port Size/LAD Data Bus Assignments											
		32-Bit				16-Bit				8-Bit			
		0–7	8–15	16–23	24–31	0–7	8–15	16–23	24–31	0–7	8–15	16–23	24–31
Half Word	000	OP0	OP1	—	—	OP0	OP1			OP0			
	001	—	OP1	OP2	—	—	OP1			OP1			
	010	—	—	OP2	OP3	OP2	OP3			OP2			
	100	OP4	OP5	—	—	OP4	OP5			OP4			
	101	—	OP5	OP6	—	—	OP5			OP5			
	110	—	—	OP6	OP7	OP6	OP7			OP6			
Word	000	OP0	OP1	OP2	OP3	OP0	OP1			OP0			
	100	OP4	OP5	OP6	OP7	OP4	OP5			OP4			

[1] Address state is the calculated address for port size.

[2] OP*n*: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a doubleword operand and OP7 is the least-significant byte.

[3] — Denotes a byte not driven during that read cycle.

14.5.4 Command Sequence Examples for NAND Flash EEPROM

In order 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 (BR*n*[MSEL] = 001), with base address, port size, ECC mode, and timing parameters configured in accordance with the device's hardware specifications.

14.5.4.1 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 Table 14-43. 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 14-43. FCM Register Settings for Soft Reset (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FCR	0xFF000000	CMD0 = 0xFF = reset command; other commands unused
FBAR	—	unused

Table 14-43. FCM Register Settings for Soft Reset (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FPAR	—	unused
FBCR	—	unused
MDR	—	unused
FIR	0x40000000	OP0 = CM0 = command 0; OP1–OP7 = NOP

14.5.4.2 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 Table 14-44. This sequence does not require use of the shared FCM buffer RAM, but reads the NAND Flash status into register MDR[AS0] for 8-bit devices. 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 14-44. FCM Register Settings for Status Read (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FCR	0x70000000	CMD0 = 0x70 = read status command; other commands unused
FBAR	—	unused
FPAR	—	unused
FBCR	—	unused
MDR	—	Status returned in AS0
FIR	0x4B000000	OP0 = CM0 = command 0; OP1 = RS = read status to MDR; OP2–OP7 = NOP

14.5.4.3 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 Table 14-45. 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 14-45. FCM Register Settings for ID Read (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FCR	0x90000000	CMD0 = 0x90 = read ID command; other commands unused
FBAR	—	unused
FPAR	—	unused

Table 14-45. FCM Register Settings for ID Read (OR*n*[PGS] = 1) (continued)

Register	Initial Contents	Description
FBCR	—	unused
MDR	0x00000000	AS0 = 0x00 = dummy address for read ID command; AS0–AS3 return with first 4 bytes of ID code
FIR	0x43BBBBB0	OP0 = CM0 = command 0; OP1 = UA = user address from MDR; OP2–OP6 = RS = read 4 bytes ID into MDR[AS3–AS0]; OP7 = NOP

14.5.4.4 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 Table 14-46. 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] = 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 14-46. FCM Register Settings for Page Read (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FCR	0x00300000	CMD0 = 0x00 = random read address entry; CMD1 = 0x30 = read page
FBAR	block index (e.g. block 0x00010ab4)	BLK locates index of 128-Kbyte block
FPAR	page offset (e.g. 0x00005000 locates page 5, buffer 1)	PI locates page index in BLK; PI mod 2 indexes FCM buffer RAM; MS = 0 and CI = 0
FBCR	0x00000000	BC = 0 to read entire 2112-byte page with ECC check
MDR	—	unused
FIR	0x4125E000	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

14.5.4.5 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 Table 14-47. 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] = 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.

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 14-47. FCM Register Settings for Block Erase (OR*n*[PGS] = 1)

Register	Initial Contents	Description
FCR	0x6070D000	CMD0 = 0x60 = block address entry; CMD1 = 0x70 = read status CMD2 = 0xD0 = erase block;
FBAR	block index (e.g. block 0x00010AB4)	BLK locates index of 128-Kbyte block
FPAR	0x00000000	PI = 0 to locate block boundary
FBCR	—	unused
MDR	—	returns with AS0 holding erase status
FIR	0x426DB000	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

14.5.4.6 NAND Flash Program Command Sequence Example

An example of configuring FCM to execute a program command to large-page NAND Flash is shown in Table 14-48. 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] = 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. 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 14-48. FCM Register Settings for Page Program (OR*n*[PGS] = 1)

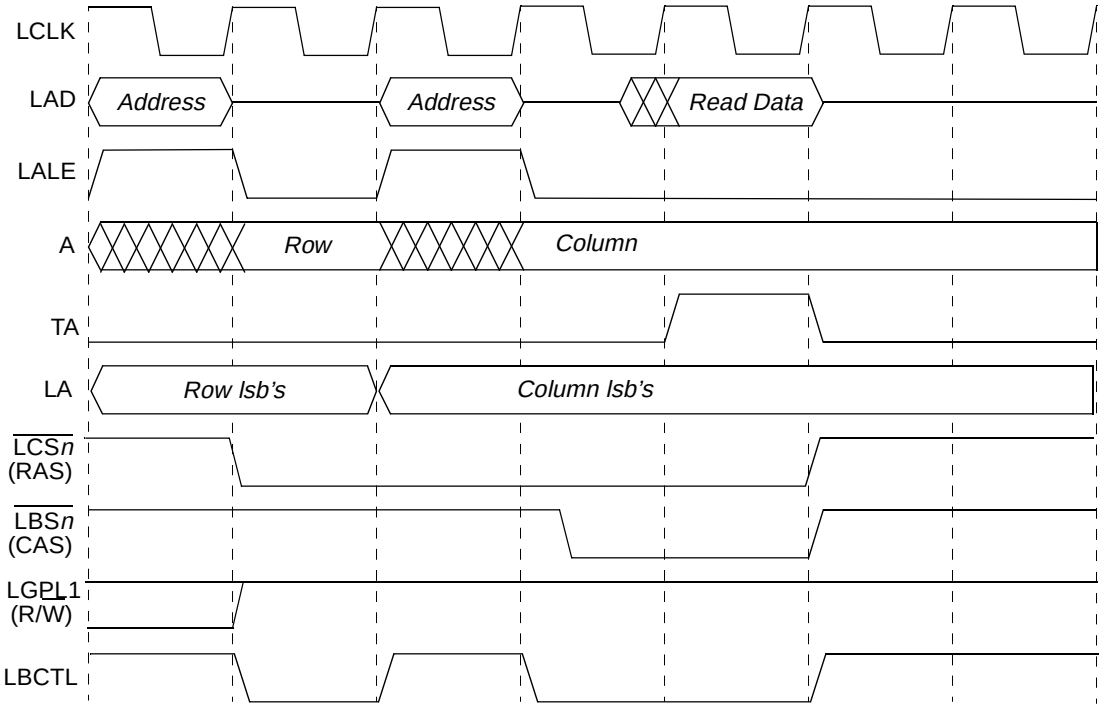
Register	Initial Contents	Description
FCR	0x80701000	CMD0 = 0x80 = page address and data entry; CMD1 = 0x70 = read status CMD2 = 0x10 = program page;
FBAR	block index (e.g. block 0x00010AB4)	BLK locates index of 128-Kbyte block

Table 14-48. FCM Register Settings for Page Program (OR*n*[PGS] = 1) (continued)

Register	Initial Contents	Description
FPAR	page offset (e.g. 0x00005000 locates page 5, buffer 1)	PI locates page index in BLK; PI mod 2 indexes FCM buffer RAM; MS = 0 and CI = 0
FBCR	0x00000000	BC = 0 to write entire 2112-Byte page with ECC generation
MDR	—	returns with AS0 holding program status
FIR	0x41286DB0	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

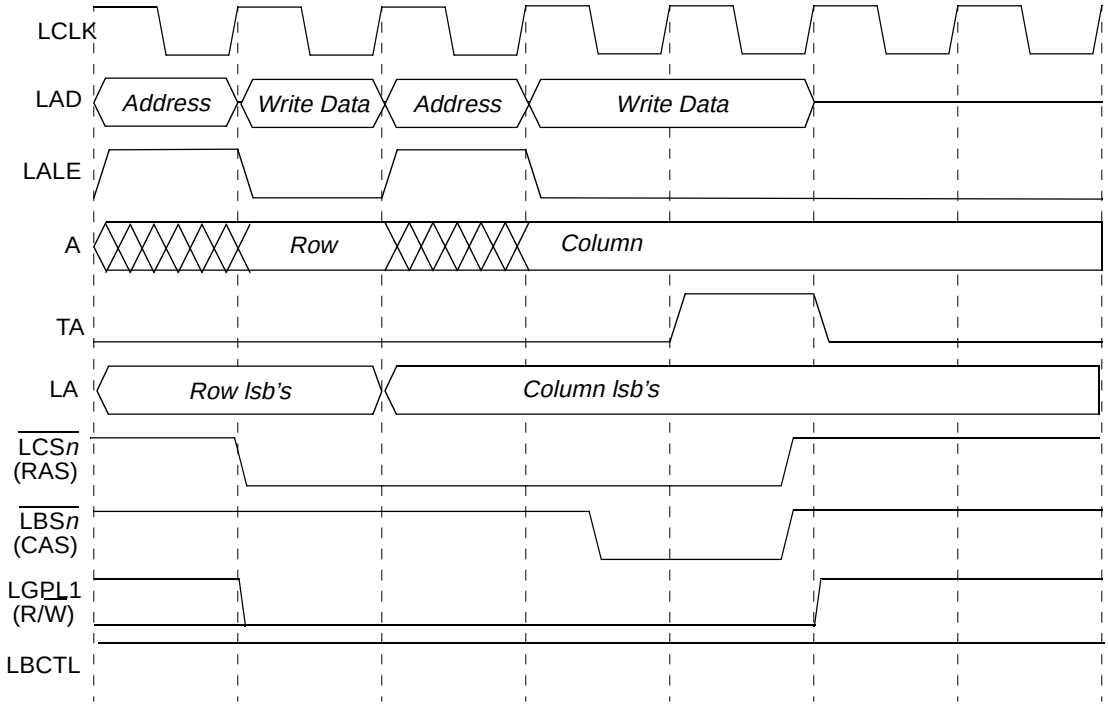
14.5.5 Interfacing to Fast-Page Mode DRAM Using UPM

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 8) or 8 (clock ratio of 16). These illustrative examples may not represent the timing necessary for any specific device used with the eLBC. Here, LGPL1 is programmed to drive R/$\overline{W}$ of the DRAM, although any LGPL*n* signal may be used for this purpose.



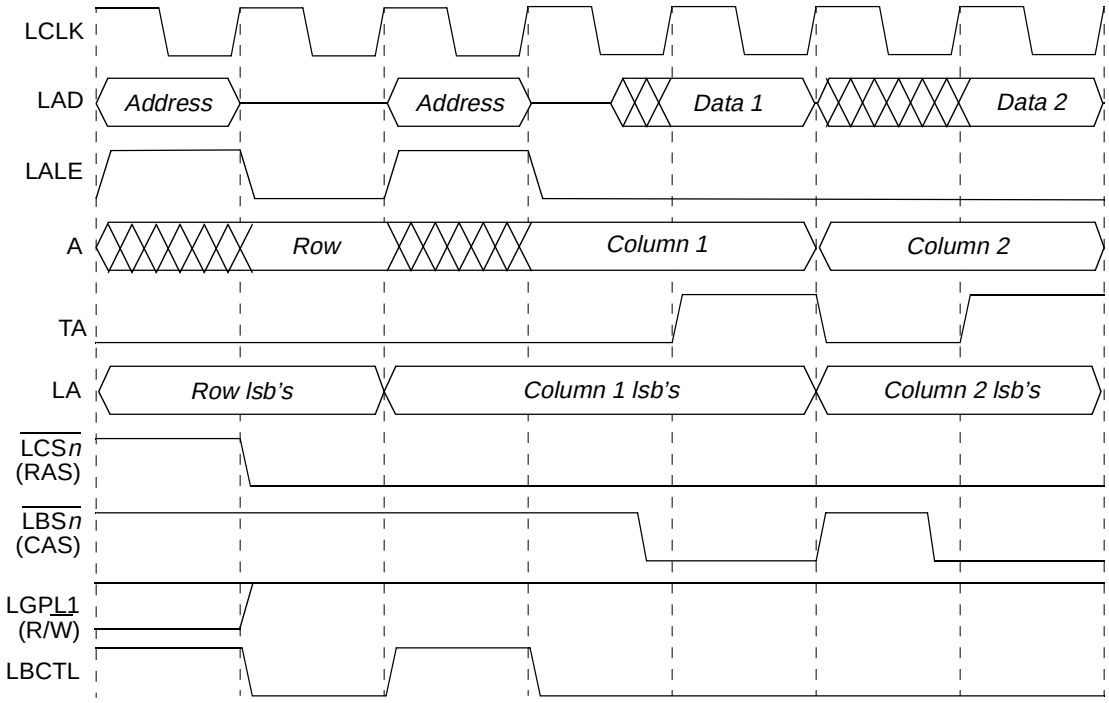
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 0 0 0	0 0 0 0	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 16 Bit 17
g4t1 g4t3					Bit 18 Bit 19
g5t1 g5t3					Bit 20 Bit 21
redo[0] redo[1]					Bit 22 Bit 23
loop exen	0 0		0 0	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	RSS	**RSS+1**	**RSS+1**	**RSS+2**	

Figure 14-72. Single-Beat Read Access to FPM DRAM



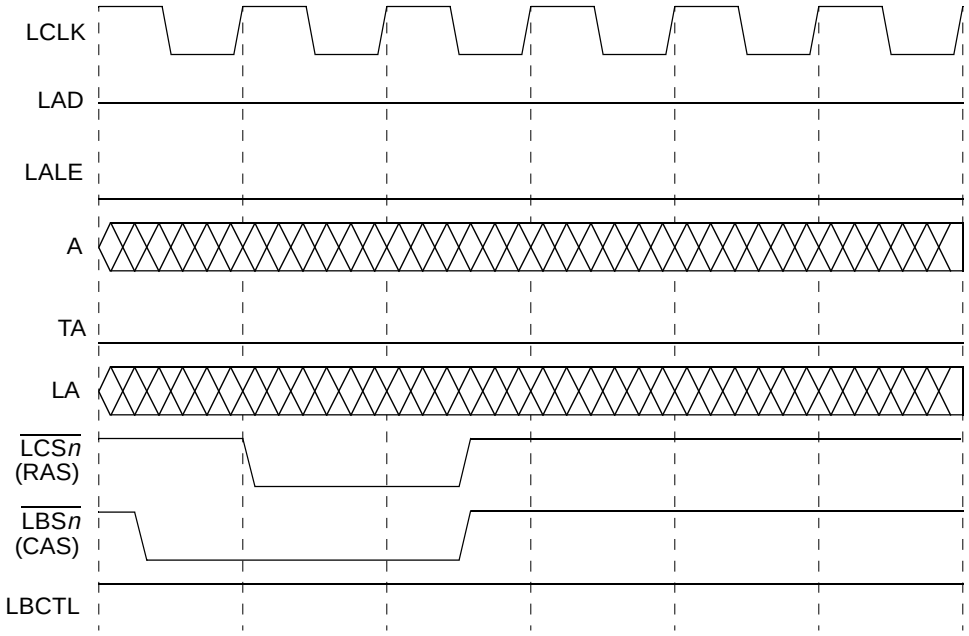
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit 16 Bit 17
g4t1 g4t3					Bit 18 Bit 19
g5t1 g5t3					Bit 20 Bit 21
redo[0] redo[1]					Bit 22 Bit 23
loop exen	0 0		0 0	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	WSS	**WSS+1**	**WSS+1**	**WSS+2**	

Figure 14-73. Single-Beat Write Access to FPM DRAM



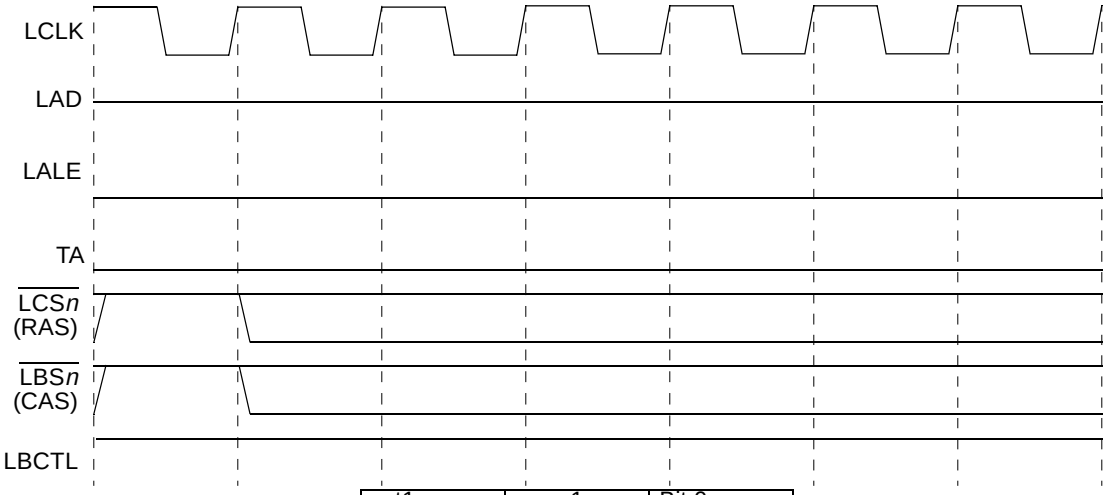
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 1 0	0 0 0 0	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1						Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	1 1	Bit 12 Bit 13
g2t1 g2t3						Bit 14 Bit 15
g3t1 g3t3						Bit 16 Bit 17
g4t1 g4t3						Bit 18 Bit 19
g5t1 g5t3						Bit 20 Bit 21
redo[0] redo[1]						Bit 22 Bit 23
loop exen	0 0		1 0	1 1	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	1 0 0 0 0 0		0 0 0 0 0 0	0 0 1 1 0 0	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	RBS		**RBS+1**	**RBS+2**	**RBS+3**	

Figure 14-74. Burst Read Access to FPM DRAM Using LOOP (Two Beats Shown)



cst1 cst2 cst3 cst4	1 1 1 1	0 0 0 0	0 0 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 0 0 0	0 0 0 0	0 0 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1				Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3				Bit 12 Bit 13
g2t1 g2t3				Bit 14 Bit 15
g3t1 g3t3				Bit 16 Bit 17
g4t1 g4t3				Bit 18 Bit 19
g5t1 g5t3				Bit 20 Bit 21
redo[0] redo[1]				Bit 22 Bit 23
loop exen	0 0	0 0	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	PTS	**PTS+1**	**PTS+2**	

Figure 14-75. Refresh Cycle (CBR) to FPM DRAM



cst1 cst2 cst3 cst4	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1		Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3		Bit 12 Bit 13
g2t1 g2t3		Bit 14 Bit 15
g3t1 g3t3		Bit 16 Bit 17
g4t1 g4t3		Bit 18 Bit 19
g5t1 g5t3		Bit 20 Bit 21
redo[0] redo[1]		Bit 22 Bit 23
loop exen	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	EXS	

Figure 14-76. Exception Cycle

14.5.6 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. Figure 14-77 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. Because ZBT

SRAMs will mostly be used by performance-critical applications, we assume here that, typically, the maximum width of the local bus of 32 bits will be used.

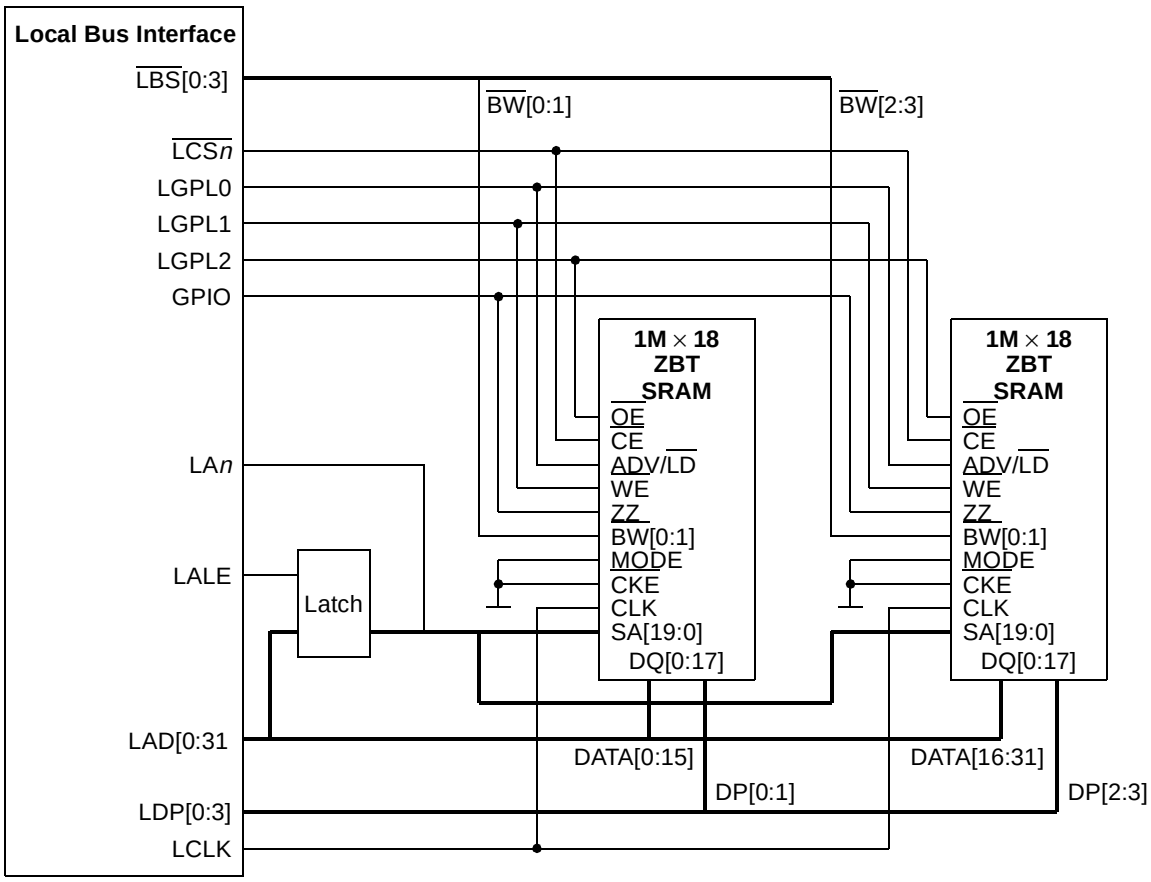


Figure 14-77. Interface to ZBT SRAM

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. $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the eLBC generates eight-beat transactions (for 32-bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the eLBC generates the new LA bits for each 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]. Therefore, two consecutive linear bursts of the ZBT SRAM with {A27, A28} = {0,0} for the first burst, and {A27, A28} = {1,0} for the second burst give the desired burst pattern.

The UPM also supports single beat accesses. Because the ZBT SRAM does not support this and always responds with a burst, the UPM pattern has to take care that data for the critical beat is provided (for write) or sampled (for read), and that the rest of the burst is ignored (by negating $\overline{\text{WE}}$). The UPM controller basically has to wait for the end of the SRAM burst to avoid bus contention with further bus activities.

Chapter 15
Enhanced Three-Speed Ethernet Controllers

15.1 Overview

The enhanced three-speed Ethernet controllers (eTSECs) of the device interface to 10 Mbps, 100 Mbps, and 1 Gbps Ethernet/IEEE 802.3 networks and devices featuring generic 8- or 16-bit FIFO ports. For Ethernet, an external PHY or SerDes device is required to complete the interface to the media. Each eTSEC supports multiple standard media-independent interfaces, of which the FIFO interface bypasses the Ethernet MAC. Multiple eTSECs are available, providing flexible options for connectivity and control access at different speeds.

The eTSEC provides the flexibility to accelerate the identification and retrieval of standard and non-standard protocols carried over Ethernet, including both IP versions 4 and 6 and TCP/UDP. CPU-intensive parsing and checksum operations can be optionally off-loaded to an eTSEC to accelerate existing TCP/IP stacks. On transmission, varying fractions of link bandwidth can be allocated to each of multiple transmit queues through a modified weighted round-robin scheduler. On receive, an arbitrary set of queue selection rules can be programmed into each eTSEC to implement flexible quality of service or firewall strategies based on high-level protocol identification. Without enabling these advanced features, each eTSEC emulates a PowerQUICC III TSEC, allowing existing driver software to be re-used with minimal change. Each eTSEC is organized as shown in Figure 15-1.

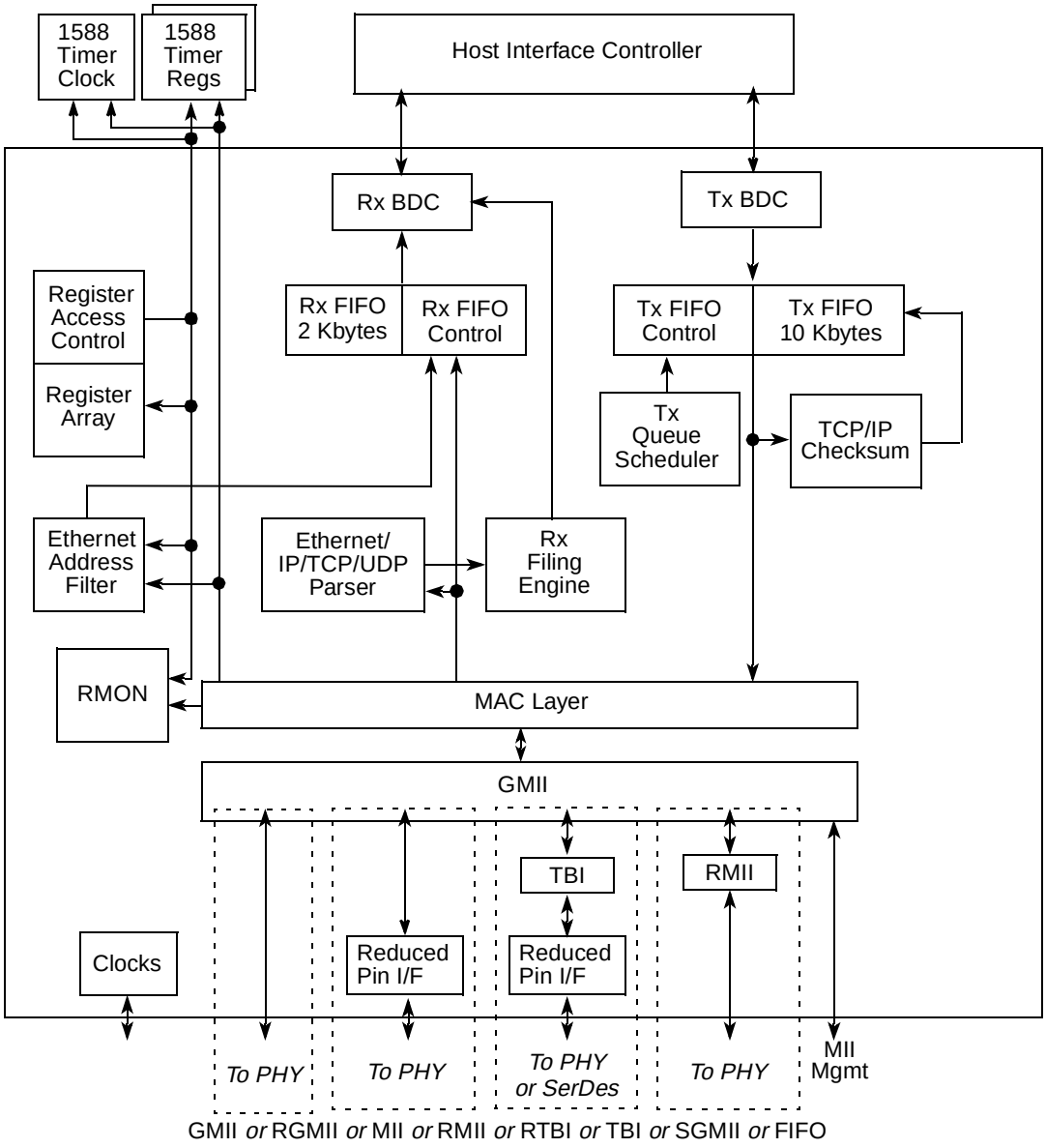


Figure 15-1. eTSEC Block Diagram

15.2 Features

The eTSECs of the device include the following distinctive features:

- IEEE 802.3, 802.3u, 820.3x, 802.3z, 802.3ac, 802.3ab compatible
- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE 802.3 GMII
  - 1000 Mbps full-duplex IEEE 802.3 GMII
  - 10/100 Mbps IEEE 802.3 MII and RMII

  - 10/100 Mbps RGMII
  - 1000 Mbps full-duplex RGMII and RTBI
  - 10/100 Mbps SGMII
  - 1000 Mbps full-duplex SGMII
  - 1000 Mbps IEEE 802.3z TBI
  - Single-clock TBI
- Support for four full-duplex FIFO interface modes
  - Two 8-bit modes—GMII style and encoded packet
  - Two 16-bit modes—GMII style and encoded packet
  - Inter-packet and intra-packet flow control
  - Optional CRC-32 generation and checking
  - Minimal glue logic required to support POS PHY Level 3 conversion
  - TCP/IP off-load and QoS features available in all FIFO modes, at up to OC-48 rates
- TCP/IP off-load
  - IP v4 and IP v6 header recognition on receive
  - IP v4 header checksum verification and generation
  - TCP and UDP checksum verification and generation
  - Per-packet configurable off-load
  - Recognition of VLAN, stacked-VLAN, 802.2, PPPoE session, MPLS stacks, and ESP/AH IP-Security headers
- Quality of service (QoS) support
  - Transmission from up to eight queues
    - Priority-based queue selection
    - Modified weighted round-robin queue selection with fair bandwidth allocation
  - Reception to up to eight physical queues
    - 64 virtual receive queues overlaid on 8 physical buffer descriptor rings
    - Table-oriented queue filing strategy based on 16 header fields or flags
    - Frame rejection support for filtering applications
    - Filing based on Ethernet, IP, and TCP/UDP properties, including VLAN fields, Ether-type, IP protocol type, IP TOS or differentiated services, IP source and destination addresses, TCP/UDP port numbers
- Interrupt coalescing
  - Packet-count-based thresholds for both receive and transmit
  - Timer-based thresholds
- Full- and half-duplex Ethernet support (1000 Mbps supports only full duplex):
  - IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)

  - Programmable maximum frame length supports jumbo frames (up to 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) tags and priority
  - VLAN insertion and deletion
    - Per-frame VLAN control word or default VLAN for each eTSEC
    - Extracted VLAN control word passed to software separately
    - Programmable VLAN tag to support metropolitan bridging
  - Retransmission following a collision
  - Support for CRC generation and verification of inbound/outbound packets
  - Programmable Ethernet preamble insertion and extraction of up to 7 bytes
- MAC address recognition:
  - Exact match on primary and virtual 48-bit unicast addresses
    - VRRP and HSRP support for seamless router fail-over
    - In addition to primary station address, up to fifteen additional exact-match MAC addresses supported
  - Broadcast address (accept/reject)
  - Hash table match on up to 256 unicast/multicast or 512 multicast-only addresses
  - Promiscuous mode
- Remote network monitoring (RMON) statistics support
  - 32-bit byte counters
  - Carry/Overflow of counter interrupts
- Backward compatibility with MPC8540E/MPC8560E (PowerQUICC III) TSEC
  - PowerQUICC III buffer descriptor (BD) format and rings supported
  - Common register memory map, with specific exceptions:
    - Out-of-sequence transmit BD not supported
    - Internal DMA BD pointers and data counts not visible
    - MINFLR register not supported
  - Reset state of eTSEC defaults to common PowerQUICC III TSEC subset
  - TSEC_ID register permits TSEC versus enhanced TSEC differentiation
- Hardware assist for 1588 compliant timestamping
  - Per packet timestamp tag for Receive
  - Programmable timestamp capture for Transmit
  - Recognition of PTP packet
  - Periodic Pulse Generation
  - Self-correcting precision timer with nano-second resolution
  - Phase aligned adjustable (divide by N) clock output
  - Two 64-bit alarm (future time) registers for future time comparison

15.3 Modes of Operation

The eTSEC's primary operational modes are the following:

- Ethernet and FIFO operation

  The ECNTRL register's FIFO mode enable bit (ECNTRL[FIFM]) allows bypass of the Ethernet MAC and enables I/O through the FIFO interface sharing the normal GMII signals. Each eTSEC supports an 8-bit FIFO interface independently. Pairs of GMII ports can be combined to create a 16-bit, full-duplex interface. If configured in FIFO mode, the FIFOCFG register determines operation. In FIFO mode data is transferred synchronously with respect to the external data clock. See the device hardware specifications document for maximum supported frequencies.

- Full- and half-duplex operation

  This is determined by the MACCFG2 register's full-duplex bit (MACCFG2[Full Duplex]). Full-duplex mode is intended for use on point-to-point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

  If configured in half-duplex mode (10- and 100-Mbps operation; MACCFG2[Full Duplex] is cleared), the MAC complies with the IEEE CSMA/CD access method.

  If configured in full-duplex mode (10/100/1000 Mbps operation; MACCFG2[Full Duplex] is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.

- 10- and 100-Mbps MII interface operation

  The MAC–PHY interface operates in MII mode by setting MACCFG2[I/F Mode] = 01. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK and TSEC*n*_RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed, or it may be controlled by software using the serial management interface (MDC/MDIO signals) to the transceiver.

  Clause 22.2.4 of the IEEE 802.3 specification describes the MII management interface.

- 10- and 100-Mbps RMII interface operation

  The RMII is the reduced media-independent interface defined by the RMII Consortium (March 1998) for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK signal, which is driven by the transceiver.

- 1000 Mbps GMII and TBI interface operation

  The MAC–PHY interface operates in GMII mode by setting MACCFG2[I/F Mode] = 10. The GMII is the gigabit media-independent interface defined by the 802.3 standard for 1000-Mbps operation.

  Independently, the MAC-PHY interface can also operate in TBI mode. Note that either the TBI or GMII interface is chosen, not both at the same time. TBI is the 10-bit interface which contains PCS functions (10-bit encoding/decoding) as defined by the 802.3 standard.

  In reduced-pin count mode (RGMII or RTBI), the MAC remains configured in GMII or TBI but the eTSEC muxes and decodes the input signals and provides the MAC with the expected interface.

  eTSEC provides the TSEC*n*_GTX_CLK to the PHY in either GMII or TBI mode of operation.

- MAC address recognition options

  The options supported are promiscuous, broadcast, exact unicast address match, exact unicast virtual address match to support router redundancy, and multicast hash match. For detailed descriptions refer to Section 15.6.3.7, “Frame Recognition.”

  eTSEC supports automatic LAN-initiated wake-up during power management via the AMD Magic Packet™ protocol, as described in Section 15.6.3.8, “Magic Packet Mode.”
- Receive frame parsing options

  Frame parsing options are to disable parsing (no TCP/IP off-load), IP header parsing, and TCP or UDP parsing. Parsing must be enabled to make use of receive queue filing algorithms. The options are detailed in Section 15.6.4, “TCP/IP Off-Load.”
- Receive queue selection options

  Received frames are by default sent to a single buffer descriptor ring. If multiple receive queues are enabled, a receive queue filer can be programmed with selection criteria to differentiate received frames and file them to different buffer descriptor rings. See Section 15.6.5, “Quality of Service (QoS) Provision,” for detailed descriptions.
- TCP/IP transmit options

  Frames for transmission may be sent as-is, with IP header processing, or TCP header processing. The transmit buffer descriptors, described in Section 15.6.8.2, “Transmit Data Buffer Descriptors (TxBD),” enable these options and operate with parameters prepended to frame buffers, as described in Section 15.6.4, “TCP/IP Off-Load.”
- Transmit queue selection options

  The options supported are single transmit queue, priority-based queue selection, and modified weighted round-robin queueing. These options are described further in Section 15.5.3.2.1, “Transmit Control Register (TCTRL).”
- RMON support

  Standard Ethernet interface management information base (MIBs) can be generated via the RMON MIB counters.
- Internal loop back supported for all interfaces except when configured for half-duplex operation

  Internal loop back mode is selected via the loop back bit in the MACCFG1 register. See Section 15.7.1, “Interface Mode Configuration,” for details.

15.4 External Signals Description

This section defines the eTSEC interface signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention. (That is, TxD[7:0] means 0 is the lsb.) Note that except for external physical interfaces the buses and registers follow a big-endian format, where 0 denotes the msb.

Each eTSEC network interface supports multiple options:

- The MII option requires 18 I/O signals (including the MDIO and MDC MII management interface) and supports both a data and a management interface to the PHY (transceiver) device. The MII option supports both 10- and 100-Mbps Ethernet rates.

- The GMII option is a superset of the MII signals and supports a 1000-Mbps Ethernet rate.
- The TBI interface shares signals with the GMII interface signals.
- The RGMII, RTBI, and RMII options are reduced-pin implementations of the GMII, TBI, and MII interfaces, respectively.
- SGMII interfaces are offered via the SerDes interface signals.
- 1588 timer signals
- Finally, the FIFO interfaces share the GMII signals—8 bits of data plus 3 bits of control signals.

Table 15-1 lists the network interface signals.

Table 15-1. eTSEC*n* Network Interface Signal Properties

Signal Name	Function	Reset State
TSEC*n*_COL	MII—collision, input FIFO—transmit flow control, input	—
TSEC*n*_CRS	MII—carrier sense, input TBI—signal detect, input FIFO—receive flow control, output	—
TSEC*n*_GTX_CLK	RTBI, RGMII—inverted transmit clock feedback, output TBI, FIFO—continuous transmit clock feedback, output GMII, MII, RMII—transmit clock feedback when transmission is enabled, zero otherwise, output	0
EC_GTX_CLK125	Oscillator source for GMII, TBI, RGMII, RTBI transmit clock, input, shared by all eTSECs	—
EC1_MDC	Management clock, output, associated with eTSEC controller 1 memory-mapped registers.	0
EC1_MDIO	Management data, bidirectional, associated with eTSEC controller 1 memory-mapped registers.	Hi-Z (input)
EC3_MDC	Management clock, output, associated with eTSEC controller 3 memory-mapped registers.	0
EC3_MDIO	Management data, bidirectional, associated with eTSEC controller 3 memory-mapped registers.	Hi-Z (input)
TSEC*n*_RX_CLK	GMII, MII, RGMII—receive clock, input TBI—PMA receive clock 0, input FIFO—receive clock, input	—
TSEC*n*_RX_DV	GMII, MII—receive data valid, input TBI—receive code group (RCG) bit 8, input RGMII (RX_CLK rising)—receive data valid, input RGMII (RX_CLK falling)—receive error, input RTBI (RX_CLK rising)—receive code group (RCG) bit 4, input RTBI (RX_CLK falling)—receive code group (RCG) bit 9, input RMII—CRS_DV carrier sense/data valid, input FIFO—receive data valid or receive control bit, input	—
TSEC*n*_RXD[7:4]	GMII—receive data bits 7:4 input TBI—RCG bits 7:4, input FIFO—receive data bits 7:4 input MII, RGMII, RTBI, RMII—unused	—

Table 15-1. eTSEC*n* Network Interface Signal Properties (continued)

Signal Name	Function	Reset State
TSEC*n*_RXD[3:0]	GMII, MII—Receive data bits 3:0, input TBI—RGC bits 3:0, input RGMII (RX_CLK rising) —Receive data bits 3:0, input RGMII (RX_CLK falling)—Receive data bits 7:4, input RTBI (RX_CLK rising)—RCG bits 3:0, input RTBI (RX_CLK falling)—RCG bits 8:5, input RMII—RXD[1:0] receive data bits, input RMII—RXD[3:2] are unused FIFO—Receive data bits 3:0, input	—
TSEC*n*_RX_ER	GMII, MII, RMII—Receive error, input TBI—RGC bit 9, input FIFO—Receive error or receive frame control bit, input RGMII, RTBI—Unused	—
TSEC*n*_TX_CLK	MII—transmit clock, input TBI—PMA receive clock 1, input RMII—reference transmit and receive clock, input FIFO—transmit clock, input RGMII, RTBI—unused	—
TSEC*n*_TXD[7:4]	GMII—transmit data bit 7:4, output TBI—transmit code group (TCG) bit 7:4, output FIFO—transmit data bit 7:4, output MII, RGMII, RTBI, RMII—unused, output driven zero	0000
TSEC*n*_TXD[3:0]	GMII, MII—Transmit data bits 3:0, output TBI—TCG bits 3:0, output RGMII (TX_CLK rising)—Transmit data bits 3:0, output RGMII (TX_CLK falling)—Transmit data bits 7:4, output RTBI (TX_CLK rising)—TCG bits 3:0, output RTBI (TX_CLK falling)—TCG bits 8:5, output RMII—TXD[1:0] transmit data bits, output RMII—TXD[3:2] unused, output driven zero FIFO—Transmit data bits 3:0, output	0000
TSEC*n*_TX_ER	GMII, MII—transmit error, output RGMII, RTBI, RMII—unused, output driven zero TBI—TCG bit 9, output FIFO—transmit error or transmit frame control bit, output	0
TSEC*n*_TX_EN	GMII, MII, RMII—Transmit data valid, output TBI—TCG bit 8, output RGMII (TX_CLK rising)—Transmit data enabled, output RGMII (TX_CLK falling)—Transmit error, output RTBI (TX_CLK rising)—TCG bit 4, output RTBI (TX_CLK falling)—TCG bit 9, output FIFO—Transmit data valid or transmit control bit, output	0
TSEC_1588_CLK	1588—Clock input	—
TSEC_1588_GCLK	1588—Clock out	0
TSEC_1588_TRIG_IN	1588—Trigger in 1	—
TSEC_1588_TRIG_IN2	1588—Trigger in 2	—

Table 15-1. eTSEC*n* Network Interface Signal Properties (continued)

Signal Name	Function	Reset State
TSEC_1588_ PULSE_OUT1	1588—Pulse out 1	0
TSEC_1588_ PULSE_OUT2	1588—Pulse out 2	0
TSEC_1588_ TRIG_OUT	1588—Timer alarm 1	0
TSEC_1588_ TRIG_OUT2	1588—Timer alarm 2	0
SD2_TX[*n-1*], $\overline{\text{SD2\_TX[}n\text{-}1\text{]}}$	SGMII transmit data (and complement)	—
SD2_RX[*n-1*], $\overline{\text{SD2\_RX[}n\text{-}1\text{]}}$	SGMII receive data (and complement)	—
SD2_REF_CLK, $\overline{\text{SD2\_REF\_CLK}}$	SGMII SerDes2 PLL reference clock (and complement)	—

15.4.1 Detailed Signal Descriptions

Below is a description of the eTSEC interface signals. For RGMII mode details please refer to the Hewlett-Packard reduced gigabit media-independent interface (RGMII) specification version 1.2a, dated 9/22/2000. RMII mode details follow the RMII Consortium Specification, dated 3/20/1998. All other modes follow IEEE 802.3, 2000 Edition. Input signals not used are internally disabled. Except for TSEC*n*_GTX_CLK, output signals not used are driven low.

Table 15-2. eTSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TSEC*n*_COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.	
		State Meaning	Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists. In FIFO mode this signal is used to effect flow control on the transmitter. This signal is not used in the following modes: • RMII • GMII • TBI • RTBI • RGMII
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_CRS	I	Carrier sense input. In TBI and RTBI modes, this signal is used as SDET (signal detect).In TBI mode SDET must be tied high externally on the board. In RTBI mode SDET is tied high internally. This signal is not used in the following modes: • RMII • GMII • RGMII	
		State Meaning	Asserted/Negated—In MII mode, TSEC*n*_CRS is asserted while the transmit or receive medium is not idle. In the event of a collision, TSEC*n*_CRS must remain asserted for the duration of the collision.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
	O	Receiver flow control signal in FIFO mode. This signal is not used in the eTSEC Ethernet modes.	
		State Meaning	Asserted/Negated—TSEC*n*_CRS is asserted while the FIFO receiver is unprepared to accept additional receive data.
		Timing	Asserted/Negated—This signal transitions synchronously with TSEC*n*_RX_CLK.
TSEC*n*_GTX_CLK	O	Gigabit transmit clock. This signal is an output from the eTSEC into the PHY. TSEC*n*_GTX_CLK is a 125-MHz clock that provides a timing reference for TX_EN, TXD, and TX_ER in the following modes: • GMII • TBI • RTBI In RGMII mode, TSEC*n*_GTX_CLK becomes the transmit clock and provides timing reference during 1000Base-T (125 MHz), 100Base-T (25 MHz) and 10Base-T (2.5 MHz) transmissions. This signal feeds back the uninverted transmit clock in MII or FIFO modes, but feeds back an inverted transmit clock in RTBI or RGMII modes. This signal is driven low unless transmission is enabled, or the eTSEC is in TBI or FIFO mode.	

Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
EC_GTX_CLK125	I	Gigabit transmit 125-MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY. EC_GTX_CLK125 is a 125-MHz input into the eTSEC and is used to generate all 125-MHz related signals and clocks in the following modes: • GMII • TBI • RTBI • RGMII This input is not used in these modes: • FIFO • RMII • SGMII • MII
EC1_MDC	O	Management data clock. This signal is a clock (typically 2.5 MHz) supplied by the MAC (IEEE set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed.) The frequency can be modified by writing to MIIMCFG[28:31] of the eTSEC1 controller.
EC1_MDIO	I/O	Management data input/output. **State Meaning**: Asserted/Negated—EC1_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles. Addressed using eTSEC1 memory-mapped registers. **Timing**: Asserted/Negated—This signal is required to be synchronous with the EC1_MDC signal.
EC3_MDC	O	Management data clock. EC3_MDC is a clock (typically 2.5 MHz) supplied by the MAC. (IEEE set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed.) The frequency can be modified by writing to MIIMCFG[28:31] of the eTSEC3 controller.
EC3_MDIO	I/O	Management data input/output, BIDI **State Meaning**: Asserted/Negated—EC3_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles. Addressed using eTSEC3 memory-mapped registers. **Timing**: Asserted/Negated—This signal is required to be synchronous with the EC3_MDC signal.
TSEC*n*_RX_CLK	I	Receive clock. In GMII, MII, or RGMII mode, the receive clock TSEC*n*_RX_CLK is a continuous clock (2.5, 25, or 125 MHz) that provides a timing reference for TSEC*n*_RX_DV, TSEC*n*_RXD, and TSEC*n*_RX_ER. In TBI mode, TSEC*n*_RX_CLK is the input for a 62.5 MHz PMA receive clock, 0 split phase with PMA_RX_CLK1 and is supplied by the SerDes. In RTBI mode it is a 125-MHz receive clock. In RMII mode this clock is not used for the receive clock, as RMII uses a shared reference clock. However, note that due to pin limitations on the MPC8572E, eTSEC4 must be configured differently from the other eTSECs in RMII mode. For eTSEC1–3, the RMII reference clock is obtained from TSEC*n*_TX_CLK. For eTSEC4, however, this clock comes from TSEC4_RX_CLK (TSEC3_COL). In FIFO mode the receive clock is a continuous clock. See the device hardware specifications document for maximum supported frequencies.

Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_RX_DV	I	Receive data valid. In GMII or MII mode, if TSEC*n*_RX_DV is asserted, the PHY is indicating that valid data is present on the GMII and MII interfaces. In RGMII mode, TSEC*n*_RX_DV becomes RX_CTL. The RX_DV and RX_ERR are received on this signal on the rising and falling edges of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 8. Together, with RCG[9] and RCG[7:0], they represents the 10-bit encoded symbol of GMII receive signals. In RTBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 4 and 9. On the positive edge of the TSEC*n*_RX_CLK, RCG[4] and RCG[3:0] represent the first half of the 10-bit encoded symbol. On the negative edge of the TSEC*n*_RX_CLK, RCG[9] and RCG[8:5] represent the second half of the 10-bit encoded symbol. In RMII mode the PHY asserts TSEC*n*_RX_DV (CRS_DV) when the receive medium is non-idle. This signal asserts asynchronously with respect to the RMII reference clock, but negates synchronously to indicate loss of carrier. In FIFO mode TSEC*n*_RX_DV is used to indicate valid data (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols).
TSEC*n*_RXD[7:0]	I	Receive data in. In GMII mode, TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0], represent one complete octet of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. In TBI mode, TSEC*n*_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals. In GMII or MII mode, TSEC*n*_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While TSEC*n*_RX_DV is not asserted, TSEC*n*_RXD has no meaning. In RGMII mode, data bits 3:0 are received on the rising edge of TSEC*n*_RX_CLK. In RTBI mode, TSEC*n*_RXD[3:0] represents RCG[3:0] on the rising edge of TSEC*n*_RX_CLK and RCG[8:5] are received on the falling edge of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals. In RMII mode TSEC*n*_RXD[1:0] represents RXD[1:0], which is considered valid when TSEC*n*_RX_DV (CRS_DV) is asserted, or invalid otherwise. In FIFO mode TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0] represent one complete octet of data to be received from the external FIFO device. For 16-bit FIFO operation the TSEC*n*_RXD[7:0] signals of a pair of eTSECs are combined to represent two octets of data.
TSEC*n*_RX_ER	I	Receive error **State Meaning**: Asserted/Negated—In GMII, MII, or RMII mode, if TSEC*n*_RX_ER and TSEC*n*_RX_DV are asserted, the PHY has detected an error in the current frame. In TBI mode, this signal represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals. In FIFO mode, this signal represents either receive data error (GMII-style protocols) or forms part of the receive control flags (encoded packet protocols). This signal is not used in the RTBI or RGMII modes.

Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_TX_CLK	I	Transmit clock in. In MII mode, TSEC*n*_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC*n*_TX_EN, TSEC*n*_TXD, and TSEC*n*_TX_ER signals. In GMII mode, this signal provides the 2.5 or 25 MHz timing reference during 10Base-T and 100Base-T and comes from the PHY. In 1000Base-T this clock is not used and TSEC*n*_GTX_CLK (125 MHz) becomes the timing reference. The TSEC*n*_GTX_CLK is generated in the eTSEC and provided to the PHY and the MAC. The TSEC*n*_TX_CLK is generated in the PHY and provided to the MAC. In TBI mode, this signal is PMA receive clock 1 at 62.5 MHz, split phase with PMA_RX_CLK0, and is supplied by the SerDes. In RMII mode this signal is the reference clock shared between transmit and receive, and is supplied by the PHY. In FIFO mode the transmit clock is a continuous clock. See the device hardware specifications document for maximum supported frequencies. This signal is not used in the eTSEC RTBI or RGMII modes.
TSEC*n*_TXD[7:4]	O	Transmit data out. In GMII mode, TSEC*n*_TXD[7:0] represents one complete octet of data to be sent from the MAC to the PHY when TSEC_TX_DV is asserted and has no meaning while TSEC*n*_TX_EN is negated. In TBI mode, TSEC*n*_TXD[7:4] represents transmit code group (TCG) bits 7:4. Together, with TCG[9:8] and TCG[3:0], they represent the 10-bit encoded symbol. In GMII or MII mode, TSEC*n*_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSEC*n*_TX_EN is asserted and have no meaning while TSEC*n*_TX_EN is negated. In RGMII or RTBI mode, data bits 3:0 are transmitted on the rising edge of TSEC*n*_TX_CLK, and data bits 7:4 are transmitted on the falling edge of TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol. In RMII mode, TSEC*n*_TXD[1:0] represents TXD[1:0], which is valid data sent to the PHY when TSEC*n*_TX_EN is asserted, or undefined otherwise. In FIFO mode, TSEC*n*_TXD[7:4] with TSEC*n*_TXD[3:0] represent one complete octet of data to be received from the external FIFO device. For 16-bit FIFO operation the TSEC*n*_TXD[7:0] signals of a pair of eTSECs are combined to represent two octets of data. Note that some of these signals are also used during reset to configure the eTSEC interface mode.
TSEC*n*_TX_EN	O	Transmit data valid. In GMII, MII, or RMII mode, if TSEC*n*_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII's or the MII's TSEC*n*_TXD signals. In RGMII mode, TSEC*n*_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this signal on rising and falling edges of the TSEC*n*_GTX_CLK, respectively. In TBI mode, TSEC*n*_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol. In RTBI mode, TSEC*n*_TX_EN represents TCG[4] on the rising edge and TCG[9] on the falling edge of TSEC*n*_GTX_CLK, respectively. Together with TCG[3:0] and TCG[8:5], they represent the 10-bit encoded symbol. In FIFO mode TSEC*n*_TX_EN is used to indicate valid data (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols).

Table 15-2. eTSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_TX_ER	O	Transmit error. In GMII or MII mode, assertion of TSEC*n*_TX_ER for one or more clock cycles while TSEC*n*_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC*n*_TX_ER has no effect while operating at 10 Mbps or while TSEC*n*_TX_EN is negated. This signal transitions synchronously with respect to TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TX_ER represents TCG[9]. Together, with TCG[8:0], they represents the 10-bit encoded symbol. In FIFO mode TSEC*n*_TX_ER represents either transmit data error (GMII-style protocols) or forms part of the transmit control flags (encoded packet protocols). This signal is not used in the eTSEC RMII, RTBI, or RGMII modes and is driven low.
TSEC_1588_CLK	I	1588 clock in. External high precision timer reference clock input (chip external input pin).
TSEC_1588_GCLK	O	1588 clock out. Phase aligned timer clock divider output (chip external output pin).
TSEC_1588_ TRIG_IN	I	1588 trigger in 1. External timer trigger input 1.This is an asynchronous general purpose input (chip external input pin).
TSEC_1588_ TRIG_IN2	i	1588 trigger in 2. External timer trigger input 2.This is an asynchronous general purpose input (chip external input pin).
TSEC_1588_ PULSE_OUT1	O	1588 pulse out 1. Timer pulse per period 1. It is phase aligned with 1588 timer clock (chip external output pin)
TSEC_1588_ PULSE_OUT2	O	1588 pulse out 2. Timer pulse per period 2. It is phase aligned with 1588 timer clock (chip external output pin)
TSEC_1588_ TRIG_OUT	O	1588 timer alarm 1. Timer current time is equal to or greater than alarm time comparator register. User reprograms the TMR_ALARMn_H/L register to deactivate this output (chip external output pin)
TSEC_1588_ TRIG_OUT2	O	1588 timer alarm 2. Timer current time is equal to or greater than alarm time comparator register. User reprograms the TMR_ALARMn_H/L register to deactivate this output (chip external output pin)
SD2_TX*n*, $\overline{\text{SD2\_TX}n}$	O	SGMII transmit data (and complement) When in SGMII interface mode: • eTSEC1 utilizes SD2_TX[0] and $\overline{\text{SD2\_TX[0]}}$ • eTSEC2 utilizes SD2_TX[1] and $\overline{\text{SD2\_TX[1]}}$ • eTSEC3 utilizes SD2_TX[2] and $\overline{\text{SD2\_TX[2]}}$ • eTSEC4 utilizes SD2_TX[3] and $\overline{\text{SD2\_TX[3]}}$
SD2_RX*n*, $\overline{\text{SD2\_RX}n}$	I	SGMII receive data (and complement) When in SGMII interface mode: • eTSEC1 utilizes SD2_RX[0] and $\overline{\text{SD2\_RX[0]}}$ • eTSEC2 utilizes SD2_RX[1] and $\overline{\text{SD2\_RX[1]}}$ • eTSEC3 utilizes SD2_RX[2] and $\overline{\text{SD2\_RX[2]}}$ • eTSEC4 utilizes SD2_RX[3] and $\overline{\text{SD2\_RX[3]}}$
SD2_REF_CLK, $\overline{\text{SD2\_REF\_CLK}}$	I	SGMII SerDes2 PLL reference clock (and complement)

15.5 Memory Map/Register Definition

The eTSECs use a software model that is a superset of the PowerQUICC III TSEC functionality and is similar to that employed by the Fast Ethernet function supported on the Freescale MPC8260 CPM FCC and in the FEC of the MPC860T.

The eTSEC device is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs are used for mode control, interrupts, and to extract status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always store 0, as writing 1 to reserved bits may have unintended side-effects. Reads from unmapped register addresses return zero. Unless otherwise specified, the read value of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in Section 15.6.8, “Buffer Descriptors.”

The ten-bit interface (TBI) module MII registers are also described in this section. The TBI registers are defined like PHY registers and, as such, are accessed via the MII management interface in the same way the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) refer to Section 15.5.4, “Ten-Bit Interface (TBI).”

15.5.1 Top-Level Module Memory Map

Each of the eTSECs is allocated 4 Kbytes of memory-mapped space. The space for each eTSEC is divided as indicated in Table 15-3.

Table 15-3. Module Memory Map Summary

Address Offset	Function
000–0FF	eTSEC general control/status registers
100–2FF	eTSEC transmit control/status registers
300–4FF	eTSEC receive control/status registers
500–5FF	MAC registers
600–7FF	RMON MIB registers
800–8FF	Hash table registers
900–9FF	—
A00–AFF	FIFO control/status registers
B00–BFF	DMA system registers
C00–C3F	Lossless Flow Control registers
C40–DFF	—
E00–EFF	1588 Hardware Assist

15.5.2 Detailed Memory Map

The eTSEC memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in CCSRBAR as defined in Chapter 2, "Memory Map.") plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers must only be accessed as 32-bit quantities.

Table 15-4 lists the offset, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are applicable to each eTSEC. Block base addresses are as follows:

- eTSEC1 starts at 0x2_4000 address offset
- eTSEC2 starts at 0x2_5000 address offset
- eTSEC3 starts at 0x2_6000 address offset
- eTSEC4 starts at 0x2_7000 address offset

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 15-4. Module Memory Map

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
eTSEC General Control and Status Registers				
0x2_4000	TSEC_ID*—Controller ID register	R	0x0124_0000	15.5.3.1.1/15-26
0x2_4004	TSEC_ID2*—Controller ID register	R	0x0030_00F0	15.5.3.1.2/15-27
0x2_4008–0x2_400C	Reserved	—	—	—
0x2_4010	IEVENT—Interrupt event register	w1c	0x0000_0000	15.5.3.1.3/15-28
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	15.5.3.1.4/15-32
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	15.5.3.1.5/15-34
0x2_401C	Reserved	—	—	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	15.5.3.1.6/15-35
0x2_4024	Reserved	—	—	—
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	15.5.3.1.7/15-38
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	15.5.3.1.8/15-39
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	15.5.3.1.9/15-40
0x2_4034–0x2_40FC	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
eTSEC Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	15.5.3.2.1/15-41
0x2_4104	TSTAT—Transmit status register	w1c	0x0000_0000	15.5.3.2.2/15-42
0x2_4108	DFVLAN*—Default VLAN control word	R/W	0x8100_0000	15.5.3.2.3/15-46
0x2_410C	Reserved	—	—	—
0x2_4110	TXIC—Transmit interrupt coalescing register	R/W	0x0000_0000	15.5.3.2.4/15-47
0x2_4114	TQUEUE*—Transmit queue control register	R/W	0x0000_8000	15.5.3.2.5/15-48
0x2_4118–0x2_413C	Reserved	—	—	—
0x2_4140	TR03WT*—TxBD Rings 0–3 round-robin weightings	R/W	0x0000_0000	15.5.3.2.6/15-49
0x2_4144	TR47WT*—TxBD Rings 4–7 round-robin weightings	R/W	0x0000_0000	15.5.3.2.7/15-49
0x2_4148–0x2_417C	Reserved	—	—	—
0x2_4180	TBDBPH*—Tx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.2.8/15-50
0x2_4184	TBPTR0—TxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_4188	Reserved	—	—	—
0x2_418C	TBPTR1*—TxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_4190	Reserved	—	—	—
0x2_4194	TBPTR2*—TxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_4198	Reserved	—	—	—
0x2_419C	TBPTR3*—TxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_41A0	Reserved	—	—	—
0x2_41A4	TBPTR4*—TxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_41A8	Reserved	—	—	—
0x2_41AC	TBPTR5*—TxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_41B0	Reserved	—	—	—
0x2_41B4	TBPTR6*—TxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_41B8	Reserved	—	—	—
0x2_41BC	TBPTR7*—TxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.2.9/15-50
0x2_41C0–0x2_41FC	Reserved	—	—	—
0x2_4200	TBASEH*—TxBD base address high bits	R/W	0x0000_0000	15.5.3.2.10/15-51
0x2_4204	TBASE0—TxBD base address of ring 0	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4208	Reserved	—	—	—
0x2_420C	TBASE1*—TxBD base address of ring 1	R/W	0x0000_0000	15.5.3.2.11/15-52

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4210	Reserved	—	—	—
0x2_4214	TBASE2*—TxBD base address of ring 2	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4218	Reserved	—	—	—
0x2_421C	TBASE3*—TxBD base address of ring 3	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4220	Reserved	—	—	—
0x2_4224	TBASE4*—TxBD base address of ring 4	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4228	Reserved	—	—	—
0x2_422C	TBASE5*—TxBD base address of ring 5	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4230	Reserved	—	—	—
0x2_4234	TBASE6*—TxBD base address of ring 6	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4238	Reserved	—	—	—
0x2_423C	TBASE7*—TxBD base address of ring 7	R/W	0x0000_0000	15.5.3.2.11/15-52
0x2_4240–0x2_427C	Reserved	—	—	—
0x2_4280	TMR_TXTS1_ID* - Tx time stamp identification tag (set 1)	R/W	0x0000_0000	15.5.3.2.12/15-52
0x2_4284	TMR_TXTS2_ID* - Tx time stamp identification tag (set 2)	R/W	0x0000_0000	15.5.3.2.12/15-52
0x2_4288–0x2_42BC	Reserved	—	—	—
0x2_42C0	TMR_TXTS1_H* - Tx time stamp high (set 1)	R/W	0x0000_0000	15.5.3.2.13/15-53
0x2_42C4	TMR_TXTS1_L* - Tx time stamp high (set 1)	R/W	0x0000_0000	15.5.3.2.13/15-53
0x2_42C8	TMR_TXTS2_H* - Tx time stamp high (set 2)	R/W	0x0000_0000	15.5.3.2.13/15-53
0x2_42CC	TMR_TXTS2_L* - Tx time stamp high (set 2)	R/W	0x0000_0000	15.5.3.2.13/15-53
0x2_42D0–0x2_42FC	Reserved	—	—	—
eTSEC Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	15.5.3.3.1/15-54
0x2_4304	RSTAT—Receive status register	w1c	0x0000_0000	15.5.3.3.2/15-56
0x2_4308–0x2_430C	Reserved	—	—	—
0x2_4310	RXIC—Receive interrupt coalescing register	R/W	0x0000_0000	15.5.3.3.3/15-59
0x2_4314	RQUEUE*—Receive queue control register.	R/W	0x0080_0080	15.5.3.3.4/15-60
0x2_4318–0x2_432C	Reserved	—	—	—
0x2_4330	RBIFX*—Receive bit field extract control register	R/W	0x0000_0000	15.5.3.3.5/15-61
0x2_4334	RQFAR*—Receive queue filing table address register	R/W	0x0000_0000	15.5.3.3.6/15-63
0x2_4338	RQFCR*—Receive queue filing table control register	R/W	0x0000_0000	15.5.3.3.7/15-64

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_433C	RQFPR*—Receive queue filing table property register	R/W	0x0000_0000	15.5.3.3.8/15-65
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	15.5.3.3.9/15-68
0x2_4344–0x2_437C	Reserved	—	—	—
0x2_4380	RBDBPH*—Rx data buffer pointer high bits	R/W	0x0000_0000	15.5.3.3.10/15-69
0x2_4384	RBPTR0—RxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_4388	Reserved	—	—	—
0x2_438C	RBPTR1*—RxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_4390	Reserved	—	—	—
0x2_4394	RBPTR2*—RxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_4398	Reserved	—	—	—
0x2_439C	RBPTR3*—RxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_43A0	Reserved	—	—	—
0x2_43A4	RBPTR4*—RxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_43A8	Reserved	—	—	—
0x2_43AC	RBPTR5*—RxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_43B0	Reserved	—	—	—
0x2_43B4	RBPTR6*—RxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_43B8	Reserved	—	—	—
0x2_43BC	RBPTR7*—RxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.3.11/15-69
0x2_43C0–0x2_43FC	Reserved	—	—	—
0x2_4400	RBASEH*—RxBD base address high bits	R/W	0x0000_0000	15.5.3.3.12/15-70
0x2_4404	RBASE0—RxBD base address of ring 0	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4408	Reserved	—	—	—
0x2_440C	RBASE1*—RxBD base address of ring 1	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4410	Reserved	—	—	—
0x2_4414	RBASE2*—RxBD base address of ring 2	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4418	Reserved	—	—	—
0x2_441C	RBASE3*—RxBD base address of ring 3	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4420	Reserved	—	—	—
0x2_4424	RBASE4*—RxBD base address of ring 4	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4428	Reserved	—	—	—
0x2_442C	RBASE5*—RxBD base address of ring 5	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4430	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4434	RBASE6*—RxBD base address of ring 6	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4438	Reserved	—	—	—
0x2_443C	RBASE7*—RxBD base address of ring 7	R/W	0x0000_0000	15.5.3.3.13/15-71
0x2_4440–0x2_44BC	Reserved	—	—	—
0x2_44C0	TMR_RXTS_H* - Rx timer time stamp register high	R/W	0x0000_0000	15.5.3.3.14/15-71
0x2_44C4	TMR_RXTS_L* - Rx timer time stamp register low	R/W	0x0000_0000	15.5.3.3.14/15-71
0x2_44C8–0x2_44FC	Reserved	—	—	—
eTSEC MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	15.5.3.5.1/15-75
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	15.5.3.5.2/15-76
0x2_4508	IPGIFG—Inter-packet/inter-frame gap register	R/W	0x4060_5060	15.5.3.5.3/15-78
0x2_450C	HAFDUP—Half-duplex control	R/W	0x00A1_F037	15.5.3.5.4/15-79
0x2_4510	MAXFRM—Maximum frame length	R/W	0x0000_0600	15.5.3.5.5/15-80
0x2_4514–0x2_451C	Reserved	—	—	—
0x2_4520	MIIMCFG—MII management configuration	R/W	0x0000_0007	15.5.3.5.6/15-80
0x2_4524	MIIMCOM—MII management command	R/W	0x0000_0000	15.5.3.5.7/15-82
0x2_4528	MIIMADD—MII management address	R/W	0x0000_0000	15.5.3.5.8/15-82
0x2_452C	MIIMCON—MII management control	WO	0x0000_0000	15.5.3.5.9/15-83
0x2_4530	MIIMSTAT—MII management status	R	0x0000_0000	15.5.3.5.10/15-83
0x2_4534	MIIMIND—MII management indicator	R	0x0000_0000	15.5.3.5.11/15-84
0x2_4538	Reserved	—	—	—
0x2_453C	IFSTAT—Interface status	R	0x0000_0000	15.5.3.5.12/15-84
0x2_4540	MACSTNADDR1—MAC station address register 1	R/W	0x0000_0000	15.5.3.5.13/15-85
0x2_4544	MACSTNADDR2—MAC station address register 2	R/W	0x0000_0000	15.5.3.5.14/15-86

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4548	MAC01ADDR1*—MAC exact match address 1, part 1	R/W	0x0000_0000	15.5.3.5.15/15-86 15.5.3.5.16/15-87
0x2_454C	MAC01ADDR2*—MAC exact match address 1, part 2	R/W	0x0000_0000	
0x2_4550	MAC02ADDR1*—MAC exact match address 2, part 1	R/W	0x0000_0000	
0x2_4554	MAC02ADDR2*—MAC exact match address 2, part 2	R/W	0x0000_0000	
0x2_4558	MAC03ADDR1*—MAC exact match address 3, part 1	R/W	0x0000_0000	
0x2_455C	MAC03ADDR2*—MAC exact match address 3, part 2	R/W	0x0000_0000	
0x2_4560	MAC04ADDR1*—MAC exact match address 4, part 1	R/W	0x0000_0000	
0x2_4564	MAC04ADDR2*—MAC exact match address 4, part 2	R/W	0x0000_0000	
0x2_4568	MAC05ADDR1*—MAC exact match address 5, part 1	R/W	0x0000_0000	
0x2_456C	MAC05ADDR2*—MAC exact match address 5, part 2	R/W	0x0000_0000	
0x2_4570	MAC06ADDR1*—MAC exact match address 6, part 1	R/W	0x0000_0000	
0x2_4574	MAC06ADDR2*—MAC exact match address 6, part 2	R/W	0x0000_0000	
0x2_4578	MAC07ADDR1*—MAC exact match address 7, part 1	R/W	0x0000_0000	
0x2_457C	MAC07ADDR2*—MAC exact match address 7, part 2	R/W	0x0000_0000	
0x2_4580	MAC08ADDR1*—MAC exact match address 8, part 1	R/W	0x0000_0000	
0x2_4584	MAC08ADDR2*—MAC exact match address 8, part 2	R/W	0x0000_0000	
0x2_4588	MAC09ADDR1*—MAC exact match address 9, part 1	R/W	0x0000_0000	
0x2_458C	MAC09ADDR2*—MAC exact match address 9, part 2	R/W	0x0000_0000	
0x2_4590	MAC10ADDR1*—MAC exact match address 10, part 1	R/W	0x0000_0000	
0x2_4594	MAC10ADDR2*—MAC exact match address 10, part 2	R/W	0x0000_0000	15.5.3.5.15/15-86 15.5.3.5.16/15-87
0x2_4598	MAC11ADDR1*—MAC exact match address 11, part 1	R/W	0x0000_0000	
0x2_459C	MAC11ADDR2*—MAC exact match address 11, part 2	R/W	0x0000_0000	
0x2_45A0	MAC12ADDR1*—MAC exact match address 12, part 1	R/W	0x0000_0000	
0x2_45A4	MAC12ADDR2*—MAC exact match address 12, part 2	R/W	0x0000_0000	
0x2_45A8	MAC13ADDR1*—MAC exact match address 13, part 1	R/W	0x0000_0000	
0x2_45AC	MAC13ADDR2*—MAC exact match address 13, part 2	R/W	0x0000_0000	
0x2_45B0	MAC14ADDR1*—MAC exact match address 14, part 1	R/W	0x0000_0000	
0x2_45B4	MAC14ADDR2*—MAC exact match address 14, part 2	R/W	0x0000_0000	
0x2_45B8	MAC15ADDR1*—MAC exact match address 15, part 1	R/W	0x0000_0000	
0x2_45BC	MAC15ADDR2*—MAC exact match address 15, part 2	R/W	0x0000_0000	
0x2_45C0–0x2_467C	Reserved	—	—	—
eTSEC Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter	R/W	0x0000_0000	15.5.3.6.1/15-88
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	15.5.3.6.2/15-89
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	15.5.3.6.3/15-89

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	15.5.3.6.4/15-90
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	15.5.3.6.5/15-90
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	15.5.3.6.6/15-91
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	15.5.3.6.7/15-91
eTSEC Receive Counters				
0x2_469C	RBYT—Receive byte counter	R/W	0x0000_0000	15.5.3.6.8/15-92
0x2_46A0	RPKT—Receive packet counter	R/W	0x0000_0000	15.5.3.6.9/15-92
0x2_46A4	RFCS—Receive FCS error counter	R/W	0x0000_0000	15.5.3.6.10/15-93
0x2_46A8	RMCA—Receive multicast packet counter	R/W	0x0000_0000	15.5.3.6.11/15-93
0x2_46AC	RBCA—Receive broadcast packet counter	R/W	0x0000_0000	15.5.3.6.12/15-94
0x2_46B0	RXCF—Receive control frame packet counter	R/W	0x0000_0000	15.5.3.6.13/15-94
0x2_46B4	RXPF—Receive PAUSE frame packet counter	R/W	0x0000_0000	15.5.3.6.14/15-95
0x2_46B8	RXUO—Receive unknown OP code counter	R/W	0x0000_0000	15.5.3.6.15/15-95
0x2_46BC	RALN—Receive alignment error counter	R/W	0x0000_0000	15.5.3.6.16/15-96
0x2_46C0	RFLR—Receive frame length error counter	R/W	0x0000_0000	15.5.3.6.17/15-96
0x2_46C4	RCDE—Receive code error counter	R/W	0x0000_0000	15.5.3.6.18/15-97
0x2_46C8	RCSE—Receive carrier sense error counter	R/W	0x0000_0000	15.5.3.6.19/15-97
0x2_46CC	RUND—Receive undersize packet counter	R/W	0x0000_0000	15.5.3.6.20/15-98
0x2_46D0	ROVR—Receive oversize packet counter	R/W	0x0000_0000	15.5.3.6.21/15-98
0x2_46D4	RFRG—Receive fragments counter	R/W	0x0000_0000	15.5.3.6.22/15-99
0x2_46D8	RJBR—Receive jabber counter	R/W	0x0000_0000	15.5.3.6.23/15-99
0x2_46DC	RDRP—Receive drop counter	R/W	0x0000_0000	15.5.3.6.24/15-100
eTSEC Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter	R/W	0x0000_0000	15.5.3.6.25/15-100
0x2_46E4	TPKT—Transmit packet counter	R/W	0x0000_0000	15.5.3.6.26/15-101
0x2_46E8	TMCA—Transmit multicast packet counter	R/W	0x0000_0000	15.5.3.6.27/15-101
0x2_46EC	TBCA—Transmit broadcast packet counter	R/W	0x0000_0000	15.5.3.6.28/15-102
0x2_46F0	TXPF—Transmit PAUSE control frame counter	R/W	0x0000_0000	15.5.3.6.29/15-102
0x2_46F4	TDFR—Transmit deferral packet counter	R/W	0x0000_0000	15.5.3.6.30/15-103
0x2_46F8	TEDF—Transmit excessive deferral packet counter	R/W	0x0000_0000	15.5.3.6.31/15-103
0x2_46FC	TSCL—Transmit single collision packet counter	R/W	0x0000_0000	15.5.3.6.32/15-104
0x2_4700	TMCL—Transmit multiple collision packet counter	R/W	0x0000_0000	15.5.3.6.33/15-104
0x2_4704	TLCL—Transmit late collision packet counter	R/W	0x0000_0000	15.5.3.6.34/15-105

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4708	TXCL—Transmit excessive collision packet counter	R/W	0x0000_0000	15.5.3.6.35/15-105
0x2_470C	TNCL—Transmit total collision counter	R/W	0x0000_0000	15.5.3.6.36/15-106
0x2_4710	Reserved	—	—	—
0x2_4714	TDRP—Transmit drop frame counter	R/W	0x0000_0000	15.5.3.6.37/15-106
0x2_4718	TJBR—Transmit jabber frame counter	R/W	0x0000_0000	15.5.3.6.38/15-107
0x2_471C	TFCS—Transmit FCS error counter	R/W	0x0000_0000	15.5.3.6.39/15-107
0x2_4720	TXCF—Transmit control frame counter	R/W	0x0000_0000	15.5.3.6.40/15-108
0x2_4724	TOVR—Transmit oversize frame counter	R/W	0x0000_0000	15.5.3.6.41/15-108
0x2_4728	TUND—Transmit undersize frame counter	R/W	0x0000_0000	15.5.3.6.42/15-109
0x2_472C	TFRG—Transmit fragments frame counter	R/W	0x0000_0000	15.5.3.6.43/15-109
eTSEC Counter Control and TOE Statistics Registers				
0x2_4730	CAR1—Carry register one register[3]	R	0x0000_0000	15.5.3.6.44/15-110
0x2_4734	CAR2—Carry register two register [3]	R	0x0000_0000	15.5.3.6.45/15-111
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE03_FFFF	15.5.3.6.46/15-112
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFD	15.5.3.6.47/15-114
0x2_4740	RREJ*—Receive filer rejected packet counter	R/W	0x0000_0000	15.5.3.6.48/15-115
0x2_4744–0x2_47FC	Reserved	—	—	—
Hash Function Registers				
0x2_4800	IGADDR0—Individual/group address register 0	R/W	0x0000_0000	15.5.3.7.1/15-116
0x2_4804	IGADDR1—Individual/group address register 1	R/W	0x0000_0000	
0x2_4808	IGADDR2—Individual/group address register 2	R/W	0x0000_0000	
0x2_480C	IGADDR3—Individual/group address register 3	R/W	0x0000_0000	
0x2_4810	IGADDR4—Individual/group address register 4	R/W	0x0000_0000	
0x2_4814	IGADDR5—Individual/group address register 5	R/W	0x0000_0000	
0x2_4818	IGADDR6—Individual/group address register 6	R/W	0x0000_0000	
0x2_481C	IGADDR7—Individual/group address register 7	R/W	0x0000_0000	
0x2_4820–0x2_487C	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	15.5.3.7.2/15-116
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0–0x2_49FC	Reserved	—	—	—
eTSEC FIFO Control Registers				
0x2_4A00	FIFOCFG*—FIFO interface configuration register	R/W	0x0000_00C0	15.5.3.8.1/15-117
0x2_4A04–0x2_4AFC	Reserved	—	—	—
eTSEC DMA Attribute Registers				
0x2_4B00–0x2_4BF4	Reserved	—	—	—
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	15.5.3.9.1/15-119
0x2_4BFC	ATTRELI*—Attribute extract length and extract index register	R/W	0x0000_0000	15.5.3.9.2/15-120
eTSEC Lossless Flow Control Registers				
0x2_4C00	RQPRM0*—Receive Queue Parameters register 0	R/W	0x0000_0000	15.5.3.10.1/15-121
0x2_4C04	RQPRM1*—Receive Queue Parameters register 1	R/W	0x0000_0000	
0x2_4C08	RQPRM2*—Receive Queue Parameters register 2	R/W	0x0000_0000	
0x2_4C0C	RQPRM3*—Receive Queue Parameters register 3	R/W	0x0000_0000	
0x2_4C10	RQPRM4*—Receive Queue Parameters register 4	R/W	0x0000_0000	
0x2_4C14	RQPRM5*—Receive Queue Parameters register 5	R/W	0x0000_0000	
0x2_4C18	RQPRM6*—Receive Queue Parameters register 6	R/W	0x0000_0000	
0x2_4C1C	RQPRM7*—Receive Queue Parameters register 7	R/W	0x0000_0000	
0x2_4C20–0x2_4C40	Reserved	—	—	—
0x2_4C44	RFBPTR0*—Last Free RxBD pointer for ring 0	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C48	Reserved	—	—	—
0x2_4C4C	RFBPTR1*—Last Free RxBD pointer for ring 1	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C50	Reserved	—	—	—
0x2_4C54	RFBPTR2*—Last Free RxBD pointer for ring 2	R/W	0x0000_0000	15.5.3.10.2/15-121

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4C58	Reserved	—	—	—
0x2_4C5C	RFBPTR3*—Last Free RxBD pointer for ring 3	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C60	Reserved	—	—	—
0x2_4C64	RFBPTR4*—Last Free RxBD pointer for ring 4	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C68	Reserved	—	—	—
0x2_4C6C	RFBPTR5*—Last Free RxBD pointer for ring 5	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C70	Reserved	—	—	—
0x2_4C74	RFBPTR6*—Last Free RxBD pointer for ring 6	R/W	0x0000_0000	15.5.3.10.2/15-121
0x2_4C78	Reserved	—	—	—
0x2_4C7C	RFBPTR7*—Last Free RxBD pointer for ring 7	R/W	0x0000_0000	15.5.3.10.2/15-121
eTSEC Future Expansion Space				
0x2_4CC0 – 0x2_4D94	Reserved	—	—	—
eTSEC IEEE 1588 Registers				
0x2_4E00	TMR_CTRL* - Timer control register	R/W	0x0001_0000	15.5.3.11.1/15-122
0x2_4E04	TMR_TEVENT* - time stamp event register	R/W	0x0000_0000	15.5.3.11.2/15-124
0x2_4E08	TMR_TEMASK* - Timer event mask register	R/W	0x0000_0000	15.5.3.11.3/15-125
0x2_4E0C	TMR_PEVENT* - time stamp event register	R/W	0x0000_0000	15.5.3.11.4/15-126
0x2_4E10	TMR_PEMASK* - Timer event mask register	R/W	0x0000_0000	15.5.3.11.5/15-127
0x2_4E14	TMR_STAT* - time stamp status register	R/W	0x0000_0000	15.5.3.11.6/15-127
0x2_4E18	TMR_CNT_H* - timer counter high register	R/W	0x0000_0000	15.5.3.11.7/15-128
0x2_4E1C	TMR_CNT_L* - timer counter low register	R/W	0x0000_0000	15.5.3.11.7/15-128
0x2_4E20	TMR_ADD* - Timer drift compensation addend register	R/W	0x0000_0000	15.5.3.11.8/15-129
0x2_4E24	TMR_ACC* - Timer accumulator register	R/W	0x0000_0000	15.5.3.11.9/15-130
0x2_4E28	TMR_PRSC* - timer prescale	R/W	0x0000_0002	15.5.3.11.10/15-130
0x2_4E2C	Reserved	—	—	—
0x2_4E30	TMROFF_H* - Timer offset high	R/W	0x0000_0000	15.5.3.11.11/15-131
0x2_4E34	TMROFF_L* - Timer offset low	R/W	0x0000_0000	15.5.3.11.11/15-131
0x2_4E40	TMR_ALARM1_H* - Timer alarm 1 high register	R/W	0xFFFF_FFFF	15.5.3.11.12/15-131
0x2_4E44	TMR_ALARM1_L* - Timer alarm 1 high register	R/W	0xFFFF_FFFF	
0x2_4E48	TMR_ALARM2_H* - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x2_4E4C	TMR_ALARM2_L* - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x2_4E50– 0x2_4E7C	Reserved	—	—	—

Table 15-4. Module Memory Map (continued)

eTSEC1 Offset	Name[1]	Access [2]	Reset	Section/Page
0x2_4E80	TMR_FIPER1* - Timer fixed period interval	R/W	0xFFFF_FFFF	15.5.3.11.13/15-132
0x2_4E84	TMR_FIPER2* - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x2_4E88	TMR_FIPER*3 - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x2_4EA0	TMR_ETTS1_H* - Time stamp of general purpose external trigger	R/W	0x0000_0000	15.5.3.11.14/15-133
0x2_4EA4	TMR_ETTS1_L* - Time stamp of general purpose external trigger	R/W	0x0000_0000	
0x2_4EA8	TMR_ETTS2_H* - Time stamp of general purpose external trigger	R/W	0x0000_0000	
0x2_4EAC	TMR_ETTS2_L* - Time stamp of general purpose external trigger	R/W	0x0000_0000	
0x2_4EB0 – 0x2_4FFF	Reserved	—	—	—
Other eTSECs				
0x2_5000– 0x2_5FFF	eTSEC2 REGISTERS[4]			
0x2_6000– 0x2_6FFF	eTSEC3 REGISTERS[5]			
0x2_7000– 0x2_7FFF	eTSEC4 REGISTERS[6]			

1 Registers denoted * are new to the enhanced TSEC and not supported by PowerQUICC III TSECs.

2 Key: R = read only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing.

3 Cleared on read.

4 eTSEC2 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFF, except the offsets are from 0x 2_5000 to 0x2_5FFF.

5 eTSEC3 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFF, except the offsets are from 0x 2_6000 to 0x2_6FFF.

6 eTSEC4 has the same memory-mapped registers that are described for eTSEC1 from 0x 2_4000 to 0x2_4FFF, except the offsets are from 0x 2_7000 to 0x2_7FFF.

15.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the eTSEC registers. Because all of the eTSEC registers are 32 bits wide, only 32-bit register accesses are supported.

15.5.3.1 eTSEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

15.5.3.1.1 Controller ID Register (TSEC_ID)

The controller ID register (TSEC_ID) is a read-only register. The TSEC_ID register is used to identify the eTSEC block and revision.

Offset eTSEC1:0x2_4000; eTSEC2:0x2_5000; eTSEC3:0x2_6000; eTSEC4:0x2_7000 — Access: Read only

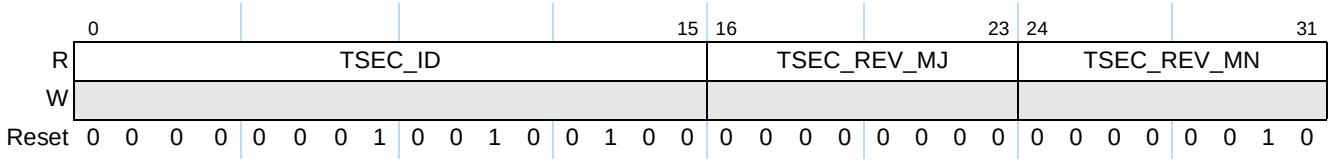


Figure 15-2. TSEC_ID Register

Table 15-9 describes the fields of the TSEC_ID register.

Table 15-5. TSEC_ID Field Descriptions

Bits	Name	Description
0–15	TSEC_ID	Value identifying the eTSEC (10/100/1000 Ethernet MAC). 0124 Unique identifier for eTSEC with 8 Rx and 8 Tx BD rings.
16–23	TSEC_REV_MJ	Value identifies the major revision of the eTSEC. 00 Initial revision
24–31	TSEC_REV_MN	Value identifies the minor revision of the eTSEC.

15.5.3.1.2 Controller ID Register (TSEC_ID2)

The controller ID register (TSEC_ID2) is a read-only register. The TSEC_ID2 register is used to identify the eTSEC block configuration.

Offset eTSEC1:0x2_4004; eTSEC2:0x2_5004; eTSEC3:0x2_6004; eTSEC4:0x2_7004 — Access: Read only

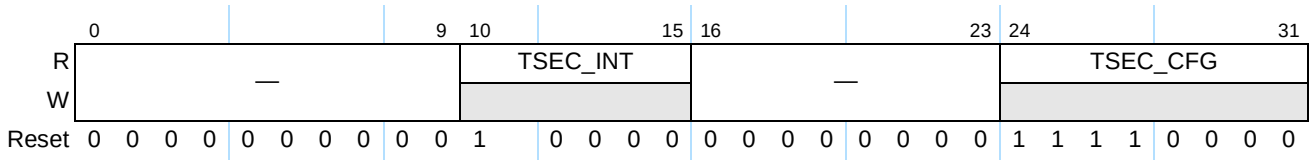


Figure 15-3. TSEC_ID2 Register

Table 15-6 describes the fields of the TSEC_ID2 register.

Table 15-6. TSEC_ID2 Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–9</td><td>—</td><td>Reserved</td></tr>
<tr><td>10–15</td><td>TSEC_INT</td><td>Interface mode support.
<table>
<tr><th>Bit</th><th>Mode</th></tr>
<tr><td>10</td><td>0 Ethernet mode not supported
1 Ethernet mode supported</td></tr>
<tr><td>11</td><td>0 FIFO mode not supported
1 FIFO mode supported</td></tr>
<tr><td>12</td><td>0 Can be configured to run in FIFO 16-bit mode
1 FIFO 16-bit mode off</td></tr>
<tr><td>12</td><td>Reserved</td></tr>
<tr><td>12-13</td><td>Reserved0Can be configured to run in FIFO 8-bit mode
1 FIFO 8-bit mode off</td></tr>
<tr><td>14</td><td>0 Can be configured to run in Ethernet normal/full mode
1 Ethernet normal/full mode off</td></tr>
<tr><td>15</td><td>0 Can be configured to run in Ethernet reduced mode
1 Ethernet reduced mode off</td></tr>
</table></td></tr>
<tr><td>16–23</td><td>—</td><td>Reserved</td></tr>
<tr><td>24–31</td><td>TSEC_CFG</td><td>Value identifies configuration options of the eTSEC.
00 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are off
F0 eTSEC multiple ring, Rx TOE, Filer and Tx TOE supports are on
30 eTSEC multiple ring support is OFF and Rx TOE, Filer and Tx TOE supports are on
50 eTSEC multiple ring and filer supports are OFF and Rx TOE and Tx TOE supports are on</td></tr>
</table>

15.5.3.1.3 Interrupt Event Register (IEVENT)

Interrupt events cause bits in the IEVENT register to be set. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the interrupt mask register (IMASK), the event also causes a hardware interrupt at the PIC. A bit in the interrupt event register is cleared by writing a 1 to that bit position. A write of 0 has no effect.

Each eTSEC can issue three kinds of hardware interrupt to the PIC:

1. Transmit interrupts—Issued whenever bits TXB or TXF of IEVENT are set to 1 and either transmit interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for TXF. To negate this hardware interrupt, software must clear both TXB and TXF bits.
2. Receive interrupts—Issued whenever bits RXB or RXF of IEVENT are set to 1 and either receive interrupt coalescing is disabled or the interrupt coalescing thresholds have been met for RXF. To negate this hardware interrupt, software must clear both RXB and RXF bits.

3. Error and diagnostic interrupts—Issued whenever bits MAG, GTSC, GRSC, TXC, RXC, BABR, BABT, LC, CRL, FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN or BSY of IEVENT are set to 1. Software must clear all of these bits to negate an error/diagnostic hardware interrupt.
   - — Magic Packet reception event is: MAG
   - — Operational diagnostics are events on: GTSC, GRSC, TXC and RXC
   - — Interrupts resulting from errors/problems detected in the network or transceiver are: BABR, BABT, LC and CRL
   - — Interrupts resulting from internal errors are: FIR, FIQ, DPE, PERR, EBERR, TXE, XFUN and BSY

Some of the error interrupts are independently counted in the MIB block counters. Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Figure 15-4 describes the definition for the IEVENT register.

Offset eTSEC1:0x2_4010; eTSEC2: 0x2_5010; eTSEC3:0x2_6010; eTSEC4:0x2_7010 — Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	—	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	—	LC	CRL	XFUN
W	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c
Reset	All zeros															

	16	17	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXB	—		MAG	MMRD	MMWR	GRSC	RXF	—		FGPI	FIR	FIQ	DPE	PERR
W	w1c			w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c
Reset	All zeros														

Figure 15-4. IEVENT Register Definition

Table 15-7 describes the fields of the IEVENT register.

Table 15-7. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received. 1 Excessive frame received.
1	RXC	Receive control interrupt. A control frame was received while MACCFG1[Rx_Flow] is set. As soon as the transmitter finishes sending the current frame, a pause operation is performed. 0 Control frame not received. 1 Control frame received.
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. 0 No frame received and discarded. 1 Frame received and discarded.
3	EBERR	Internal bus error. This bit indicates that a system bus error occurred while a DMA transaction was underway. As a result, transferred data is expected to be partially or completely invalid. 0 No system bus error occurred. 1 System bus error occurred.

Table 15-7. IEVENT Field Descriptions (continued)

Bits	Name	Description
4	—	Reserved
5	MSRO	MIB counter overflow. This interrupt is asserted if the count for one of the MIB counters has exceeded the size of its register. 0 MIB count not exceeding its register size. 1 MIB count exceeds its register size.
6	GTSC	Graceful transmit stop complete. This interrupt is asserted for one of two reasons. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. • A graceful stop, which was initiated by setting DMACTRL[GTS], is now complete. • A transmission of a flow control PAUSE frame, which was initiated by setting TCTRL[TFC_PAUSE], is now complete. 0 No graceful stop interrupt. 1 Graceful stop requested.
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's maximum frame length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. 0 Transmitted frame length not exceeding maximum frame length. 1 Transmitted frame length exceeding maximum frame length when MACCFG2[Huge Frame] = 0.
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted. 1 Control frame transmitted.
9	TXE	Transmit error. This bit indicates that an error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by the eTSEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL, XFUN). 0 No transmit channel error occurred. 1 Transmit channel error occurred.
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated. 1 Transmit buffer descriptor updated.
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (interrupt) bit in the status word of the buffer descriptor is set. The specific transmit queue that was updated has its TXF bit set in TSTAT. 0 No frame transmitted/TxBD not updated. 1 Frame transmitted/TxBD updated.
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred.
14	CRL	Collision retry limit. This bit indicates that the number of successive transmission collisions has exceeded the MAC's half-duplex register's retransmission maximum count (HAFDUP[Retransmission Maximum]). The frame is discarded without being transmitted and transmission of the next frame commences. This only occurs while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum. 1 Successive transmission collisions exceed maximum.

Table 15-7. IEVENT Field Descriptions (continued)

Bits	Name	Description
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun. 1 Transmit FIFO underrun.
16	RXB	Receive buffer. This bit indicates that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated. 1 Receiver buffer descriptor updated.
17–19	—	Reserved
20	MAG	Magic Packet detected when the eTSEC is in Magic Packet detection mode (MACCFG2[MPEN] = 1). 0 No Magic Packet received, or Magic Packet mode was not enabled. 1 A Magic Packet was received while in Magic Packet mode. MACCFG2[MPEN] is also cleared upon receiving the Magic Packet.
21	MMRD	MII management read completion 0 MII management read not issued or in process. 1 MII management read completed that was initiated by a user through the MII Scan or Read cycle command.
22	MMWR	MII management write completion 0 MII management write not issued or in process. 1 MII management write completed that was initiated by a user write to the MIIMCON register.
23	GRSC	Graceful receive stop complete. This interrupt is asserted if a graceful receive stop is completed. It allows the user to know if the system has completed the stop and it is safe to write to receive registers (status, control or configuration registers) that are used by the system during normal operation. 0 Graceful stop not completed. 1 Graceful stop completed.
24	RXF	Receive frame interrupt. This bit indicates that a frame was received and the last receive buffer descriptor (RxBD) in that frame was updated. This occurs either if the I (interrupt) bit in the buffer descriptor status word is set, or an overrun error occurs. The specific receive queue that was updated has its RXF bit set in RSTAT. 0 Frame not received. 1 Frame received.
25–26	—	Reserved
27	FGPI	Filer generated general purpose interrupt on a set of filer rule match. This bit is set upon reception of a frame that matches a GPI rule sequence that is specified in the filer. It is synchronized with the setting of RXF. 0 No filer generated interrupt has occurred. 1 The filer has accepted a frame via a matching rule that the RQFCR[GPI] bit set.
28	FIR	The receive queue filer result is invalid, either because not enough time between frames was available to find a matching rule, or no entry in the filer table could be matched. 0 Receive queue filer reached a definite result; however, bit FIQ may still be set if a frame was filed to a disabled RxBD ring. 1 Receive queue filer was unable to reach a definite result. In this case, bit FIQ also is set if no entry in the filer table could provide a rule match.
29	FIQ	Filed frame to invalid receive queue. This bit indicates that either the receive queue filer chose to DMA a received frame to a disabled RxBD ring, or that no rule in the filer table could be matched. 0 Received frames filed to valid queues or rejected. Note that a frame may be rejected if the filer has insufficient time to reach a conclusive result between frames, in which case bit FIR is set. 1 Received frames filed to RxBD rings that are not enabled. The frame is discarded. If bit FIR is also set this indicates that the filer exhausted all of its table entries without a rule match.

Table 15-7. IEVENT Field Descriptions (continued)

Bits	Name	Description
30	DPE	Internal data parity error. This bit indicates that the eTSEC has detected a parity error on its stored data, which is likely to compromise the validity of recently transferred frames. 0 No parity errors detected. 1 Data held in the FIFO or filer arrays is expected to be corrupted due to a parity error.
31	PERR	Receive frame parse error for TCP/IP off-load. This bit indicates that a received frame could not be parsed unambiguously, due to encapsulated header type fields contradicting each other. 0 Received frame parsed successfully. 1 Received frame parse revealed header inconsistencies.

15.5.3.1.4 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events in the IEVENT register are permitted to participate in generating hardware interrupts to the PIC. All implemented bits in this register are R/W and cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, the PIC receives an interrupt (for each eTSEC these are grouped into transmit, receive, and error/diagnostic interrupts). The interrupt signal remains asserted until either the IEVENT bit is cleared, by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Figure 15-5 describes the IMASK register.

Offset eTSEC1:0x2_4014; eTSEC2:0x2_5014; eTSEC3:0x2_6014; eTSEC4:0x2_7014 — Access: Read/Write

	0	1	2	3	4	5	6	7
R W	BREN	RXCEN	BSYEN	EBERREN	—	MSROEN	GTSCEN	BTEN
Reset	All zeros							

	8	9	10	11	12	13	14	15
R W	TXCEN	TXEEN	TXBEN	TXFEN	—	LCEN	CRLEN	XFUNEN
Reset	All zeros							

	16	17	19	20	21	22	23
R W	RXBEN	—		MAGEN	MMRDEN	MMWREN	GRSCEN
Reset	All zeros						

	24	25	26	27	28	29	30	31
R W	RXFEN	—		FGPIEN	FIREN	FIQEN	DPEEN	PERREN
Reset	All zeros							

Figure 15-5. IMASK Register Definition

Table 15-8 describes the fields of the IMASK register.

Table 15-8. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MIB counter overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRLEN	Collision retry limit enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–19	—	Reserved
20	MAGEN	Magic packet received interrupt enable
21	MMRDEN	MII management read completion interrupt enable
22	MMWREN	MII management write completion interrupt enable
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–26	—	Reserved
27	FGPIEN	Filer general purpose interrupt enable
28	FIREN	Filer invalid result interrupt enable
29	FIQEN	Filed frame to invalid queue interrupt enable
30	DPEEN	Data parity error interrupt enable
31	PERREN	Receive frame parse error enable

15.5.3.1.5 Error Disabled Register (EDIS)

Figure 15-6 describes the definition for the EDIS register. The error disabled register allows the user to disable an error interruption, possibly to avoid spurious error indications external to the eTSECs.

Offset eTSEC1:0x2_4018; eTSEC2:0x2_5018; eTSEC3:0x2_6018; eTSEC4:0x2_7018 — Access: Read/Write

	0	1	2	3	4	6	7	8	9	10	12	13	14	15
R	—		BSYDIS	EBERRDIS	—		BABTDIS	—	TXEDIS	—		LCDIS	CRLDIS	XFUNDIS
W														
Reset	All zeros													

	16	27	28	29	30	31
R	—		FIRDIS	FIQDIS	DPEDIS	PERRDIS
W						
Reset	All zeros					

Figure 15-6. EDIS Register Definition

Table 15-9 describes the fields of the EDIS register.

Table 15-9. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable. 0 Allow eTSEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable. 0 Allow eTSEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–6	—	Reserved
7	BABTDIS	Babbling transmit error disable. 0 Allow eTSEC to report IEVENT[BABT] status and set the buffer descriptor TR field. 1 Do not set IEVENT[BABT] nor the buffer descriptor TR field.
8	—	Reserved
9	TXEDIS	Transmit error disable. 0 Allow eTSEC to report IEVENT[TXE] status. 1 Do not set IEVENT[TXE] if TXE condition occurs.
10–12	—	Reserved
13	LCDIS	Late collision disable. 0 Allow eTSEC to report IEVENT[LC] status, set the buffer descriptor LC field, and halt buffer descriptor queue if LC condition occurs. 1 Do not set IEVENT[LC] nor the buffer descriptor LC field, and do not halt buffer descriptor queue if LC condition occurs.

Table 15-9. EDIS Field Descriptions (continued)

Bits	Name	Description
14	CRLDIS	Collision retry limit disable. 0 Allow eTSEC to report IEVENT[CRL] status, set the buffer descriptor RL field, and halt buffer descriptor queue if CRL condition occurs. 1 Do not set IEVENT[CRL] nor the buffer descriptor RL field, and do not halt buffer descriptor queue if CRL condition occurs.
15	XFUNDIS	Transmit FIFO underrun disable. 0 Allow eTSEC to report IEVENT[XFUN] status, set the buffer descriptor UN field, and halt buffer descriptor queue if XFUN condition occurs. 1 Do not set IEVENT[XFUN] nor the buffer descriptor UN field, and do not halt buffer descriptor queue if XFUN condition occurs.
16–27	—	Reserved
28	FIRDIS	Filer invalid result error disable. 0 Allow eTSEC to report IEVENT[FIR] status. 1 Do not set IEVENT[FIR] if eTSEC fails to reach a definite filer result when attempting to file a received frame, but discard the frame silently.
29	FIQDIS	Filed frame to invalid queue error disable. 0 Allow eTSEC to report IEVENT[FIQ] status. 1 Do not set IEVENT[FIQ] if eTSEC attempts to file a received frame to an invalid (disabled) RxBD ring, but discard the frame silently.
30	DPEDIS	Data parity error disable. 0 Allow eTSEC to report IEVENT[DPE] status. 1 Do not set IEVENT[DPE] if a parity error occurs in eTSEC's FIFO or filer arrays.
31	PERRDIS	Receive frame parse error disable. 0 Allow eTSEC to report IEVENT[PERR] status. 1 Do not set IEVENT[PERR] if a parse error occurs on a received frame.

15.5.3.1.6 Ethernet Control Register (ECNTRL)

ECNTRL is a register writable by the user to reset, configure, and initialize the eTSEC. Note that the FIFM, GMIIM, TBIM, RPM, and RMM fields are read-only, having been set after sampling signals at power-on-reset.

Figure 15-7 describes the definition for the ECNTRL register.

Offset eTSEC1:0x2_4020; eTSEC2:0x2_5020; eTSEC3:0x2_6020; eTSEC4:0x2_7020 Access: Mixed

	0	15
R	—	
W		
Reset	All zeros	

	16	17	18	19	20	24	25	26	27	28	29	30	31
R	FIFM	CLRCNT	AUTOZ	STEN	—		GMIIM	TBIM	RPM	R100M	RMM	SGMIIM	—
W													
Reset	All zeros												

Figure 15-7. ECNTRL Register Definition

Table 15-10 describes the fields of the ECNTRL register.

Table 15-10. ECNTRL Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	FIFM	FIFO mode enable. If this bit is set, 8- or 16-bit FIFO interface mode is enabled. This bit can be pin configured at reset to set or clear. See Section 4.4.3, "Power-On Reset Configuration." The FIFO width is configured according to RPM. 0 Interface to external signals via the Ethernet MAC. 1 Interface to external signals via the 8/16-bit FIFO interface, bypassing the Ethernet MAC. Frame parsing in this mode automatically assumes that IP packets are being received and transmitted. See FIFOCFG register for configuration of the FIFO interface.
17	CLRCNT	Clear all statistics counters 0 Allow MIB counters to continue to increment. 1 Reset all MIB counters. This bit is self-resetting.
18	AUTOZ	Automatically zero MIB counter values. 0 The user must write the addressed counter zero after a host read. 1 The addressed counter value is automatically cleared to zero after a host read. This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
19	STEN	MIB counter statistics enabled. 0 Statistics not enabled 1 Enables internal counters to update This is a steady state signal and must be set prior to enabling the Ethernet controller and must not be changed without proper care.
20–24	—	Reserved
25	GMIIM	GMII interface mode. If this bit is set, a PHY with a GMII or RGMII interface is expected to be connected. If cleared, a PHY with an MII or RMII interface is expected. The user should then set MACCFG2[I/F Mode] accordingly. The state of this status bit is defined during power-on reset. See Section 4.4.3, "Power-On Reset Configuration." 0 MII or RMII mode interface expected 1 GMII or RGMII mode interface expected
26	TBIM	Ten-bit interface mode. If this bit is set, ten-bit interface mode is enabled. This bit can be pin-configured at reset to set or clear. See Section 4.4.3, "Power-On Reset Configuration." 0 GMII or MII or RMII mode interface 1 TBI mode interface
27	RPM	Reduced-pin mode for Gigabit interfaces. If this bit is set, a reduced-pin interface is expected on either Ethernet and FIFO interfaces. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3, "Power-On Reset Configuration." 0 GMII or MII or TBI in non-reduced-pin mode configuration 1 RGMII or RTBI reduced-pin mode FIFO configured for 8-bit operation
27	RPM	Reduced-pin mode for Gigabit interfaces. If this bit is set, a reduced-pin interface is expected on either Ethernet and FIFO interfaces. RPM and RMM are never set together. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3, "Power-On Reset Configuration." 0 GMII or MII or TBI in non-reduced-pin mode configuration FIFO configured for 16-bit operation 1 RGMII or RTBI reduced-pin mode FIFO configured for 8-bit operation

Table 15-10. ECNTRL Field Descriptions (continued)

Bits	Name	Description
28	R100M	RGMII/RMII 100 mode. This bit is ignored unless SGMIIM, RPM or RMM are set and MACCFG2[I/F Mode] is assigned to 10/100 (01). If this bit is set, the eTSEC interface is in 100 Mbps speed. 0 RGMII is in 10 Mbps mode; RMII is in 10 Mbps mode, and every 10th RMII Reference clock is used to transfer data SGMII is in 10 Mbps mode, and every 100th SGMII Reference clock is used to transfer data 1 RGMII is in 100 Mbps mode; RMII is in 100 Mbps mode, and data is transferred on every Reference clock SGMII is in 100 Mbps mode, and every 10th SGMII Reference clock is used to transfer data
29	RMM	Reduced-pin mode for 10/100 interfaces. If this bit is set, an RMII pin interface is expected. RMM must be 0 if RPM = 1. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3, “Power-On Reset Configuration.” 0 Non-RMII interface mode 1 RMII interface mode
30	SGMIIM	Serial GMII mode. If this bit is set, a SGMII pin interface is expected to be connected via an on chip SERDES. This register can be pin-configured at reset to 0 or 1. See Section 4.4.3.11, “eTSEC SGMII Mode.” 0 SGMII mode disabled. eTSEC connected via a parallel interface. 1 SGMII mode enabled.
31	—	Reserved

The different interface configurations indicated by registers ECNTRL and MACCFG2 are summarized in Table 15-11.

Table 15-11. eTSEC Interface Configurations

Interface Mode	ECNTRL Field							MACCFG2 Field
	FIFM	GMIIM	TBIM	RPM	R100M	RMM	SGMIIM	I/F Mode
FIFO 8-bits	1	0	0	1	—	—	0	—
FIFO 16-bits	1	0	0	0	—	—	0	—
TBI 1Gbps	0	0	1	0	—	—	0	10
RTBI 1Gbps	0	0	1	1	—	—	0	10
GMII 1Gbps[1]	0	1	0	0	—	—	0	10
RGMII 1Gbps	0	1	0	1	—	—	0	10
RGMII 100 Mbps	0	1	0	1	1	—	0	01
RGMII 10 Mbps	0	1	0	1	0	—	0	01
MII 10/100 Mbps	0	0	0	0	—	0	0	01
RMII 100 Mbps	0	0	0	0	1	1	0	01
RMII 10 Mbps	0	0	0	0	0	1	0	01
SGMII 1 Gbps	0	0	1	0	—	—	1	10

Table 15-11. eTSEC Interface Configurations (continued)

Interface Mode	ECNTRL Field							MACCFG2 Field
	FIFM	GMIIM	TBIM	RPM	R100M	RMM	SGMIIM	I/F Mode
SGMII 100 Mbps	0	0	1	0	1	—	1	01
SGMII 10 Mbps	0	0	1	0	0	—	1	01

[1] See MII 10/100 Mbps mode for GMII 10/100 Mbps 'fall-back' mode.

15.5.3.1.7 Pause Time Value Register (PTV)

PTV is a 32-bit register written by the user to store the pause duration used when the eTSEC initiates an IEEE 802.3 PAUSE control frame via TCTRL[TFC_PAUSE]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of *pause_quanta*, equal to 512 bit times. The pause time can range from 0 to 65,535 *pause_quanta*, or 0 to 33,553,920 bit times. See Section 15.6.3.9, "Flow Control," for additional details. Figure 15-8 describes the definition for the PTV register.

Offset eTSEC1:0x2_4028; eTSEC2:0x2_5028; eTSEC3:0x2_6028; eTSEC4:0x2_7028 Access: Read/Write

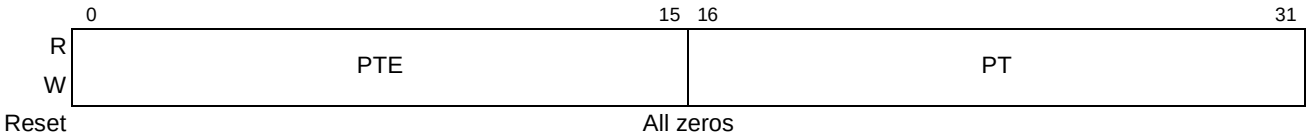


Figure 15-8. PTV Register Definition

Table 15-12 describes the fields of the PTV register.

Table 15-12. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 PAUSE frame format requires this parameter to be cleared.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (for example, 512 bit times). This pause value is used as part of the PAUSE frame to be sent when TCTRL[TFC_PAUSE] is set. See Section 15.6.3.9, "Flow Control," on page 15-175 for more information.

15.5.3.1.8 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. Figure 15-9 describes the definition for the DMACTRL register.

Offset eTSEC1:0x2_402C; eTSEC2:0x2_502C; eTSEC3:0x2_602C; eTSEC4:0x2_702C — Access: Read/Write

	0–15	16	17–23	24	25	26	27	28	29	30	31
R W	—	LE	—	TDSEN	TBDSEN	—	GRS	GTS	TOD	WWR	WOP
Reset	All zeros										

Figure 15-9. DMACTRL Register

Table 15-13 describes the fields of the DMACTRL register.

Table 15-13. DMACTRL Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	LE	Little-endian descriptor mode enable. This bit controls both the reading and writing of descriptors; data buffers are always transferred in network byte order. 0 RxBDs and TxBDs are interpreted with big-endian byte ordering, as shown in Section 15.6.8.1, “Data Buffer Descriptors.” 1 RxBDs and TxBDs are interpreted with little-endian byte ordering. That is, the 16 bits of flags are considered a complete half-word unit, the buffer length is considered another complete half-word unit, and the buffer pointer is considered a complete word unit.
17–23	—	Reserved
24	TDSEN	Tx Data snoop enable. 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable. 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received. (That is, after a valid end of frame was received). The contents of the Rx FIFO are then written to memory, and the IEVENT[GRSC] is set to indicate that all current receive buffers have been closed. Because the receive enable bit of the MAC may still be set, the MAC may continue to receive but the eTSEC ignores the receive data until GRS is cleared. If this bit is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the eTSEC hardware without fear of conflict. 0 eTSEC scans input data stream for valid frame. 1 eTSEC stops receiving frames following completion of current frame.

Table 15-13. DMACTRL Field Descriptions (continued)

Bits	Name	Description
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after all frames that are currently in the Tx FIFO or scheduled have been transmitted, and the GTSC interrupt in the IEVENT register is asserted. A frame that has started reading buffer descriptors or data from memory is read to completion and transmitted before the GTSC interrupt occurs. However, if no frame has been scheduled for transmission and the Tx FIFO is empty, the GTSC interrupt is asserted immediately. Once transmission has completed, clearing GTS “restarts” transmit. 0 Controller continues. 1 Controller stops transmission after completion of current frame.
29	TOD	Transmit on demand for TxBD ring 0. This bit is applicable only to the transmitter, and requires both TCTRL[TXSCHED] = 00 and DMACTRL[WOP] = 0. If 1 is written to this bit, the eTSEC immediately begins fetching the next TxBD from ring 0, avoiding waiting the normal polling time to check the TxBD’s R bit. This bit is always read as 0. 0 eTSEC continues waiting for the TxBD ring 0 poll timer to expire. 1 eTSEC immediately fetches a new TxBD from ring 0, and resets the poll timer.
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, XFUN, LC, CRL, RXB, RXF, the eTSEC waits for acknowledgement from system that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll for TxBD ring 0. This bit, which is applicable only to the transmitter and when TCTRL[TXSCHED] = 00, provides the user the option for the eTSEC to periodically poll TxBDs or to wait for software to tell eTSEC to fetch a buffer descriptor. While operating in the “Wait” mode, the eTSEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven. To resume transmission, software must clear TSTAT[THLT]. 0 Poll TxBD on ring 0 every 512 serial clocks. 1 Do not poll, but wait for TSTAT[THLT] to be cleared by the user.

15.5.3.1.9 TBI Physical Address Register (TBIPA)

The TBIPA, shown in Figure 15-10, is writable by the user to assign a physical address to the TBI for MII management configuration. The TBI registers are accessed at the offset of TBIPA. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 15.5.4, “Ten-Bit Interface (TBI).”

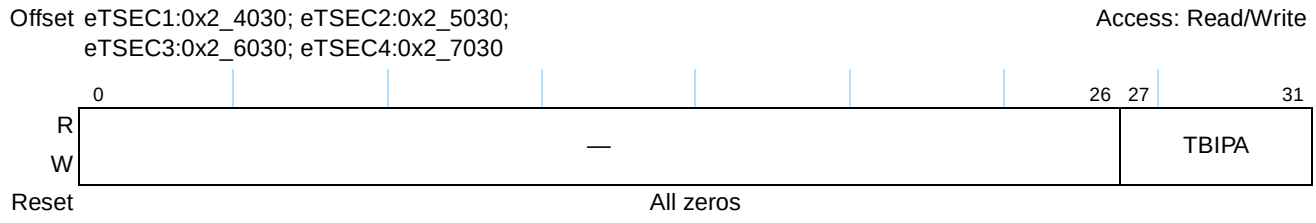


Figure 15-10. TBIPA Register Definition

Table 15-14 describes the fields of the TBIPA register.

Table 15-14. TBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	TBIPA	This field is used to program the PHY address of the ten-bit interface's MII management bus. To access the TBI register the user must write the TBIPA value to the MIIMADD [PHY Address] register located in the MAC register section. PHY Address 0 is reserved. Refer to Section 15.5.3.5.8, "MII Management Address Register (MIIMADD)."

15.5.3.2 eTSEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

15.5.3.2.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. Figure 15-11 describes the TCTRL register.

Offset eTSEC1:0x2_4100; eTSEC2:0x2_5100; eTSEC3:0x2_6100; eTSEC4:0x2_7100 — Access: Mixed

	0–16	17	18	19	20	21–26	27	28	29	30	31
R	—	IPCSEN	TUCSEN	VLINS	THDF	—	RFC_PAUSE	TFC_PAUSE	TXSCHED	—	
W											
Reset	All zeros										

Figure 15-11. TCTRL Register Definition

Table 15-15 describes the fields of the TCTRL register.

Table 15-15. TCTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	IPCSEN	IP header checksum generation enable. When set, the eTSEC offloads IPv4 header checksum generation. See Section 15.6.4.2, "Transmit Path Off-Load and Tx PTP Packet Parsing," on page 15-182. 0 IP header checksum generation is disabled even if enabled in a transmit frame control block. 1 IP header checksum generation is performed for IPv4 headers as determined by the settings in the current transmit frame control block.
18	TUCSEN	TCP/UDP header checksum generation enable. When set, the eTSEC offloads TCP or UDP header checksum generation. See Section 15.6.4.2, "Transmit Path Off-Load and Tx PTP Packet Parsing," on page 15-182. 0 TCP or UDP header checksum generation is disabled even if enabled in a transmit frame control block. 1 TCP or UDP header checksum generation is performed as determined by the settings in the current transmit frame control block.
19	VLINS	VLAN (IEEE 802.1Q) tag insertion enable. Applicable only for transmission via the Ethernet MAC. 0 Do not insert a VLAN tag into the frame. 1 Insert a VLAN tag into the frame. If the frame FCB has a valid VLAN field, use the FCB to source the VLAN control word, otherwise take the default VLAN control word from register DFVLAN.

Table 15-15. TCTRL Field Descriptions (continued)

Bits	Name	Description
20	THDF	Transmit half-duplex flow control under software control for 10-/100-Mbps half-duplex media. This bit is not self-resetting. 0 Disable back pressure 1 Back pressure is applied to media by raising carrier
21–26	—	Reserved
27	RFC_PAUSE	Receive flow control pause frame (written by the eTSEC). This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete. 1 Flow control pause frame received.
28	TFC_PAUSE	Transmit flow control pause frame. Set this bit to transmit a PAUSE frame. If this bit is set, the MAC stops transmission of data frames after the currently transmitting frame completes. Next, the MAC transmits a pause control frame with the duration value obtained from the PTV register. The TXC event occurs after sending the pause control frame. Finally, the controller clears TFC_PAUSE and resumes transmitting data frames as before. Note that pause control frames can still be transmitted if the Tx controller is stopped due to user assertion of DMACTRL[GTS] or reception of a PAUSE frame. 0 No request for Tx PAUSE frame pending or transmission complete. 1 Software request for Tx PAUSE frame pending.
29–30	TXSCHED	Transmit ring scheduling algorithm. This field determines which scheme the transmit scheduler uses to arbitrate between the enabled TxBD rings. The scheme chosen also controls how the DMACTRL and TQUEUE bits are interpreted. Ring polling is supported only by mode 00; the other modes require software to restart rings with the TSTAT register. TCP/IP offload can be enabled with any scheduling mode. 00 Single polled ring mode. TxBD ring 0 is the only ring serviced, even if other rings are enabled and ready. In this scheduler mode, the DMACTRL[WOP] and DMACTRL[TOD] bits control polling and retry behavior. This mode supports ring polling, and allows fetching of a non-ready TxBD to be retried twice. 01 Priority scheduling mode. All enabled TxBD rings are serviced in ascending ring index order. Once a non-ready TxBD has been fetched from the lowest-numbered ring, the eTSEC attempts to fetch TxBDs from the next enabled ring having a higher index, until transmission stops for lack of data. TSTAT records whenever a TxBD ring is exhausted. 10 Modified weighted round-robin scheduling mode. Each TxBD ring is polled in sequence for frames that are ready for transmission. If a non-ready TxBD is fetched from a ring, that ring is removed from the scheduling pool until software re-enables it. Ready frames are repeatedly transmitted from a chosen ring until its transmission quota is exhausted. The transmission quota for TxBD ring *n* is set to WT*n* × 64 bytes, where WT*n* is a weight from the TR03WT/TR47WT registers. If a ring transmits more data than its quota allows, the excess is deducted from its quota on the next transmission opportunity, thereby preventing large frames from monopolizing the eTSEC bandwidth. 11 Reserved
31	—	Reserved

15.5.3.2.2 Transmit Status Register (TSTAT)

This register is read/write-one-to-clear and is written by the eTSEC to convey DMA status information for each TxBD ring. The halt bit only has meaning for enabled rings. After processing transmit-related interrupts, software should use TSTAT to restart transmission from rings that may have been affected by the interrupt condition. In particular, an error condition that prevents eTSEC from continuing transmission

halts DMA from all rings, including the ring that gave rise to the error. Figure 15-12 describes the TSTAT register.

Offset eTSEC1:0x2_4104; eTSEC2:0x2_5104; eTSEC3:0x2_6104; eTSEC4:0x2_7104 Access: w1c

	0	1	2	3	4	5	6	7	8–15
R	THLT0	THLT1	THLT2	THLT3	THLT4	THLT5	THLT6	THLT7	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros								

	16	17	18	19	20	21	22	23	24–31
R	TXF0	TXF1	TXF2	TXF3	TXF4	TXF5	TXF6	TXF7	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros								

Figure 15-12. TSTAT Register Definition

Table 15-16 describes the fields of the TSTAT register.

Table 15-16. TSTAT Field Descriptions

Bits	Name	Description
0	THLT0	Transmit halt of ring 0. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN0], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
1	THLT1	Transmit halt of ring 1. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN1], or if no ready TxBDs can be fetched.DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
2	THLT2	Transmit halt of ring 2. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN2], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
3	THLT3	Transmit halt of ring 3. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN3], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
4	THLT4	Transmit halt of ring 4. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN4], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
5	THLT5	Transmit halt of ring 5. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN5], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
6	THLT6	Transmit halt of ring 6. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN6], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
7	THLT7	Transmit halt of ring 7. Set by the eTSEC if is no longer processing transmit frames from this TxBD ring, and DMA from this ring is disabled. To re-start transmission from this TxBD ring, this bit must be cleared by writing 1 to it. This bit is set only on a general error condition (as in IEVENT[TXE]), regardless of TQUEUE[EN7], or if no ready TxBDs can be fetched. DMACTRL[GTS] being set by the user does not cause this bit to be set. Software should examine the halted queue's buffer descriptors for repeatable error conditions before taking it out of the halt state. Failure to do so may cause an effective livelock, in which the error condition recurs and halts all queues again. Repeatable error conditions which cause halt include: Bus error: • Invalid BD or data address • Uncorrectable error on BD or data read TxBD programming errors: • Ready=1 and length=0
8–15	—	Reserved
16	TXF0	Transmit frame event occurred on ring 0. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
17	TXF1	Transmit frame event occurred on ring 1. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
18	TXF2	Transmit frame event occurred on ring 2. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
19	TXF3	Transmit frame event occurred on ring 3. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.

Table 15-16. TSTAT Field Descriptions (continued)

Bits	Name	Description
20	TXF4	Transmit frame event occurred on ring 4. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
21	TXF5	Transmit frame event occurred on ring 5. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
22	TXF6	Transmit frame event occurred on ring 6. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
23	TXF7	Transmit frame event occurred on ring 7. Set by the eTSEC if IEVENT[TXF] was set in relation to transmitting a frame from this ring.
24–31	—	Reserved

15.5.3.2.3 Default VLAN Control Word Register (DFVLAN)

This register defines the default value for the VLAN Ethertype and control word when VLAN tags are automatically inserted by the eTSEC, and no per-frame VLAN data is supplied by software. On receive, this register defines a customizable VLAN Ethertype for automatic deletion. Note that an Ethertype of 0x8808 (Control Word) is not permitted as a custom VLAN tag. Frames with an Ethertype of 0x8808 are dropped by the receiver unless RCNTRL[CFA] is set. In the case of frames containing stacked VLAN tags, this register defines the tag associated with the outer or metropolitan area VLAN. Figure 15-13 describes the DFVLAN register.

Offset eTSEC1:0x2_4108; eTSEC2:0x2_5108; eTSEC3:0x2_6108; eTSEC4:0x2_7108 — Access: Read/Write

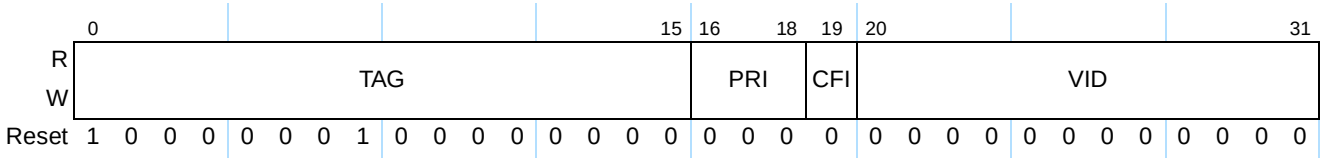


Figure 15-13. DFVLAN Register Definition

Table 15-17 describes the fields of the DFVLAN register.

Table 15-17. DFVLAN Field Descriptions

Bits	Name	Description
0–15	TAG	This is the default Ethertype used to tag VLAN frames. On transmit, this tag is inserted ahead of the VLAN control word; TAG should be set to 0x8100 for IEEE 802.1Q VLAN. On receive, an Ethertype matching TAG or an Ethertype of 0x8100 marks a VLAN-tagged frame. Note that if using DFVLAN to set a custom ethertype (that is, using a value other than 0x8100), packets received with a custom tag are not counted by any of the RMON counters. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF.
16–18	PRI	This is the default value used for the IEEE 802.1p frame priority.
19	CFI	This is the default value used for the IEEE 802.1Q canonical format indicator.
20–31	VID	This is the default value used for the virtual-LAN identifier in VLAN-tagged frames. A value of zero is defined as the null VLAN, however field PRI may be still set independently.

15.5.3.2.4 Transmit Interrupt Coalescing Register (TXIC)

The TXIC register enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Figure 15-14 describes the definition for the TXIC register.

Offset eTSEC1:0x2_4110; eTSEC2:0x2_5110; eTSEC3:0x2_6110; eTSEC4:0x2_7110 — Access: Read/Write

	0	1	2	3 – 10	11 – 15	16 – 31
R W	ICEN	ICCS	—	ICFT	—	ICTT
Reset	All zeros					

Figure 15-14. TXIC Register Definition

Table 15-18 describes the fields of the TXIC register.

Table 15-18. TXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by TXIC[ICFT]) or when the threshold timer expires (determined by TXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Tx interface clocks (TSEC*n*_GTX_CLK). 1 The coalescing timer advances count every 64 system clocks[1]. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines how many frames are transmitted before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero to avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines the maximum amount of time after transmitting a frame before raising an interrupt. If frames have been transmitted but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon transmission of the first frame having its TxBD[I] bit set. The threshold value is represented in units of 64 clock periods as specified by the timer clock source (TXIC[ICCS[). The value of ICTT must be greater than zero to avoid unpredictable behavior.

[1] The term ‘system clock’ refers to CCB clock/2.

15.5.3.2.5 Transmit Queue Control Register (TQUEUE)

The TQUEUE register, shown in Figure 15-15, selectively enables each of the TxBD rings 0–7. By default, TxBD ring 0 is enabled.

Offset eTSEC1:0x2_4114; eTSEC2:0x2_5114; eTSEC3:0x2_6114; eTSEC4:0x2_7114 — Access: Read/Write

	0–15	16	17	18	19	20	21	22	23	24–31
R W	—	EN0	EN1	EN2	EN3	EN4	EN5	EN6	EN7	—
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0

Figure 15-15. TQUEUE Register Definition

Table 15-19 describes the TQUEUE register.

Table 15-19. TQUEUE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	EN0	Transmit queue 0 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
17	EN1	Transmit queue 1 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
18	EN2	Transmit queue 2 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
19	EN3	Transmit queue 3 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
20	EN4	Transmit queue 4 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
21	EN5	Transmit queue 5 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
22	EN6	Transmit queue 6 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
23	EN7	Transmit queue 7 enable. 0 TxBD ring is not queried for transmission. In effect the transmit queue is disabled. 1 TxBD ring is queried for transmission.
24–31	—	Reserved

15.5.3.2.6 TxBD Ring 0–3 Weighting Register (TR03WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHED] = 10), this register determines the weighting applied to each transmit queue for queues 0 to 3. For priority-based scheduling, TR03WT has no effect. A description of how queue weights affect eTSEC's round-robin algorithm appears in Section 15.6.5.2.2, "Modified Weighted Round-Robin Queuing (MWRR)." Figure 15-16 describes the TR03WT register.

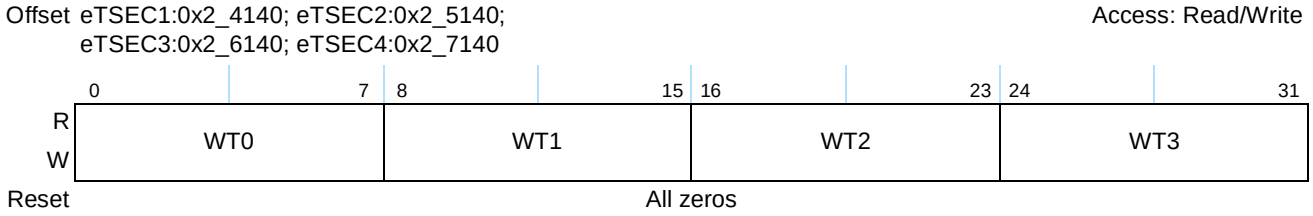


Figure 15-16. TR03WT Register Definition

Table 15-20 describes the fields of the TR03WT register.

Table 15-20. TR03WT Field Descriptions

Bits	Name	Description
0–7	WT0	Weighting value for TxBD ring 0 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT0 × 64 bytes of data are scheduled for transmission from TxBD ring 0. Clearing this field prevents transmission.
8–15	WT1	Weighting value for TxBD ring 1 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT1 × 64 bytes of data are scheduled for transmission from TxBD ring 1. Clearing this field prevents transmission.
16–23	WT2	Weighting value for TxBD ring 2 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT2 × 64 bytes of data are scheduled for transmission from TxBD ring 2. Clearing this field prevents transmission.
24–31	WT3	Weighting value for TxBD ring 3 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT3 × 64 bytes of data are scheduled for transmission from TxBD ring 3. Clearing this field prevents transmission.

15.5.3.2.7 TxBD Ring 4–7 Weighting Register (TR47WT)

When modified weighted round-robin Tx scheduling is enabled (TCTRL[TXSCHED] = 10), this register determines the weighting applied to each enabled transmit queue for queues 4 to 7. For priority-based scheduling, TR47WT has no effect. A description of how queue weights affect eTSEC's modified weighted round-robin algorithm appears in Section 15.6.5.2.2, "Modified Weighted Round-Robin Queuing (MWRR)." Figure 15-17 describes the definition for the TR47WT register.

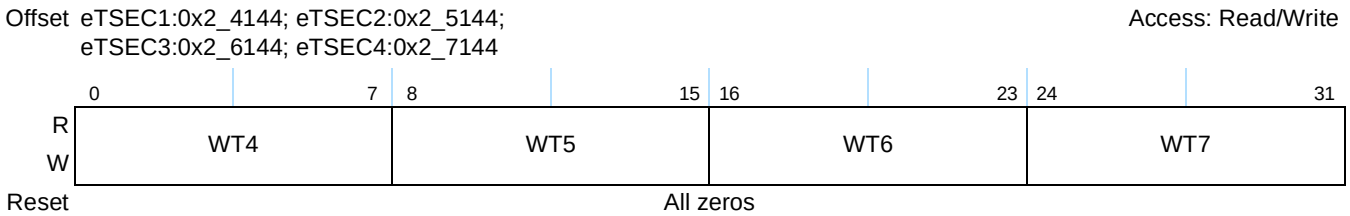


Figure 15-17. TR47WT Register Definition

Table 15-21 describes the fields of the TR47WT register.

Table 15-21. TR47WT Field Descriptions

Bits	Name	Description
0–7	WT4	Weighting value for TxBD ring 4 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT4 × 64 bytes of data are scheduled for transmission from TxBD ring 4. Clearing this field prevents transmission.
8–15	WT5	Weighting value for TxBD ring 5 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT5 × 64 bytes of data are scheduled for transmission from TxBD ring 5. Clearing this field prevents transmission.
16–23	WT6	Weighting value for TxBD ring 6 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT6 × 64 bytes of data are scheduled for transmission from TxBD ring 6. Clearing this field prevents transmission.
24–31	WT7	Weighting value for TxBD ring 7 when TCTRL[TXSCHED] = 10. On each round of the Tx scheduler, a minimum of WT7 × 64 bytes of data are scheduled for transmission from TxBD ring 7. Clearing this field prevents transmission.

15.5.3.2.8 Transmit Data Buffer Pointer High Register (TBDBPH)

The TBDBPH register is written by the user with the most significant address bits common to all TxBD buffer addresses, TxBD[Data Buffer Pointer]. As a consequence, all Tx buffers must be placed in a 4 gigabyte segment of memory whose base address is prefixed by the bits in TBDBPH. The TxBD ring itself can reside in a different memory region (based at TBASEH). Figure 15-18 describes the definition for the TBDBPH register.

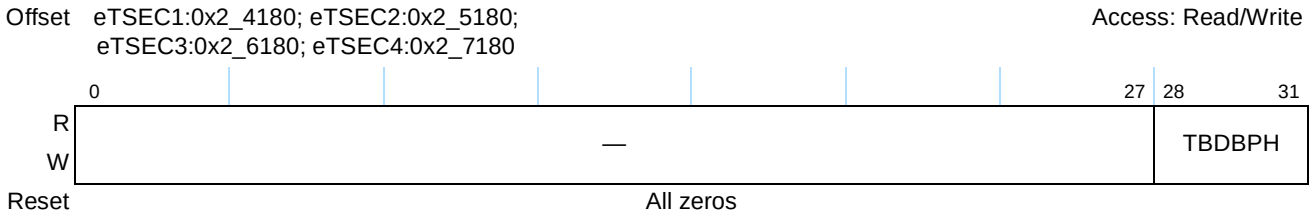


Figure 15-18. TBDBPH Register Definition

Table 15-24 describes the fields of the TBDBPH register.

Table 15-22. TBDBPH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TBDBPH	Most significant bits common to all data buffer addresses contained in TxBDs. The user must initialize TBDBPH before enabling the eTSEC transmit function.

15.5.3.2.9 Transmit Buffer Descriptor Pointers 0–7 (TBPTR0–TBPTR7)

TBPTR0–TBPTR7 each contains the low-order 32 bits of the next transmit buffer descriptor address for their respective TxBD ring. Figure 15-19 describes the TBPTR registers. These registers takes on the value of their ring's associated TBASE when the TBASE register is written by software. Software must not write TBPTR0–TBPTR7 while eTSEC is actively transmitting frames. However, TBPTR0– TBPTR7 can be

modified when the transmitter is disabled or when no Tx buffer is in use (after a GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission) in order to change the next TxBD eTSEC transmits.

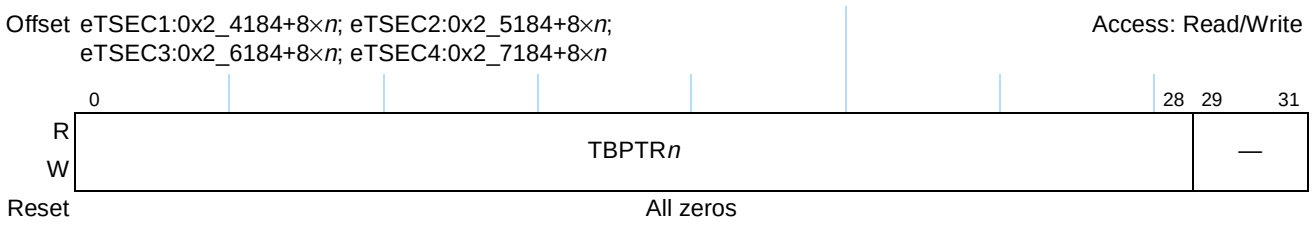


Figure 15-19. TBPTR0–TBPTR7 Register Definition

Table 15-23 describes the fields of the TBPTR*n* register.

Table 15-23. TBPTR*n* Field Descriptions

Bits	Name	Description
0–28	TBPTR*n*	Current TxBD pointer for TxBD ring *n*. Points to the current BD being processed or to the next BD the transmitter uses when it is idling. When the end of the TxBD ring is reached, eTSEC initializes TBPTR*n* to the value in the corresponding TBASE*n*. The TBPTR register is internally written by the eTSEC's DMA controller during transmission. The pointer increments by eight (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the three least significant bits of this register are read-only and zero. After an error condition, the eTSEC returns TBPTR*n* to point to the first BD of the frame partially transmitted.
29–31	—	Reserved

15.5.3.2.10 Transmit Descriptor Base Address High Register (TBASEH)

The TBASEH register is written by the user with the most significant address bits common to all TxBD addresses, including TBASE0–TBASE7 and TBPTR0–TBPTR7. As a consequence, all TxBD rings must be placed in a 4-Gbyte segment of memory whose base address is prefixed by the bits in TBASEH. Data buffers are located in a potentially different region, based at TBDBPH. Figure 15-20 describes the TBASEH register.

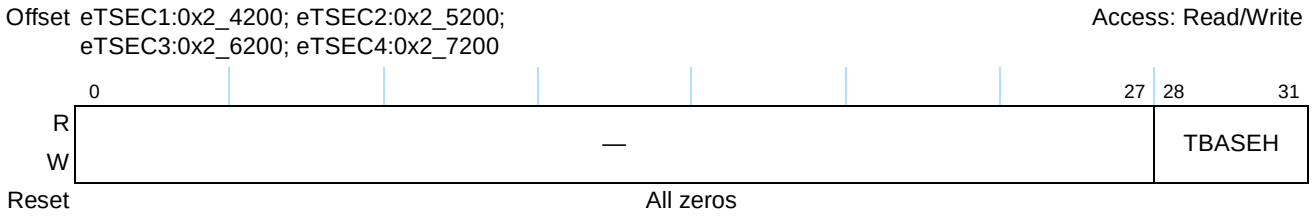


Figure 15-20. TBASEH Register Definition

Table 15-24 describes the fields of the TBASEH register.

Table 15-24. TBASEH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TBASEH	Most significant bits common to all TxBD addresses—except data buffer pointers. The user must initialize TBASEH before enabling the eTSEC transmit function.

15.5.3.2.11 Transmit Descriptor Base Address Registers (TBASE0–TBASE7)

The TBASE*n* registers are written by the user with the base address of each TxBD ring *n*. Each such value must be divisible by eight, since the three least significant bits always write as 000. Figure 15-21 describes the definition for the TBASE*n* registers.

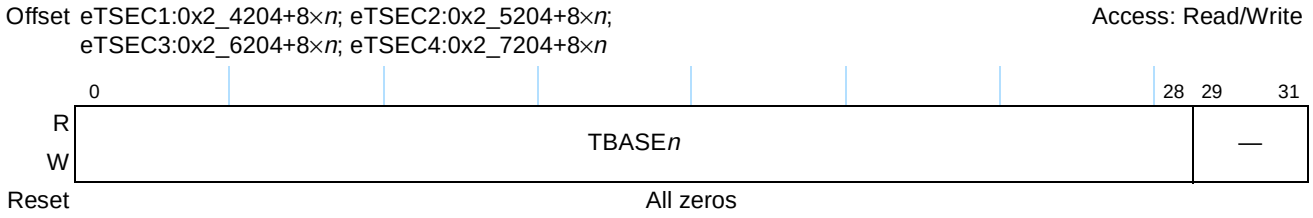


Figure 15-21. TBASE Register Definition

Table 15-25 describes the fields of the TBASE*n* registers.

Table 15-25. TBASE0–TBASE7 Field Descriptions

Bits	Name	Description
0–28	TBASE*n*	Transmit base for ring *n*. TBASE defines the starting location in the memory map for the eTSEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the eTSEC transmit function on the associated ring.
29–31	—	Reserved

15.5.3.2.12 Transmit Time Stamp Identification Register (TMR_TXTS1–2_ID)

Transmit time stamp identification register (TMR_TXTS*n*_ID). This register holds the identification number of the transmitted frame corresponding to the timestamp captured in TMR_TXTS*n*_H/L. Each time the eTSEC is instructed to capture the timestamp of an outgoing frame via TxFCB[PTP] the associated field in TxFCB[PTP_ID] is stored in this register, overwriting the previous value.

This register is read only in normal operation. Figure 15-22 describes the definition for the TMR_TXTS*n*_ID register.

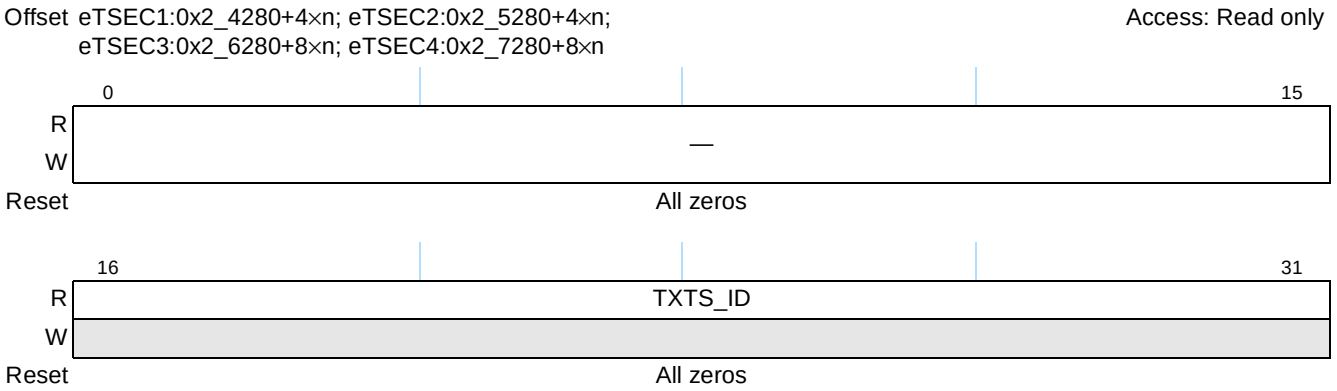


Figure 15-22. TMR_TXTS*n*_ID Register Definition

Table 15-26 describes the fields of the TMR_TXTS*n*_ID register.

Table 15-26. TMR_TXTS*n*_ID Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TXTS_ID	Tx time stamp identification field

15.5.3.2.13 Transmit Time Stamp Register (TMR_TXTS1–2_H/L)

Transmit stamp register (TMR_TXTS*n*_H/L). This register holds the value of the TMR_CNT_H/L when a frame tagged for timestamp capture (via Tx FCB[PTP]) is transmitted. Upon transmission of the start of frame symbol of such a frame, the value in TMR_CNT_H/L is copied into TMR_TXTS*n*_H/L.

This register is read only in normal operation. Figure 15-23 depicts TMR_TXTS*n*_H/L.

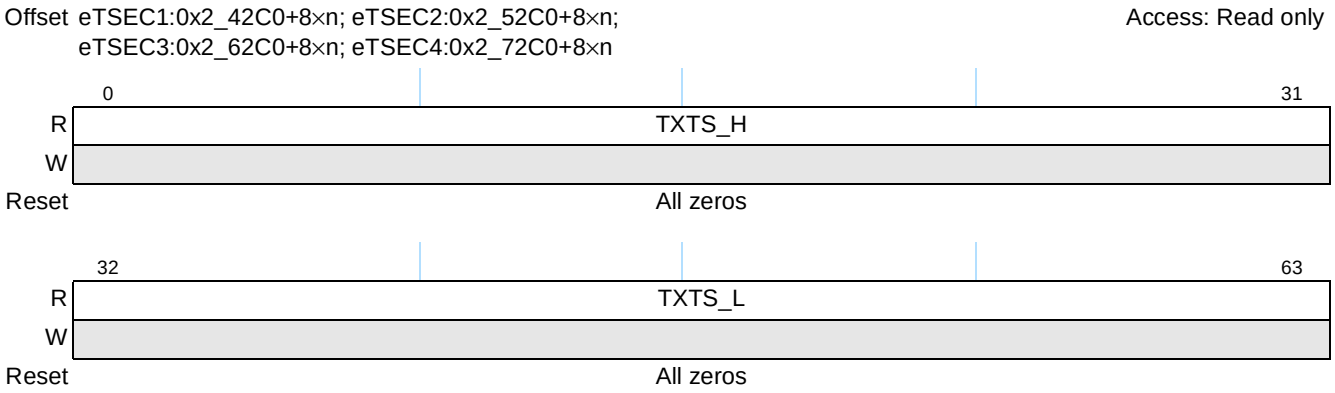


Figure 15-23. TMR_TXTS*n*_H/L Register Definition

Table 15-27 describes the fields of the TMR_TXTS*n*_H/L register.

Table 15-27. TMR_TXTS*n*_H/L Register Field Descriptions

Bits	Name	Description
0–63	TXTS_H/L	Time stamp field of the transmitted PTP packet's start of frame detection.

15.5.3.3 eTSEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

15.5.3.3.1 Receive Control Register (RCTRL)

The RCTRL register is programmed by the user and controls the operational mode of the receiver. It must be written only after a system reset (at initialization) or after a graceful receive stop has completed. Figure 15-24 describes the RCTRL register.

Offset eTSEC1:0x2_4300; eTSEC2:0x2_5300; eTSEC3:0x2_6300; eTSEC4:0x2_7300 Access: Read/Write

	0	7	8	10	11	15
R W	L2OFF	TS	—		PAL	
Reset	All zeros					

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CFA	LFC	VLEX	FILREN	FSQEN	GHTX	IPCSEN	TUCSEN	PRSDEP		PRSFM	BC_REJ	PROM	RSF	EMEN	—
Reset	All zeros															

Figure 15-24. RCTRL Register Definition

Table 15-28 describes the fields of the RCTRL register.

Table 15-28. RCTRL Field Descriptions

Bits	Name	Description
0–6	L2OFF	Layer 2 offset. The number of octet pairs from the start of the frame that the parser should expect to see before the first byte of the Ethernet DA. For frames received over Ethernet, the start of frame is regarded as the SFD symbol. For packets received through the FIFO packet interface the start of frame is regarded as the first octet of receive data. The user may think of this value as representing the length - in multiples of two bytes - of a ‘shim’ header that is inserted between the SFD and DA. By writing to RCTRL with a mask of 0xFE00_0000, the even byte length restriction is guaranteed. For normal frames, this field should be left as 0.
7	TS	Time stamp incoming packets as padding bytes. PAL field is set to 8 if the PAL field is programmed to less than 8. Must be set to zero if TMR_CTRL[TE]=0.
8–10	—	Reserved
11–15	PAL	Packet alignment padding length. If not zero, PAL (1–31) bytes of zero padding are inserted before the start of each received frame, but following the RxFCB if TOE is enabled. For Ethernet where optional preamble extraction is enabled, the padding appears before the preamble, otherwise the padding precedes the layer 2 header. The value of PAL can be set so that the start of the IP header in the receive data buffer is aligned to a 32-bit boundary. Normally, setting PAL = 2 provides minimal padding to ensure such alignment of the IP header. Note that the minimum zero padding value for this field should be PAL–8 if the TS field is set and 0 when PAL is < 8.
16	CFA	Control frame accept. Overrides 802.3 standard control frame behavior. When set, all ethernet frames that have an ethertype of 0x8808 are treated as normal ethernet frames and passed up to the stack on a DA match. This bit has no effect on frames received over the FIFO packet interface. 0 Control frames are sunk in the MAC layer and not passed up the stack. 1 Ignore 802.3 specified behavior and DMA received frames to memory on a DA match.

Table 15-28. RCTRL Field Descriptions (continued)

Bits	Name	Description
17	LFC	Lossless flow control. When set, the eTSEC determines the number of free BDs (via RQPARM*n*[LEN] and RBTPTR*n*) in each active ring. Should the free BD count in an active ring drop below its setting for RQPARM*n*[FBTHR], the eTSEC asserts link layer flow control. For full-duplex ethernet connections, the eTSEC emits a pause frame as if TCTRL[TFC_PAUSE] was set. For FIFO packet interface connections, the RFC signal is asserted. 0 Disabled. This is the default 1 Enabled, calculate the free BDs in each active ring and assert link layer flow control if required.
18	VLEX	Enable automatic VLAN tag extraction and deletion from Ethernet frames. 0 Do not delete VLAN tags from received Ethernet frames. 1 If a VLAN tag is seen after the Ethernet source address, and PRSDEP is non-zero, delete the VLAN tag and return the VLAN control word in the frame control block returned with this frame. Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.)
19	FILREN	Filer enable. When set, the receive frame filer is enabled. This files accepted frames to a particular RxBD ring according to rules defined in the filer table. In this case, PRSDEP must not be cleared. 0 Do not search the receive queue filer table for received frames. All received frames are sent to RxBD ring 0 by default. 1 Search the receive queue filer table for received frames, and let the filer determine the index of the RxBD ring for each frame. Note that if PRSDEP is cleared, FILREN must be cleared as well.
20	FSQEN	Enable single-queue mode for the receive frame filer. This bit is ignored unless FILREN is also set. 0 The filer chooses the RxBD ring using the least significant bits of the virtual queue ID as a ring index. 1 The filer always attempts to file received frames to ring 0, regardless of virtual queue ID. This mode is intended for operating the filer as a packet classification engine.
21	GHTX	Group address hash table extend. By default, the group address hash table is 256 entries (as defined by registers GADDR0–GADDR7); registers IGADDR0–IGADDR7 are then used to define the individual address hash table. When this bit is set, the hash table is extended to a total of 512 entries (IGADDR0–IGADDR7 are then the first 256 entries of the extended 512-entry group address hash table). 0 Both the individual and group hash functions are the 8 MSBs of the CRC-32 of the Ethernet destination address. 1 The group hash function is the 9 MSBs of the CRC-32 of the Ethernet destination address. The individual address hash function is unavailable.
22	IPCSEN	IP Checksum verification enable. See Section 15.6.4.3, “Receive Path Off-Load.” 0 IPv4 header checksums are not verified by the eTSEC—even if layer 3 parsing is enabled. 1 Perform IPv4 header checksum verification if PRSDEP > 01.
23	TUCSEN	TCP or UDP Checksum verification enable. See Section 15.6.4.3, “Receive Path Off-Load.” 0 TCP or UDP checksums are not verified by the eTSEC—even if layer 4 parsing is enabled. 1 Perform TCP or UDP checksum verification if PRSDEP = 11.

Table 15-28. RCTRL Field Descriptions (continued)

Bits	Name	Description
24–25	PRSDEP	Parser control. The level of parser layer recognition is determined as follows: 00 Parser disabled. Receive frame filer must also be disabled by clearing RCTRL[FILREN]. This should be the setting for raw (non-IP) packets received over a FIFO interface. 01 Only L2 (Ethernet) protocols are recognized. For packets received over a FIFO interface, this parse level is available only if RCTRL[PRSFM]=1. 10 L2 and L3 (IP) protocols are recognized over any interface not configured as a FIFO interface. If RCTRL[PRSFM]=0, this encoding means L3 (IP) only protocols are recognized for packets received over a FIFO interface. If RCTRL[PRSFM]=1, this encoding means L2 and L3 (IP) protocols are recognized for packets received over a FIFO interface. 11 L2, L3, and L4 (TCP/UDP) protocols are recognized over any interface not configured as a FIFO interface. If RCTRL[PRSFM]=0, this encoding means L3 and L4 (TCP/UDP) protocols are recognized for packets received over a FIFO interface. If RCTRL[PRSDEP]=1, this encoding means L2, L3, and L4 (TCP/UDP) protocols are recognized for packets received over a FIFO interface. If this field is non-zero, a TOE frame control block is prepended to the received frame, and the first RxBD points to the FCB. Note that if PRSDEP is cleared, VLEX must be cleared as well. (VLAN tag extraction is only supported when the parser is enabled.) Also, if PRSDEP is cleared, FILREN must also be cleared.
26	PRSFM	FIFO-mode parsing 0 L2 parsing in FIFO mode is not available. Must be 0 for non-FIFO modes. 1 L2 parsing in FIFO mode is available
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD.
28	PROM	Promiscuous mode. All Ethernet frames, regardless of destination address, are accepted.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than 64 bytes. For packets received over the FIFO packet interface, this bit has no effect (packets shorter than 64 bytes are always accepted). 0 Ethernet frames less than 64B in length are silently dropped. 1 Frames more than 16B and less than 64B in length are accepted upon a DA match. Note that frames less than or equal to 16B in length are always silently dropped.
30	EMEN	Exact match MAC address enable. If this bit is set, the MAC01ADDR1–MAC15ADDR1 and MAC01ADDR2–MAC15ADDR2 registers are recognized as containing MAC addresses aliasing the MAC's station address. Setting this bit therefore allows eTSEC to receive Ethernet frames having a destination address matching one of these 15 addresses.
31	—	Reserved

15.5.3.3.2 Receive Status Register (RSTAT)

The eTSEC writes to this register under the following conditions:

- A frame interrupt event occurred on one or more RxBD rings
- The receiver runs out of descriptors due to a busy condition on a RxBD ring
- The receiver was halted because an error condition was encountered while receiving a frame

Writing 1 to any bit of this register clears it. Software should clear the QHLT bit to take eTSEC's receiver function out of halt state for the associated queue. Figure 15-25 describes the definition for the RSTAT register.

Offset eTSEC1:0x2_4304; eTSEC2:0x2_5304; eTSEC3:0x2_6304; eTSEC4:0x2_7304 Access: w1c

	0–7	8	9	10	11	12	13	14	15
R	—	QHLT0	QHLT1	QHLT2	QHLT3	QHLT4	QHLT5	QHLT6	QHLT7
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset		All zeros							

	16–23	24	25	26	27	28	29	30	31
R	—	RXF0	RXF1	RXF2	RXF3	RXF4	RXF5	RXF6	RXF7
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset		All zeros							

Figure 15-25. RSTAT Register Definition

Table 15-29 describes the fields of the RSTAT register.

Table 15-29. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT0	RxBD queue 0 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT0 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
9	QHLT1	RxBD queue 1 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT1 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
10	QHLT2	RxBD queue 2 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT2 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
11	QHLT3	RxBD queue 3 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT3 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
12	QHLT4	RxBD queue 4 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT4 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.

Table 15-29. RSTAT Field Descriptions (continued)

Bits	Name	Description
13	QHLT5	RxBD queue 5 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT5 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
14	QHLT6	RxBD queue 6 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT6 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
15	QHLT7	RxBD queue 7 is halted. It is a hardware-initiated stop indication. (DMACTRL[GRS] being set by the user does not cause a QHLT7 to be set.). The current frame and all other frames directed to a halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 This queue is enabled for reception. (That is, it is not halted) 1 All controller receive activity to this queue is halted.
16–23	—	Reserved
24	RXF0	Receive frame event occurred on ring 0. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
25	RXF1	Receive frame event occurred on ring 1. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
26	RXF2	Receive frame event occurred on ring 2. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
27	RXF3	Receive frame event occurred on ring 3. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
28	RXF4	Receive frame event occurred on ring 4. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
29	RXF5	Receive frame event occurred on ring 5. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
30	RXF6	Receive frame event occurred on ring 6. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.
31	RXF7	Receive frame event occurred on ring 7. Set by the eTSEC if IEVENT[RXF] was set in relation to receiving a frame to this ring.

15.5.3.3.3 Receive Interrupt Coalescing Register (RXIC)

The RXIC register enables and configures the operational parameters for interrupt coalescing associated with received frames. Figure 15-26 describes the RXIC register.

Offset eTSEC1:0x2_4310; eTSEC2:0x2_5310; eTSEC3:0x2_6310; eTSEC4:0x2_7310 Access: Read/Write

	0	1	2	3–10	11–15	16–31
R	ICEN	ICCS	—	ICFT	—	ICTT
W						
Reset	All zeros					

Figure 15-26. RXIC Register Definition

Table 15-30 describes the fields of the RXIC register.

Table 15-30. RXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received. 1 Interrupt coalescing is enabled. If the eTSEC receive frame interrupt is enabled (IMASK[RXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by RXIC[ICFT]) or when the threshold timer expires (determined by RXIC[ICTT]).
1	ICCS	Interrupt coalescing timer clock source. 0 The coalescing timer advances count every 64 eTSEC Rx interface clocks (TSECn_GTX_CLK). 1 The coalescing timer advances count every 64 system clocks[1]. This mode is recommended for FIFO operation.
2	—	Reserved
3–10	ICFT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines how many frames are received before raising an interrupt. The eTSEC threshold counter is reset to ICFT following an interrupt. The value of ICFT must be greater than zero avoid unpredictable behavior.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (RXIC[ICE] is set), this value determines the maximum amount of time after receiving a frame before raising an interrupt. If frames have been received but the frame count threshold has not been met, an interrupt is raised when the threshold timer reaches zero. The threshold timer is reset to the value in this field and begins counting down upon receiving the first frame having its RxBD[I] bit set. The threshold value is represented in units equal to 64 periods of the clock specified by RXIC[ICCS]. ICTT must be greater than zero to avoid unpredictable behavior.

[1] The term ‘system clock’ refers to CCB clock/2.

15.5.3.3.4 Receive Queue Control Register (RQUEUE)

The RQUEUE register enables each of the RxBD rings 0–7. By default, RxBD ring 0 is enabled. Figure 15-27 describes the definition for the RQUEUE register.

Offset eTSEC1:0x2_4314; eTSEC2:0x2_5314; eTSEC3:0x2_6314; eTSEC4:0x2_7314 Access: Read/Write

	0–7	8	9	10	11	12	13	14	15	16–23	24	25	26	27	28	29	30	31
R W	—	EX0	EX1	EX2	EX3	EX4	EX5	EX6	EX7	—	EN0	EN1	EN2	EN3	EN4	EN5	EN6	EN7
Reset	0 0 0 0 0 0 0 0	1	0	0	0	0	0	0	0	0 0 0 0 0 0 0 0	1	0	0	0	0	0	0	0

Figure 15-27. RQUEUE Register Definition

Table 15-31 describes the RQUEUE register.

Table 15-31. RQUEUE Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	EX0	Receive queue 0 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
9	EX1	Receive queue 1 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
10	EX2	Receive queue 2 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
11	EX3	Receive queue 3 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
12	EX4	Receive queue 4 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
13	EX5	Receive queue 5 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
14	EX6	Receive queue 6 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
15	EX7	Receive queue 7 extract enable. 0 Data transferred by DMA to this RxBD ring is not extracted to cache. 1 Data transferred by DMA to this RxBD ring undergoes extraction according to ATTR register.
16–23	—	Reserved
24	EN0	Receive queue 0 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.

Table 15-31. RQUEUE Field Descriptions (continued)

Bits	Name	Description
25	EN1	Receive queue 1 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
26	EN2	Receive queue 2 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
27	EN3	Receive queue 3 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
28	EN4	Receive queue 4 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
29	EN5	Receive queue 5 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
30	EN6	Receive queue 6 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.
31	EN7	Receive queue 7 enable. 0 RxBD ring is not queried for reception. In effect the receive queue is disabled. 1 RxBD ring is queried for reception.

15.5.3.3.5 Receive Bit Field Extract Control Register (RBIFX)

The RBIFX register provides a set of four 6-bit offsets for locating up to four octets in a received frame and passing them to the receive queue filer as the user-defined ARB property. Through RBIFX a custom ARB filer property can be constructed from arbitrary bytes, which allows frame filing on the basis of bitfields not ordinarily provided to the filer, such as bits from the Ethernet preamble or TCP flags. The value of property ARB is the concatenation of {B0, B1, B2, B3} to 32-bits, where B0–B3 are the bytes as defined by RBIFX.

Figure 15-28 describes the definition for the RBIFX register.Note: when the eTSEC is configured to receive frame via the FIFO packet interface, a value of B*n*CTL = 01 is not supported unless RCTRL[PRSFM]=1. In addition, the byte extraction level cannot exceed the parser depth: a value of BnCTL=10 requires RCTRL[PRSDEP]=1x and a value of BnCTL=11 requires RCTRL[PRSDEP]=11.

For values of BnCTL=10 or BCTL=11, the controller extracts the defined bytes even if it does not recognize the L3 or L4 header, respectively.

Offset eTSEC1:0x2_4330; eTSEC2:0x2_5330; eTSEC3:0x2_6330; eTSEC4:0x2_7330 Access: Read/Write

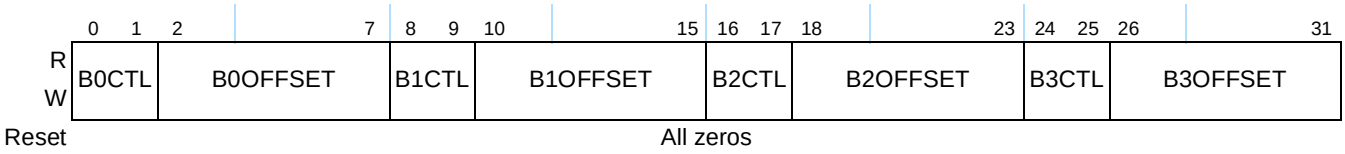


Figure 15-28. RBIFX Register Definition

Table 15-32 describes the RBIFX register.

Table 15-32. RBIFX Field Descriptions

Bits	Name	Description
0–1	B0CTL	Location of byte 0 of property ARB. 00 Byte 0 is not extracted, and appears as zero in property ARB. 01 Byte 0 is located in the received frame at offset (B0OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B0OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B0OFFSET bytes from the byte after the last byte of the layer 3 header.
2–7	B0OFFSET	Offset relative to the header defined by B0CTL that locates byte 0 of property ARB. An effective offset of zero points to the first byte of the specified header.
8–9	B1CTL	Location of byte 1 of property ARB. 00 Byte 1 is not extracted, and appears as zero in property ARB. 01 Byte 1 is located in the received frame at offset (B1OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B1OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B1OFFSET bytes from the byte after the last byte of the layer 3 header.
10–15	B1OFFSET	Offset relative to the header defined by B1CTL that locates byte 1 of property ARB. An effective offset of zero points to the first byte of the specified header.
16–17	B2CTL	Location of byte 2 of property ARB. 00 Byte 2 is not extracted, and appears as zero in property ARB. 01 Byte 2 is located in the received frame at offset (B2OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B2OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B2OFFSET bytes from the byte after the last byte of the layer 3 header.
18–23	B2OFFSET	Offset relative to the header defined by B2CTL that locates byte 2 of property ARB. An effective offset of zero points to the first byte of the specified header.

Table 15-32. RBIFX Field Descriptions (continued)

Bits	Name	Description
24–25	B3CTL	Location of byte 3 of property ARB. 00 Byte 3 is not extracted, and appears as zero in property ARB. 01 Byte 3 is located in the received frame at offset (B3OFFSET – 8) bytes from the first byte of the Ethernet DA. In non-FIFO modes, a negative effective offset points to bytes of the standard Ethernet preamble. Values of B3OFFSET less than 8 are reserved in FIFO modes. 10 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 2 header. 11 Byte 0 is located in the received frame at offset B3OFFSET bytes from the byte after the last byte of the layer 3 header.
26–31	B3OFFSET	Offset relative to the header defined by B3CTL that locates byte 3 of property ARB. An effective offset of zero points to the first byte of the specified header.

15.5.3.3.6 Receive Queue Filer Table Address Register (RQFAR)

RQFAR, shown in Figure 15-29, contains the index of the current, indirectly accessible entry of the received queue filer table. Each table entry occupies a pair of 32-bit words, denoted RQCTRL and RQPROP. To access the RQCTRL and RQPROP words of entry *n*, write *n* to RQFAR. Then read or write the indexed RQCTRL and RQPROP words by reading or writing the RQFCR and RQFPR registers, respectively.

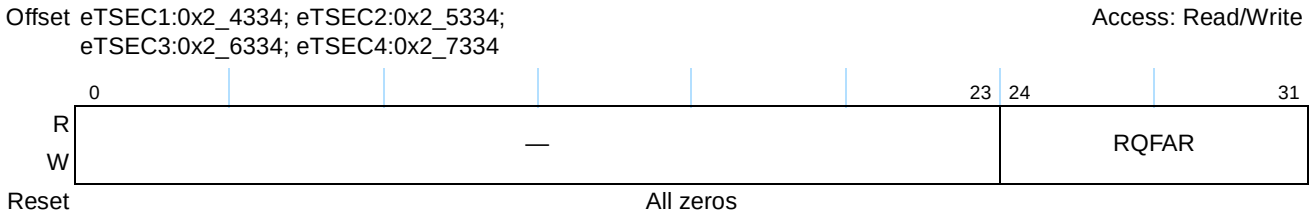


Figure 15-29. Receive Queue Filer Table Address Register Definition

Table 15-33 describes the fields of the RQFAR register.

Table 15-33. RQFAR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RQFAR	Current index of receive queue filer table, which spans a total of 256 entries.

15.5.3.3.7 Receive Queue Filer Table Control Register (RQFCR)

RQFCR is accessed to read or write the RQCTRL words in entries of the receive queue filer table. The table entries are described in greater detail in Section 15.6.5.1, "Receive Queue Filer." The word accessed via RQFCR is defined by the current value of RQFAR.

Figure 15-30 describes the definition for the RQFCR register.

Offset eTSEC1:0x2_4338; eTSEC2:0x2_5338; eTSEC3:0x2_6338; eTSEC4:0x2_7338 — Access: Read/Write

	0	1–15	16–21	22	23	24	25–26	27	28–31
R W	GPI	—	Q	CLE	REJ	AND	CMP	—	PID
Reset	All zeros								

Figure 15-30. Receive Queue Filer Table Control Register Definition

Table 15-34 describes the fields of the RQFCR register.

Table 15-34. RQFCR Field Descriptions

Bit	Name	Description
0	GPI	General purpose interrupt. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the filer instructs the Rx descriptor controller to set IEVENT[FGPI] when the corresponding receive frame is written to memory. If the timer is enabled (TMR_CTRL[TE] = 1), then TMR_PEVENT[RXP] is also set.
1–15	—	Reserved, should be written with zero.
16–21	Q	Receive queue index, from 0 to 63, inclusive, written into the Rx frame control block associated with the received frame. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the frame is sent to either RxBD ring 0 (if RCTRL[FSQEN] = 1) or the RxBD ring with index (Q mod 8) and the filing table search is terminated. In the case where RCTRL[FSQEN] = 0, 8 virtual receive queues are overlaid on every RxBD ring, and software needs to consult the RQ field of the Rx frame control block to determine which virtual receive queue was chosen.
22	CLE	Cluster entry/exit (used in combination with AND bit). This bit brackets clusters, marking the start and end entries of a cluster. Clusters cannot be nested. 0 Regular RQCTRL entry. 1 If entry matches and AND = 1, treat subsequent entries as belonging to a nested cluster and enter the cluster; otherwise skip all entries up to and including the next cluster exit. If AND = 0, exit current cluster.
23	REJ	Reject frame. This bit and its specified action are ignored if AND = 1. 0 If entry matches, accept frame and file it to RxBD ring Q. 1 If entry matches, reject frame and discard it, ignoring Q.
24	AND	Match this entry and the next entry as a pair. 0 Match property[PID] against RQPROP, independent of the next entry. 1 Match property[PID] against RQPROP. If matched and CLE = 0, attempt to match next entry, otherwise, skip all entries up to and including the entry with AND = 0. If matched and CLE = 1, enter cluster of entries, otherwise, skip all entries up to and including the entry with CLE = 1 (cluster exit).

Table 15-34. RQFCR Field Descriptions (continued)

Bit	Name	Description
25–26	CMP	Comparison operation to perform on the RQPROP entry at this index when PID > 0. The property value extracted by the frame parser is masked by the 32-bit *mask_register* prior to comparison against RQPROP. However, the property value is not permanently altered by the value in *mask_register*. By default, *mask_register* is initialized to 0xFFFF_FFFF before each frame is processed. In the case where PID = 0, CMP is interpreted as follows: 00/01 Filer *mask_register* is set to all 32 bits of RQPROP, and this entry always *matches*. 10/11 Filer *mask_register* is set to all 32 bits of RQPROP, and this entry always *fails to match*. In the case where PID > 0, CMP is interpreted as follows (& is bit-wise AND operator): 00 *property*[PID] & *mask_register* = RQPROP 01 *property*[PID] & *mask_register* >= RQPROP 10 *property*[PID] & *mask_register* != RQPROP 11 *property*[PID] & *mask_register* < RQPROP
27	—	Reserved, should be written with zero.
28–31	PID	Property identifier. The value in the RQPROP entry at this index is interpreted according to PID (see Table 15-35).

15.5.3.3.8 Receive Queue Filer Table Property Register (RQFPR)

RQFPR (see Figure 15-31) is accessed to read or write the RQPROP words in entries of the receive queue filer table. The table entries are described in greater detail in Section 15.6.5.1, "Receive Queue Filer." The word accessed via RQFPR is defined by the current value of RQFAR. Figure 15-31 and Figure 15-32 describe the fields of the RQFPR register according to property ID.

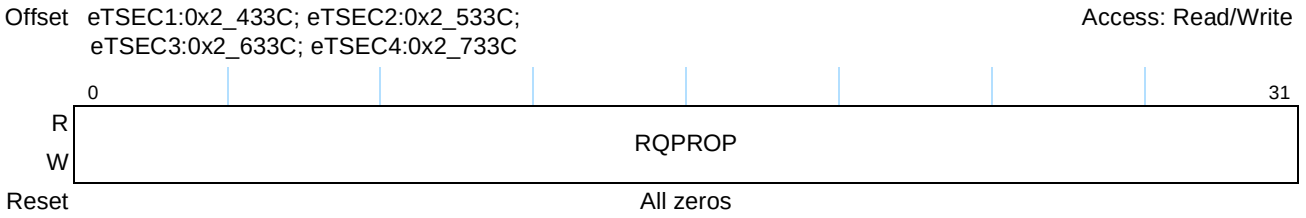


Figure 15-31. Receive Queue Filer Table Property IDs 0, 2–15 Register Definition

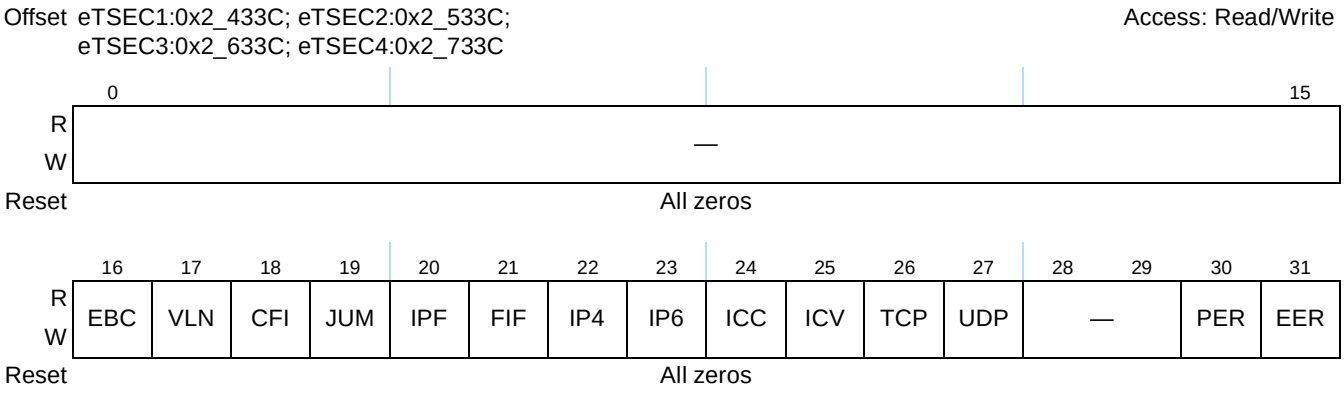


Figure 15-32. Receive Queue Filer Table Property ID1 Register Definition

Table 15-35 describes the fields of the RQFPR register.

Table 15-35. RQFPR Field Descriptions

PID[1]	Bit	Name	Description
0000	0–31	MASK	Mask bits to be written to Filer *mask_register* for masking of property values. The rule match/fail status for this PID is determined by RQCTRL[CMP]. Since *mask_register* is bit-wise ANDed with properties, every bit of MASK that is cleared also results in the corresponding property bit being cleared in comparisons. Therefore setting MASK to 0xFFFF_FFFF ensures that all property bits participate in rule matches.
0001	0–15	—	Reserved
	16	EBC	Set if the destination Ethernet address is to the broadcast address.
	17	VLN	Set if a VLAN tag (Ethertype DFVLAN[TAG] or 0x8100) was seen in the frame.
	18	CFI	Set to the value of the Canonical Format Indicator in the VLAN control tag if VLAN is set, zero otherwise.
	19	JUM	Set if a jumbo Ethernet frame was parsed.
	20	IPF	Set if a fragmented IPv4 or IPv6 header was encountered. See the descriptions of receive FCB fields IP and PRO in Section 15.6.4.3, “Receive Path Off-Load,” for more information on determining the status of received packets for which IPF is set.
	21	FIF	Set if the packet entered on eTSEC’s FIFO interface.
	22	IP4	Set if an IPv4 header was parsed.
	23	IP6	Set if an IPv6 header was parsed.
	24	ICC	Set if the IPv4 header checksum was checked.
	25	ICV	Set if the IPv4 header checksum was verified correct.
	26	TCP	Set if a TCP header was parsed.
	27	UDP	Set if a UDP header was parsed.
	28-29	—	Reserved.
	30	PER	Set on a parse error, such as header inconsistency.
	31	EER	Set on an Ethernet framing error that prevents parsing.
0010	0–7	ARB	User-defined arbitrary bit field property: byte 0 extracted. Defaults to 0x00.
	8–15		User-defined arbitrary bit field property: byte 1 extracted. Defaults to 0x00.
	16–23		User-defined arbitrary bit field property: byte 2 extracted. Defaults to 0x00.
	24–31		User-defined arbitrary bit field property: byte 3 extracted. Defaults to 0x00.
0011	0–7	—	Reserved, should be written with zero.
	8–31	DAH	Destination MAC address, most significant 24 bits. Defaults to 0x000000.
0100	0–7	—	Reserved, should be written with zero.
	8–31	DAL	Destination MAC address, least significant 24 bits. Defaults to 0x000000.
0101	0–7	—	Reserved, should be written with zero.
	8–31	SAH	Source MAC address, most significant 24 bits. Defaults to 0x000000.

Table 15-35. RQFPR Field Descriptions (continued)

PID[1]	Bit	Name	Description
0110	0–7	—	Reserved, should be written with zero.
	8–31	SAL	Source MAC address, least significant 24 bits. Defaults to 0x000000.
0111	0–15	—	Reserved, should be written with zero.
	16–31	ETY	Ethertype of next layer protocol, for example, last ethertype if layer 2 headers nest. Defaults to 0xFFFF. Using the filer to match ETY does not work in the case of PPPoE packets, because the PPPoE ethertype in the original packet, 0x8864, is always overwritten with the PPP protocol field. Thus, matches on ETY == 0x8864 always fail. Instead, software should use PID=1 fields IP4 (ETY = 0x0021) and IP6 (ETY = 0x0057) to distinguish PPPoE session packets carrying IPv4 and IPv6 datagrams. Other PPP protocols are encoded in the ETY field, but many of them overlap with real ethertype definitions. Consult IANA and IEEE for possible ambiguities. Packets with a value in the length/type field greater than 1500 and less than 1536 are treated as payload length. If the eTSEC is used in a network where there are packets carrying a type designation between 1500 and 1536 (note there are none currently publicly defined by IANA), then the S/W must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range. Note that the eTSEC filer gets multiple packet attributes as a result of parsing the packet. The behavior of the eTSEC is that it pulls the innermost ethertype found in the packet; this means that in many supported protocols, it is impossible to create a filer rule that matches on the outer ethertype. There are four cases that need to be highlighted. 1. The jumbo ethertype (0x8870)—In this case, the eTSEC assumes that the following header is LLC/SNAP. LLC/SNAP has an associated Ethertype, and the ETY field is populated with that ethertype. This makes it impossible to file on jumbo frames. In this case, one can use arbitrary extracted bytes to pull the outermost Ethertype. 2. The PPPoE ethertype described above. 3. The VLAN tag ethertype (0x8100)—In this case, one can use the PID1 VLN bit to indicate that the packet had a VLAN tag. 4. The MPLS tagged packets. In this case, one can use arbitrary extraction bytes to compare to the actual ethertype if a filer rule is intending to file based on an MPLS label existence.
1000	0–19	—	Reserved, should be written with zero.
	20–31	VID	VLAN network identifier (as per IEEE 802.1Q). This value defaults to 0x000 if no VLAN tag was found, or the VLAN tag contained only priority information.
1001	0–28	—	Reserved, should be written with zero.
	29–31	PRI	VLAN user priority (as per IEEE 802.1p). This value defaults to 000 (best effort priority) if no VLAN tag was found.
1010	0–23	—	Reserved, should be written with zero.
	24–31	TOS	IPv4 header Type Of Service field or IPv6 Traffic Class field. This value defaults to 0x00 (default RFC 2474 best-effort behavior) if no IP header appeared. Note that for IPv6 the Traffic Class field is extracted using the IP header definition in RFC 2460. IPv6 headers formed using the earlier RFC 1883 have a different format and must be handled with software.
1011	0–23	—	Reserved, should be written with zero.
	24–31	L4P	Layer 4 protocol identifier as per published IANA specification. This is the last recognized protocol type recognized in the case of IPv6 extension headers. This value defaults to 0xFF to indicate that no layer 4 header was recognized (possibly due to absence of an IP header).

Table 15-35. RQFPR Field Descriptions (continued)

PID[1]	Bit	Name	Description
1100	0–31	DIA	Destination IP address. If an IPv4 header was found, this is the entire destination address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit destination address. This value defaults to 0x0000_0000 if no IP header appeared.
1101	0–31	SIA	Source IP address. If an IPv4 header was found, this is the entire source address. If an IPv6 header was found, this is the 32 most significant bits of the 128-bit source address. This value defaults to 0x0000_0000 if no IP header appeared.
1110	0–15	—	Reserved, should be written with zero.
	16–31	DPT	Destination port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.
1111	0–15	—	Reserved, should be written with zero.
	16–31	SPT	Source port number for TCP or UDP headers. This value defaults to 0x0000 if no TCP or UDP headers were recognized.

1 PID is the property identifier field of the filer table control entry (see RQFCR[PID]) at the same index.

15.5.3.3.9 Maximum Receive Buffer Length Register (MRBLR)

The MRBLR register is written by the user. It informs the eTSEC how much space is in the receive buffer pointed to by the RxBD. Figure 15-33 describes the definition for the MRBLR.

Offset eTSEC1:0x2_4340; eTSEC2:0x2_5340; eTSEC3:0x2_6340; eTSEC4:0x2_7340 — Access: Read/Write

	0	15	16	25	26	31
R	—		MRBL		—	
W						
Reset	All zeros					

Figure 15-33. MRBLR Register Definition

Table 15-36 describes the fields of the MRBLR register.

Table 15-36. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the eTSEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. The eTSEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL. MRBL must be set, together with the number of buffer descriptors, to ensure adequate space for received frames. See Section 15.5.3.5.5, “Maximum Frame Length Register (MAXFRM),” for further discussion.
26–31	—	To ensure that MRBL is a multiple of 64, these bits are reserved and should be cleared.

15.5.3.3.10 Receive Data Buffer Pointer High Register (RBDBPH)

The RBDBPH register is written by the user with the most significant address bits common to all RxBD buffer addresses, RxBD[Data Buffer Pointer]. As a consequence, Rx buffers must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBDBPH. The RxBD ring itself can reside in a different memory region (based at RBASEH). Figure 15-34 describes the definition for the RBDBPH register.

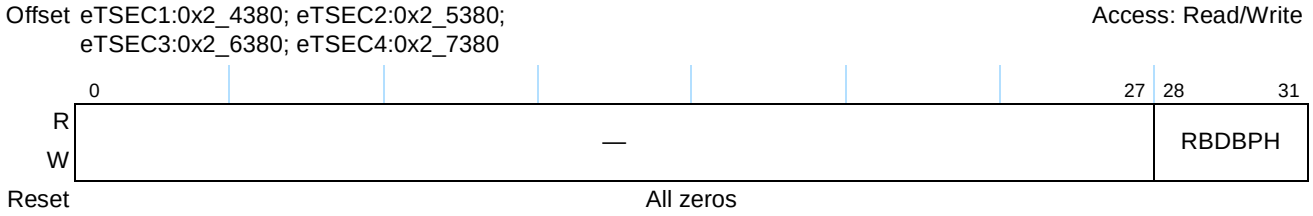


Figure 15-34. RBDBPH Register Definition

Table 15-37 describes the fields of the RBDBPH register.

Table 15-37. RBDBPH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	RBDBPH	Most significant bits common to all data buffer addresses contained in RxBDs. The user must initialize RBDBPH before enabling the eTSEC receive function.

15.5.3.3.11 Receive Buffer Descriptor Pointers 0–7 (RBPTR0–RBPTR7)

RBPTR0–RBPTR7 each contains the low-order 32 bits of the next receive buffer descriptor address for their respective RxBD ring. Figure 15-35 describes the RBPTR registers. These registers takes on the value of their ring's associated RBASE when the RBASE register is written by software. Software must not write RBPTR*n* while eTSEC is actively receiving frames. However, RBPTR*n* can be modified when the receiver is disabled or when no Rx buffer is in use (after a GRACEFUL STOP RECEIVE command is issued and the frame completes its reception) in order to change the next RxBD eTSEC receives.

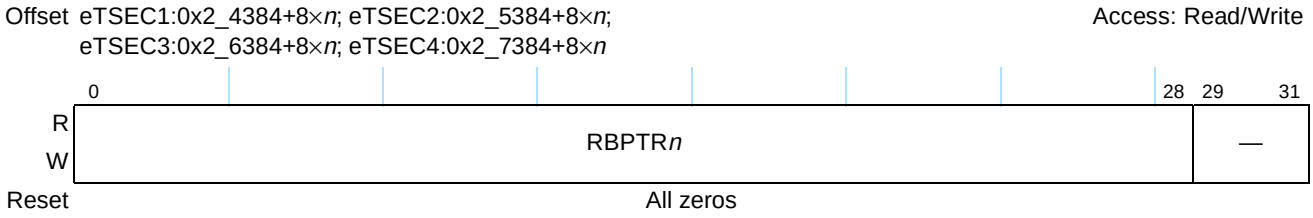


Figure 15-35. RBPTR0–RBPTR7 Register Definition

Table 15-23 describes the fields of the RBPTR*n* register.

Table 15-38. RBPTR*n* Field Descriptions

Bits	Name	Description
0–28	RBPTR*n*	Current RxBD pointer for RxBD ring *n*. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the RxBD ring is reached, eTSEC initializes RBPTR*n* to the value in the corresponding RBASE*n*. The RBPTR register is internally written by the eTSEC's DMA controller during reception. The pointer increments by 8 (bytes) each time a descriptor is closed successfully by the eTSEC. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved

15.5.3.3.12 Receive Descriptor Base Address High Register (RBASEH)

The RBASEH register is written by the user with the most significant address bits common to all RxBD addresses, including RBASE0–RBASE7 and RBPTR0–RBPTR7. As a consequence, RxBD rings must be placed in a 4 Gbyte segment of memory whose base address is prefixed by the bits in RBASEH. However, Rx data buffers may potentially reside in a different memory region based at RBDBPH. Figure 15-36 describes the definition for the RBASEH register.

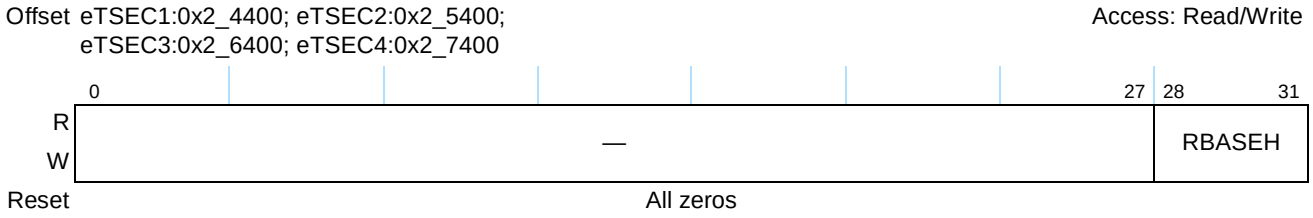


Figure 15-36. RBASEH Register Definition

Table 15-24 describes the fields of the RBASEH register.

Table 15-39. RBASEH Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	RBASEH	Most significant bits common to all RxBD addresses—except data buffer pointers. The user must initialize RBASEH before enabling the eTSEC receive function.

15.5.3.3.13 Receive Descriptor Base Address Registers (RBASE0–RBASE7)

The RBASE*n* registers are written by the user with the base address of each RxBD ring *n*. Each such value must be divisible by eight, since the 3 least-significant bits always write as 000. Figure 15-37 describes the RBASE*n* registers.

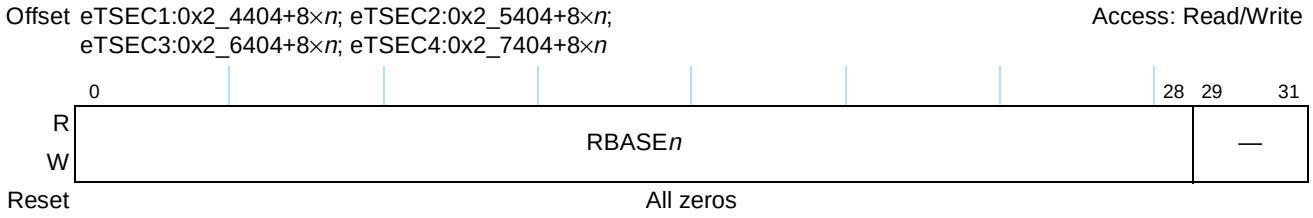


Figure 15-37. RBASE Register Definition

Table 15-25 describes the fields of the RBASE*n* registers.

Table 15-40. RBASE0–RBASE7 Field Descriptions

Bits	Name	Description
0–28	RBASE*n*	Receive base for ring *n*. RBASE defines the starting location in the memory map for the eTSEC RxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the receive packets. The user must initialize RBASE before enabling the eTSEC receive function on the associated ring.
29–31	—	Reserved

15.5.3.3.14 Receive Stamp Register (TMR_RXTS_H/L)

Receive time stamp register (RXTS_H/L). This register holds the value present in TMR_CNT_H/L when the eTSEC detects a new incoming Ethernet frame. This register is only updated when the precision time stamp logic is enable via TMR_CTRL[TE]. This register is read only in normal operation. Figure 15-38 describes the definition for the RXTS_H/L register.

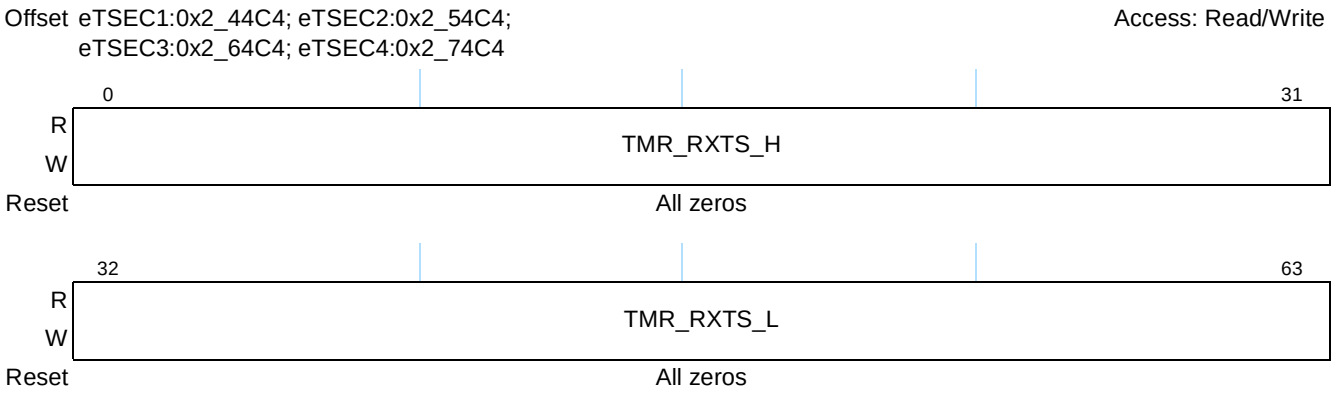


Figure 15-38. TMR_RXTS_H/L Register Definition

Table 15-41 describes the fields of the TMR_RXTS_H/L register.

Table 15-41. TMR_RXTS_H/L Register Field Descriptions

Bits	Name	Description
0–63	TMR_RXTS_H/L	Value of the eTSEC precision timer upon detection of a start of frame symbol for the received frame.

15.5.3.4 MAC Functionality

This section describes the MAC registers and provides a brief overview of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by IEEE 802.3. All of the MAC registers are 32 bits wide.

15.5.3.4.1 Configuring the MAC

The MAC configuration registers 1 and 2 provide for configuring the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value. Should custom preamble insertion/extraction be configured, then this register must by left at its default value.
- Varying pad/CRC combinations—Three different pad/CRC combinations are provided to handle a variety of system requirements. Simplest are frames that already have a valid frame check sequence (FCS) field. The other two options include appending a valid CRC or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

15.5.3.4.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the eTSEC. Half-duplex mode is only supported for 10- and 100-Mbps operation. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

In full-duplex mode both the carrier sense (CRS) and collision (COL) indications from the PHY are ignored, but in half-duplex mode the eTSEC defers to CRS, and following a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if CRS is sensed. During the final one-third of the IPG, CRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

15.5.3.4.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the eTSEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating

that a collision occurred and that the start of the frame is needed for retransmission. The eTSEC then backs off of the medium for a time determined by the truncated binary exponential back off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured via the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system should flush the frame and move to the next one in line. If the system requests to send a packet while the eTSEC is deferring to a carrier, the eTSEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

If packet transmission attempts experience collisions, the eTSEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly-distributed random integer *r* in the range:

$$0 \leq r \leq 2^{k}, \text{ where } k = \min(n,10).$$

So after the first collision, the eTSEC backs off either 0 or 1 slot times. After the fifth collision, the eTSEC backs off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back off is 1024. This can be adjusted via the half-duplex register. An alternate truncation point, such as 7 for instance, can be programmed. On average, the MAC is more aggressive after seven collisions than other stations on the network.

15.5.3.4.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within eTSEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of the packet (including preamble), the end occurs around the 56th byte of the frame data. Slot time in 1000-Mbps mode is not supported.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The eTSEC implements the optional back pressure mechanism using the raise carrier method. If the system receive logic wishes to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is set (TCTRL[THDF]). If the medium is idle, the eTSEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, the eTSEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the back-pressure-no-back-off configuration bit HAFDUP[BP No BackOff]. These transmitting-preamble-for-back pressure collisions are not counted. If HAFDUP[BP No BackOff] is set, the eTSEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If HAFDUP[BP No BackOff] is cleared, the eTSEC adheres to the truncated BEB algorithm that allows the possibility of packets being received. This also can be detrimental in that packets can now experience excessive collisions, causing them to be dropped in the stations from which they

originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, HAFDUP[BP No BackOff] must be set.

The eTSEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the eTSEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. HAFDUP[BP No BackOff] applies for any collision that occurs during the sending of this packet. Collisions for packets while half duplex back pressure is asserted are counted. The eTSEC does not defer while attempting to send packets while in back pressure. Again, back pressure is non-standard, yet it can be effective in reducing the flow of receive packets.

15.5.3.4.5 Controlling PHY Links

Control and status to and from the PHY is provided via the two-wire MII management interface described in IEEE 802.3u. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) are used to exercise this interface between a host processor and one or more PHY devices (including the TBI).

The eTSEC MII's registers provide the ability to perform continuous read cycles (called a scan cycle); although, scan cycles are not explicitly defined in the standard. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY scan].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the eTSEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

15.5.3.5 MAC Registers

This section describes the MAC registers.

15.5.3.5.1 MAC Configuration 1 Register (MACCFG1)

MACCFG1 is written by the user. Figure 15-39 describes the definition for the MACCFG1 register.

Offset eTSEC1:0x2_4500; eTSEC2:0x2_5500; eTSEC3:0x2_6500; eTSEC4:0x2_7500 Access: Mixed

	0	1	11	12	13	14	15
R	Soft_Reset	—		Reset Rx MC	Reset Tx MC	Reset Rx Fun	Reset Tx Fun
W							
Reset	All zeros						

	16	22	23	24	25	26	27	28	29	30	31
R	—		Loop Back	—		Rx_Flow	Tx_Flow	Sync'd Rx EN	Rx_EN	Sync'd Tx EN	Tx_EN
W											
Reset	All zeros										

Figure 15-39. MACCFG1 Register Definition

Table 15-42 describes the fields of the MACCFG1 register.

Table 15-42. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 15.6.3.2, "Soft Reset and Reconfiguring Procedure," for more information on setting this bit. 0 Normal operation. 1 Place the entire MAC in reset except for the host interface.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the receive part of the MAC in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit part of the MAC in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation. 1 Place the receive function in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit function in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop Back	Loop back. This bit is cleared by default. 0 Normal operation. 1 Loop back the MAC transmit outputs to the MAC receive inputs.

Table 15-42. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
24–25	—	Reserved
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream. (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream. (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

15.5.3.5.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. Figure 15-40 describes the definition for the MACCFG2 register.

Offset eTSEC1:0x2_4504; eTSEC2:0x2_5504; eTSEC3:0x2_6504; eTSEC4:0x2_7504 — Access: Read/Write

	0–15
R W	—
Reset	All zeros

	16–19	20–21	22–23	24	25	26	27	28	29	30	31
R W	Preamble Length	—	I/F Mode	PreAmRxEN	PreAmTxEN	Huge Frame	Length check	MPEN	PAD/CRC	CRC EN	Full Duplex
Reset	0 1 1 1	0 0	0 0	0	0	0	0	0	0	0	0

Figure 15-40. MACCFG2 Register Definition

Table 15-43 describes the fields of the MACCFG2 register.

Table 15-43. MACCFG2 Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–15</td><td>—</td><td>Reserved</td></tr>
<tr><td>16–19</td><td>Preamble Length</td><td>This field determines the length in bytes of the preamble field preceding each Ethernet start-of-frame delimiter byte. Values from 0x3 to 0xF are supported by the controller. The default value of 0x7 should not be altered in order to guarantee reliable operation with IEEE 802.3 compatible hardware.</td></tr>
<tr><td>20–21</td><td>—</td><td>Reserved</td></tr>
<tr><td>22–23</td><td>I/F Mode</td><td>This field determines the type of interface to which the MAC is connected. Its default is 00.
00 Reserved bit mode (not supported) (10 Mbps GENDEC/GPSI)
01 Nibble mode (MII) (10/100 Mbps MII/RMII)
10 Byte mode (GMII/TBI) (1000 Mbps GMII/TBI). Reserved if neither GMII or TBI are supported.
11 Reserved</td></tr>
<tr><td>24</td><td>PreAM RxEN</td><td>User defined preamble enable for received frames. This bit is cleared by default.
0 The MAC skips the Ethernet preamble without returning it.
1 The MAC recovers the received Ethernet preamble and passes it to the driver at the start of each received frame. Not applicable to FIFO or RMII 10/100 modes.</td></tr>
<tr><td>25</td><td>PreAM TxEN</td><td>User defined preamble enable for transmitted frames. This bit is cleared by default.
0 The MAC generates a standard Ethernet preamble.
1 If a user-defined preamble has been passed to the MAC it is transmitted instead of the standard preamble. Otherwise the standard Ethernet preamble is generated. The Preamble Length field should be left at its default setting if a user-defined preamble is transmitted. Not applicable to FIFO or RMII 10/100 modes.</td></tr>
<tr><td>26</td><td>Huge Frame</td><td>Huge frame enable. This bit is cleared by default.
0 Limit the length of frames received to less than or equal to the maximum frame length value (MAXFRM[Maximum Frame]) and limit the length of frames transmitted to less than the maximum frame length.
See Section 15.6.8, “Buffer Descriptors,” for further details of buffer descriptor bit updating.
<table>
<tr><th>Frame type</th><th>Frame length</th><th>Packet truncation</th><th>Buffer descriptor updated</th></tr>
<tr><td>Receive or transmit</td><td>> maximum frame length</td><td>yes</td><td>yes</td></tr>
<tr><td>Receive</td><td>= maximum frame length</td><td>no</td><td>yes</td></tr>
<tr><td>Transmit</td><td>= maximum frame length</td><td>no</td><td>no</td></tr>
<tr><td>Receive or transmit</td><td>< maximum frame length</td><td>no</td><td>no</td></tr>
</table>
1 Frames are transmitted and received regardless of their relationship to the maximum frame length.
Note that if Huge Frame is cleared, the user must ensure that adequate buffer space is allocated for received frames. See Section 15.5.3.5.5, “Maximum Frame Length Register (MAXFRM),” for further information.</td></tr>
<tr><td>27</td><td>Length check</td><td>Length check. This bit is cleared by default.
0 No length field checking is performed.
1 The MAC checks the frame’s length field on receive to ensure it matches the actual data field length. Transmitted frames are not checked.</td></tr>
</table>

Table 15-43. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
28	MPEN	Magic packet enable for Ethernet modes. This bit is cleared by default. MPEN should be enabled only after GRACEFUL RECEIVE STOP and GRACEFUL TRANSMIT STOP are completed successfully (in other words, transmission and reception have stopped). Note that sleep mode is not supported with Magic Packet. 0 Normal receive behavior on receive, or Magic Packet mode has exited with reception of a valid Magic Packet. 1 Commence Magic Packet detection by the MAC provided that frame reception is enabled in MACCFG1. In this mode the MAC ignores all received frames until the specific Magic Packet frame is received, at which point this bit is cleared by the eTSEC, and a maskable interrupt via IEVENT[MAG] occurs.
29	PAD/CRC	Pad and append CRC. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.
30	CRC EN	CRC enable. If the configuration bit PAD/CRC ENABLE or the per-packet PAD/CRC ENABLE is set, CRC ENABLE is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	Full Duplex	Full duplex configure. This bit is cleared by default. 0 The MAC operates in half-duplex mode only. 1 The MAC operates in full-duplex mode.

15.5.3.5.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. Figure 15-41 describes the definition for IPGIFG.

Offset eTSEC1:0x2_4508; eTSEC2:0x2_5508; eTSEC3:0x2_6508; eTSEC4:0x2_7508 — Access: Read/Write

	0	1–7	8	9–15	16–23	24	25–31
R W	—	Non-Back-to-Back Inter-Packet-Gap, Part 1	—	Non-Back-to-Back Inter-Packet-Gap, Part 2	Minimum IFG Enforcement	—	Back-to-Back Inter-Packet-Gap
Reset	0	1 0 0 0 0 0 0	0	1 1 0 0 0 0 0	0 1 0 1 0 0 0 0	0	1 1 0 0 0 0 0

Figure 15-41. IPGIFG Register Definition

Table 15-44 describes the fields of the IPGIFG register.

Table 15-44. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	This is a programmable field representing the optional carrier sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.

Table 15-44. IPGIFG Field Descriptions (continued)

Bits	Name	Description
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	This is a programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	This is a programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	This is a programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

15.5.3.5.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 15-42 describes the HAFDUP register.

Offset eTSEC1:0x2_450C; eTSEC2:0x2_550C; eTSEC3:0x2_650C; eTSEC4:0x2_750C — Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—								Alternate BEB Truncation				Alt BEB	BP No BackOff	No BackOff	Excess Defer
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Retransmission Maximum				—				Collision Window							
Reset	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1

Figure 15-42. Half-Duplex Register Definition

Table 15-45 describes the fields of the HAFDUP register.

Table 15-45. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while ALTERNATE BINARY EXPONENTIAL BACKOFF ENABLE is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential backoff. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back off rule. 1 The Tx MAC uses the ALTERNATE BINARY EXPONENTIAL BACKOFF TRUNCATION setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum backoff time.
13	BP No BackOff	Back pressure no backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.

Table 15-45. HAFDUP Field Descriptions (continued)

Bits	Name	Description
14	No BackOff	No backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF.
20–21	—	Reserved
22–31	Collision Window	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

15.5.3.5.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 15-43 shows the MAXFRM register.

Offset eTSEC1:0x2_4510; eTSEC2:0x2_5510; eTSEC3:0x2_6510; eTSEC4:0x2_7510 — Access: Read/Write

	0															15	16															31
R	—																Maximum Frame															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Figure 15-43. Maximum Frame Length Register Definition

Table 15-46 describes the fields of the MAXFRM register.

Table 15-46. MAXFRM Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536 bytes). It sets the maximum Ethernet frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].). It does not affect the size of packets sent or received via the FIFO packet interface. Note that if MACCFG2[Huge Frame] = 0, the value of this field must be less than or equal to MRBLR[MRBL] × (minimum number of RxBDs per ring). See Section 15.5.3.5.2, “MAC Configuration 2 Register (MACCFG2),” Section 15.5.3.3.9, “Maximum Receive Buffer Length Register (MRBLR),” and Section 15.6.8.3, “Receive Buffer Descriptors (RxBD).”

15.5.3.5.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user to configure all MII management operations. Note that MII management hardware is shared by all eTSECs. Thus, only through the MIIM registers of eTSEC1 and eTSEC3 can external PHYs be accessed and configured. (eTSEC1 registers apply to MIIM interface 1 and

eTSEC3 registers apply to MIIM interface 3.) Note: when an eTSEC is configured to use TBI/RTBI, configuration of the TBI/RTBI (described in Section 15.5.4, “Ten-Bit Interface (TBI)”) is done via the MIIM registers for that eTSEC. For example, if a TBI/RTBI interface is required on eTSEC2, then the MIIM registers starting at offset 0x2_5520 are used to configure it.

Figure 15-44 describes the definition for the MIIMCFG register.

Offset eTSEC1:0x2_4520 Access: Read/Write

	0	1–26	27	28	29–31
R W	Reset Mgmt	—	No Pre	—	MgmtClk
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

Figure 15-44. MII Management Configuration Register Definition

Table 15-47 describes the fields of the MIIMCFG register.

Table 15-47. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII MGMT to perform mgmt read/write cycles if requested via the host interface. 1 Reset the MII MGMT.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII MGMT performs Mgmt read/write cycles with 32 clocks of preamble. 1 The MII MGMT suppresses preamble generation and reduces the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	MgmtClk	This field determines the clock frequency of the MII management clock (EC_MDC). Its default value is 111. **Note:** The eTSEC system clock is derived from (CCB Clock)/2. 000 1/4 of the eTSEC system clock divided by 8 001 1/4 of the eTSEC system clock divided by 8 010 1/6 of the eTSEC system clock divided by 8 011 1/8 of the eTSEC system clock divided by 8 100 1/10 of the eTSEC system clock divided by 8 101 1/14 of the eTSEC system clock divided by 8 110 1/20 of the eTSEC system clock divided by 8 111 1/28 of the eTSEC system clock divided by 8

15.5.3.5.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 15-45 describes the definition for MIIMCOM.

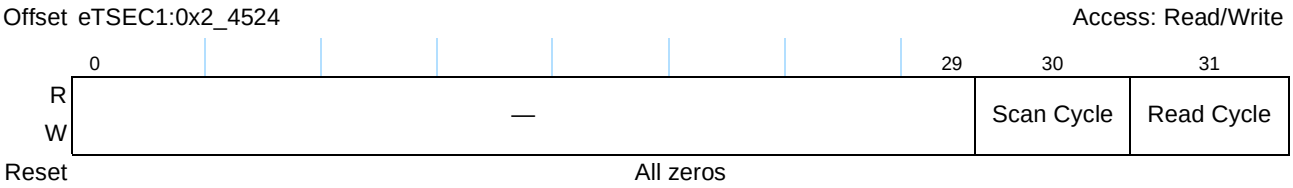


Figure 15-45. MIIMCOM Register Definition

Table 15-48 describes the fields of the MIIMCOM register.

Table 15-48. MIIMCOM Descriptions

Bits	Name	Description
0–29	—	Reserved
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation. 1 The MII management continuously performs read cycles. This is useful for monitoring link fail, for example.
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation. 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0-to-1 transition of this bit also causes the MIIMIND[Busy] bit to be set. The read is complete when the MIIMIND[Busy] bit clears. Data is returned in register MIIMSTAT[PHY Status].

15.5.3.5.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 15-46 shows the MIIMADD register.

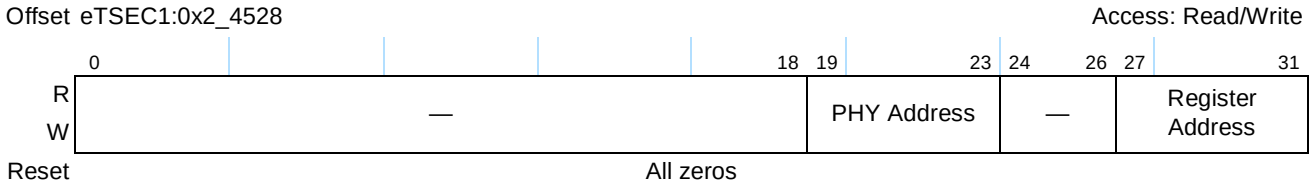


Figure 15-46. MIIMADD Register Definition

Table 15-49 describes the fields of the MIIMADD register.

Table 15-49. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved). Its default value is 0x00.

Table 15-49. MIIMADD Field Descriptions (continued)

Bits	Name	Description
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of Mgmt cycles. Up to 32 registers can be accessed. Its default value is 0x00.

15.5.3.5.9 MII Management Control Register (MIIMCON)

MIIMCON, shown in Figure 15-47, is written by the user.

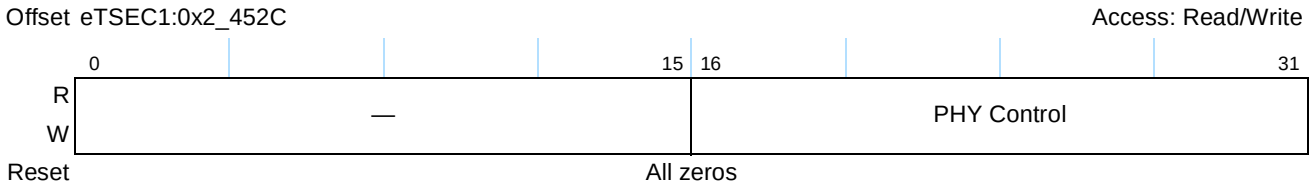


Figure 15-47. MII Mgmt Control Register Definition

Table 15-50 describes the fields of the MIIMCON register.

Table 15-50. MIIMCON Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Control	If written, an MII Mgmt write cycle is performed using this 16-bit data, the pre-configured PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). Its default value is 0x0000.

15.5.3.5.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read only by the user. Figure 15-48 describes the definition for the MIIMSTAT register.

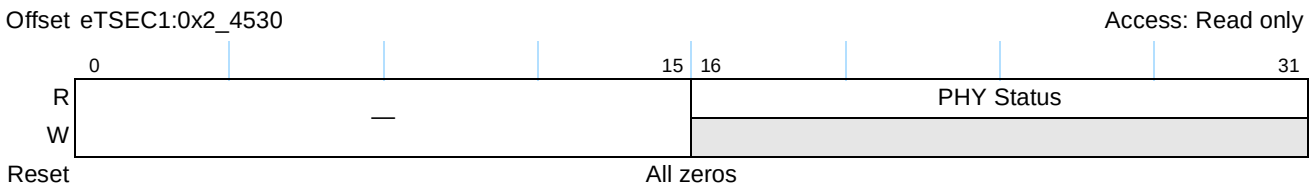


Figure 15-48. MIIMSTAT Register Definition

Table 15-51 describes the fields of the MIIMSTAT register.

Table 15-51. MIIMSTAT Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII Mgmt read cycle, the 16-bit data can be read from this location. Its default value is 0x0000.

15.5.3.5.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 15-49 describes the definition for the MIIMIND register.

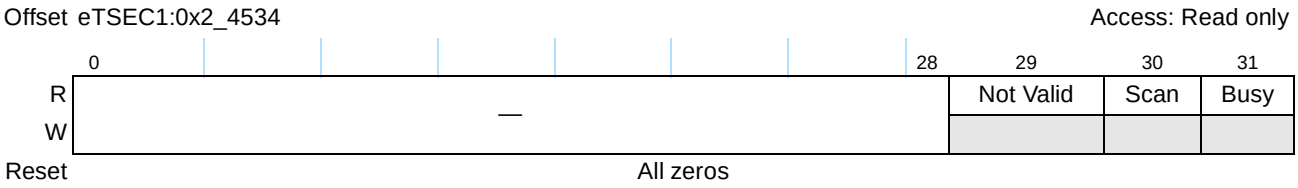


Figure 15-49. MII Mgmt Indicator Register Definition

Table 15-52. MIIMIND Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	Not Valid	Not valid. 0 MII Mgmt read cycle has completed and the read data is valid. 1 MII Mgmt read cycle has not completed and the read data is not yet valid.
30	Scan	Scan in progress. 0 A scan operation (continuous MII Mgmt read cycles) is not in progress. 1 A scan operation (continuous MII Mgmt read cycles) is in progress.
31	Busy	Busy. 0 MII Mgmt block is not currently performing an MII Mgmt read or write cycle. 1 MII Mgmt block is currently performing an MII Mgmt read or write cycle.

15.5.3.5.12 Interface Status Register (IFSTAT)

Figure 15-50 shows the IFSTAT register.

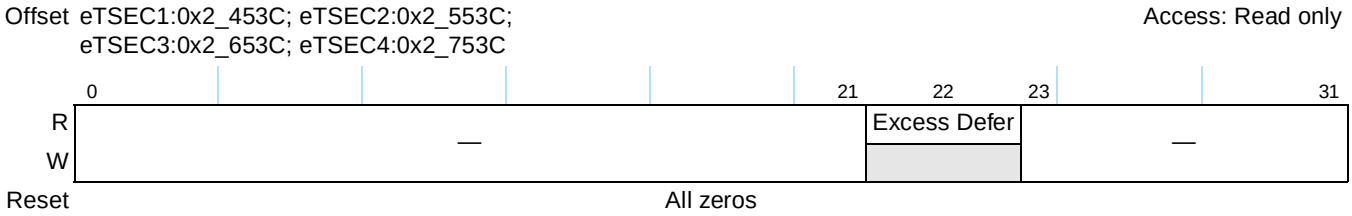


Figure 15-50. Interface Status Register Definition

Table 15-53 describes the fields of the FSTAT register.

Table 15-53. IFSTAT Field Descriptions

Bits	Name	Description
0–21	—	Reserved

Table 15-53. IFSTAT Field Descriptions (continued)

Bits	Name	Description
22	Excess Defer	Excessive transmission defer. This bit latches high and is cleared when read. This bit is cleared by default. 0 Normal operation. 1 The MAC excessively defers a transmission.
23–31	—	Reserved

15.5.3.5.13 MAC Station Address Part 1 Register (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. The value of the station address written into MACSTNADDR1 and MACSTNADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, MACSTNADDR1 is set to 0xCDAB7856 and MACSTNADDR2 is set to 0x34120000.

Figure 15-51 shows the MACSTNADDR1 register.

Offset eTSEC1:0x2_4540; eTSEC2:0x2_5540; eTSEC3:0x2_6540; eTSEC4:0x2_7540 — Access: Read/Write

	0–7	8–15	16–23	24–31
R W	Station Address, 6th Octet	Station Address, 5th Octet	Station Address, 4th Octet	Station Address, 3rd Octet
Reset	All zeros			

Figure 15-51. MAC Station Address Part 1 Register Definition

Table 15-54 describes the fields of the MACSTNADDR1 register.

Table 15-54. MACSTNADDR1 Field Descriptions

Bit	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x0.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x0.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x0.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x0.

15.5.3.5.14 MAC Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 15-52 describes the definition for the MACSTNADDR2 register.

Offset eTSEC1:0x2_4544; eTSEC2:0x2_5544; eTSEC3:0x2_6544; eTSEC4:0x2_7544 — Access: Read/Write

	0–7	8–15	16–31
R W	Station Address, 2nd Octet	Station Address, 1st Octet	—
Reset	All zeros		

Figure 15-52. MAC Station Address Part 2 Register Definition

Table 15-55 describes the fields of the MACSTNADDR2 register.

Table 15-55. MACSTNADDR2 Field Descriptions

Bit	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x0.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

15.5.3.5.15 MAC Exact Match Address 1–15 Part 1 Registers (MAC01ADDR1–MAC15ADDR1)

The MAC01ADDR1–MAC15ADDR1 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 15-53 describes the definition for all of the fifteen MAC*n*ADDR1 registers. The value of the address written into MAC*x*ADDR1 and MAC*n*ADDR2 is byte reversed from how it would appear in the DA field of a frame in memory. For example, for a MAC address of 0x12345678ABCD, MACnADDR1 is set to 0xCDAB7856 and MACnADDR2 is set to 0x34120000. For any valid, non-zero MAC address received, exact match registers can be excluded individually by clearing them to all zero bytes.

Offset eTSEC1:0x2_4548+8×*n*; eTSEC2:0x2_5548+8×*n*; eTSEC3:0x2_6548+8×*n*; eTSEC4:0x2_7548+8×*n* — Access: Read/Write

	0–7	8–15	16–23	24–31
R W	Exact Match Address, 6th Octet	Exact Match Address, 5th Octet	Exact Match Address, 4th Octet	Exact Match Address, 3rd Octet
Reset	All zeros			

Figure 15-53. MAC Exact Match Address *n* Part 1 Register Definition

Table 15-54 describes the fields of a MAC*n*ADDR1 register.

Table 15-56. MAC*n*ADDR1 Field Descriptions

Bit	Name	Description
0–7	Exact Match Address, 6th Octet	Holds the sixth octet of the exact match address. The sixth octet (destination address bits 40–47) defaults to a value of 0x0.
8–15	Exact Match Address, 5th Octet	Holds the fifth octet of the exact match address. The fifth octet (destination address bits 32–39) defaults to a value of 0x0.
16–23	Exact Match Address, 4th Octet	Holds the fourth octet of the exact match address. The fourth octet (destination address bits 24–31) defaults to a value of 0x0.
24–31	Exact Match Address, 3rd Octet	Holds the third octet of the exact match address. The third octet (destination address bits 16–23) defaults to a value of 0x0.

15.5.3.5.16 MAC Exact Match Address 1–15 Part 2 Registers (MAC01ADDR2–MAC15ADDR2)

The MAC01ADDR2–MAC15ADDR2 registers are written by the user with the unicast or multicast addresses aliasing the MAC. Figure 15-54 describes the definition for all of the fifteen MACxADDR2 registers.

Offset eTSEC1:0x2_454C+8×*n*; eTSEC2:0x2_554C+8×*n*; eTSEC3:0x2_654C+8×*n*; eTSEC4:0x2_754C+8×*n* Access: Read/Write

	0–7	8–15	16–31
R	Exact Match Address, 2nd Octet	Exact Match Address, 1st Octet	—
W			
Reset	All zeros		

Figure 15-54. MAC Exact Match Address *x* Part 2 Register Definition

Table 15-55 describes the fields of a MAC*x*ADDR2 register.

Table 15-57. MAC01ADDR2–MAC15ADDR2 Field Descriptions

Bit	Name	Description
0–7	Exact Match Address, 2nd Octet	This field holds the second octet of the exact match address. The second octet (destination address bits 8–15) defaults to a value of 0x0.
8–15	Exact Match Address, 1st Octet	This field holds the first octet of the exact match address. The first octet (destination address bits 0–7) defaults to a value of 0x0.
16–31	—	Reserved

15.5.3.6 MIB Registers

This section describes the MIB registers. The eTSEC RMON module has 37 separate statistics counters, which simply count or accumulate statistical events that occur as packets transmitted and received. These counters support RMON MIB group 1, RMON MIB group 2 if table counters, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB.

An interrupt can be generated upon any one counter's rollover condition via a carry interrupt output from the RMON. Each counter's rollover condition can be discretely masked from causing an interrupt by internal masking registers. In addition, each individual counter value may be reset on read access, or all counters may be simultaneously reset by setting ECNTRL[CLRCNT].

The majority of MIB counters are Ethernet-specific.

In FIFO modes, only the following registers are updated:

- Transmit: TBYT, TPKT, TDRP
- Receive: RBYT, RPKT, RFCS

NOTE

RMON counters do not comprehend custom VLAN tagged frames. Affected counters include TRMGV, RMCA, RBCA, RXCF, RXPF, RXUO, RALN, RFLR, ROVR, RJBR, TMCA, TBCA, TXPF, TXCF. Specifically, custom VLAN tagged frames are not afforded the ability to be greater than 1518, as the IEEE standard tagged frames are.

15.5.3.6.1 Transmit and Receive 64-Byte Frame Counter (TR64)

Figure 15-55 describes the definition for the TR64 register.

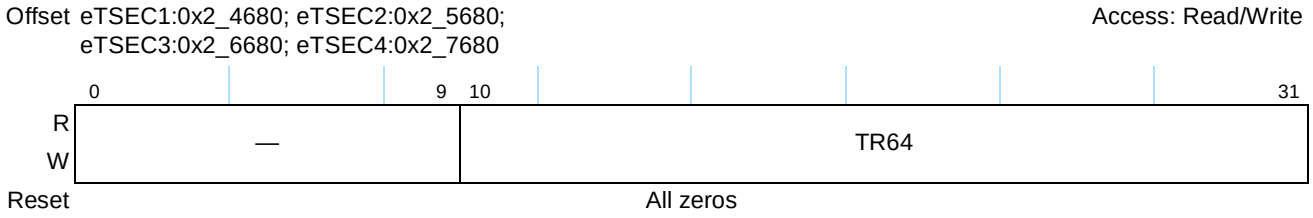


Figure 15-55. Transmit and Receive 64-Byte Frame Register Definition

Table 15-58 describes the fields of the TR64 register.

Table 15-58. TR64 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR64	Transmit and receive 64-byte frame counter—Increment for each good or bad frame transmitted and received which is 64 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.2 Transmit and Receive 65- to 127-Byte Frame Counter (TR127)

Figure 15-56 describes the definition for the TR127 register.

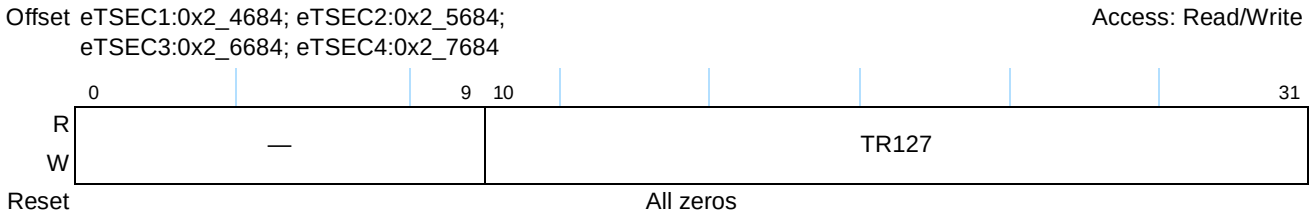


Figure 15-56. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 15-59 describes the fields of the TR127 register.

Table 15-59. TR127 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR127	Transmit and receive 65- to 127-byte frame counter—Increments for each good or bad frame transmitted and received which is 65–127 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.3 Transmit and Receive 128- to 255-Byte Frame Counter (TR255)

Figure 15-57 describes the definition for the TR255 register.

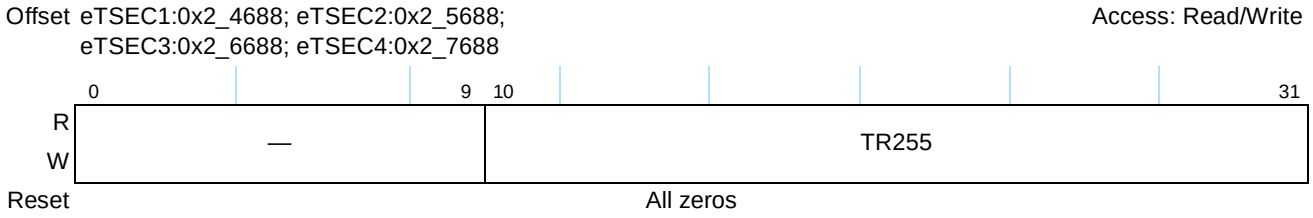


Figure 15-57. Transmit and Received 128- to 255-Byte Frame Register Definition

Table 15-60 describes the fields of the TR255 register.

Table 15-60. TR255 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR255	Transmit and receive 128- to 255-byte frame counter—Increments for each good or bad frame transmitted and received which is 128–255 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.4 Transmit and Receive 256- to 511-Byte Frame Counter (TR511)

Figure 15-58 describes the definition for the TR511 register.

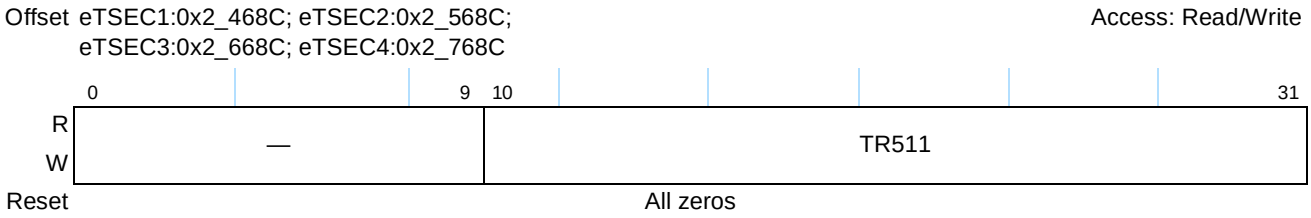


Figure 15-58. Transmit and Received 256- to 511-Byte Frame Register Definition

Table 15-61 describes the fields of the TR511 register.

Table 15-61. TR511 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR511	Increments for each good or bad frame transmitted and received which is 256–511 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.5 Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)

Figure 15-59 shows the TR1K register.

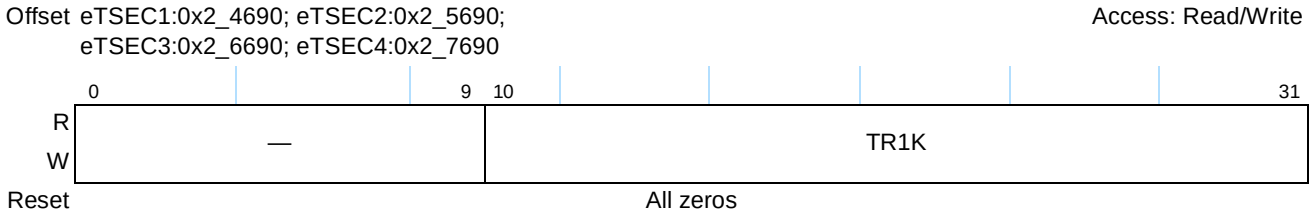


Figure 15-59. Transmit and Received 512- to 1023-Byte Frame Register Definition

Table 15-62 describes the fields of the TR1K register.

Table 15-62. TR1K Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR1K	Increments for each good or bad frame transmitted and received which is 512–1023 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.6 Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)

Figure 15-60 describes the definition for the TRMAX register.

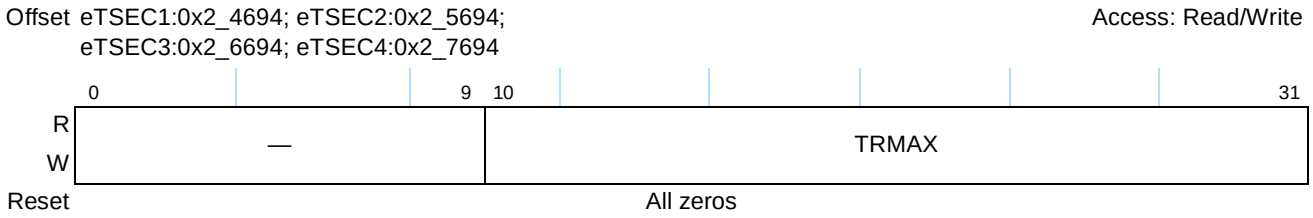


Figure 15-60. Transmit and Received 1024- to 1518-Byte Frame Register Definition

Table 15-63 describes the fields of the TRMAX register.

Table 15-63. TRMAX Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMAX	Increments for each good or bad frame transmitted and received which is 1024–1518 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.7 Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)

Figure 15-61 describes the definition for the TRMGV register.

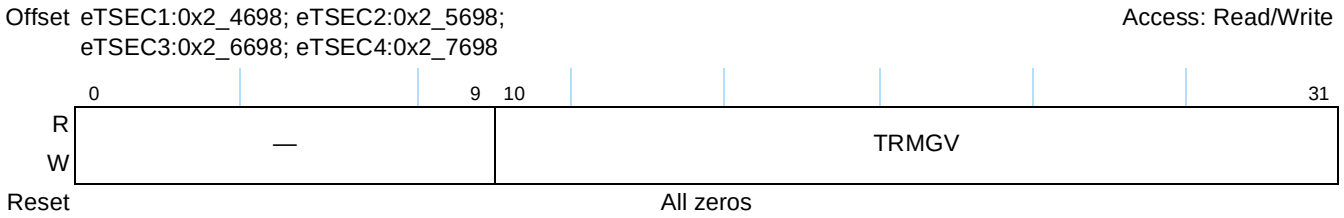


Figure 15-61. Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition

Table 15-64 describes the fields of the TRMGV register.

Table 15-64. TRMGV Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TRMGV	Increments for each good or bad frame transmitted and received which is 1519–1522 bytes in length, inclusive (excluding preamble and SFD but including FCS bytes).

15.5.3.6.8 Receive Byte Counter (RBYT)

Figure 15-62 shows the RBYT register.

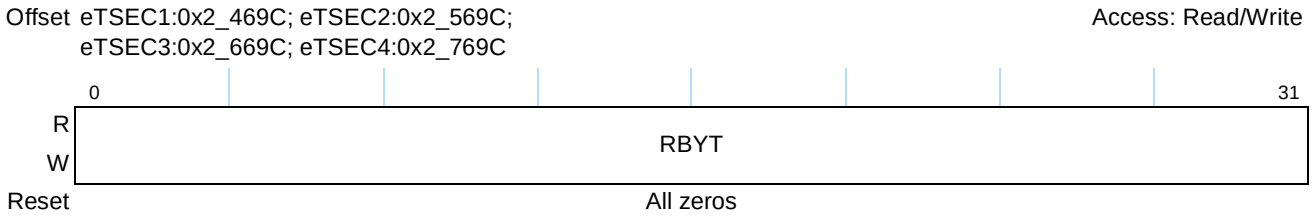


Figure 15-62. Receive Byte Counter Register Definition

Table 15-65 describes the fields of the RBYT register.

Table 15-65. RBYT Field Descriptions

Bits	Name	Description
0–31	RBYT	Receive byte counter. The statistic counter register increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes. In FIFO mode, all bytes (including FCS bytes) are counted.

15.5.3.6.9 Receive Packet Counter (RPKT)

Figure 15-63 describes the definition for the RPKT register.

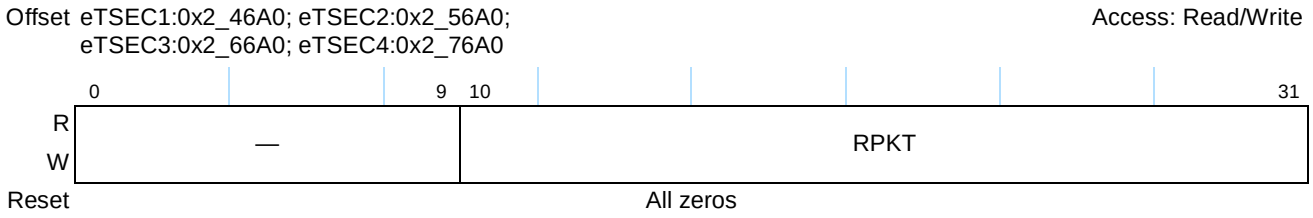


Figure 15-63. Receive Packet Counter Register Definition

Table 15-66 describes the fields of the RPKT register.

Table 15-66. RPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10-31	RPKT	Receive packet counter. Increments for each frame received packet (including bad packets, all unicast, broadcast, and multicast packets).

15.5.3.6.10 Receive FCS Error Counter (RFCS)

Figure 15-64 describes the definition for the RFCS register.

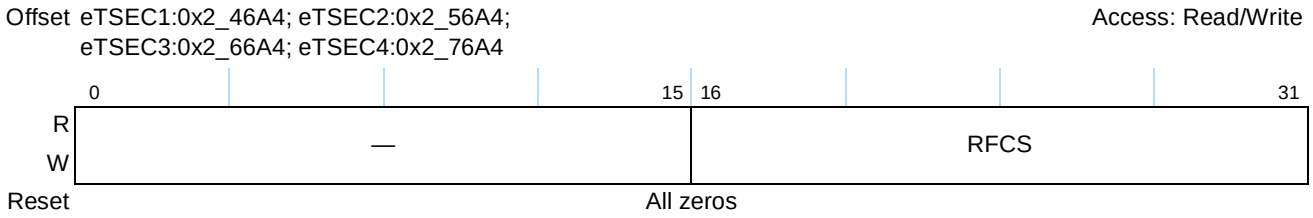


Figure 15-64. Receive FCS Error Counter Register Definition

Table 15-67 describes the fields of the RFCS register.

Table 15-67. RFCS Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFCS	Receive FCS error counter. In Ethernet mode, increments for each frame received that has an integral 64–1518 length and contains a frame check sequence error. In FIFO mode, increments for each frame received that contains a frame check sequence error (regardless of size).

15.5.3.6.11 Receive Multicast Packet Counter (RMCA)

Figure 15-65 describes the definition for the RMCA register.

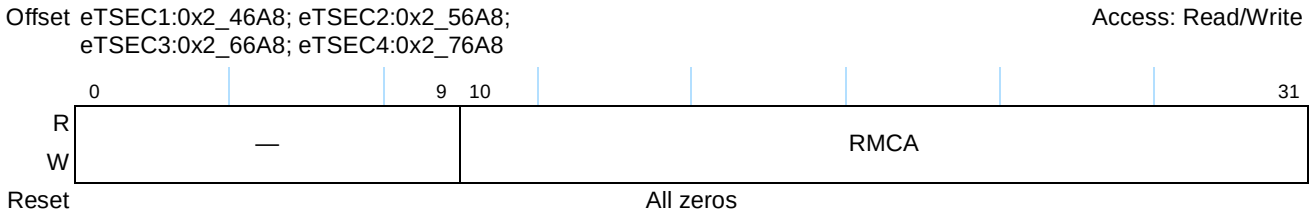


Figure 15-65. Receive Multicast Packet Counter Register Definition

Table 15-68 describes the fields of the RMCA register.

Table 15-68. RMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RMCA	Receive multicast packet counter. Increments for each multicast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

15.5.3.6.12 Receive Broadcast Packet Counter (RBCA)

Figure 15-66 describes the definition for the RBCA register.

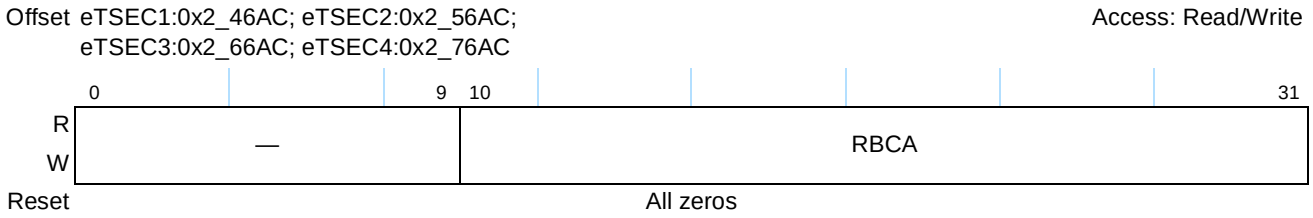


Figure 15-66. Receive Broadcast Packet Counter Register Definition

Table 15-69 describes the fields of the RBCA register.

Table 15-69. RBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RBCA	Receive broadcast packet counter. Increments for each broadcast frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

15.5.3.6.13 Receive Control Frame Packet Counter (RXCF)

Figure 15-67 describes the definition for the RXCF register.

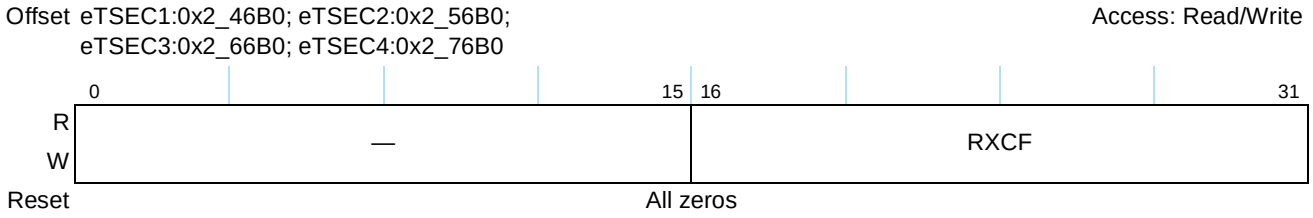


Figure 15-67. Receive Control Frame Packet Counter Register Definition

Table 15-70 describes the fields of the RXCF register.

Table 15-70. RXCF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXCF	Receive control frame packet counter. Increments for each MAC control frame received (PAUSE and unsupported) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.14 Receive Pause Frame Packet Counter (RXPF)

Figure 15-68 describes the definition for the RXPF register.

Offset eTSEC1:0x2_46B4; eTSEC2:0x2_56B4; eTSEC3:0x2_66B4; eTSEC4:0x2_76B4 — Access: Read/Write

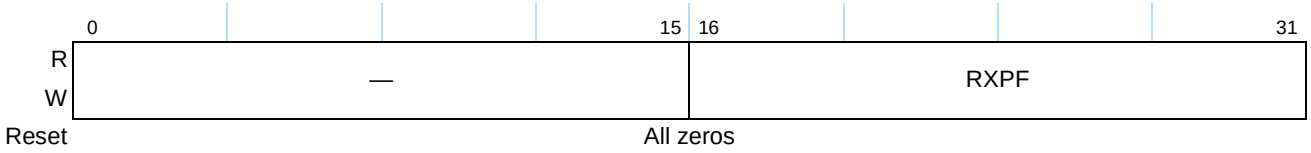


Figure 15-68. Receive Pause Frame Packet Counter Register Definition

Table 15-71 describes the fields of the RXPF register.

Table 15-71. RXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXPF	Receive PAUSE frame packet counter. Increments each time a PAUSE MAC control frame is received with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.15 Receive Unknown Opcode Packet Counter (RXUO)

Figure 15-69 describes the definition for the RXUO register.

Offset eTSEC1:0x2_46B8; eTSEC2:0x2_56B8; eTSEC3:0x2_66B8; eTSEC4:0x2_76B8 — Access: Read/Write

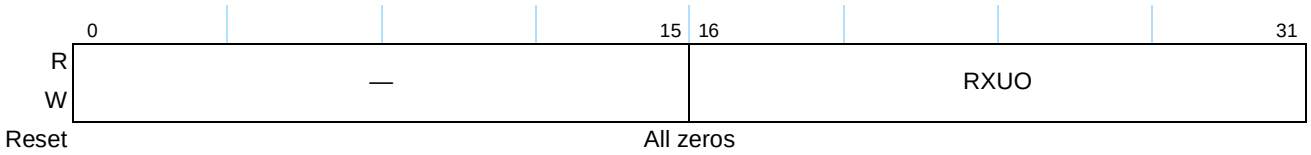


Figure 15-69. Receive Unknown OPCode Packet Counter Register Definition

Table 15-72 describes the fields of the RXUO register.

Table 15-72. RXUO Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXUO	Receive unknown opcode counter. Increments each time a MAC control frame is received which contains an opcode other than PAUSE, but the frame has valid CRC and length 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.16 Receive Alignment Error Counter (RALN)

Figure 15-70 describes the definition for the RALN register.

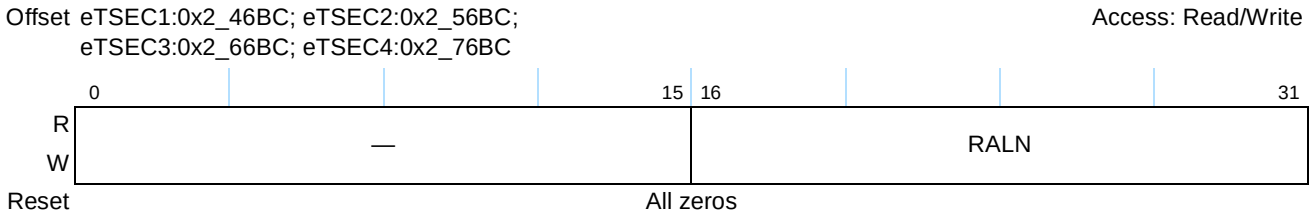


Figure 15-70. Receive Alignment Error Counter Register Definition

Table 15-73 describes the fields of the RALN register.

Table 15-73. RALN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RALN	Receive alignment error counter. Increments for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.

15.5.3.6.17 Receive Frame Length Error Counter (RFLR)

Figure 15-71 describes the definition for the RFLR register.

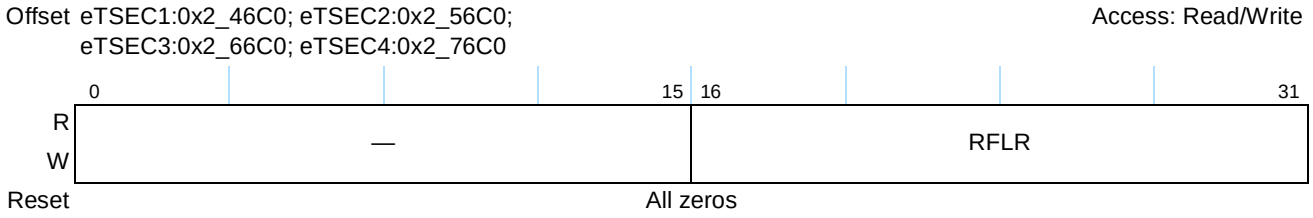


Figure 15-71. Receive Frame Length Error Counter Register Definition

Table 15-74 describes the fields of the RFLR register.

Table 15-74. RFLR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFLR	Receive frame length error counter. Increments for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46–1500 bytes). The counter does not increment if the length field is not a valid 802.3 length, such as an Ethertype value.

15.5.3.6.18 Receive Code Error Counter (RCDE)

Figure 15-72 describes the definition for the RCDE register.

Offset eTSEC1:0x2_46C4; eTSEC2:0x2_56C4; eTSEC3:0x2_66C4; eTSEC4:0x2_76C4 Access: Read/Write

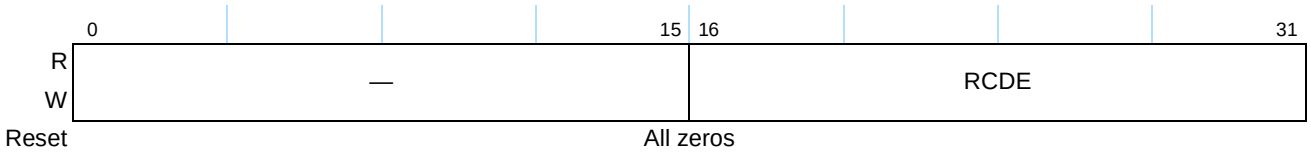


Figure 15-72. Receive Code Error Counter Register Definition

Table 15-75 describes the fields of the RCDE register.

Table 15-75. RCDE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCDE	Receive code error counter. Increments each time a valid carrier is present and at least one invalid data symbol is detected.

15.5.3.6.19 Receive Carrier Sense Error Counter (RCSE)

Figure 15-73 describes the definition for the RCSE register.

Offset eTSEC1:0x2_46C8; eTSEC2:0x2_56C8; eTSEC3:0x2_66C8; eTSEC4:0x2_76C8 Access: Read/Write

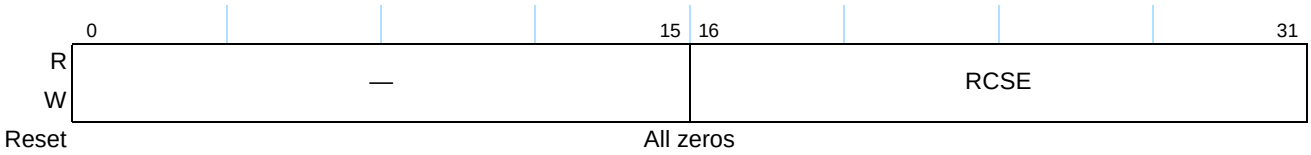


Figure 15-73. Receive Carrier Sense Error Counter Register Definition

Table 15-76 describes the fields of the RCSE register.

Table 15-76. RCSE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCSE	Receive false carrier counter. Counts the number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on a particular interface. The count represented by an instance of this object is incremented at most once per transmission attempt, even if the carrier sense condition fluctuates during a transmission attempt. The event is reported along with the statistics generated on the next received frame, as defined by a 1 on TSEC*n*_RX_ER and an 0xE on TSEC*n*_RXD. Only one false carrier condition can be detected and logged between frames.

15.5.3.6.20 Receive Undersize Packet Counter (RUND)

Figure 15-74 describes the definition for the RUND register.

Offset eTSEC1:0x2_46CC; eTSEC2:0x2_56CC; eTSEC3:0x2_66CC; eTSEC4:0x2_76CC — Access: Read/Write

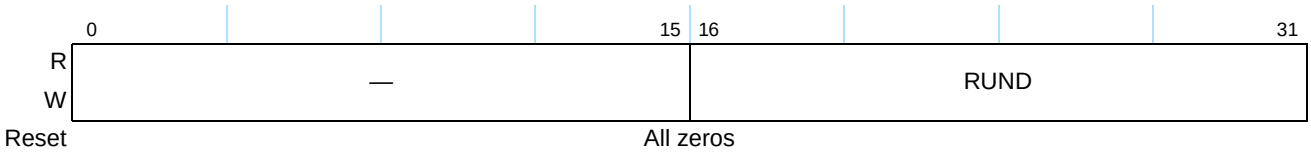


Figure 15-74. Receive Undersize Packet Counter Register Definition

Table 15-77 describes the fields of the RUND register.

Table 15-77. RUND Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RUND	Receive undersize packet counter. Increments each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This count does not include range length errors.

15.5.3.6.21 Receive Oversize Packet Counter (ROVR)

Figure 15-75 describes the definition for the ROVR register.

Offset eTSEC1:0x2_46D0; eTSEC2:0x2_56D0; eTSEC3:0x2_66D0; eTSEC4:0x2_76D0 — Access: Read/Write

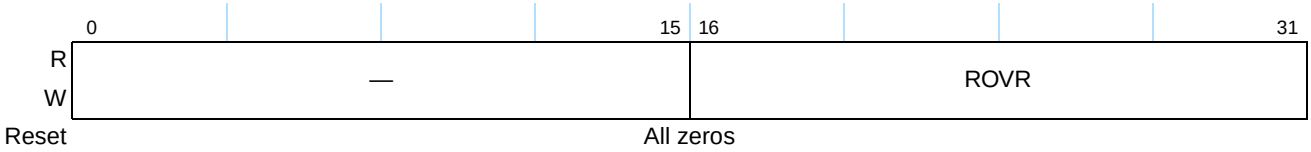


Figure 15-75. Receive Oversize Packet Counter Register Definition

Table 15-78 describes the fields of the ROVR register.

Table 15-78. ROVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ROVR	Receive oversize packet counter. Increments each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and was otherwise well formed.

15.5.3.6.22 Receive Fragments Counter (RFRG)

Figure 15-76 describes the definition for the RFRG register.

Offset eTSEC1:0x2_46D4; eTSEC2:0x2_56D4; eTSEC3:0x2_66D4; eTSEC4:0x2_76D4 Access: Read/Write

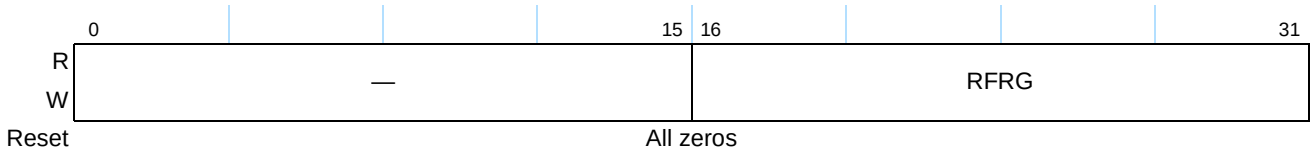


Figure 15-76. Receive Fragments Counter Register Definition

Table 15-79 describes the fields of the RFRG register.

Table 15-79. RFRG Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFRG	Receive fragments counter. Increments for each frame received which is less than 64 bytes in length and contains an invalid FCS. This includes integral and non-integral lengths.

15.5.3.6.23 Receive Jabber Counter (RJBR)

Figure 15-77 describes the definition for the RJBR register.

Offset eTSEC1:0x2_46D8; eTSEC2:0x2_56D8; eTSEC3:0x2_66D8; eTSEC4:0x2_76D8 Access: Read/Write

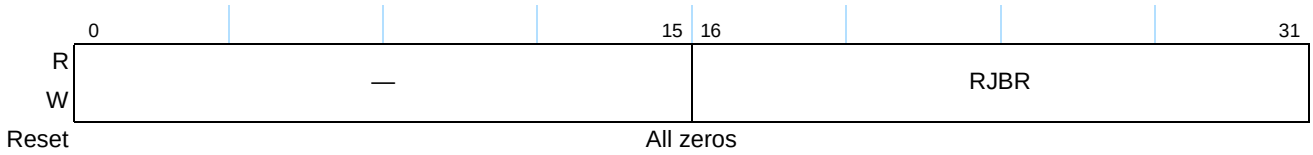


Figure 15-77. Receive Jabber Counter Register Definition

Table 15-80 describes the fields of the RJBR register.

Table 15-80. RJBR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RJBR	Receive jabber counter. Increments for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contain an invalid FCS. This includes alignment errors.

15.5.3.6.24 Receive Dropped Packet Counter (RDRP)

Figure 15-78 describes the definition for the RDRP register.

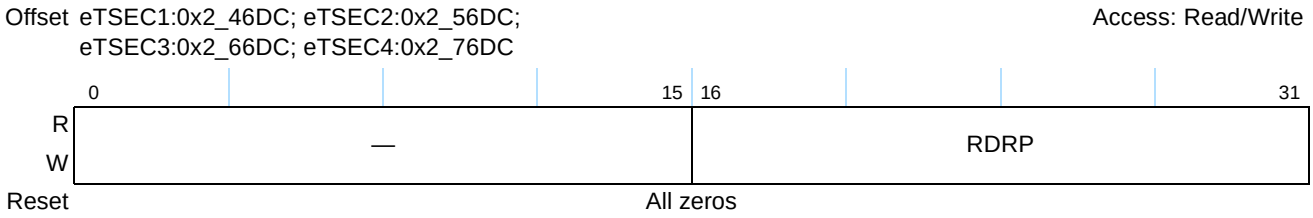


Figure 15-78. Receive Dropped Packet Counter Register Definition

Table 15-81 describes the fields of the RDRP register.

Table 15-81. RDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RDRP	Receive dropped packets counter. Increments for frames received which are streamed to system but are later dropped due to lack of system resources.

15.5.3.6.25 Transmit Byte Counter (TBYT)

Figure 15-79 describes the definition for the TBYT register.

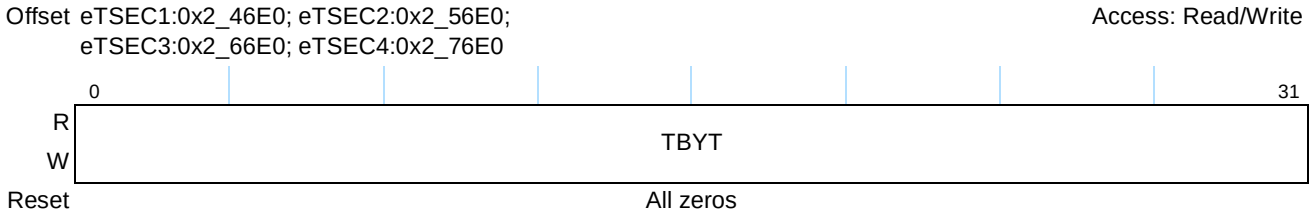


Figure 15-79. Transmit Byte Counter Register Definition

Table 15-82 describes the fields of the TBYT register.

Table 15-82. TBYT Field Descriptions

Bits	Name	Description
0–31	TBYT	Transmit byte counter. Increments by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes. Note that the value of TBYT may be greater than the actual number of bytes transmitted if the frame is truncated because it exceeds MAXFRM.

15.5.3.6.26 Transmit Packet Counter (TPKT)

Figure 15-80 describes the definition for the TPKT register.

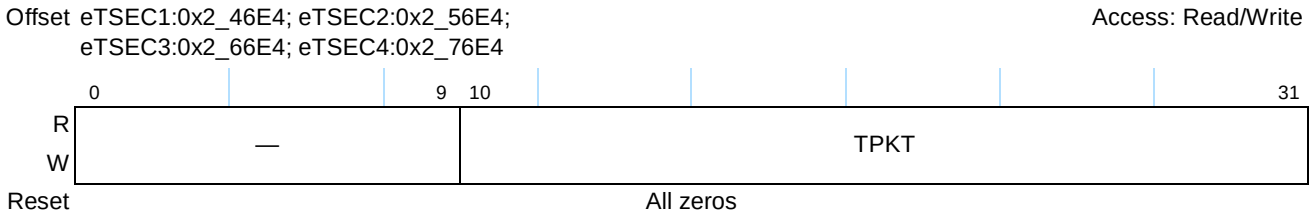


Figure 15-80. Transmit Packet Counter Register Definition

Table 15-83 describes the fields of the TPKT register.

Table 15-83. TPKT Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TPKT	Transmit packet counter. Increments for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all unicast, broadcast, and multicast packets).

15.5.3.6.27 Transmit Multicast Packet Counter (TMCA)

Figure 15-81 describes the definition for the TMCA register.

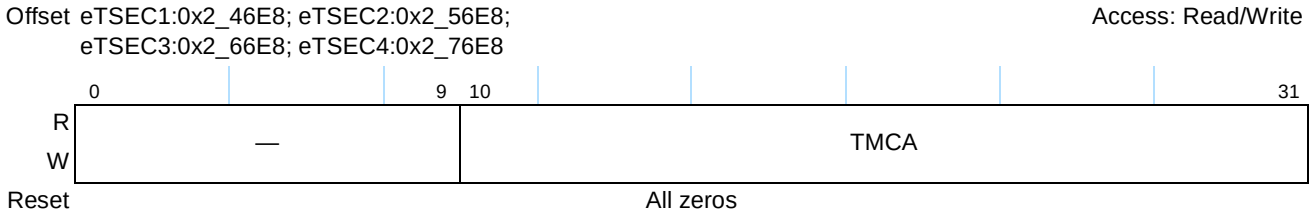


Figure 15-81. Transmit Multicast Packet Counter Register Definition

Table 15-84 describes the fields of the TMCA register.

Table 15-84. TMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TMCA	Transmit multicast packet counter. Increments for each multicast valid frame transmitted (excluding broadcast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.28 Transmit Broadcast Packet Counter (TBCA)

Figure 15-82 describes the definition for the TBCA register.

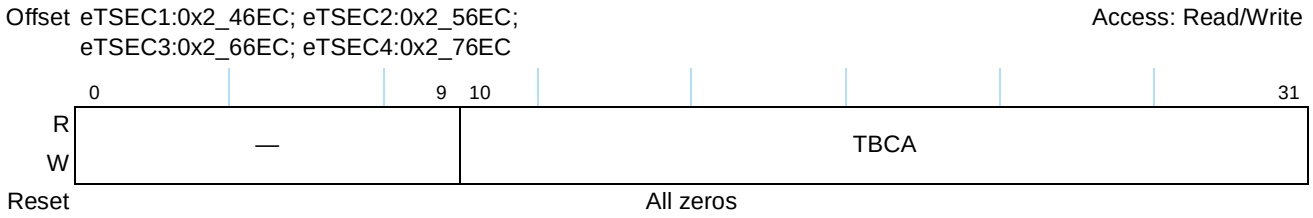


Figure 15-82. Transmit Broadcast Packet Counter Register Definition

Table 15-85 describes the fields of the TBCA register.

Table 15-85. TBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TBCA	Transmit broadcast packet counter. Increments for each broadcast frame transmitted (excluding multicast frames) with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.29 Transmit Pause Control Frame Counter (TXPF)

Figure 15-83 describes the definition for the TXPF register.

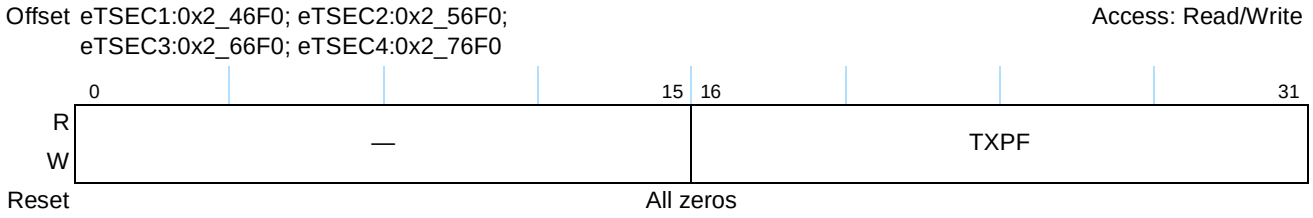


Figure 15-83. Transmit Pause Control Frame Counter Register Definition

Table 15-86 describes the fields of the TXPF register.

Table 15-86. TXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TXPF	Transmit PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is transmitted with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.30 Transmit Deferral Packet Counter (TDFR)

Figure 15-84 describes the definition for the TDFR register.

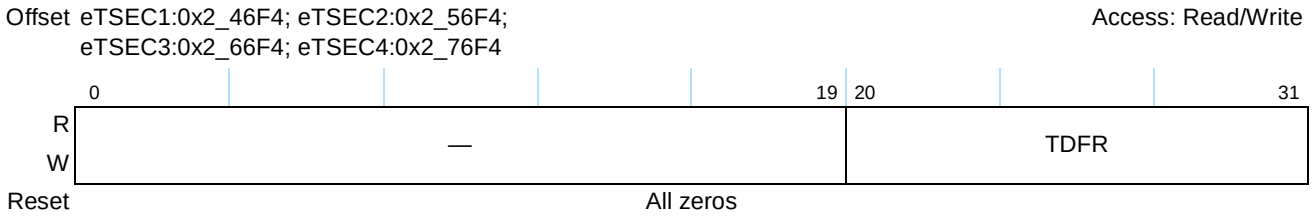


Figure 15-84. Transmit Deferral Packet Counter Register Definition

Table 15-87 describes the fields of the TDFR register.

Table 15-87. TDFR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit deferral packet counter. Increments for each frame, which was deferred on its first transmission attempt. This count does not include frames involved in collisions.

15.5.3.6.31 Transmit Excessive Deferral Packet Counter (TEDF)

Figure 15-85 describes the definition for the TEDF register.

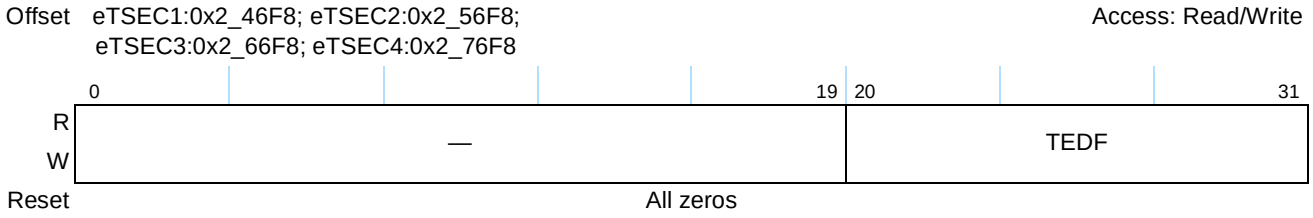


Figure 15-85. Transmit Excessive Deferral Packet Counter Register Definition

Table 15-88 describes the fields of the TEDF register.

Table 15-88. TEDF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TEDF	Transmit excessive deferral packet counter. Increments for frames aborted which were deferred for an excessive period of time (3036 byte times).

15.5.3.6.32 Transmit Single Collision Packet Counter (TSCL)

Figure 15-86 describes the definition for the TSCL register.

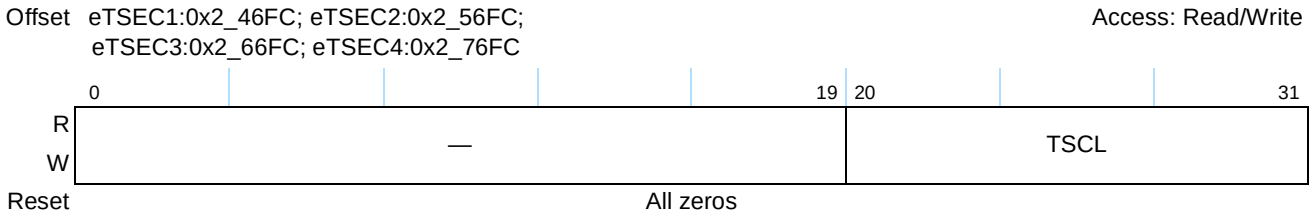


Figure 15-86. Transmit Single Collision Packet Counter Register Definition

Table 15-89 describes the fields of the TSCL register.

Table 15-89. TSCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TSCL	Transmit single collision packet counter. Increments for each frame transmitted which experienced exactly one collision during transmission.

15.5.3.6.33 Transmit Multiple Collision Packet Counter (TMCL)

Figure 15-87 describes the definition for the TMCL register.

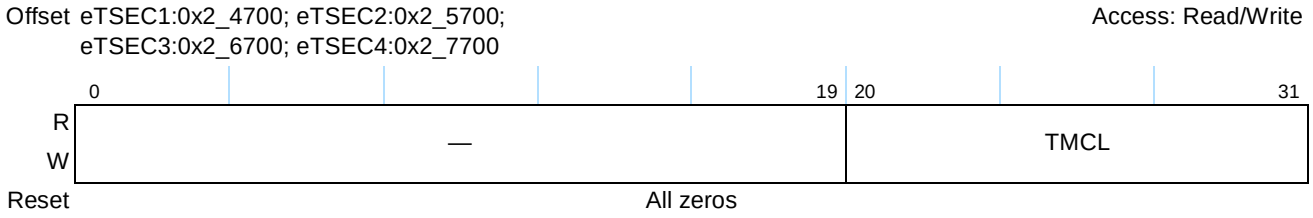


Figure 15-87. Transmit Multiple Collision Packet Counter Register Definition

Table 15-90 describes the fields of the TMCL register.

Table 15-90. TMCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TMCL	Transmit multiple collision packet counter. Increments for each frame transmitted which experienced 2–15 collisions (including any late collisions) during transmission as defined using the Half_Duplex[RETRANSMISSION MAXIMUM] field.

15.5.3.6.34 Transmit Late Collision Packet Counter (TLCL)

Figure 15-88 describes the definition for the TLCL register.

Offset eTSEC1:0x2_4704; eTSEC2:0x2_5704; eTSEC3:0x2_6704; eTSEC4:0x2_7704 Access: Read/Write

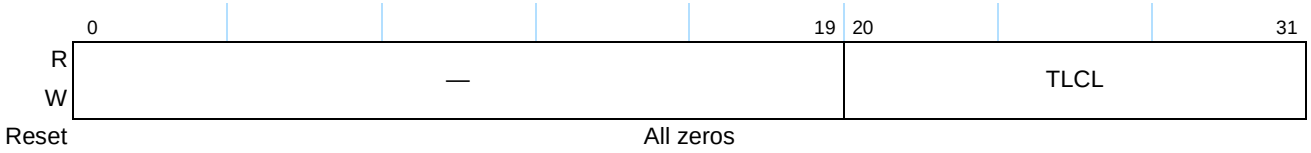


Figure 15-88. Transmit Late Collision Packet Counter Register Definition

Table 15-91 describes the fields of the TLCL register.

Table 15-91. TLCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TLCL	Transmit late collision packet counter. Increments for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the collision window field of the half-duplex [26:31] register.

15.5.3.6.35 Transmit Excessive Collision Packet Counter (TXCL)

Figure 15-89 describes the definition for the TXCL register.

Offset eTSEC1:0x2_4708; eTSEC2:0x2_5708; eTSEC3:0x2_6708; eTSEC4:0x2_7708 Access: Read/Write

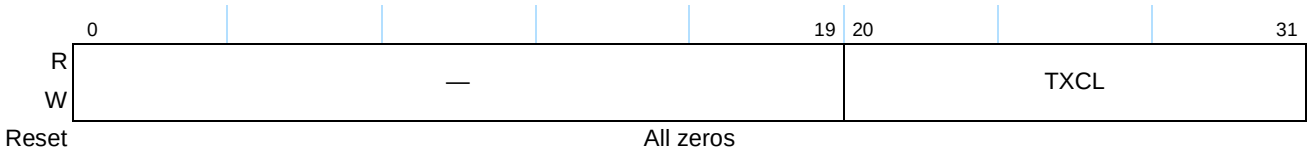


Figure 15-89. Transmit Excessive Collision Packet Counter Register Definition

Table 15-92 describes the fields of the TXCL register.

Table 15-92. TXCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCL	Transmit excessive collision packet counter. Increments for each frame that experienced 16 collisions during transmission and was aborted.

15.5.3.6.36 Transmit Total Collision Counter (TNCL)

Figure 15-90 describes the definition for the TNCL register.

Offset eTSEC1:0x2_470C; eTSEC2:0x2_570C; eTSEC3:0x2_670C; eTSEC4:0x2_770C — Access: Read/Write

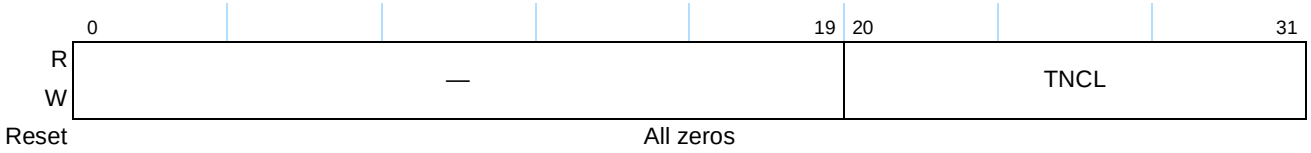


Figure 15-90. Transmit Total Collision Counter Register Definition

Table 15-93 describes the fields of the TNCL register.

Table 15-93. TNCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TNCL	Transmit total collision counter. Increments by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (That is, transmitting and receiving at the same time). **Note:** This count does not include collisions that result in an excessive collision condition.

15.5.3.6.37 Transmit Drop Frame Counter (TDRP)

Figure 15-91 describes the definition for the TDRP register.

Offset eTSEC1:0x2_4714; eTSEC2:0x2_5714; eTSEC3:0x2_6714; eTSEC4:0x2_7714 — Access: Read/Write

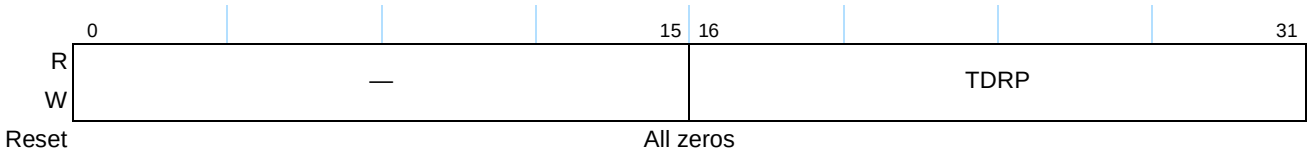


Figure 15-91. Transmit Drop Frame Counter Register Definition

Table 15-94 describes the fields of the TDRP register.

Table 15-94. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time a memory error or an underrun has occurred.

15.5.3.6.38 Transmit Jabber Frame Counter (TJBR)

Figure 15-92 describes the definition for the TJBR register.

Offset eTSEC1:0x2_4718; eTSEC2:0x2_5718; eTSEC3:0x2_6718; eTSEC4:0x2_7718 Access: Read/Write

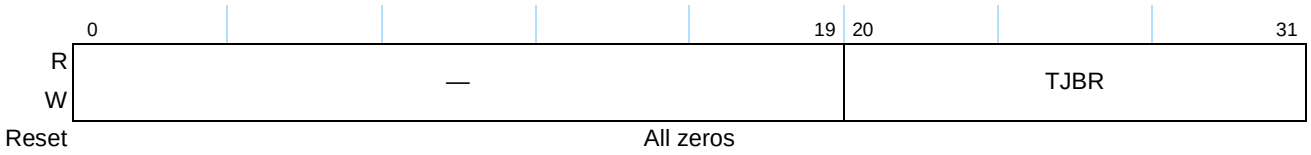


Figure 15-92. Transmit Jabber Frame Counter Register Definition

Table 15-95 describes the fields of the TJBR register.

Table 15-95. TJBR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TJBR	Transmit jabber frame counter. Increments for each oversized transmitted frame with an incorrect FCS value.

15.5.3.6.39 Transmit FCS Error Counter (TFCS)

Figure 15-93 describes the definition for the TFCS register.

Offset eTSEC1:0x2_471C; eTSEC2:0x2_571C; eTSEC3:0x2_671C; eTSEC4:0x2_771C Access: Read/Write

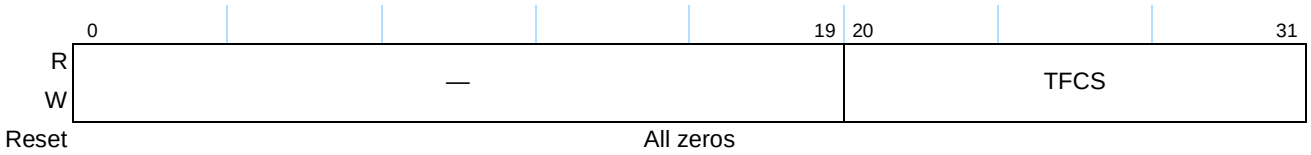


Figure 15-93. Transmit FCS Error Counter Register Definition

Table 15-96 describes the fields of the TFCS register.

Table 15-96. TFCS Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFCS	Transmit FCS error counter. Increments for every valid sized packet with an incorrect FCS value.

15.5.3.6.40 Transmit Control Frame Counter (TXCF)

Figure 15-94 describes the definition for the TXCF register.

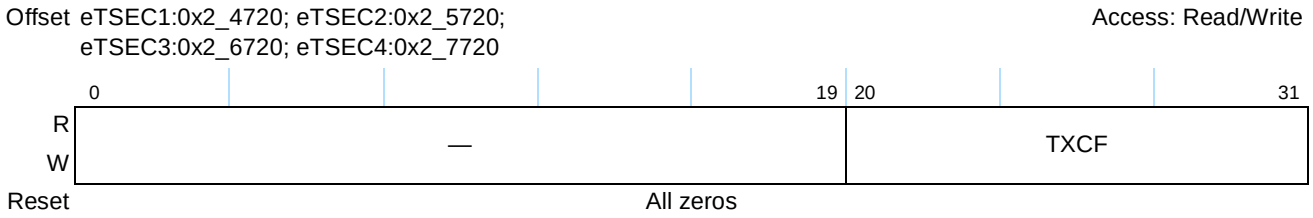


Figure 15-94. Transmit Control Frame Counter Register Definition

Table 15-97 describes the fields of the TXCF register.

Table 15-97. TXCF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCF	Transmit control frame counter. Increments for every control frame with valid CRC and of lengths 64 to 1518 (non VLAN) or 1522 (VLAN).

15.5.3.6.41 Transmit Oversize Frame Counter (TOVR)

Figure 15-95 describes the definition for the TOVR register.

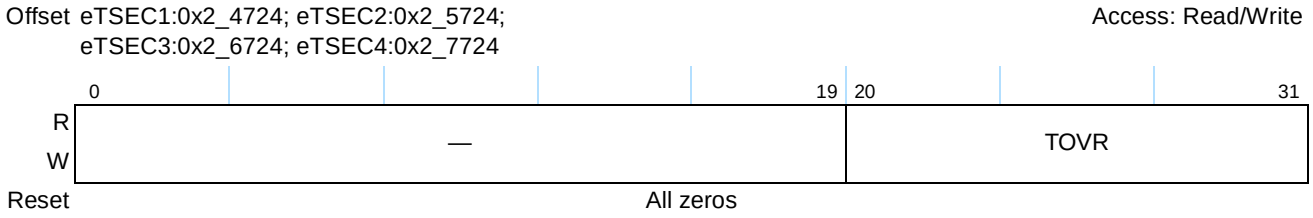


Figure 15-95. Transmit Oversized Frame Counter Register Definition

Table 15-98 describes the fields of the TOVR register.

Table 15-98. TOVR Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TOVR	Transmit oversize frame counter. Increments for each oversized transmitted frame with a correct FCS value.

15.5.3.6.42 Transmit Undersize Frame Counter (TUND)

Figure 15-96 describes the definition for the TUND register.

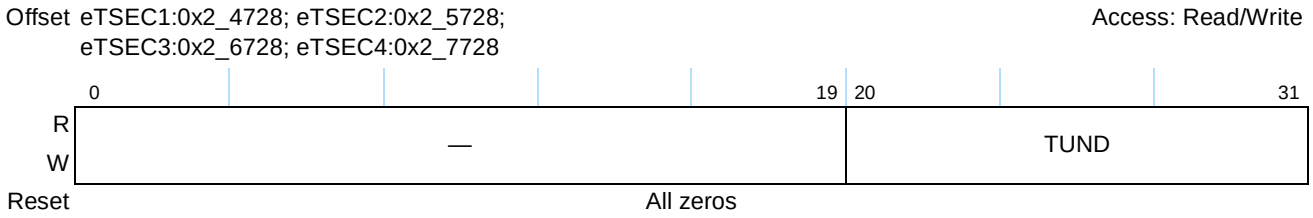


Figure 15-96. Transmit Undersize Frame Counter Register Definition

Table 15-99 describes the fields of the TUND register.

Table 15-99. TUND Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TUND	Transmit undersize frame counter. Increments for every frame less then 64 bytes, with a correct FCS value.

15.5.3.6.43 Transmit Fragment Counter (TFRG)

Figure 15-97 describes the definition for the TFRG register.

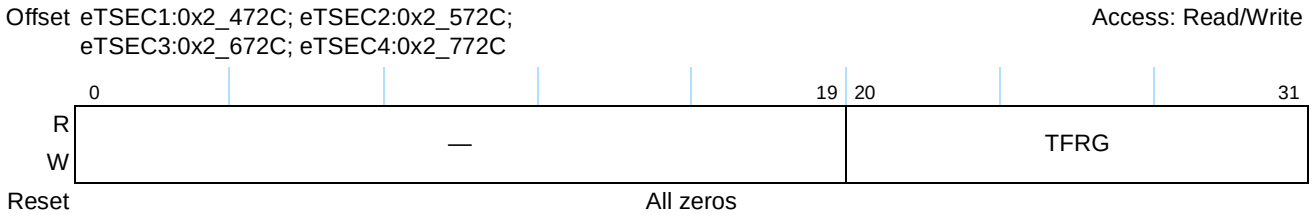


Figure 15-97. Transmit Fragment Counter Register Definition

Table 15-100 describes the fields of the TFRG register.

Table 15-100. TFRG Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less then 64 bytes, with an incorrect FCS value.

15.5.3.6.44 Carry Register 1 (CAR1)

Carry register bits are cleared on carry register writes when the respective bits are set. Figure 15-98 describes the definition for the CAR1 register.

Offset eTSEC1:0x2_4730; eTSEC2:0x2_5730; eTSEC3:0x2_6730; eTSEC4:0x2_7730 — Access: Read/Write

	0	1	2	3	4	5	6	7	13	14	15
R W	C1 64	C1 127	C1 255	C1 511	C1 1K	C1 MAX	C1 MGV	—		C1 REJ	C1 RBY
Reset	All zeros										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1 RXP	C1 RXU	C1 RAL	C1 RFL	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	C1 RDR
Reset	All zeros															

Figure 15-98. Carry Register 1 (CAR1) Register Definition

Table 15-101 describes the fields of the CAR1 register.

Table 15-101. CAR1 Field Descriptions

Bits	Name	Description
0	C164	Carry register 1 TR64 counter carry bit
1	C1127	Carry register 1 TR127 counter carry bit
2	C1255	Carry register 1 TR255 counter carry bit
3	C1511	Carry register 1 TR511 counter carry bit
4	C11K	Carry register 1 TR1K counter carry bit
5	C1MAX	Carry register 1 TRMAX counter carry bit
6	C1MGV	Carry register 1 TRMGV counter carry bit
7–13	—	Reserved
14	C1REJ	Carry register 1 RREJ counter carry bit
15	C1RBY	Carry register 1 RBYT counter carry bit
16	C1RPK	Carry register 1 RPKT counter carry bit
17	C1RFC	Carry register 1 RFCS counter carry bit
18	C1RMC	Carry register 1 RMCA counter carry bit
19	C1RBC	Carry register 1 RBCA counter carry bit
20	C1RXC	Carry register 1 RXCF counter carry bit
21	C1RXP	Carry register 1 RXPF counter carry bit
22	C1RXU	Carry register 1 RXUO counter carry bit
23	C1RAL	Carry register 1 RALN counter carry bit
24	C1RFL	Carry register 1 RFLR counter carry bit

Table 15-101. CAR1 Field Descriptions (continued)

Bits	Name	Description
25	C1RCD	Carry register 1 RCDE counter carry bit
26	C1RCS	Carry register 1 RCSE counter carry bit
27	C1RUN	Carry register 1 RUND counter carry bit
28	C1ROV	Carry register 1 ROVR counter carry bit
29	C1RFR	Carry register 1 RFRG counter carry bit
30	C1RJB	Carry register 1 RJBR counter carry bit
31	C1RDR	Carry register 1 RDRP counter carry bit

15.5.3.6.45 Carry Register 2 (CAR2)

Figure 15-99 describes the definition for the CAR2 register.

Offset eTSEC1:0x2_4734; eTSEC2:0x2_5734; eTSEC3:0x2_6734; eTSEC4:0x2_7734 — Access: Read/Write

	0												12	13	14	15
R	—												C2 TJB	C2 TFC	C2 TCF	C2 TOV
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	—	C2 TDP
W																
Reset	All zeros															

Figure 15-99. Carry Register 2 (CAR2) Register Definition

Carry register bits are cleared on carry register write when the respective bits are set. Table 15-102 describes the fields of the CAR2 register.

Table 15-102. CAR2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	C2TJB	Carry register 2 TJBR counter carry bit
13	C2TFC	Carry register 2 TFCS counter carry bit
14	C2TCF	Carry register 2 TXCF counter carry bit
15	C2TOV	Carry register 2 TOVR counter carry bit
16	C2TUN	Carry register 2 TUND counter carry bit
17	C2TFG	Carry register 2 TFRG counter carry bit
18	C2TBY	Carry register 2 TBYT counter carry bit
19	C2TPK	Carry register 2 TPKT counter carry bit

Table 15-102. CAR2 Field Descriptions (continued)

Bits	Name	Description
20	C2TMC	Carry register 2 TMCA counter carry bit
21	C2TBC	Carry register 2 TBCA counter carry bit
22	C2TPF	Carry register 2 TXPF counter carry bit
23	C2TDF	Carry register 2 TDFR counter carry bit
24	C2TED	Carry register 2 TEDF counter carry bit
25	C2TSC	Carry register 2 TSCL counter carry bit
26	C2TMA	Carry register 2 TMCL counter carry bit
27	C2TLC	Carry register 2 TLCL counter carry bit
28	C2TXC	Carry register 2 TXCL counter carry bit
29	C2TNC	Carry register 2 TNCL counter carry bit
30	—	Reserved, should be cleared
31	C2TDP	Carry register 2 TDRP counter carry bit

15.5.3.6.46 Carry Mask Register 1 (CAM1)

While one of the below mask bits are cleared, the corresponding carry bit in CAR1 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits all default to a set state. Figure 15-100 describes the definition for the CAM1 register.

Offset eTSEC1:0x2_4738; eTSEC2:0x2_5738; eTSEC3:0x2_6738; eTSEC4:0x2_7738 — Access: Read/Write

	0	1	2	3	4	5	6	7						13	14	15
R	M1	M1	M1	M1	M1	M1	M1								M1	M1
W	64	127	255	511	1K	MAX	MGV	—							REJ	RBY
Reset	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1	M1
W	RPK	RFC	RMC	RBC	RXC	RXP	RXU	RAL	RFL	RCD	RCS	RUN	ROV	RFR	RJB	RDR
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 15-100. Carry Mask Register 1 (CAM1) Register Definition

Table 15-103 describes the fields of the CAM1 register.

Table 15-103. CAM1 Field Descriptions

Bits	Name	Description
0	M164	Mask register 1 TR64 counter carry bit mask
1	M1127	Mask register 1 TR127 counter carry bit mask
2	M1255	Mask register 1 TR255 counter carry bit mask

Table 15-103. CAM1 Field Descriptions (continued)

Bits	Name	Description
3	M1511	Mask register 1 TR511 counter carry bit mask
4	M11k	Mask register 1 TR1K counter carry bit mask
5	M1MAX	Mask register 1 TRMAX counter carry bit mask
6	M1MGV	Mask register 1 TRMGV counter carry bit mask
7–13	—	Reserved
14	M1REJ	Mask register 1 RREJ counter carry bit mask
15	M1RBY	Mask register 1 RBYT counter carry bit mask
16	M1RPK	Mask register 1 RPKT counter carry bit mask
17	M1RFC	Mask register 1 RFCS counter carry bit mask
18	M1RMC	Mask register 1 RMCA counter carry bit mask
19	M1RBC	Mask register 1 RBCA counter carry bit mask
20	M1RXC	Mask register 1 RXCF counter carry bit mask
21	M1RXP	Mask register 1 RXPF counter carry bit mask
22	M1RXU	Mask register 1 RXUO counter carry bit mask
23	M1RAL	Mask register 1 RALN counter carry bit mask
24	M1RFL	Mask register 1 RFLR counter carry bit mask
25	M1RCD	Mask register 1 RCDE counter carry bit mask
26	M1RCS	Mask register 1 RCSE counter carry bit mask
27	M1RUN	Mask register 1 RUND counter carry bit mask
28	M1ROV	Mask register 1 ROVR counter carry bit mask
29	M1RFR	Mask register 1 RFRG counter carry bit mask
30	M1RJB	Mask register 1 RJBR counter carry bit mask
31	M1RDR	Mask register 1 RDRP counter carry bit mask

15.5.3.6.47 Carry Mask Register 2 (CAM2)

While one of the below mask bits are cleared, the corresponding carry bit in CAR2 is allowed to cause interrupt indications in register IEVENT[MSR0]. These bits default to a set state. Figure 15-101 describes the definition for the CAM2 register.

Offset eTSEC1:0x2_473C; eTSEC2:0x2_573C; eTSEC3:0x2_673C; eTSEC4:0x2_773C — Access: Read/Write

	0											11	12	13	14	15
R W	—												M2 TJB	M2 TFC	M2 TCF	M2 TOV
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	—	M2 TDP
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Figure 15-101. Carry Mask Register 2 (CAM2) Register Definition

Table 15-104 describes the fields of the CAM2 register.

Table 15-104. CAM2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	M2TJB	Mask register 2 TJBR counter carry bit mask
13	M2TFC	Mask register 2 TFCS counter carry bit mask
14	M2TCF	Mask register 2 TXCF counter carry bit mask
15	M2TOV	Mask register 2 TOVR counter carry bit mask
16	M2TUN	Mask register 2 TUND counter carry bit mask
17	M2TFG	Mask register 2 TFRG counter carry bit mask
18	M2TBY	Mask register 2 TBYT counter carry bit mask
19	M2TPK	Mask register 2 TPKT counter carry bit mask
20	M2TMC	Mask register 2 TMCA counter carry bit mask
21	M2TBC	Mask register 2 TBCA counter carry bit mask
22	M2TPF	Mask register 2 TXPF counter carry bit mask
23	M2TDF	Mask register 2 TDFR counter carry bit mask
24	M2TED	Mask register 2 TEDF counter carry bit mask
25	M2TSC	Mask register 2 TSCL counter carry bit mask
26	M2TMA	Mask register 2 TMCL counter carry bit mask
27	M2TLC	Mask register 2 TLCL counter carry bit mask
28	M2TXC	Mask register 2 TXCL counter carry bit mask

Table 15-104. CAM2 Field Descriptions (continued)

Bits	Name	Description
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved
31	M2TDP	Mask register 2 TDRP counter carry bit mask

15.5.3.6.48 Receive Filer Rejected Packet Counter (RREJ)

Figure 15-102 describes the definition for the RREJ register.

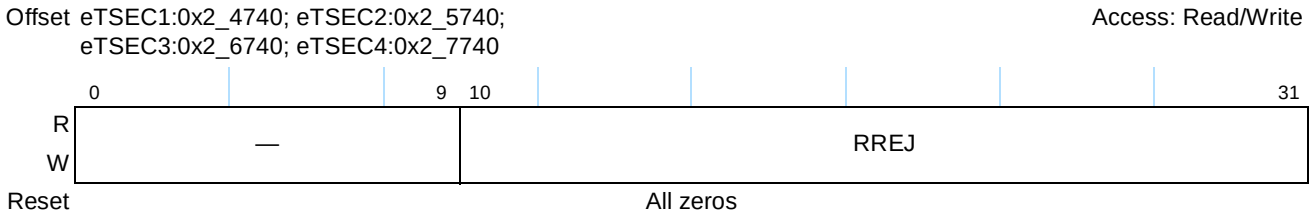


Figure 15-102. Receive Filer Rejected Packet Counter Register Definition

Table 15-69 describes the fields of the RREJ register.

Table 15-105. RREJ Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RREJ	Receive filer rejected packet counter. Increments for each frame with valid CRC received, but rejected by the receive queue filer—either due to a matching rule that asserted the REJ flag or due to filing to a RxBD ring that was not enabled (see IEVENT[FIQ] error).

15.5.3.7 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. The DA field of every received frame is processed through a 32-bit CRC generator (CRC-32 polynomial), and the 8 or 9 most significant bits of the CRC are mapped to a hash table entry. The user can enable a hash entry by setting its bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Software may need to further filter the address in order to eliminate false-positive hits in the hash table.

If RCTRL[GHTX] = 0, the 8 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 comprise a 256-entry hash table exclusively for individual (unicast) address matching, while registers GADDR0–GADDR7 comprise a 256-entry hash table for group (multicast) address matching. If RCTRL[GHTX] = 1, the group hash table is extended to all 512 entries, and the 9 most significant bits of the CRC are used as the hash table index. In this case, registers IGADDR0–IGADDR7 hold hash table entries 0–255 for group addresses, while registers GADDR0–GADDR7 hold entries 256–511 of the extended group hash table.

See Section 15.6.3.7.2, “Hash Table Algorithm” for more information on the hash algorithm.

15.5.3.7.1 Individual/Group Address Registers 0–7 (IGADDR*n*)

The IGADDR*n* registers are written by the user. Together these registers represent, depending on RCTRL[GHTX], either the 256 entries of the individual address hash table, or the first 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC-32 result points to an enabled hash entry.

Figure 15-103 describes the definition for the IGADDR*n* register.

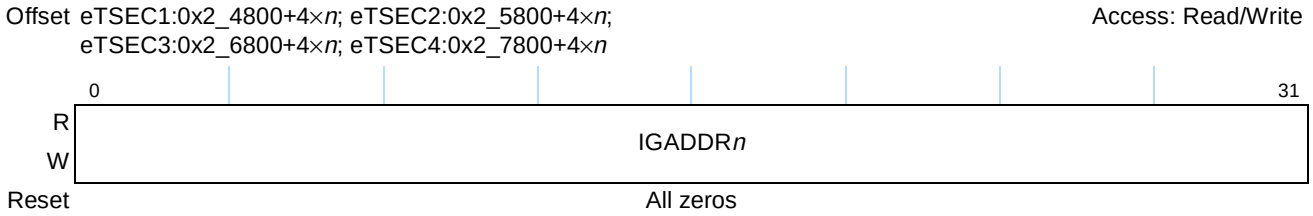


Figure 15-103. IGADDR*n* Register Definition

Table 15-107 describes the fields of the IGADDR*n* register.

Table 15-106. IGADDR*n* Field Descriptions

Bits	Name	Description
0–31	IGADDR*n*	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, IGADDR0 contains entries 0–31 of the 256-entry individual hash table and IGADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, IGADDR0 contains entries 0–31 of the 512-entry extended group hash table and IGADDR7 represents entries 224–255.

15.5.3.7.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers are written by the user. Together these registers represent, depending on RCTRL[GHTX], either the 256 entries of the group address hash table, or the last 256 entries of the extended group address hash table used in the address recognition process. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 15-104 describes the definition for the GADDR*n* register.

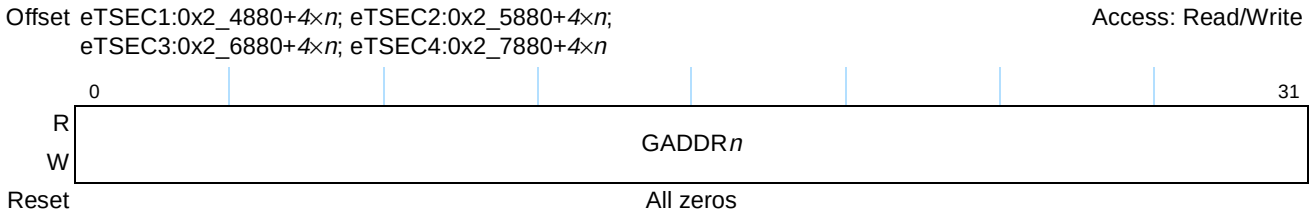


Figure 15-104. GADDR*n* Register Definition

Table 15-107 describes the fields of the GADDR*n* register.

Table 15-107. GADDR*n* Field Descriptions

Bits	Name	Description
0–31	GADDR*n*	Represents the 32-bit value associated with the corresponding register. When RCTRL[GHTX] = 0, GADDR0 contains entries 0–31 of the 256-entry group hash table and GADDR7 represents entries 224–255. When RCTRL[GHTX] = 1, GADDR0 contains entries 256–287 of the 512-entry extended group hash table and GADDR7 represents entries 480–511.

15.5.3.8 FIFO Registers

This section provides detailed descriptions of the registers used to configure the FIFO interface. All of the registers are 32 bits wide. The ECNTRL[FIFM] bit is set to indicate that data transfers take place over this interface. Please refer to Section 15.6.2, "Connecting to FIFO Interfaces," for details of the signaling protocols available.

15.5.3.8.1 FIFO Configuration Register (FIFOCFG)

The FIFO Configuration Register configures and enables the 8/16-bit FIFO interface.

The FIFO Configuration Register configures and enables the 8-bit FIFO interface.

Figure 15-105 describes the definition for the FIFOCFG register.

Offset eTSEC1:0x2_4A00; eTSEC2:0x2_5A00; eTSEC3:0x2_6A00; eTSEC4:0x2_7A00 — Access: Read/Write

	0							7	8							15
R	—								IPG							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	22	23	24	25	26	27	28	29	30	31
R	RRX	RTX	RXE	TXE	—		LPB	RFC	TFC	FFC	CRCAPP	CRCCHK	—	SIGM	
W															
Reset	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Figure 15-105. FIFOCFG Register Definition

Table 15-108 describes the fields of the FIFOCFG register.

Table 15-108. FIFOCFG Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IPG	Minimum inter packet gap. This sets the minimum number of cycles inserted between back-to-back frames transmitted over the FIFO interface. The minimum required is 3 cycles if CRCAPP=0, 5 cycles for 16-bit interfaces if CRCAPP=1 and 7 cycles for 8-bit interfaces if CRCAPP=1.
16	RRX	Enable reset of FIFO receive function. 0 Do not reset the FIFO receiver. 1 Reset the FIFO receiver for as long as this bit is set.

Table 15-108. FIFOCFG Field Descriptions (continued)

Bits	Name	Description
17	RTX	Enable reset of FIFO transmit function. 0 Do not reset the FIFO transmitter. 1 Reset the FIFO transmitter for as long as this bit is set.
18	RXE	Enable FIFO receive function. 0 Disable reception over the FIFO interface, ignoring data presented to the signals. 1 Enable normal reception over the FIFO interface.
19	TXE	Enable FIFO transmit function. 0 Disable transmission over the FIFO interface. 1 Enable normal transmission over the FIFO interface.
20–22	—	Reserved.
23	LPB	Loopback enable. 0 Do not loopback data in the FIFO interface. 1 Loopback transmitted data to the FIFO receiver rather than outputting transmitted data to signals.
24	RFC	Enable receive flow control. Setting FFC overrides this bit. 0 Do not allow the FIFO receiver to assert link-level flow control if eTSEC requires it. 1 Allow the FIFO receiver to assert link-level flow control if eTSEC requires it. This is the default setting.
25	TFC	Enable transmit flow control. 0 Do not allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable, resulting in underruns. 1 Allow the FIFO transmitter to assert link-level flow control if transmit data is unavailable and SIGM = 01. This is the default setting.
26	FFC	Force flow control. This can be used by software to stop reception on the FIFO interface. 0 Do not assert link-level flow control via the RXFC signal unless eTSEC requires flow control. 1 Force flow control on the RXFC signal in encoded FIFO packet mode regardless of eTSEC pause requirements.
27	CRCAPP	Append a CRC (CRC-32 algorithm, as per IEEE 802.3) to the end of every transmitted frame. 0 Do not automatically append a CRC to transmitted frames. Allow TxBD[TC], if set, to control when a CRC is appended. 1 Automatically append a CRC to transmitted frames. Ignore TxBD[TC].
28	CRCCHK	Check the CRC (CRC-32 algorithm, as per IEEE 802.3) at the end of every frame. 0 Do not automatically check the last 4 bytes of received frames for a valid CRC. 1 Automatically check the last 4 bytes of received frames for a valid CRC. If a CRC error is detected, or insufficient data is received to recover the CRC, the RxBD[CR] bit is set.
29	—	Reserved
30–31	SIGM	FIFO signaling mode. Determines how the GMII signals are interpreted as framing signals. 00 GMII style mode. 01 Encoded packet mode. 10 Reserved 11 Reserved

15.5.3.9 DMA Attribute Registers

This section describes the two eTSEC DMA attribute registers.

15.5.3.9.1 Attribute Register (ATTR)

The attribute register defines memory access attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data. Buffer descriptors may be written with attributes that cause allocation into the L2 cache. Similarly, broad sections of a receive frame header may have attributes attached that cause allocation in the L2 cache. This process of specifying a region of each frame to stash into the L2 cache is referred to as *extraction*, which is specified in conjunction with register ATTRELI. ATTR[ELCWT] only has meaning if ATTRELI[EL] is non-zero. It is important to note that even though portions of received frames may be stashed to L2 cache, this is only a performance optimization as entire frames are still written to off-chip memory regardless of settings in ATTR.

Figure 15-106 describes the definition for the ATTR register.

Offset eTSEC1:0x2_4BF8; eTSEC2:0x2_5BF8; eTSEC3:0x2_6BF8; eTSEC4:0x2_7BF8 — Access: Read/Write

	0–16	17–18	19	20–21	22–23	24	25	26–31
R/W	—	ELCWT	—	BDLWT	—	RDSEN	RBDSEN	—
Reset	All zeros							

Figure 15-106. ATTR Register Definition

Table 15-109 describes the fields of the ATTR register.

Table 15-109. ATTR Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Specifies the write transaction type to perform for the extracted data. For maximum performance, it is recommended that if ELCWT is set to allocate, BDLWT should also be set to allocate.Writes to cache are always performed with snoop. 00 No allocation performed. 01 Reserved 10 Allocate L2 cache line. 11 Reserved.
19	—	Reserved
20–21	BDLWT	Buffer descriptor L2 cache write type. specifies the write transaction type to perform for the bufferdescriptor for a receive frame. Writes to cache are always performed with snoop. 00 No allocation performed. 01 Reserved 10 Allocate L2 cache line. 11 Reserved.
22–23	—	Reserved

Table 15-109. ATTR Field Descriptions (continued)

Bits	Name	Description
24	RDSEN	Rx data snoop enable. 0 Disables snooping of all receive frames data to memory unless ELCWT specifies L2allocation. 1 Enables snooping of all receive frames data to memory.
25	RBDSEN	RxBD snoop enable. 0 Disables snooping of all receive BD memory accesses unless BDLWT specifies L2 allocation. 1 Enables snooping of all receive BD memory accesses.
26–31	—	Reserved

15.5.3.9.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length for extracting received frames. The extract length is typically set to the expected length of extracted packet headers. Figure 15-107 describes the definition for the ATTRELI register.

Offset eTSEC1:0x2_4BFC; eTSEC2:0x2_5BFC; eTSEC3:0x2_6BFC; eTSEC4:0x2_7BFC — Access: Read/Write

	0	1	2 – 12	13 – 17	18 – 25	26 – 31
R W	—		EL	—	EI	—
Reset	All zeros					

Figure 15-107. ATTRELI Register Definition

Table 15-110 describes the fields of the ATTRELI register.

Table 15-110. ATTRELI Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–12	EL	Extracted length. Specifies the number of bytes, as a multiple of 8 bytes, to extract from the receive frame. The DMA controller uses this field to perform extraction. If set to zero, no extraction is performed.
13–15	—	To ensure that EL is a multiple of 8 bytes, these bits should be written with zero.
16–17	—	Reserved
18–25	EI	Extracted index. Points to the first byte, as a multiple of 64 bytes, within the receive frame from which to begin extracting data.
26–31	—	To ensure that EI is a multiple of 8 bytes, these bits should be written with zero.

15.5.3.10 Lossless Flow Control Configuration Registers

When enabled via RCTRL[LFC], the eTSEC tracks location of the last free BD in each Rx BD ring via the value of RFBPTR*n*. Using this pointer and the ring length stored in RQPRM*n*[LEN], the eTSEC continuously calculates the number of free BDs in the ring. Whenever the calculated number of free BDs in the ring drops below the pause threshold specified in RQPRM*n*[FBTHR], the eTSEC issues link layer flow control. It continue to assert flow control until the free BD count for each active ring reaches or

exceeds RQPRM*n*[FBTHR]. See Section 15.6.6.1, “Back Pressure Determination via Free Buffers,” for the theory of operation of these registers.

15.5.3.10.1 Receive Queue Parameters 0–7 (RQPRM0–PQPRM7)

The RQPRM*n* registers specify the minimum number of BDs required to prevent flow control being asserted and the total number of Rx BDs in their respective ring. Whenever the free BD count calculated by the eTSEC for any active ring drops below the value of RQPRM*n*[FBTHR] for that ring, link level flow control is asserted. Software must not write to RQPRM*n* while LFC is enabled and the eTSEC is actively receiving frames. However, software may modify these registers after disabling LFC by clearing RCTRL[LFC]. Note that packets may be lost due to lack of RxBDs while RCTRL[LFC] is clear. Software can prevent packet loss by manually generating pause frames (via TCTRL[TFC_PAUSE]) to cover the time when RCTRL[LFC] is clear. Figure 15-108 describes the definition for the RQPRM*n* register.

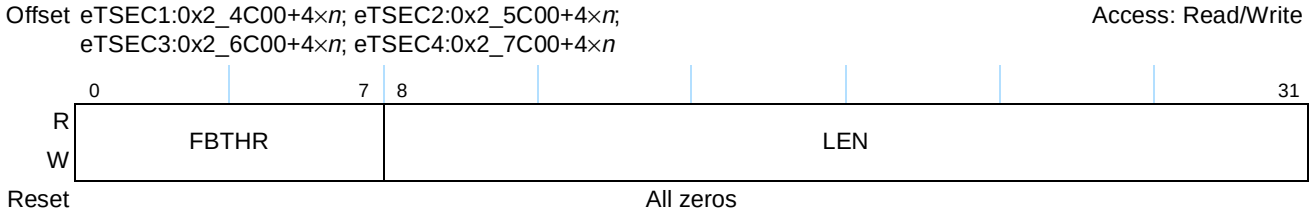


Figure 15-108. RQPRM Register Definition

Table 15-111 describes the fields of the RQPRM register.

Table 15-111. RQPRM Field Descriptions

Bits	Name	Description
0–7	FBTHR	Free BD threshold. Minimum number of BDs required for normal operation. If the eTSEC calculated number of free BDs drops below this threshold, link layer flow control is asserted.
8–31	LEN	Ring length. Total number of Rx BDs in this ring.

15.5.3.10.2 Receive Free Buffer Descriptor Pointer Registers 0–7 (RFBPTR0–RFBPTR7)

The RFBPTR*n* registers specify the location of the last free buffer descriptor in their respective ring. These registers live in the same 32b address space – and must share the same 4 most significant bits – as RBPTR*n*. That is, RFBPTR*n* and its associated RBPTR*n* must remain in the same 256MB page. Like RBPTR*n*, whenever RBASE*n* is updated, RFBPTR*n* is initialized to the value of RBASE*n*. This indicates that the ring is completely empty. As buffers are freed and their respective BDs are returned (by setting the EMPTY bit) to the ring, software is expected to update this register. The eTSEC then performs modulo arithmetic involving RBASE*n*, RBPTR*n* and RFBPTR*n* to determine the number of free BDs remaining in the ring. If, at any stage, the value written to RFBPTR*n* matches that of the respective RBPTR*n* the eTSEC free BD calculation assumes that the ring is now completely empty. For more information on the recommended use of these registers, see Section 15.6.6.1, “Back Pressure Determination via Free Buffers.” Figure 15-109 describes the definition for the RFBPTR*n* register.

Offset eTSEC1:0x2_4C40+8×*n*; eTSEC2:0x2_5C40+8×*n*; eTSEC3:0x2_6C40+8×*n*; eTSEC4:0x2_7C40+8×*n* Access: Read/Write

	0	28	29	31
R W	RFBPTR*n*		—	
Reset	All zeros			

Figure 15-109. RFBPTR0–RFBPTR7 Register Definition

Table 15-112 describes the fields of the RFBPTR*n* registers.

Table 15-112. RFBPTR0–RFBPTR7 Field Descriptions

Bits	Name	Description
0–28	RFBPTR	Pointer to the last free BD in RxBD Ring *n*. When RBASE*n* is updated, eTSEC initializes RFBPTR*n* to the value in the corresponding RBASE*n*. Software may update this register at any time to inform the eTSEC the location of the last free BD in the ring. Note that the 3 least-significant bits of this register are read only and zero.
29–31	—	Reserved.

15.5.3.11 Hardware Assist for IEEE 1588 Compatible Timestamping

IEEE 1588 compatible timestamping on this device is accomplished using the per-port transmit timestamping registers within each Ethernet controller memory space (See Section 15.5.3.2.12, "Transmit Time Stamp Identification Register (TMR_TXTS1–2_ID)," and Section 15.5.3.2.13, "Transmit Time Stamp Register (TMR_TXTS1–2_H/L).") in conjunction with the following common registers, which are located within the memory space for eTSEC1. Because the common 1588 timestamping registers exist within the eTSEC1 memory space, the eTSEC1 controller must remain enabled in order to use 1588 timestamping for any Ethernet port.

15.5.3.11.1 Timer Control Register (TMR_CTRL)

This register is used to reset, configure, and initialize the eTSEC precision timer clock. The control of all timer function is performed via programming eTSEC1.The register in eTSEC1 is shared for all eTSECs. Figure 15-7 describes the definition for the TMR_CTRL register.

Offset eTSEC1:0x2_4E00 Access: Mixed

	0	1	2	3	4	5	6	15
R W	ALM1P	ALM2P	—	FS	—		TCLK_PERIOD	
Reset	0	0	0	0	0	0	0 0 0 0 0 0 0 0	1

	16	21	22	23	24	25	26	27	28	29	30	31
R W	—		ETE P2	ETE P1	COPH	CIPH	TMSR	—	BYP	TE	CKSEL	
Reset	All zeros											

Figure 15-110. TMR_CTRL Register Definition

Table 15-113 describes the fields of the TMR_CTRL register.

Table 15-113. TMR_CTRL Register Field Descriptions

Bits	Name	Description
0	ALM1P	Alarm1 output polarity 0 active high output 1 active low output
1	ALM2P	Alarm2 output polarity 0 active high output 1 active low output
2	—	Reserved
3	FS	FIPER start indication 0 Fiper is enabled through timer enable 1 Fiper is enabled through timer enable and alarm indication.
4–5	—	Reserved
6–15	TCLK_ PERIOD	1588 timer reference clock period. The timer clock counter increments by TCLK_PERIOD every time the accumulator register overflows. This clock period must be larger than the clock period of the timer reference clock. For applications where user does not want the clock period to be added, they can program this field to 1 to count the clock ticks. This field defaulted to 1 to count overflow ticks.
16–21	—	Reserved
22	ETEP2	External trigger 2 edge polarity 0 Time stamp on the rising edge of the external trigger 1 Time stamp on the falling edge of the external trigger
23	ETEP1	External trigger 1 edge polarity 0 Time stamp on the rising edge of the external trigger 1 Time stamp on the falling edge of the external trigger
24	COPH	Generated clock (TSEC_1588_GCLK) output phase. 0 non-inverted divided clock is output 1 inverted divided clock is output
25	CIPH	External oscillator input clock phase. 0 non-inverted frequency tuned timer input clock 1 inverted frequency tuned timer input clock (NOTE: this setting is reserved if CKSEL=01.)
26	TMSR	Timer soft reset. When enabled, it resets all the timer registers and state machines. 0 normal operation 1 place entire timer in reset except control and config registers NOTE: Prior to initiating timer reset (setting TMSR), must gracefully stop receiver (See MACCFG1[RX_EN] description). User programmable registers are not reset by the soft reset; for example, TMR_CTRL, TMR_TEMASK, TMR_PEMASK, TMR_ADD, TMR_PRSC, TMROFF_H/L, TMR_ALARMn, and TMR_FIPERn.
27	—	Reserved
28	BYP	Bypass drift compensated clock 0 64-bit clock counter is incremented on the accumulator overflow 1 64-bit clock counter is directly driven from the external oscillator ignoring accumulator overflow

Table 15-113. TMR_CTRL Register Field Descriptions (continued)

Bits	Name	Description
29	TE	1588 timer enable. If not enabled, all the timer registers and state machines are disabled. 0 timer not enabled 1 timer enabled and resume normal operation
30–31	CKSEL	1588 Timer reference clock source select. 00 External high precision timer reference clock (TSEC_1588_CLK) 01 eTSEC system clock 10 eTSEC1 transmit clock 11 RTC clock input Note that the 1588 reference clock must be no slower than1/5 the Rx_clk frequency.

15.5.3.11.2 Timer Event Register (TMR_TEVENT)

The eTSEC precision timer implementation can generate additional interrupts that are independent of the frame based events that controlled via IEVENT. The timer interrupts are not affected by any interrupt coalescing that may be specified in TXIC/RXIC. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the event mask register (TEMASK), the event also causes a hardware interrupt at the PIC. A bit in the timer event register is cleared by writing a 1 to that bit position. Figure 15-4 describes the definition for the TMR_TEVENT register.

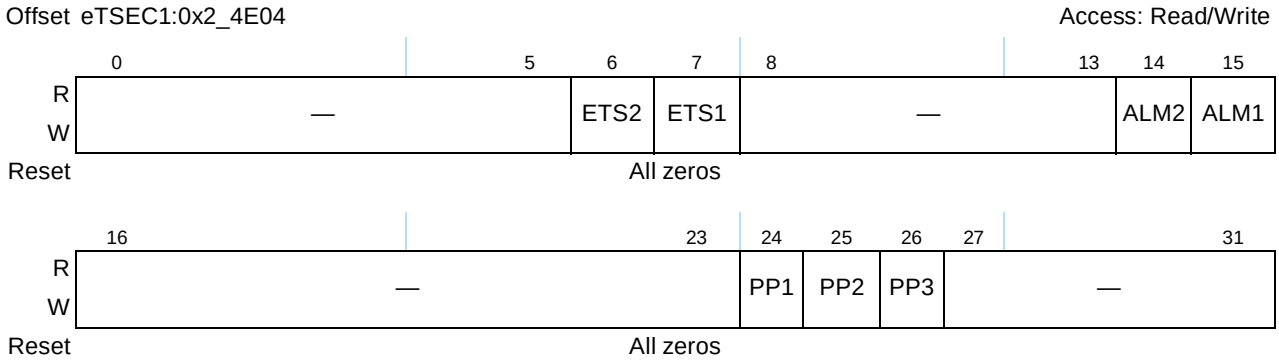


Figure 15-111. TMR_TEVENT Register Definition

Table 15-114 describes the fields of the TMR_TEVENT register fields for the timer.

Table 15-114. TMR_TEVENT Register Field Descriptions

Bits	Name	Description
0–6	—	Reserved
6	ETS2	External trigger 2 timestamp sampled 0 external trigger timestamp not sampled 1 external trigger timestamp sampled
7	ETS1	External trigger 1 timestamp sampled 0 external trigger timestamp not sampled 1 external trigger timestamp sampled
8–13	—	Reserved

Table 15-114. TMR_TEVENT Register Field Descriptions (continued)

Bits	Name	Description
14	ALM2	Current time equaled alarm time register 2 0 alarm time has not be reached yet 1 alarm time has been reached
15	ALM1	Current time equaled alarm time register 1 0 alarm time has not be reached yet 1 alarm time has been reached
16–23	—	Reserved
24	PP1	Indicates that a periodic pulse has been generated based on FIPER1 register. 0 periodic pulse not generated 1 periodic pulse generated
25	PP2	Indicates that a periodic pulse has been generated based on FIPER2 register. 0 periodic pulse not generated 1 periodic pulse generated
26	PP3	Indicates that a periodic pulse has been generated based on FIPER3 register. 0 periodic pulse not generated 1 periodic pulse generated
27–31	—	Reserved

15.5.3.11.3 Timer Event Mask Register (TMR_TEMASK)

Timer event mask register. The event mask register provides control over which possible interrupt events in the TMR_TEVENT register are permitted to participate in generating hardware interrupts to the PIC. All implemented bits in this register are R/W and cleared upon a hardware reset. Figure 15-115 describes the definition for the TMR_TEMASK register.

Offset eTSEC1:0x2_4E08 Access: Read/Write

	0	5	6	7	8	13	14	15
R W	—		ETS2EN	ETS1EN	—		ALM2EN	ALM1EN
Reset	All zeros							

	16	23	24	25	26	31
R W	—		PP1EN	PP2EN	—	
Reset	All zeros					

Table 15-115. TMR_TEMASK Register Definition

Table 15-116 describes the fields of the TMR_TEMASK register fields for the timer.

Table 15-116. TMR_TEMASK Register Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	ETS2EN	External trigger 2 timestamp sample event enable

Table 15-116. TMR_TEMASK Register Field Descriptions (continued)

Bits	Name	Description
7	ETS1EN	External trigger 1 timestamp sample event enable
8–13	—	Reserved
14	ALM2EN	Timer ALM1 event enable
15	ALM1EN	Timer ALM2 event enable
16–23	—	Reserved
24	PP1EN	Periodic pulse event 1 enable
25	PP2EN	Periodic pulse event 2 enable
26–31	—	Reserved

15.5.3.11.4 Timer PTP Packet Event Register (TMR_PEVENT)

The eTSEC precision timer logic can generate interrupts upon the capture of a timestamp due to either transmission or reception of a frame. If an event occurs and its corresponding enable bit is set in the event mask register (EMASK), the event also causes a hardware interrupt at the PIC. A bit in the timer event register is cleared by writing a 1 to that bit position. Figure 15-112 describes the definition for the TMR_PEVENT register.

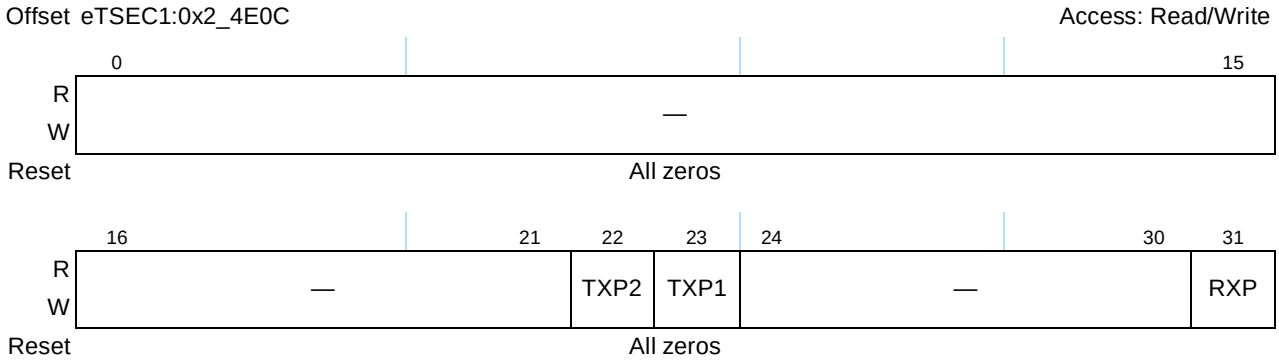


Figure 15-112. TMR_PEVENT Register Definition

Table 15-117 describes the fields of the TMR_PEVENT register fields for the timer.

Table 15-117. TMR_PEVENT Register Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	TXP2	Indicates that a PTP frame has been transmitted and its timestamp is stored in TXTS2 register. 0 PTP packet not transmitted 1 PTP packet has been transmitted
23	TXP1	Indicates that a PTP frame has been transmitted and its timestamp is stored in TXTS1 register. 0 PTP packet not transmitted 1 PTP packet has been transmitted

Table 15-117. TMR_PEVENT Register Field Descriptions (continued)

Bits	Name	Description
24–30	—	Reserved
31	RXP	Indicates that a PTP frame has been received 0 PTP packet not received 1 PTP packet has been received

15.5.3.11.5 Timer Event Mask Register (TMR_PEMASK)

Timer event mask register. The event mask register provides control over which possible interrupt events in the TMR_PEVENT register are permitted to participate in generating hardware interrupts to the PIC. All implemented bits in this register are R/W and cleared upon a hardware reset. Figure 15-113 describes the definition for the TMR_PEMASK register.

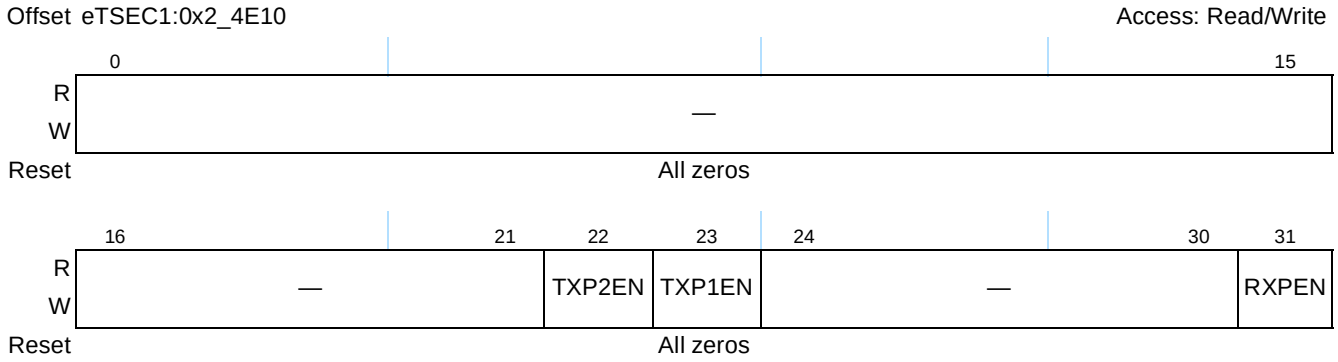


Figure 15-113. TMR_PEMASK Register Definition

Table 15-118 describes the fields of the TMR_PEMASK register fields for the timer.

Table 15-118. TMR_PEMASK Register Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	TXP2EN	Transmit PTP packet event 2 enable
23	TXP1EN	Transmit PTP packet event 1 enable
24–30	—	Reserved
31	RXPEN	Receive PTP packet event enable

15.5.3.11.6 Timer Status Register (TMR_STAT)

This register requires the eTSEC filer to be enabled (via RCTRL[FILREN]). When eTSEC generates an interrupt based on the timestamp event for a received packet, the queue ID which the incoming packet is sent to is captured in this register. This register update is synchronized with the RXF interrupt of the corresponding received packet. Writing 1 to any bit of this register clears it. Figure 15-119 describes the definition for the TMR_STAT register.

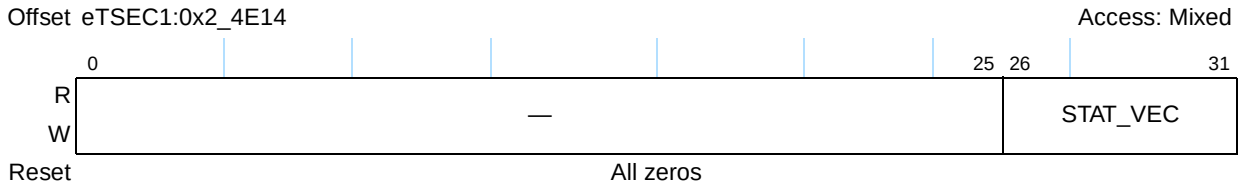


Table 15-119. TMR_STAT Register Definition

Table 15-120 describes the fields of the TMR_STAT register.

Table 15-120. TMR_STAT Register Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26–31	STAT_VEC	Timer general purpose status vector. It stores the 6-bit queue number generated by the filer. User to decode this status vector. For example, user can encode received PTP packet message types (Sync, Delay_req, Follow_up, Delay_resp, Management) in the filer virtual queue field.

15.5.3.11.7 Timer Counter Register (TMR_CNT_H/L)

The timer register (TMR_CNT_H/L) represents accurate time in terms clock ticks or in nano-seconds. Writes to these registers override the previous time. The register in eTSEC1 is shared for all eTSECs. This is a read/write register. Figure 15-114 describes the definition for the TMR_CNT_H/L register.

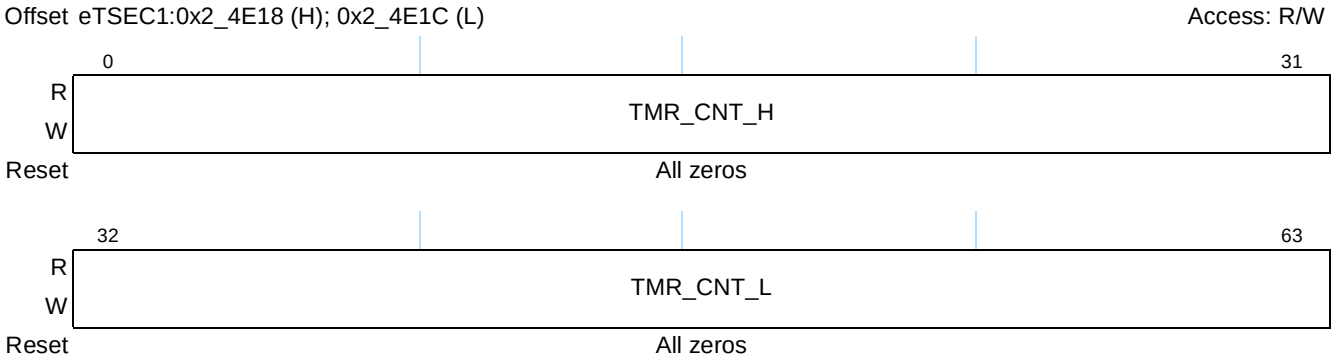


Figure 15-114. TMR_CNT_H Register Definition

Table 15-121 describes the fields of the TMR_CNT_H/L register.

Table 15-121. TMR_CNT_H/L Register Field Descriptions

Bits	Name	Description
0–63	TMR_CNT_H/L	Value of the current time counter. Current time is calculated by adding TMROFF_H/L with the TMR_CNT_H/L counter. This register can be written through the register writes.Writes to the TMR_CNT_L register copies the written value into the shadow TMR_CNT_L register. Writes to the TMR_CNT_H register copies the values written into the shadow TMR_CNT_H register. Contents of the shadow registers are copied into the TMR_CNT_L and TMR_CNT_H registers following a write into the TMR_CNT_H register. Writes to these registers have precedence over the timer increment. The user must write to TMR_CNT_L register first. Reads from the TMR_CNT_L register copies the entire 64-bit clock time of the read enable into the TMR_CNT_H/L shadow registers. Read instruction from the TMR_CNT_H register reads the value stored in the TMR_CNT_H shadow register. The user must read the TMR_CNT_L register first to get correct 64-bit TMR_CNT_H/L counter values.

15.5.3.11.8 Timer Drift Compensation Addend Register (TMR_ADD)

Timer drift compensation addend register (TMR_ADD) is used to hold timer frequency compensation value (FreqCompensationValue). The nominal frequency of the clock counter is determined by the FreqDivRatio and the clock frequency (FreqClock). This register is programmed with 2^{32}/FreqDivRatio. Frequency division ratio (FreqDivRatio) is the ratio between the frequency of the oscillator (TimerOsc) and the desired clock frequency (NominalFreq). FreqDivRatio is a design constant chosen to be greater than 1.0001. The ADDEND value is added to the 32-bit accumulator register at every rising edge of the oscillator clock (TimerOsc). The clock counter is incremented at every carry pulse of the accumulator. Only one of this register is required for the entire group of eTSECs. Figure 15-115 describes the definition of the TMR_ADD register.

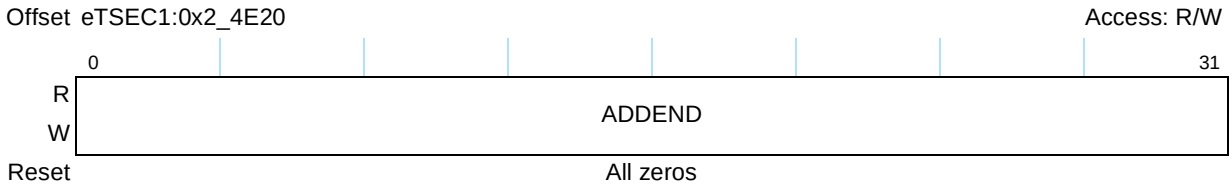


Figure 15-115. TMR_ADD Register Definition

Table 15-122 describes the fields of the TMR_ADD register fields for the timer.

Table 15-122. TMR_ADD Register Field Descriptions

Bits	Name	Description
0–31	ADDEND	Timer drift compensation addend register value. It is programmed with a value of 2^32/FreqDivRatio. For example, TimerOsc = 50 MHz NominalFreq = 40 MHz FreqDivRatio = 1.25 ADDEND = ceil(2^32/1.25) = 0xCCCC_CCCD

15.5.3.11.9 Timer Accumulator Register (TMR_ACC)

Timer accumulator register accumulates the value of the addend register into it. An overflow pulse of the accumulator is used to increment the timer clock by TMR_CTRL[TCLK_PERIOD]. This register is read only in normal operation. The register in eTSEC1 is shared for all eTSECs. Figure 15-116 describes the definition of the TMR_ACC register.

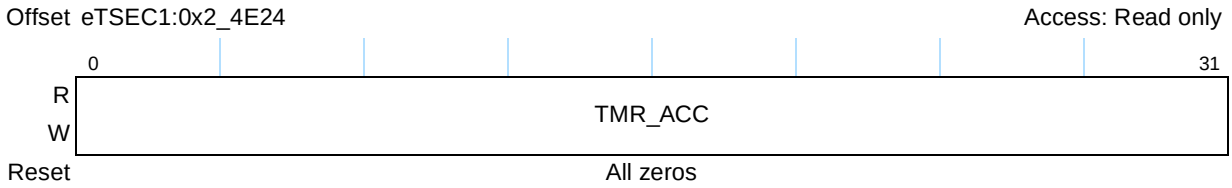


Figure 15-116. TMR_ACC Register Definition

Table 15-123 describes the fields of the TMR_ACC register.

Table 15-123. TMR_ACC Register Field Descriptions

Bits	Name	Description
0–31	TMR_ACC	32-bit timer accumulator register

15.5.3.11.10 Timer Prescale Register (TMR_PRSC)

Timer generated output clock prescale register. It is used to adjust output clock frequency that is put onto the 1588 clock output signal. The register in eTSEC1 is shared for all eTSECs. Figure 15-117 describes the definition for the TMR_PRSC register.

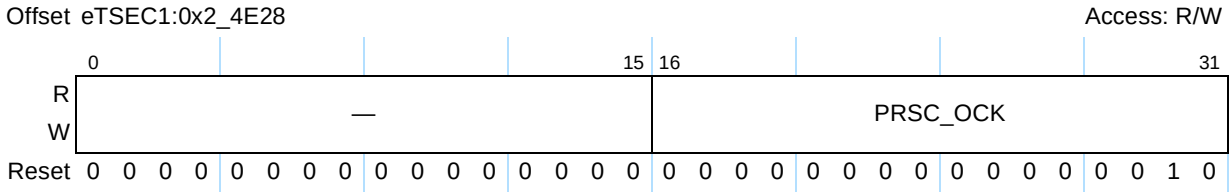


Figure 15-117. TMR_PRSC Register Definition

Table 15-124 describes the fields of the TMR_PRSC register.

Table 15-124. TMR_PRSC Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PRSC_OCK	Output clock division/prescale factor. Output clock is generated by dividing the timer input clock by this number. Programmed value in this field must be greater than 1. Any value less than 1 is treated as 2.

15.5.3.11.11 Timer Offset Register (TMROFF_H/L)

The timer offset register is used to provide current time by adding its value to the clock counter. Figure 15-118 describes the definition of the TMROFF_H/L register.

NOTE

All TMROFF_H registers in a device should be set to the same value, and all TMROFF_L registers in a device should be set to the same value. Otherwise, the precision time protocol may not work.

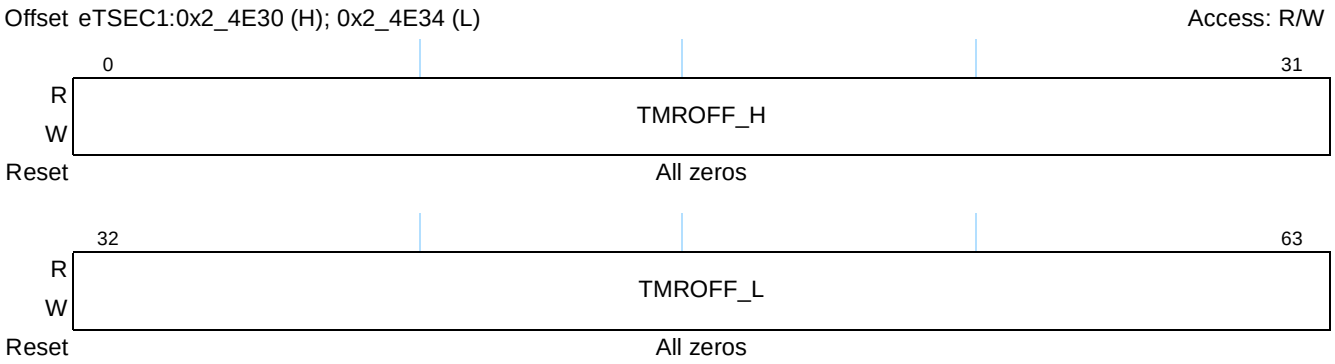


Figure 15-118. TMROFF_H/L Register Definition

Table 15-125 describes the fields of the TMROFF_H/L register.

Table 15-125. TMROFF_H/L Register Field Descriptions

Bits	Name	Description
0–63	TMROFF_H/L	Offset value of the clock counter. Current time in is calculated by adding TMROFF_H/L with the timer's counter TMR_CNT_H/L register.

15.5.3.11.12 Alarm Time Comparator Register (TMR_ALARM1–2_H/L)

Alarm time comparator register (TMR_ALARM*n*_H/L). This register holds alarm time for comparison with the current time counter. There are two of these registers for eTSEC1 which are shared amongst all eTSECs. Figure 15-119 describes the definition for the TMR_ALARM*n*_H/L register.

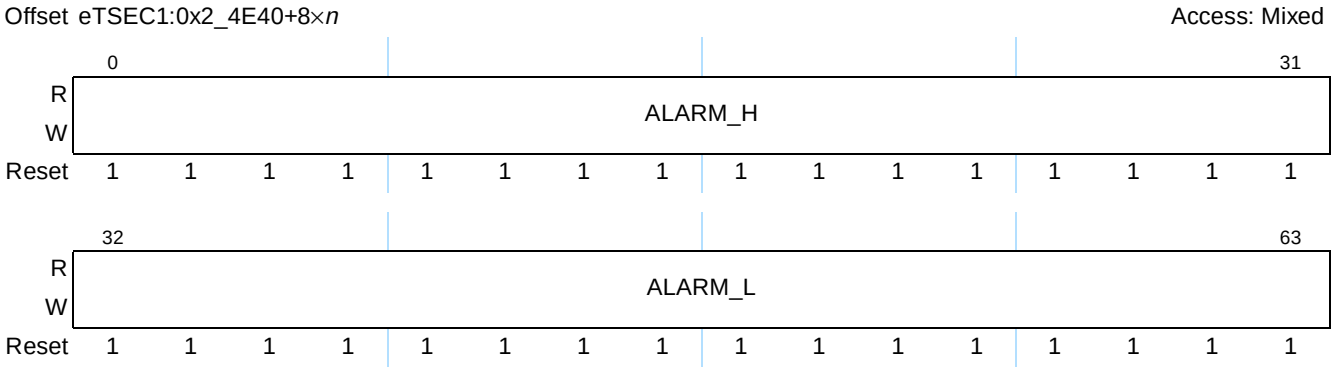


Figure 15-119. TMR_ALARM1-2_H/L Register Definition

Table 15-126 describes the fields of the TMR_ALARM*n*_H/L register.

Table 15-126. TMR_ALARM*n*_H/L Register Field Descriptions

Bits	Name	Description
0–63	ALARM_H/L	Alarm time comparator register. The corresponding alarm event in TMR_TEVENT is set when the current time counter becomes equal to or greater than this alarm time compare value in TMR_ALARM*n*_L/H. Writing the TMR_ALARM*n*_L register deactivates the alarm event after it has fired. Writing the TMR_ALARM*n*_L followed by the TMR_ALARM*n*_H register rearms the alarm function with the new compare value. The value programmed in this register must be an integer multiple of TMR_CTRL[TCLK_PERIOD] in order to get correct result. This register is reset to all ones to avoid false alarm after reset. In FS mode the alarm trigger is used as an indication to the fiper start down counting. Only alarm 1 supports this mode. In FS mode, alarm polarity bit should be configured to 0 (rising edge).

15.5.3.11.13 Timer Fixed Interval Period Register (TMR_FIPER1–3)

Timer fixed interval period pulse generator register. It is used to generate periodic pulses. This register is reset with 0xFFFF_FFFF to prevent any false pulse upon initialization. The down count register loads the value programmed in the fixed period interval (FIPER). FIPER register must be programmed before the timer is enabled. At every tick of the timer accumulator overflow, the counter decrements by the value of TMR_CTRL[TCLK_PERIOD]. It generates a pulse when the down counter value reaches zero. It reloads the down counter in the cycle following a pulse.

Should a user wish to use the TMR_FIPER1 register to generate a 1 PPS event, the following setup should be used:

- Program TMR_FIPER1 to a value that generates a pulse every second,
- Program TMR_ALARM1 to the correct time for the first PPS event
- Enable the timer

The eTSEC then waits for TMR_ALARM1 to expire before enabling the count down of TMR_FIPER1. The end result is that TMR_FIPER1 pulses every second after the original timer ALARM1 expired.

NOTE

In the case where the PPS signals are required to be phased aligned to the prescale output clock, the alarm value should be configured to **1 clock period less** than the desired value.

In order to keep tracking the prescale output clock, each time before enabling the FIPER, the user must reset the FIPER by writing a new value to the register. The ratio between the prescale register value and the FIPER value should be devisable by the clk period.

FIPER_VALUE = (prescale_value × tclk_per × N) – tclk_per

For example:

prescale = 9

clock period = 10

The FIPER can get the following values: 80, 170, 260 .....

The three registers in eTSEC1 are shared for all eTSECs. Figure 15-120 describes the definition for the TMR_FIPER register.

Figure 15-120. TMR_FIPER*n* Register Definition

Table 15-127 describes the fields of the TMR_FIPER register.

Table 15-127. TMR_FIPER Register Field Descriptions

Bits	Name	Description
0–31	FIPER	Fixed interval pulse period register. This field must be programmed to an integer multiple of TMR_CTRL[TCLK_PERIOD] value to ensure a period pulse being generated correctly.

15.5.3.11.14 External Trigger Stamp Register (TMR_ETTS1–2_H/L)

General purpose external trigger -stamp register (TMR_ETTS*n*_H/L). This register holds time at the programmable edge of the external trigger. The registers in eTSEC1 are shared for all eTSECs. This register is read only in normal operation. Figure 15-121 describes the definition for the TMR_ETTS*n*_H/L register.

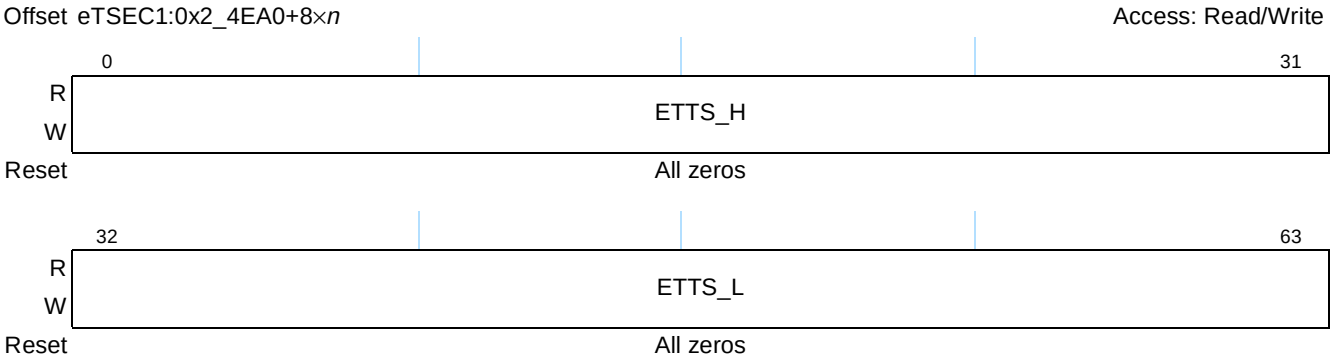


Figure 15-121. TMR_ETTS1-2_H/L Register Definition

Table 15-128 describes the fields of the TMR_ETTS*n*_H/L register.

Table 15-128. TMR_ETTS1-2_H Register Field Descriptions

Bits	Name	Description
0–63	ETTS_H/L	Time stamp field at the programmable edge of the external trigger.

15.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

15.5.4.1 TBI Transmit Process

The eTSEC's TBI implements the transmit portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. In SerDes mode, packets conveyed across the GMII are encapsulated and encoded into 10-bit symbols and output to the SerDes. In GMII mode, the GMII signals are passed through to the attached GMII PHY.

15.5.4.1.1 Packet Encapsulation

If TX_EN is de-asserted the eTSEC outputs an idle stream. If TX_EN is asserted, a Start_of_Packet symbol is output. This symbol replaces the first byte of the preamble field. All other bytes of the packet pass through an 8B10B encoding module. After the last byte of the FCS field is signaled via the GMII, the MAC de-asserts TX_EN. The eTSEC then outputs an End_of_Packet symbol. Then, depending on the position of the End_of_Packet symbols (being in either an odd or even position) the eTSEC outputs one or two Carrier_Extend symbols. Following the last Carrier_Extend symbol, the eTSEC resumes sending idle codes. If, during a packet, the eTSEC wishes to mark a byte invalid, TX_ER is asserted. The eTSEC, upon detection of TX_ER, substitutes the data symbol for an Error_Propagation symbol.

15.5.4.1.2 8B10B Encoding

Every 8-bit data octet has two (not necessarily different) ten-bit symbols associated with it. Depending on the running disparity (the cumulative difference of ones and zeroes) the eTSEC module chooses the appropriate symbol.

Special encapsulation symbols are called ordered_sets. Ordered_sets are comprised of one to four ten-bit symbols. Ordered_sets can be found in Clause 36 of the IEEE 802.3z specification.

15.5.4.1.3 Preamble Shortening

Because the idle ordered_set comprises two symbols and begins on an even symbols boundary, packets can only begin on an even boundary. However, the GMII has no such restriction and may signal TX_EN on an odd boundary. If this happens, the eTSEC delays the Start_of_Packet symbol, effectively ignoring the first byte of preamble; thus, a seven octet preamble becomes six octets on the Ten-Bit Interface.

15.5.4.2 TBI Receive Process

The eTSEC's TBI Implements the receive portion of the physical coding sublayer as found in Clause 36 of IEEE 802.3z. The Receive portion includes the Synchronization state machine. In SerDes mode, the eTSEC first attempts to acquire synchronization on the link by examining received symbols. Once synchronization is acquired, received packets are decoded and sent across the Receive GMII interface. In GMII mode, the GMII signals are passed through to the MAC.

15.5.4.2.1 Synchronization

The eTSEC examines received symbols looking for the seven bit 'comma' string embedded in some special symbols. Both the idle ordered_set and the Configuration ordered_set contain a symbol which has the comma. Once a certain number of codes with comma are detected, the eTSEC is considered to have acquired synchronization.

15.5.4.2.2 Auto-Negotiation for 1000BASE-X

Once synchronization is acquired, ordered_sets are decoded. If Configuration ordered_sets are received, the eTSEC decodes the two octet data field and the sixteen-bit Configuration data is stored and used to Auto-Negotiate with the link partner. in the Receive Configuration Register (RXCR[15:0]) an internal register used to receive all the link partners informations and used to compare to local ability during negotiation. Not visible to user. If, during Auto-Negotiation an invalid symbol is detected, Auto-Negotiation re-starts. After Auto-Negotiation is completed the TBI MII Status Register SR[AN done] in set. In this mode, packets may be received from the link partner.

15.5.4.3 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All of the TBI registers are 16 bits wide. The TBI registers are accessed at the offset of the TBI physical address. The eTSEC's TBI physical address is stored in the TBIPA register. Writing to the TBI registers is performed in a way similar to writing to an external PHY, by using the MII management interface. By using TBIPA in place of the PHY address, in the MIIMADD[PHY Address] field, and setting the MIIMADD[Register Address] to the appropriate address offset that corresponds to the register that one wants to read or write (see Table 15-129), the user can read (set MIIMCOM[read cycle]) or write (writing to MIIMCON[PHY control]) to the TBI block. Refer to the TBI physical address register in Section 15.5.3.1, "eTSEC General Control and Status Registers," and the TBI MII register set in Table 15-129. Notice that jitter diagnostics and TBI control are not IEEE 802.3 required registers and are only used for test and control of the eTSEC TBI block. The TBI's TBI control register (TBI) is for configuring the eTSEC ten-bit interface block. However, because this TBI block has an MII management interface (just like any other PHY), it has an IEEE 802.3 register called the control register (CR).

Table 15-129. TBI MII Register Set

Offset Address	Name	Access	Size	Section/page
TEN-BIT INTERFACE (TBI) REGISTERS				15.5.4/15-134
0x00	Control (CR)	R/W[1]	16 bits	15.5.4.3.1/15-136

Table 15-129. TBI MII Register Set (continued)

Offset Address	Name	Access	Size	Section/page
0x01	Status (SR)	R, LH, LL	16 bits	15.5.4.3.2/15-137
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	15.5.4.3.2/15-137
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	15.5.4.3.4/15-140
0x06	AN expansion (ANEX)	R, LH	16 bits	15.5.4.3.5/15-141
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	15.5.4.3.6/15-142
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	15.5.4.3.7/15-143
0x0F	Extended status (EXST)	R	16 bits	15.5.4.3.8/15-144
0x10	Jitter diagnostics (JD)	R/W	16 bits	15.5.4.3.9/15-144
0x11	TBI control (TBICON)	R/W	16 bits	15.5.4.3.10/15-145

[1] R = Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

15.5.4.3.1 Control Register (CR)

Figure 15-122 describes the definition for the CR register.

Offset 0x00 Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PHY Reset	—	Speed[0]	AN Enable	—		Reset AN	Full Duplex	—	Speed[1]	—					
Reset	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0

Figure 15-122. Control Register Definition

Table 15-130 describes the fields of the CR register.

Table 15-130. CR Field Descriptions

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The internal state of the TBI is reset. This in turn may change the state of the TBI link partner.
1	—	Reserved
2	Speed[0]	Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed.Setting this field controls the speed at which the TBI operates. The table for Speed[1] provides the appropriate encoding. Its default is bit[2] = ‘0’; bit[9] = ’1’.
3	AN Enable	Auto-negotiation enable. This bit is set by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled.
4–5	—	Reserved

Table 15-130. CR Field Descriptions (continued)

Bits	Name	Description
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation. 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.
7	Full Duplex	Duplex mode. This bit is set by default. 0 Reserved. 1 Full-duplex operation.
8	—	Reserved, should be cleared.
9	Speed[1]	Speed selection. This bit defaults to a set state and should always be set, which corresponds to 1000 Mbps speed.Setting this field controls the speed at which the TBI operates. The following table provides the appropriate encoding. Its default is bit[2] = ‘0’; bit[9] = ’1’. **Maximum Operating Speed / Bit 2 / Bit 9** Reserved / 0 / 0 Reserved / 1 / 0 1000 Mbps / 0 / 1 Reserved / 1 / 1
10–15	—	Reserved

15.5.4.3.2 Status Register (SR)

Figure 15-123 describes the definition for the SR register.

Offset 0x01 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	—							Extend Status	—	No Pre	AN Done	Remote Fault	AN Ability	Link Status	—	Extend Ability
W																
Reset	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	1

Figure 15-123. Status Register Definition

Table 15-131 describes the fields of the SR register.

Table 15-131. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	Extend Status	This bit indicates that PHY status information is also contained in the Register 15, Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, should be cleared.
9	No Pre	MF preamble suppression enable. This bit indicates whether or not the PHY is capable of handling MII management frames without the 32-bit preamble field. Returns 1, indicating support for suppressed preamble MII management frames. This bit is read-only.

Table 15-131. SR Descriptions (continued)

Bits	Name	Description
10	AN Done	Auto-negotiation complete. This bit is read-only and is cleared by default. 0 Either the auto-negotiation process is underway or the auto-negotiation function is disabled. 1 The auto-negotiation process has completed.
11	Remote Fault	Remote fault. This bit is read-only and is cleared by default. Each read of the status register clears this bit. 0 Normal operation. 1 A remote fault condition was detected. This bit latches high in order for software to detect the condition.
12	AN Ability	Auto-negotiation ability. While read as set, this bit indicates that the PHY has the ability to perform auto-negotiation. While read as cleared, this bit indicates the PHY lacks the ability to perform auto-negotiation. Returns 1 on read. This bit is read-only.
13	Link Status	Link status. This bit is read-only and is cleared by default. 0 A valid link is not established. This bit latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, should be cleared.
15	Extend Ability	Extended capability. This bit indicates that the PHY contains the extended set of registers (those beyond control and status). Returns 1 on read. This bit is read-only.

15.5.4.3.3 AN Advertisement Register (ANA)

Figure 15-124 describes the definition for the ANA register.

Offset 0x04 Access: Read/Write

	0	1	2 3	4 6	7	8	9	10	11 15
R W	Next Page	—	Remote Fault	—	Pause		Half Duplex	Full Duplex	—
Reset	All zeros								

Figure 15-124. AN Advertisement Register Definition

Table 15-132 describes the fields of the ANA register.

Table 15-132. ANA Field Descriptions

Bits	Name	Description
0	Next Page	Next page configuration. The local device sets this bit to either request next page transmission or advertise next page exchange capability. 0 The local device wishes not to engage in next page exchange. 1 The local device has no next pages but wishes to allow reception of next pages. If the local device has no next pages and the link partner wishes to send next pages, the local device shall send null message codes and have the message page set to 0b000_0000_0001, as defined in annex 28C.
1	—	Reserved. (Ignore on read)

Table 15-132. ANA Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The local device’s remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>The local device’s PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult Table 15-133.
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability.
0 Designates local device as not capable of half-duplex operation.
1 Designates local device as capable of half-duplex operation.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability.
0 Designates the local device as not capable of full-duplex operation.
1 Designates the local device as capable of full-duplex operation.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

Table 15-133 describes the resolution of pause priority.

Table 15-133. PAUSE Priority Resolution

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	0	x	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive

Table 15-133. PAUSE Priority Resolution (continued)

Local Device		Link Partner		Local Resolution	Link Partner Resolution
PAUSE	ASM_DIR	PAUSE	ASM_DIR		
0	1	1	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
0	1	1	1	Enable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Enable PAUSE receive
1	0	0	x	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	0	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive
1	1	0	0	Disable PAUSE transmit Disable PAUSE receive	Disable PAUSE transmit Disable PAUSE receive
1	1	0	1	Disable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Disable PAUSE receive
1	1	1	x	Enable PAUSE transmit Enable PAUSE receive	Enable PAUSE transmit Enable PAUSE receive

15.5.4.3.4 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 15-125 describes the definition for the ANLPBPA register.

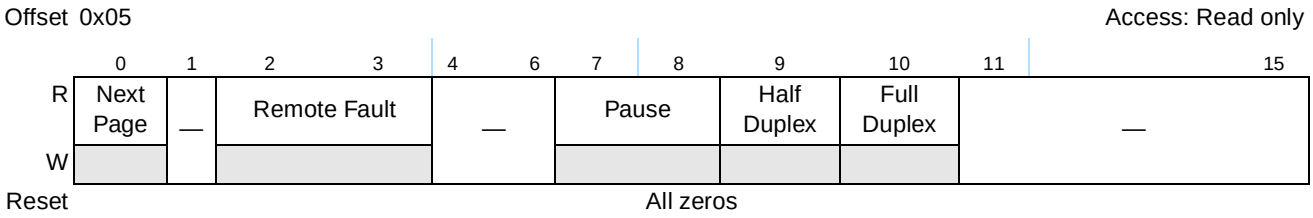


Figure 15-125. AN Link Partner Base Page Ability Register Definition

Table 15-134 describes the fields of the ANLPBPA register.

Table 15-134. ANLPBPA Field Descriptions

Bits	Name	Description
0	Next Page	Next page. This bit is read-only. The link partner sets or clears this bit. 0 Link partner has no subsequent next pages or is not capable of receiving next pages. 1 Link partner either requesting next page transmission or indicating the capability to receive next pages.
1	—	Reserved. (Ignore on read)

Table 15-134. ANLPBPA Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The link partner's remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>Encoding of the link partner's PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult. This bit is read-only
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability. This bit is read-only.
0 Link partner is not capable of half-duplex mode.
1 Link partner is capable of half-duplex mode.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability. This bit is read-only.
0 Link partner is not capable of full-duplex mode.
1 Link partner is capable of full-duplex mode.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

15.5.4.3.5 AN Expansion Register (ANEX)

Figure 15-126 describes the definition for the ANEX register.

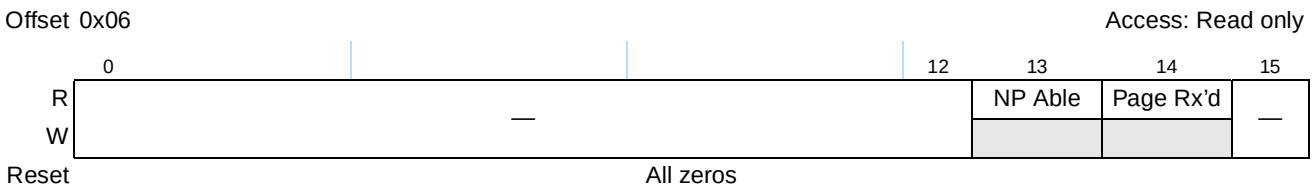


Figure 15-126. AN Expansion Register Definition

Table 15-135 describes the fields of the ANEX register.

Table 15-135. ANEX Field Descriptions

Bits	Name	Description
0–12	—	Reserved, should be cleared.
13	NP Able	Next page able. This bit is read-only and returns 1 on read. While read as set, indicates local device supports next page function.
14	Page Rx'd	Page received. This bit is read-only. The bit clears on a read to the register. 0 Normal operation. 1 A new page was received and stored in the applicable AN link partner ability or AN next page register. This bit latches high in order for software to detect while polling.
15	—	Reserved, should be cleared.

15.5.4.3.6 AN Next Page Transmit Register (ANNPT)

Figure 15-127 describes the definition for the ANNPT register.

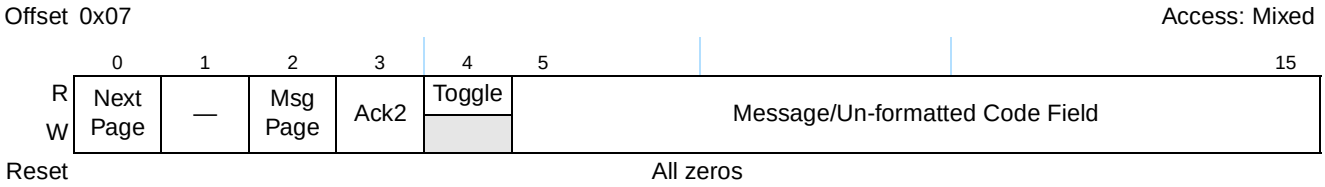


Figure 15-127. AN Next Page Transmit Register Definition

Table 15-136 describes the fields of the ANNPT register.

Table 15-136. ANNPT Field Descriptions

Bits	Name	Description
0	Next Page	Next page indication. [Reference MII bit 7.15 in IEEE 802.3, 2000 Edition Clause 28.2.4] 0 Last page. 1 Additional next pages to follow.
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. [Reference MII bit 7.13] 0 Unformatted page. 1 Message page.
3	Ack2	Acknowledge 2. Used by the next page function to indicate that the device has the ability to comply with the message. [Reference MII bit 7.12] 0 The local device cannot comply with message. 1 The local device complies with message.

Table 15-136. ANNPT Field Descriptions (continued)

Bits	Name	Description
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. [Reference MII bit 7.11] This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/ Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value. [Reference MII field 7.10:0]

15.5.4.3.7 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 15-128 describes the definition for the ANLPANP register.

Offset 0x08 Access: Read only

	0	1	2	3	4	5	15
R	Next Page	—	Msg Page	Ack2	Toggle	Message/Un-formatted Code Field	
W							
Reset	All zeros						

Figure 15-128. AN Link Partner Ability Next Page Register Definition

Table 15-137 describes the fields of the ANLPANP register.

Table 15-137. ANLPANP Field Descriptions

Bits	Name	Description
0	Next Page	Next page. The link partner sets and clears this bit. 0 Last page from link partner 1 Additional next pages to follow
1	—	Reserved. (Ignore on read)
2	Msg Page	Message page. 0 Unformatted page 1 Message page
3	Ack2	Acknowledge 2. Indicates the link partner's ability to comply with the message. 0 Link partner cannot comply with message. 1 Link partner complies with message.
4	Toggle	Toggle. Used to ensure synchronization with the link partner during next page exchange. This bit always takes the opposite value of the toggle bit of the previously-exchanged link code word. The initial value in the first next page transmitted is the inverse of bit 11 in the base link code word. This bit is read-only. 0 Toggle bit of the previously-exchanged link code word was 1. 1 Toggle bit of the previously-exchanged link code word was 0.
5–15	Message/ Un-formatted Code Field	Message pages are formatted pages that carry a pre-defined message code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted code fields take on an arbitrary value.

15.5.4.3.8 Extended Status Register (EXST)

Figure 15-129 describes the definition for the EXST register.

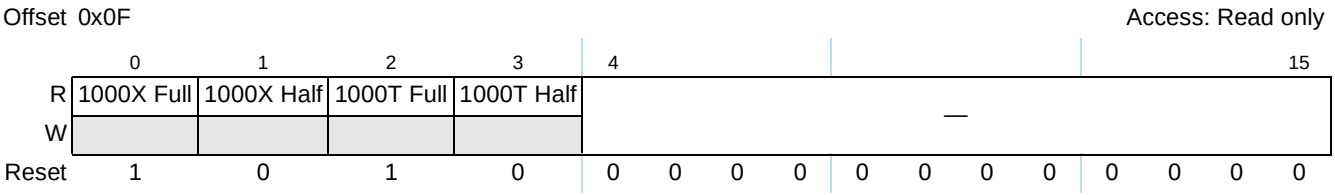


Figure 15-129. Extended Status Register Definition

Table 15-138 describes the fields of the EXST register.

Table 15-138. EXST Field Descriptions

Bits	Name	Description
0	1000X Full	1000X full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X full-duplex mode. 1 PHY can operate in 1000BASE-X full-duplex mode.
1	1000X Half	1000X half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-X half-duplex mode. 1 PHY can operate in 1000BASE-X half-duplex mode.
2	1000T Full	1000T full-duplex capability. Returns 1 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T full-duplex mode. 1 PHY can operate in 1000BASE-T full-duplex mode.
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operation in 1000BASE-T half-duplex mode. 1 PHY can operate in 1000BASE-T half-duplex mode.
4–15	—	Reserved

15.5.4.3.9 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. In may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns. Figure 15-130 describes the definition for the JD register.

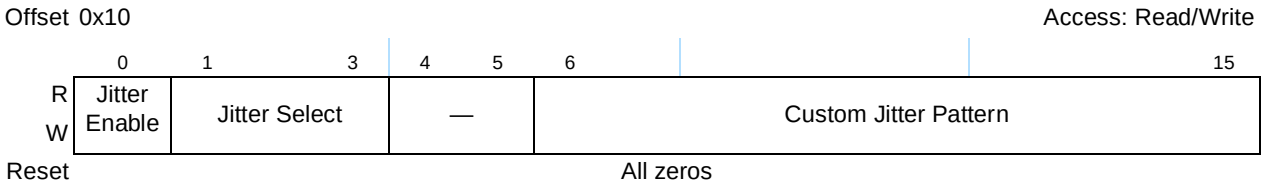


Figure 15-130. Jitter Diagnostics Register Definition

Table 15-139 describes the fields of the JD register.

Table 15-139. JD Field Descriptions

Bits	Name	Description
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation. 1 Enable the TBI to transmit the jitter test patterns defined in IEEE 802.3z 36A.
1–3	Jitter Select	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 00.

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (+/- D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (- K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0
Reserved	1	1	1

4–5	—	Reserved
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0x000.

15.5.4.3.10 TBI Control Register (TBICON)

Figure 15-131 describes the definition for the TBICON register.

Offset 0x11 Access: Mixed

	0	1	2	3	4–6	7	8–9	10	11	12–15
R	Soft_Reset	—	Disable Rx Dis	Disable Tx Dis	—	AN Sense	—	Clock Select	MII Mode	—
W										
Reset	0	0	0	0	0 0 0	0	0 0	0	1	0 0 0 0

Figure 15-131. TBI Control Register Definition

Table 15-140 describes the fields of the TBICON register.

Table 15-140. TBICON Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in the TBI.
1	—	Reserved. (Ignore on read)
2	Disable Rx Dis	Disable receive disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the receive direction.
3	Disable Tx Dis	Disable transmit disparity. This bit is cleared by default. 0 Normal operation. 1 Disables the running disparity calculation and checking in the transmit direction.
4–6	—	Reserved
7	AN Sense	Auto-negotiation sense enable. This bit is cleared by default. 0 IEEE 802.3z Clause 37 behavior is desired, which results in the link not completing. 1 Allow the auto-negotiation function to sense either a Gigabit MAC in auto-negotiation bypass mode or an older Gigabit MAC without auto-negotiation capability. If sensed, auto-negotiation complete becomes true; however, the page received is low, indicating no page was exchanged. Management can then act accordingly.
8–9	—	Reserved
10	Clock Select	Clock select. This bit is cleared by default. 0 Allow the TBI to accept dual split-phase 62.5 MHz receive clocks. 1 Configure the TBI to accept a 125 MHz receive clock from the SerDes/PHY. The 125 MHz clock must be physically connected to ‘PMA receive clock 0’ if using a parallel (non-SGMII) Ethernet protocol.
11	MI Mode	This bit describes the configuration mode of the TBI. The user reads a 1 while the TBI is configured in GMII/MII mode (connected to a GMII/MII PHY) and a 0 while configured in TBI mode (connected to a 1000BASE-X SerDes). Its value is the inverse of ECNTRL[TBIM]. 0 TBI mode. 1 GMII mode.
12–15	—	Reserved

15.6 Functional Description

15.6.1 Connecting to Physical Interfaces on Ethernet

This section describes how to connect the eTSEC to various interfaces: MII, GMII, RMII, RGMII, TBI, and RTBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification because the PHYs follow the same conventions. (For instance, in the bus TSEC*n*_TXD[7:0], bit 7 is the msb and bit 0 is the lsb). If a mode does not use all input signals available to a particular eTSEC, those inputs that are not used must be pulled low on the board.

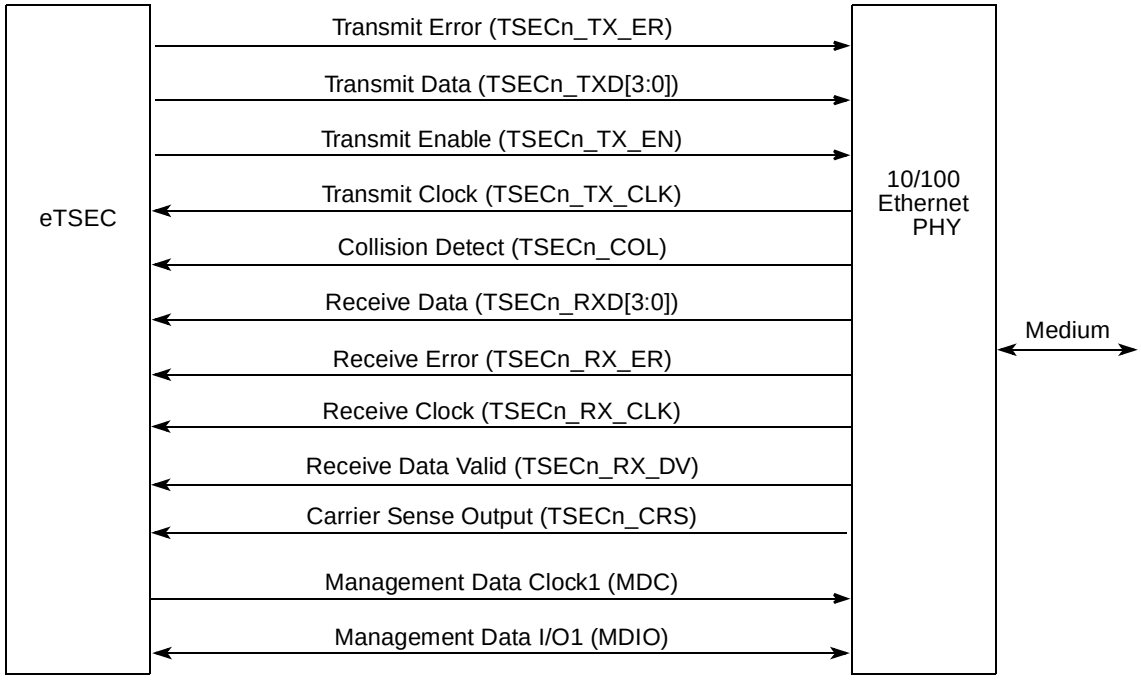
eTSEC1, eTSEC2 and eTSEC3 can be configured in all modes, whereas eTSEC4 can only operate in a reduced-pin mode and only when eTSEC3 is also configured in a reduced-pin mode. Table 15-141 shows what modes are available for each eTSEC.

Table 15-141. Ethernet Modes versus eTSEC (V = Available)

Mode	eTSEC1	eTSEC2	eTSEC3	eTSEC4
GMII	V	V	V	—
RGMII	V	V	V	V
TBI	V	V	V	—
RTBI	V	V	V	V
MII	V	V	V	—
RMII	V	V	V	V
SGMII	V	V	V	V

15.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the eTSEC. Figure 15-132 depicts the basic components of the MII including the signals required to establish eTSEC module connection with a PHY.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 15-132. eTSEC-MII Connection

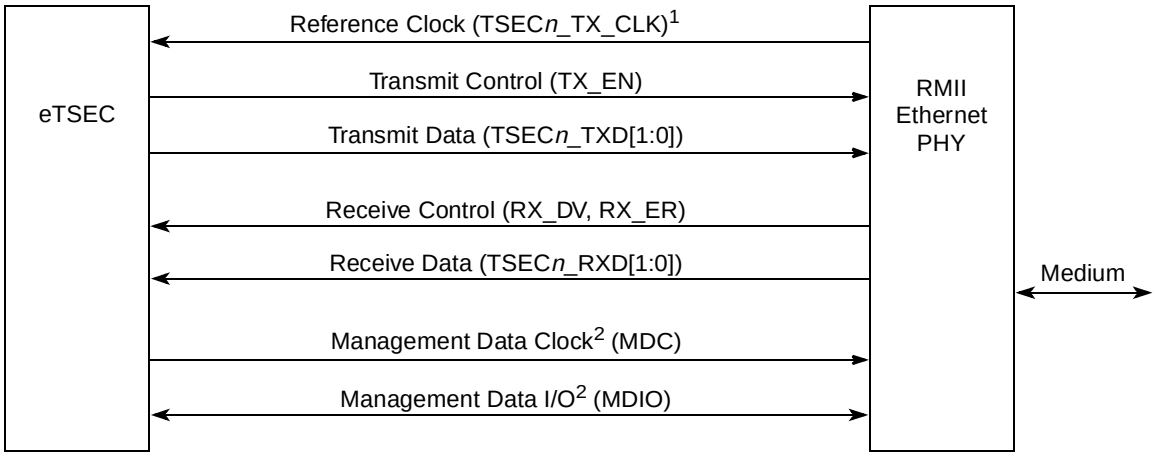
An MII interface has 18 signals (including the MDC and MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

15.6.1.2 Reduced Media-Independent Interface (RMII)

This section describes the reduced media-independent interface (RMII) intended to be used between the PHYs and the GMII MAC. The RMII is a reduced-pin alternative to the IEEE802.3u MII. The RMII reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 18 signals (MII) to 10 signals. To accomplish this objective, the data paths are halved in width and clocked at twice the MII clock frequency, while clocks, carrier sense and error signals have been partly combined. For 100 Mbps operation, the reference clock operates at 50 MHz, whereas for 10 Mbps operation, the clock remains at 50MHz, but only every 10th cycle is used. Figure 15-133 depicts the basic components of the reduced media-independent interface and the signals required to establish an eTSEC's connection with a PHY. The RMII is implemented as defined by the RMII Specification of the RMII Consortium, as of March 20, 1998.

NOTE

Due to pin limitations on the MPC8572E, eTSEC4 must be configured differently from the other eTSECs in RMII mode. For eTSEC1–3, the RMII reference clock is obtained from TSEC*n*_TX_CLK. For eTSEC4, however, this clock comes from TSEC4_RX_CLK (TSEC3_COL).

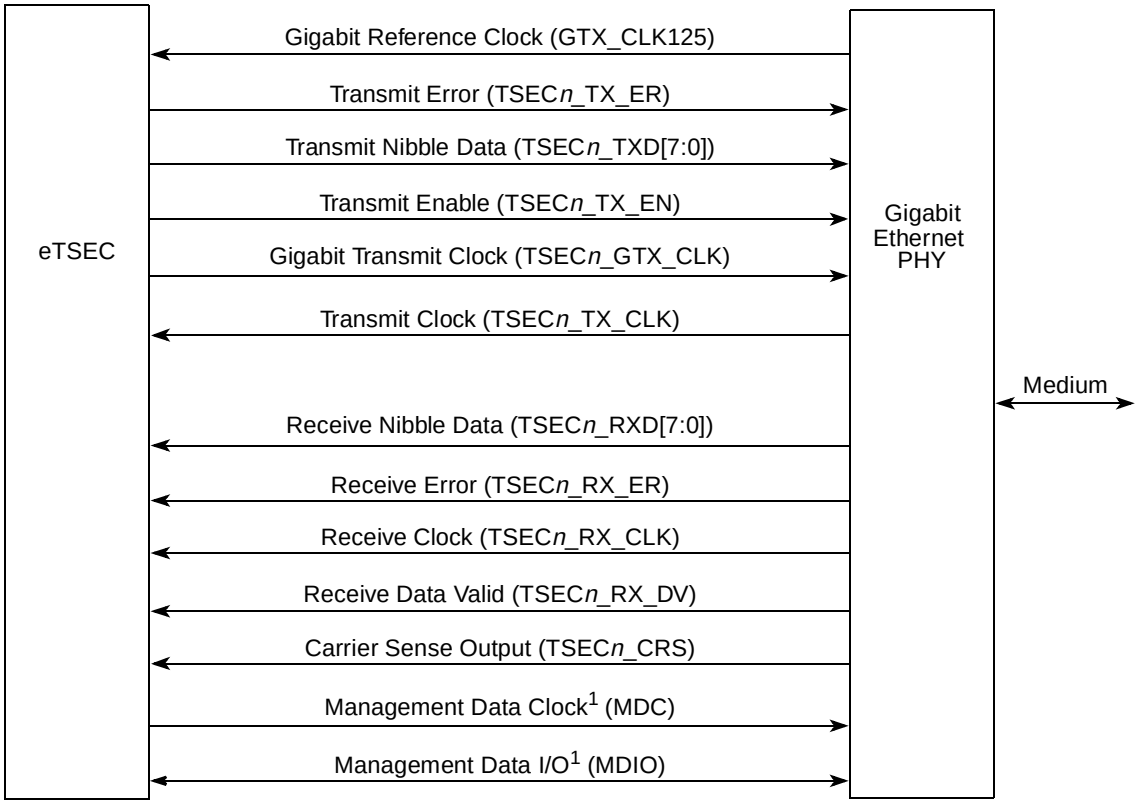


eTSEC4 of the MPC8572E uses TSEC4_RX_CLK (TSEC3_COL) as reference clock due to device pin limitations.

Figure 15-133. eTSEC-RMII Connection

15.6.1.3 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the eTSEC. Figure 15-134 depicts the basic components of the GMII including the signals required to establish the eTSEC module connection with a PHY.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

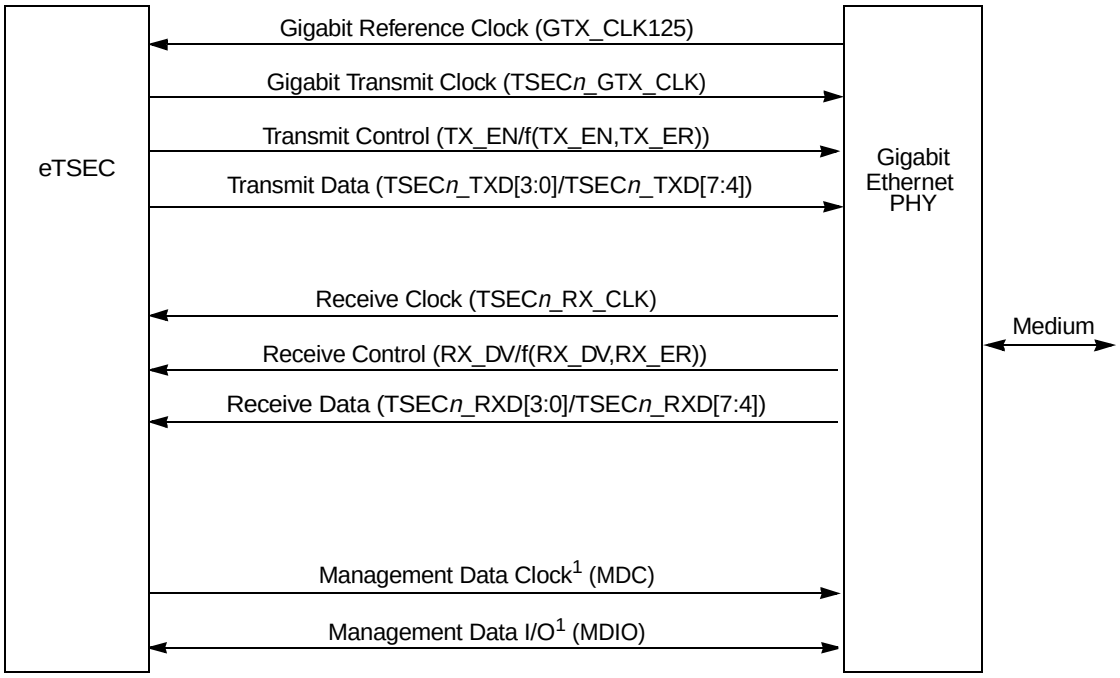
Figure 15-134. eTSEC-GMII Connection

A GMII interface has 28 signals (TSEC*n*_GTX_CLK + _GTX_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

15.6.1.4 Reduced Gigabit Media-Independent Interface (RGMII)

This section describes the reduced gigabit media-independent interface (RGMII) intended to be used between the PHYs and the GMII MAC. The RGMII is an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI. The RGMII reduces the number of signals required to interconnect the MAC and the PHY from a maximum of 28 signals (GMII) to 15 signals (GTX_CLK125 included) in a cost effective and technology independent manner. To accomplish this objective, the data paths and all associated control signals are multiplexed using both edges of the clock. For gigabit operation, the clocks operate at 125MHz, and for 10/100 operation, the clocks operate at 2.5 MHz or 25 MHz, respectively. Note that the GTX_CLK125 input must be provided at 125 MHz for an RGMII interface, regardless of operation speed (1 Gbps, 100 Mbps, or 10 Mbps). Figure 15-135 depicts the basic components of the gigabit reduced media-independent interface and the signals required to establish the gigabit Ethernet controllers' module

connection with a PHY. The RGMII is implemented as defined by the RGMII specification Version 1.2a 9/22/00.

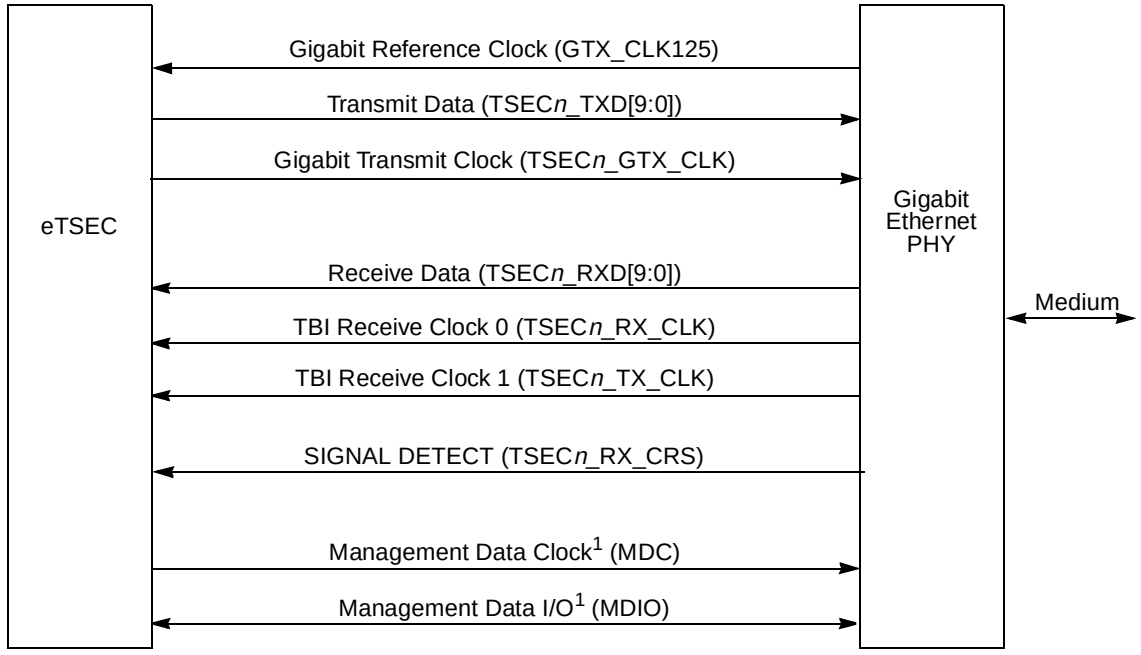


1 The management signals (MDC and MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 15-135. eTSEC-RGMII Connection

15.6.1.5 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the eTSEC to implement a standard SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 15-136 depicts the basic components of the TBI including the signals required to establish eTSEC module connection with a PHY. RBC0 and RBC1 are differential 62.5 MHz receive clocks. If not connected to the TBI PHY, the Signal Detect (SDET) input must be tied high. This causes the eTSEC to begin auto negotiation with the SERDES immediately upon the TBI module being enabled.



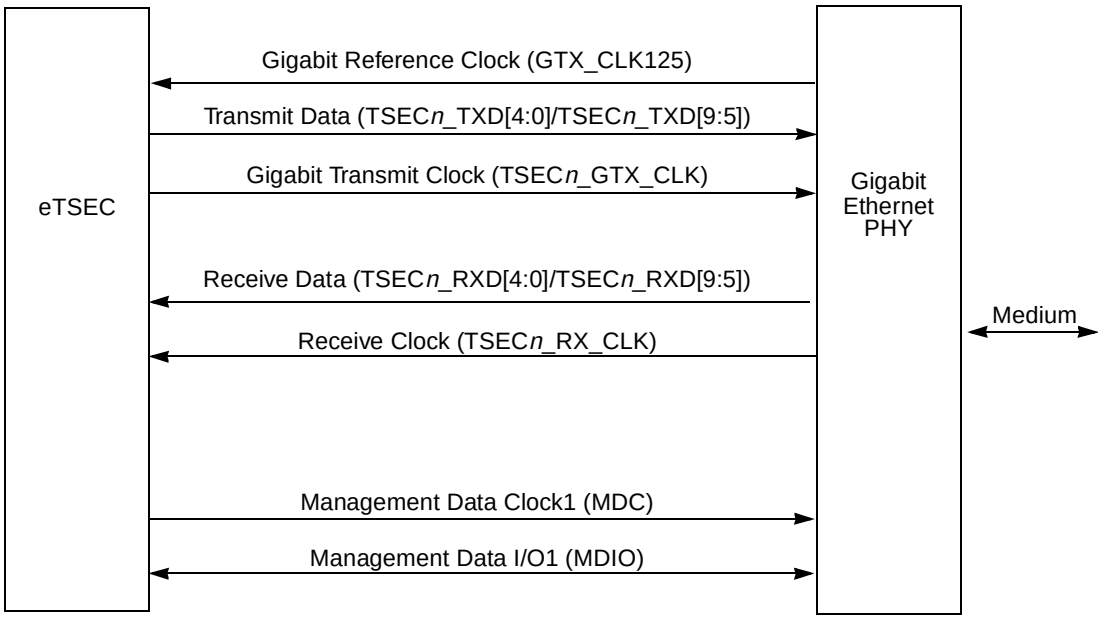
1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 15-136. eTSEC-TBI Connection

A TBI interface has 26 signals (GE_GTX_CLK125 included) for connecting to an Ethernet PHY, as defined by IEEE 802.3z GMII and TBI standards.

15.6.1.6 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the eTSEC to implement a reduced-pin count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 15-137 depicts the basic components of the RTBI including the signals required to establish eTSEC module connection with a PHY. Note that in RTBI the eTSEC immediately begins auto-negotiation with the SerDes.



1 The management signals (MDC and MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 15-137. eTSEC-RTBI Connection

A RTBI interface has 15 signals (GE_GTX_CLK125 included), as defined by the RGMII specification Version 1.2a 9/22/00, and is intended to be an alternative to the IEEE 802.3u MII, the IEEE 802.3z GMII and the TBI standard for connecting to an Ethernet PHY.

15.6.1.7 Ethernet Physical Interfaces Signal Summary

Table 15-142 describes the signal multiplexing for the following interfaces: GMII, MII, and RMII.

Table 15-142. GMII, MII, and RMII Signals Multiplexing

eTSEC Signals			GMII Interface			MII Interface			RMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 25			Frequency [MHz] 50		
Voltage[V] 3.3/2.5			Voltage[V] 3.3			Voltage[V] 3.3			Voltage[V] 3.3		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	—	—	—	—	—	—
TX_CLK	I	1	TX_CLK	I	1	TX_CLK	I	1	REF_CLK[1]	I	1
TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1	TxD[2]	O	1	—	—	—
TxD[3]	O	1	TxD[3]	O	1	TxD[3]	O	1	—	—	—
TxD[4]	O	1	TxD[4]	O	1	—			—	—	—
TxD[5]	O	1	TxD[5]	O	1	—	—	—	—	—	—
TxD[6]	O	1	TxD[6]	O	1	—	—	—	—	—	—
TxD[7]	O	1	TxD[7]	O	1	—	—	—	—	—	—
TX_EN	O	1	TX_EN	O	1	TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1	TX_ER	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1	—	—	—
RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1	RxD[2]	I	1	—	—	—
RxD[3]	I	1	RxD[3]	I	1	RxD[3]	I	1	—	—	—
RxD[4]	I	1	RxD[4]	I	1	—	—	—	—	—	—
RxD[5]	I	1	RxD[5]	I	1	—	—	—	—	—	—
RxD[6]	I	1	RxD[6]	I	1	—	—	—	—	—	—
RxD[7]	I	1	RxD[7]	I	1	—	—	—	—	—	—
RX_DV	I	1	RX_DV	I	1	RX_DV	I	1	CRS_DV	I	1
RX_ER[2]	I	1	RX_ER	I	1	RX_ER	I	1	RX_ER	I	1
COL	I	1	—	—	—	COL	I	1	—	—	—
CRS	I	1	—	—	—	CRS	I	1	—	—	—
Sum		25	**Sum**		23	**Sum**		16	**Sum**		8

1 eTSEC4 of the MPC8572E uses TSEC4_RX_CLK (TSEC3_COL) as reference clock due to device pin limitations. All the other eTSEC interfaces (1–3) use TSEC*n*_TX_CLK.

[2] eTSEC4 of the MPC8572E does not offer an RX_ER signal. All the other eTSEC interfaces (1–3) do offer TSEC*n*_RX_ER signals.

Table 15-144 describes the signal multiplexing for RGMII, TBI, and RTBI interfaces.

Table 15-143. RGMII, TBI, and RTBI Signals Multiplexing

eTSEC Signals			RGMII Interface			TBI Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 3.3			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—	RX_CLK1	I	1	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	—	—	—	TCG[4]	O	1	—	—	—
TxD[5]	O	1	—	—	—	TCG[5]	O	1	—	—	—
TxD[6]	O	1	—	—	—	TCG[6]	O	1	—	—	—
TxD[7]	O	1	—	—	—	TCG[7]	O	1	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[8]	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	—	—	—	TCG[9]	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK0	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	—	—	—	RCG[4]	I	1	—	—	—
RxD[5]	I	1	—	—	—	RCG[5]	I	1	—	—	—
RxD[6]	I	1	—	—	—	RCG[6]	I	1	—	—	—
RxD[7]	I	1	—	—	—	RCG[7]	I	1	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[8]	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	—	—	—	RCG[9]	I	1	—	—	—
COL	I	1	—	—	—				—	—	—
CRS	I	1	—	—	—	SDET	I	1	—	I	—
Sum		25	**Sum**		12	**Sum**		24	**Sum**		12

Table 15-144. RGMII and RTBI Signals Multiplexing

eTSEC Signals			RGMII Interface			RTBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	—	—	—	—	—	—
TxD[5]	O	1	—	—	—	—	—	—
TxD[6]	O	1	—	—	—	—	—	—
TxD[7]	O	1	—	—	—	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/ TX_ERR)	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	—	—	—	—	—	—
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	—	—	—	—	—	—
RxD[5]	I	1	—	—	—	—	—	—
RxD[6]	I	1	—	—	—	—	—	—
RxD[7]	I	1	—	—	—	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/ RX_ERR)	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	—	—	—	—	—	—
COL	I	1	—	—	—	—	—	—
CRS	I	1	—	—	—	—	I	—
Sum		25	**Sum**		12	**Sum**		12

Table 15-145. RGMII Signals Multiplexing

eTSEC Signals			RGMII Interface		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals (TSEC*n*_)	I/O	No. of Signals	Signals (TSEC*n*_)	I/O	No. of Signals
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	—	—	—
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1	—	—	—
TxD[5]	O	1	—	—	—
TxD[6]	O	1	—	—	—
TxD[7]	O	1	—	—	—
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1	—	—	—
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1	—	—	—
RxD[5]	I	1	—	—	—
RxD[6]	I	1	—	—	—
RxD[7]	I	1	—	—	—
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1	—	—	—
COL	I	1	—	—	—
CRS	I	1	—	—	—
Sum		25	**Sum**		12

Table 15-146 describes the signals shared by all interfaces.

Table 15-146. Shared Signals

Signals	I/O	No. of Signals	Function
MDIO	I/O	1	Management interface I/O
MDC	O	1	Management interface clock
GTX_CLK125	I	1	Reference clock
Sum		3	—

15.6.1.8 SGMII Interface

SGMII communication using the eTSEC is accomplished through the SerDes interface. See Table 15-1 on page 15-7 for specific signal assignments.

15.6.2 Connecting to FIFO Interfaces

This section describes how to connect an eTSEC to third-party communication devices, including users' ASICs and FPGAs, via the FIFO interface.

Each eTSEC provides an 8-bit or 16-bit full-duplex packet FIFO interface port that bypasses the Ethernet MAC, but re-uses the GMII signals. As a result, the FIFO interface normally does not impose the overheads of Ethernet framing. The FIFO interface operates synchronously, at a maximum frequency defined by a ratio of 4.2:1 (platform:TxClk) in GMII mode and 3.2:1 (platform:TxClk) in encoded mode providing OC-48 full-duplex transfer rates. For example, a FIFO frequency of 127 MHz in GMII mode requires a platform frequency of 533 MHz; a FIFO frequency of 200 MHz in encoded mode requires a platform frequency of 667 MHz; a FIFO frequency of 167 MHz in encoded mode requires a platform frequency of 533 MHz.

Bare IP packets—with an optional 32-bit CRC check sequence—can be transferred to the eTSEC directly. The eTSEC Tx and Rx FIFOs, TOE functions, and DMA continue to be used in packet FIFO mode.

The ECNTRL[FIFM] bit determines whether eTSEC is communicating with its Ethernet MAC or FIFO interface. There are two main modes of FIFO operation:

- 8-bit packet FIFO
  - The GMII signals of each eTSEC can be used to create a FIFO port, therefore eTSEC can support up to three simultaneous 8-bit FIFO interfaces (eTSEC4 cannot accommodate GMII Ethernet and, as such, cannot accommodate a dedicated FIFO.) Choosing between 8-bit FIFO and Ethernet affects each eTSEC independently, therefore a mix of FIFO and Ethernet interfaces can be configured.
  - The data signals of GMII and 8-bit FIFO remain the same. The data valid (RX_DV, TX_EN) and error (RX_ER, TX_ER) signals are used to signal framing information. If required, the collision (COL) and carrier sense (CRS) signals can be used in an encoded mode to provide link-level flow control.

- 16-bit packet FIFO
  - The GMII signals of eTSECs 1 & 2 may be combined to create a 16-bit FIFO port, therefore providing up to one 16-bit FIFO and one 8-bit FIFO (or Ethernet) interfaces simultaneously The eTSEC that loses access to its GMII signals (eTSEC2) in this mode cannot be used for communications.
  - The possible pairing of eTSECs to create 16-bit FIFO ports is limited to eTSEC1 paired with eTSEC2. The remaining eTSEC (or eTSEC3) must be used in an 8-bit FIFO or Ethernet mode.
  - Either a GMII-style or encoded framing protocol is available, the latter allowing relatively simple conversion to a POS PHY Level 3 interface with additional glue logic. When the FIFO interface is in 16-bit mode, RX_DV and TX_EN signals from both ports are used to determine data valid. Only one of the RX_ER and TX_ER signals are used to determine framing errors. The collision (COL) and carrier sense (CRS) signals can be used to provide link-level flow control.

The possible port mode combinations supported across all four eTSECs are shown in Table 15-147.

Table 15-147. Valid Combinations of eTSEC Signals and Interface Modes

Mode Option	eTSEC1 Signals	eTSEC2 Signals	eTSEC3 Signals	eTSEC4 Signals
Ethernet only	All Ethernets	All Ethernets	All Ethernets	—
			Reduced Ethernets	Reduced Ethernets
8-bit FIFO Only	8-bit FIFO port 1	8-bit FIFO port 2	8-bit FIFO port 3	
8-bit FIFO & Ethernet, mixed	8-bit FIFO port 1 or Reduced Ethernets	8-bit FIFO port 2 or Reduced Ethernets	8-bit FIFO port 3	
			All Ethernets	—
			Reduced Ethernets	Reduced Ethernets
16-bit FIFO & 8-bit FIFO	16-bit FIFO port 1		8-bit FIFO port 3	
16-bit FIFO & Ethernet	16-bit FIFO port 1		All Ethernets	—
			Reduced Ethernets	Reduced Ethernets

The following restrictions apply in any of the FIFO modes:

- Transferred packets must be a minimum of 10 bytes, and no more than 9600 bytes in length.
- Although TCP/IP offload is supported, the receive queue filer table must be limited to as many entries as eTSEC can search every packet. See Section 15.6.5.1.1, “Filing Rules,” on page 15-187 for guidance on how to determine maximum table size for an application.
- eTSEC requires received packets to have a minimum inter-packet gap of three cycles.
- On transmission, the minimum inter-packet gap (set in FIFOCFG[IPG]) is three cycles if CRC is not automatically appended. Each CRC data beat adds to this requirement. For 16-bit FIFO interfaces the minimum Tx IPG is 5 cycles and for 8-bit FIFO interfaces the minimum is 7 cycles.

No Ethernet-specific features (such as MAC address matching) or layer 2 properties (such as Ethertype) are available in FIFO mode.

15.6.2.1 Flow Control

In the encoded (non GMII-style) FIFO modes, link-level flow control is provided to the eTSEC transmitter on the COL signal of the controlling eTSEC, while back pressure to the remote transmitter is sent on the CRS signal (which acts as an output signal only in FIFO mode). Owing to the synchronization delay of responding to flow control on signal COL, the eTSEC cannot stop transmission immediately, but may require up to 8 clock cycles before transmission is paused. The eTSEC issues flow control either when software forces it (via the FIFOCFG[FFC] bit), or when the Rx FIFO reaches its high watermark.

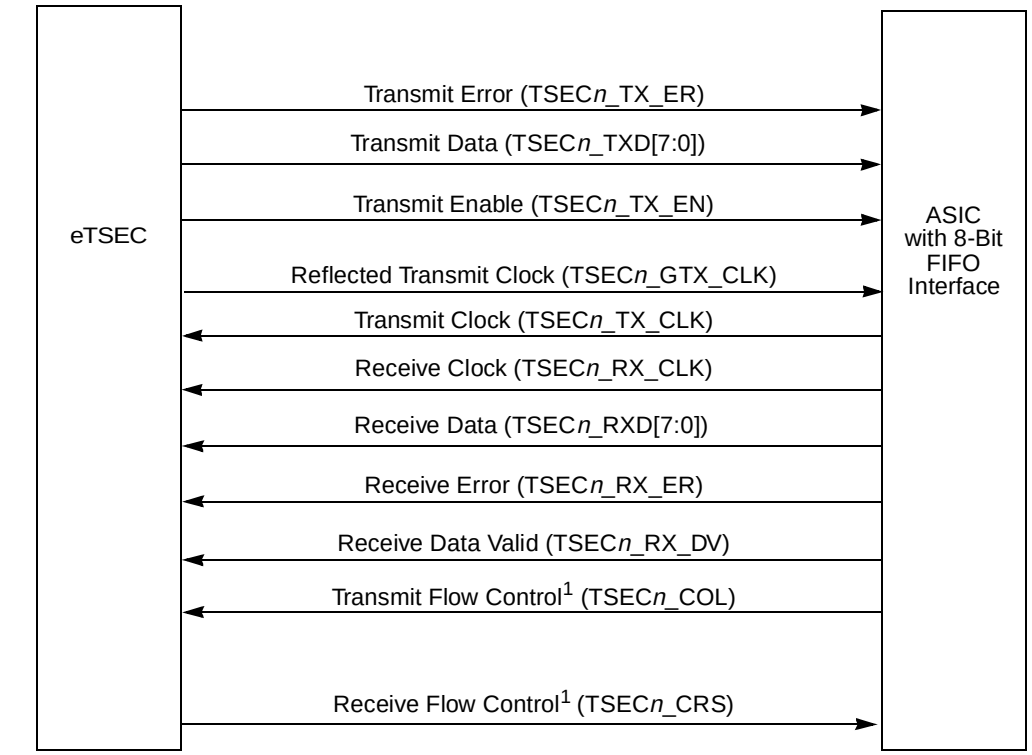
15.6.2.2 CRC Appending and Checking

If FIFOCFG[CRCAPP] is enabled, the FIFO interface automatically appends a 4-byte CRC to each transmitted packet. Alternatively, if FIFOCFG[CRCAPP] is cleared, TxBD[TC] provides a per-packet override to append CRC. The IEEE 802.3 standard CRC-32 algorithm is used, where the least significant bit of each byte (TXD[0]) is combined into the CRC ahead of the most significant bit (TXD[7]). Accordingly, the CRC result, CRC[31:0] is transmitted onto the interface in bit-reversed order, CRC[24:31], CRC[16:23], CRC[8:15], CRC[0:7].

Automatic checking of CRC-32 checksums received over the FIFO interface is enabled by setting FIFOCFG[CRCCHK]. CRC errors are recorded in the RxBD[CR] flag of every last buffer. Like transmit, the receiver combines data into the CRC in the order least significant data bit (RXD[0]) to most significant bit (RXD[7]). The last 4 bytes of the packet are assumed to be CRC whenever FIFOCFG[CRCCHK] is enabled, and these bytes are returned as part of the data buffer.

15.6.2.3 8-Bit GMII-Style Packet FIFO Mode

Figure 15-138 depicts the signals required to establish eTSEC module connection with an external device using the 8-bit FIFO interface.



[1]The flow control signals (TSEC*n*_CRS and TSEC*n*_COL) are common to all of the FIFO modes. TSEC*n*_CRS becomes an output signal in FIFO modes only.

Figure 15-138. eTSEC-FIFO (8-Bit) Connection

The 8-bit FIFO interface has 25 signals (including the flow control signals). Illustrative timing of the GMII-style FIFO mode is shown in Figure 15-139.

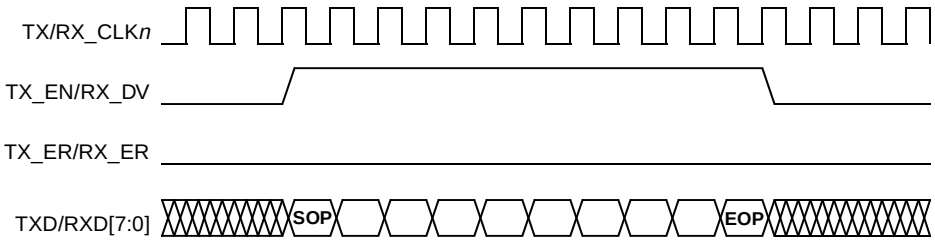


Figure 15-139. 8-Bit GMII-Style Packet FIFO Timing

The encoding of the eTSEC GMII signals in this FIFO mode is shown in Table 15-148.

Table 15-148. Signal Encoding for GMII-Style 8-Bit FIFO

Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	0 to 1 transition at start of cycle	0
Valid data	1	0
Valid data, end of packet	1 to 0 transition at end of cycle	0
Error	1	1 until TX_EN/RX_DV falls

In this mode flow control can control only the decision to continue transmitting packets, as packet transfers cannot be suspended once started.

15.6.2.4 8-Bit Encoded Packet FIFO Mode

The encoded packet 8-bit FIFO mode uses the signals shown in Figure 15-140. The control lines encode four states that can be associated with each beat of data. This mode should be used where invalid bytes can appear between the start and end of packet. Illustrative timing of the encoded packet FIFO mode is shown in Figure 15-140.

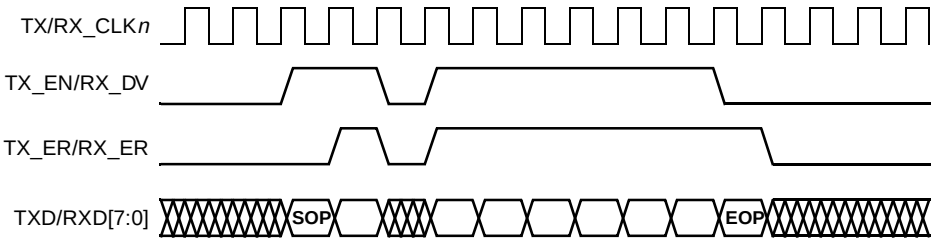


Figure 15-140. 8-Bit Encoded Packet FIFO Timing

The encoding of the eTSEC GMII signals in this FIFO mode is shown in Table 15-149.

Table 15-149. Signal Encoding for Encoded 8-Bit FIFO

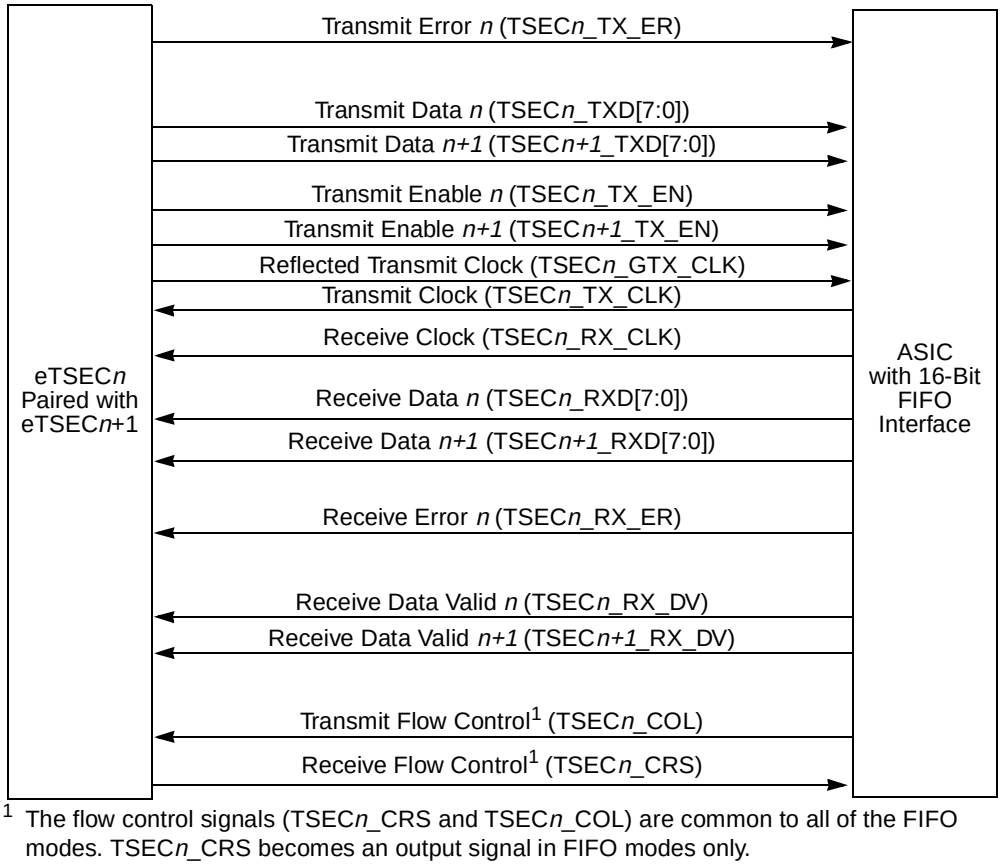
Condition	TX_EN/RX_DV	TX_ER/RX_ER
Valid data, start of packet	1	0
Valid data	1	1
Valid data, end of packet	0	1
Data not valid	0	0

In this mode flow control can cause an indefinite number of invalid data bytes to be transferred. This is the only mode in which an empty eTSEC Tx FIFO also causes a string of invalid data bytes to be transmitted rather than causing an underrun error.

15.6.2.5 16-Bit GMII-Style Packet FIFO Mode

Figure 15-141 depicts the signals required to establish eTSEC module connection with an external device using the 16-bit FIFO interface. The controlling eTSEC is marked as eTSEC*n*, with eTSEC*n+1* being

disabled in this mode. Data is transferred simultaneously on both pairs of RXD[7:0] and TXD[7:0]. The ordering of bytes in each 16 bits is such that eTSEC*n* receives/transmits the earlier byte ahead of the byte in the packet transferred by eTSEC*n+1*. In the case where an odd number of bytes are being transferred, the last byte on eTSEC*n+1* data signals are undefined.



1 The flow control signals (TSEC*n*_CRS and TSEC*n*_COL) are common to all of the FIFO modes. TSEC*n*_CRS becomes an output signal in FIFO modes only.

Figure 15-141. eTSEC-FIFO (16-Bit) Connection

The 16-bit FIFO interface has 44 signals (including the flow control signals). Illustrative timing of the GMII-style FIFO mode is shown in Figure 15-142. The eTSEC*n* control signals mark the extent of valid data, while the eTSEC*n+1* signals mark whether the paired byte is also valid. Any errors are indicated by the eTSEC*n* error signals, and held until the end of the packet.

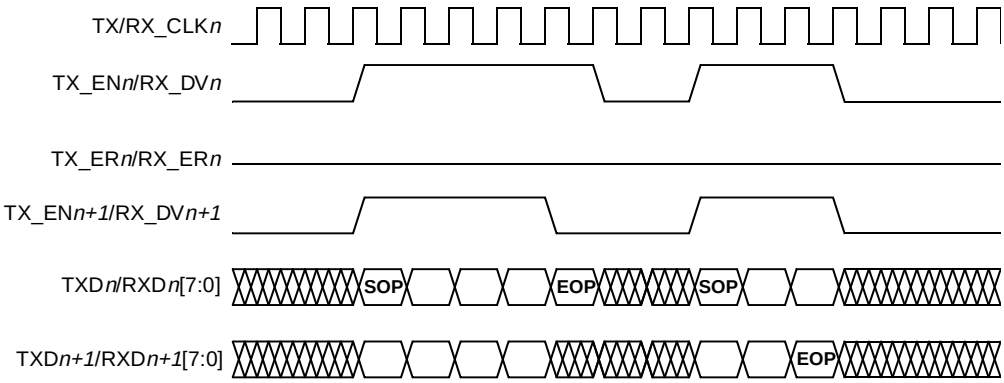


Figure 15-142. 16-Bit GMII-Style Packet FIFO Timing

The encoding of the paired eTSEC GMII signals in 16-bit GMII-style packet FIFO mode is shown in Table 15-150.

Table 15-150. Signal Encoding for GMII-Style 16-Bit FIFO

Condition	TX_EN*n*/RX_DV*n*	TX_ER*n*/RX_ER*n*	TX_EN*n*+1/RX_DV*n*+1
Data not valid	0	0	0
Start of packet, 2 bytes valid	0 to 1 transition at start cycle	0	0 to 1 transition at start cycle
Middle of packet, 2 bytes valid	1	0	1
End of packet, 1 byte valid	1 to 0 transition at end cycle	0	0
End of packet, 2 bytes valid	1 to 0 transition at end cycle	0	1 to 0 transition at end cycle
Error	1	1 held until TX_EN/RX_DV = 0 for both eTSECs	1

In 16-bit FIFO mode, one invalid byte and one valid byte received prior to the end of the packet is a signaling error which results in the CR bit of the RxBD being set for the frame in question (regardless of the state of FIFOCFG[CRCAPP]) and should be recognized by software as an indication that the receive data contains error(s).

In this mode flow control can control only the decision to continue transmitting packets, as packet transfers cannot be suspended once started.

15.6.2.6 16-Bit Encoded Packet FIFO Mode

The 16-bit encoded packet FIFO mode uses the control signals from both eTSECs to mark the status of data. The encoding scheme facilitates easy conversion to POS PHY Level 3 signaling via a PLD.

Illustrative timing of the encoded FIFO mode is shown in Figure 15-143. Note that GMII control signals from both participating eTSECs are combined into a 3-bit code, with the usual FIFO flow control signals operating in parallel.

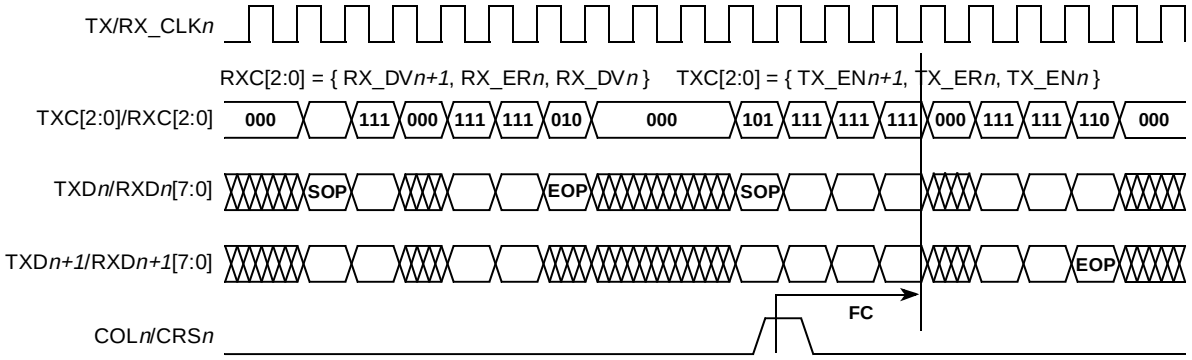


Figure 15-143. 16-Bit Encoded Packet FIFO Timing

The encoding of the paired eTSEC GMII signals in 16-bit encoded packet FIFO mode is shown in Table 15-151.

Table 15-151. Signal Encoding for Encoded 16-Bit FIFO

Condition	TXC[2:0][1]/RXC[2:0][2]
Data not valid	000
Reserved	001
End of packet, only eTSEC*n* byte valid	010
End of packet with error	011
Reserved	100
Start of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	101
End of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	110
Middle of packet, both eTSEC*n* and eTSEC*n+1* bytes valid	111

1 TXC[2:0] = {TX_EN*n+1*, TX_ER*n*, TX_EN*n*}
2 RXC[2:0] = {RX_DV*n+1*, RX_ER*n*, RX_DV*n*}

In this mode flow control can control either the decision to continue transmitting packets or insert invalid data (TXC/RXC = 000) into the data stream. Once asserted by the external receiver, transmit flow control takes effect after several (typically six) rising edges of TX_CLK. Should eTSEC momentarily run out of transmit data, it sends an indefinite stream of invalid data until normal transmission can resume; no transmit underrun can occur. In the event where the transmitter halts due to a lack of valid transmit BDs, the packet is terminated with an error.

15.6.2.7 FIFO Interface Signal Summary

Refer to Section 15.7.1.7, “8-Bit FIFO Mode” and Section 15.7.1.8, “16-Bit FIFO Mode,” for interface signal details.

15.6.3 Gigabit Ethernet Controller Channel Operation

This section describes the operation of the eTSEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Frame filtering and receive filing algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

15.6.3.1 Initialization Sequence

This sections describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the eTSEC.

15.6.3.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All eTSEC's registers and control logic are reset to their default states after a hard reset has occurred. In this state, each eTSEC behaves like a PowerQUICC III device, except for the absence of out-of-sequence TxBD features. That is, initially TCP/IP off-load is disabled and only single RxBD and TxBD rings are accessible.

15.6.3.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic eTSEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. See Table 15-3 for the register list. Table 15-152 describes the minimum steps for register initialization.

Table 15-152. Steps for Minimum Register Initialization

Description
1. Set and clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII Mgmt Interface
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT
7. Initialize IMASK
8. Initialize RCTRL
9. Initialize DMACTRL

After the initialization of registers is performed, the user must execute the following steps in the order described below to bring the eTSEC into a functional state (out of reset):

1. Write to the MACCFG1 register and set the appropriate bits. These need to include RX_EN and TX_EN. To enable flow control, Rx_Flow and Tx_Flow should also be set.

2. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE*n* registers. A minimum of two buffer descriptors per ring is required, unless the ring is disabled. Setting the ring to a size of one causes the same frame to be transmitted twice. If TCP/IP off-load is to be enabled, the TxBD[TOE] bit must be set for each frame.
3. Likewise, for the reception of Ethernet frames, the receive queue (or queues) must be ready, with its RxBD pointed to by the RBASE*n* registers. If TCP/IP off-load is to be enabled, RCTRL[PRSDEP] must be set to the required off-load level. Both transmit and receive can be gracefully stopped after transmission and reception begins.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. The DMACTRL[GRS] must be cleared if the receiver had been previously stopped. Refer to the DMACTRL register section, and Section 15.6.8.1, "Data Buffer Descriptors," for more information.

15.6.3.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft-reset to and/or reconfiguring the MAC with new parameters, the user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors are used, the user must load the pointers into the TBASE registers. Likewise if a new set of Rx buffer descriptors are used, the RBASE registers must be written with new pointers.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GRS/GTS bits in DMACTRL register
2. Poll GRSC/GTSC bits in IEVENT register until both are set
3. Set SOFT_RESET bit in MACCFG1 register (Note that SOFT_RESET must remain set for at least 3 TX clocks before proceeding.)
4. Clear SOFT_RESET bit in MACCFG1 register
5. Load TDBPH, TBASEH, TBASE0–TBASE7 with new Tx BD pointers
6. Load RDBPH, RBASEH, RBASE0–RBASE7 with new Rx BD pointers
7. Setup other MAC registers (MACCFG2, MAXFRM, etc.)
8. Setup group address hash table (GADDR0–GADDR15) if address filtering is required
9. Setup receive frame filer table (via RQFAR, RQFCR, and RQFPR) if filing to multiple RxBD rings is required
10. Setup WWR, WOP, TOD bits in DMACTRL register
11. Enable transmit queues in TQUEUE, and ensure that the transmit scheduling mode is correctly set in TCTRL.
12. Enable receive queues in RQUEUE, and optionally set TOE functionality in RCTRL.
13. Clear THLT and TXF bits in TSTAT register by writing 1 to them

14. Clear QHLT and RXF bits in RSTAT register by writing 1 to them.
15. Clear GRS/GTS bits in DMACTRL (do not change other bits)
16. Enable Tx_EN/Rx_EN in MACCFG1 register

15.6.3.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the eTSEC begins to poll the first transmit buffer descriptor (TxBD) in TxBD ring 0 every 512 transmit clocks. If TxBD[R] is set, and the TxBD ring is scheduled for transmission, the eTSEC begins copying the associated transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the GMII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, setting the DMACTRL[TOD] eliminates waiting for the next poll and a DMA transfer of the transmit data buffers can begin immediately. The transmission begins once all data for the frame is loaded into the Tx FIFO or sufficient transmit data (determined by the Tx FIFO threshold register) is in the Tx FIFO. If the line is not busy, the MAC transmit logic asserts TX_EN and sends the 7-octet preamble sequence, 1-octet start of frame delimiter, and frame information in that order. If the line is busy, the controller waits for the carrier sense signal, CRS, to remain inactive for 60 bit times (60 clocks) and transmission begins after an additional 36 bit times (96 bit times after CRS became active). In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap (96 bit times) regardless of CRS.

In half-duplex mode (MACCFG2[Full Duplex] is cleared) the MAC defers transmission if the line is busy (CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after CRS originally became negated). If CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is re-transmitted in case of a collision. This avoids unnecessary memory traffic.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap must be enforced.

The transmitter implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter’s request to send data until the pause duration is over. If the MAC is currently sending data after a pause frame has been received and processed, the MAC finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. In addition, the transmitter supports transmission of flow control frames via TCTRL[TFC_PAUSE]. The transmit pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD

- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC] is set

The Tx_EN is negated after the FCS is sent. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes with zero bytes to make up the minimum length.

To pause transmission, or rearrange the transmit queue, set DMACTRL[GTS]. This can be useful for transmitting expedited data ahead of previously-linked buffers or for error situations. If this bit is set, the eTSEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until all queued frames in the Tx FIFO have been disposed of. The IEVENT[GTSC] interrupt occurs once the graceful transmit stop operation is completed. After the DMACTRL[GTS] is cleared, the eTSEC resumes transmission with the next frame.

While the eTSEC is in 10/100Mbps mode it sends bytes least-significant nibble first and each nibble is sent lsb first. While it is in 1000Mbps mode it sends bytes LSB first.

15.6.3.4 Gigabit Ethernet Frame Reception

The eTSEC Ethernet receiver is designed to work with little core intervention and can perform data extraction, address recognition, CRC checking, short frame checking, and maximum frame-length checking.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. The MAC checks for when TSEC*n*_RX_DV is asserted and as long as TSEC*n*_COL remains negated (full-duplex mode ignores TSEC*n*_COL), the MAC looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped (unless MACCFG2[PreAM RxEN] is set) and the frame begins to be processed. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the eTSEC controller begins to perform the frame recognition function through destination address (DA) recognition (see Section 15.6.3.7, “Frame Recognition”). Based on this match the frame can be accepted or rejected. The receiver can filter frames based on individual (unicast), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, system bus usage is not wasted on frames unwanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from either queue 0 or the queue determined by the filer. If the RxBD is not being used by software (RxBD[E] is set), the eTSEC starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame. If the current RxBD is not available for the received frame, a receive busy error condition is raised in IEVENT[BSY].

After the buffer is filled, the eTSEC clears RxBD[E] and, if RxBD[I] is set, generates an interrupt. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table. If it is empty, the controller continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicate LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBLR). The smallest valid value is 64 bytes, with larger values being be some integral multiple of 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize an oversized frame condition.

Receive frames are not truncated when they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the eTSEC receiver performs a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signaled after the graceful receive stop operation is completed. While in this mode the user can write to registers that are accessible to both the user and the eTSEC hardware without fear of conflict, and finally clear IEVENT[GRSC]. After DMACTRL[GRS] is cleared, the eTSEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

15.6.3.5 Ethernet Preamble Customization

By default eTSEC generates a standard Ethernet preamble sequence prior to transmitting frames. However, the user can substitute a custom preamble sequence for the purpose of controlling switching equipment at the receiver, particularly at 100/1000Mbps speeds. In FIFO mode preamble customization is ignored; in any RMII mode only the standard preamble can be transmitted.

eTSEC normally searches for and discards the standard Ethernet preamble sequence upon receiving frames. Part of the received preamble sequence can be optionally recovered and returned as part of the frame data, making it visible to user software. Note however, that no preamble is received in FIFO mode, and preamble cannot be recovered in any RMII mode. Note that it is also possible for the first two bytes of custom preamble (PreOct0 and PreOct1) to be lost in during conversion to ten-bit code groups in the PCS sub-layer. Thus is it recommended that any custom preamble start at PreOct2.

15.6.3.5.1 User-Defined Preamble Transmission

To substitute a custom preamble, the user must ensure that:

- MACCFG2[PreAm TxEN] bit is set
- The first TxBD of every frame containing a custom preamble has its PRE bit set
- An 8-byte custom preamble sequence appears before the Ethernet DA field in the first transmit data buffer

The definition of the 8-byte custom preamble sequence is shown in Figure 15-144.

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0								PreOct1							
2–3	PreOct2								PreOct3							
4–5	PreOct4								PreOct5							
6–7	PreOct6															

Figure 15-144. Definition of Custom Preamble Sequence

The fields of the custom preamble sequence are described in Table 15-153. It should be noted that use of preamble octets matching the standard start of frame delimiter (0xD5) can be expected to trigger premature frame reception by the receiving station.

Table 15-153. Custom Preamble Field Descriptions

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of custom transmit preamble. This is the first octet of preamble sent.
	8–15	PreOct1	Octet #1 of custom transmit preamble. This is the second octet of preamble sent.
2–3	0–7	PreOct2	Octet #2 of custom transmit preamble. This is the third octet of preamble sent.
	8–15	PreOct3	Octet #3 of custom transmit preamble. This is the fourth octet of preamble sent.
4–5	0–7	PreOct4	Octet #4 of custom transmit preamble. This is the fifth octet of preamble sent.
	8–15	PreOct5	Octet #5 of custom transmit preamble. This is the sixth octet of preamble sent.
6–7	0–7	PreOct6	Octet #6 of custom transmit preamble. This is the seventh octet of preamble sent. The last octet (the start of frame delimiter) is generated by the MAC automatically.
	8–15	—	Reserved; should be cleared.

15.6.3.5.2 User-Visible Preamble Reception

To return the received preamble, the user must ensure that:

- MACCFG2[PreAm RxEN] bit is set
- Space for an 8-byte preamble sequence is allowed before the Ethernet DA field in the first receive data buffer of each frame

The definition of the 8-byte received preamble sequence is shown in Figure 15-145.

Byte Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0–1	PreOct0								PreOct1							
2–3	PreOct2								PreOct3							
4–5	PreOct4								PreOct5							
6–7	PreOct6															

Figure 15-145. Definition of Received Preamble Sequence

The fields of the received preamble sequence are described in Table 15-154. Should the received preamble be shorter than the 7-octet sequence defined by IEEE 802.3, initial bytes of the received preamble sequence hold undefined values. The standard start of frame delimiter (0xD5) is always omitted. Note that preamble extraction is not possible in RMII mode.

Table 15-154. Received Preamble Field Descriptions

Bytes	Bits	Name	Description
0–1	0–7	PreOct0	Octet #0 of received preamble. This is the first octet of preamble received.
	8–15	PreOct1	Octet #1 of received preamble. This is the second octet of preamble received.
2–3	0–7	PreOct2	Octet #2 of received preamble. This is the third octet of preamble received.
	8–15	PreOct3	Octet #3 of received preamble. This is the fourth octet of preamble received.
4–5	0–7	PreOct4	Octet #4 of received preamble. This is the fifth octet of preamble received.
	8–15	PreOct5	Octet #5 of received preamble. This is the sixth octet of preamble received.
6–7	0–7	PreOct6	Octet #6 of received preamble. This is the seventh octet of preamble received. The last octet (the start of frame delimiter) is discarded.
	8–15	—	Reserved

15.6.3.6 RMON Support

By using promiscuous mode, the eTSEC can automatically gather network statistics required for remote network interface monitoring. The RMON MIB group 1, RMON MIB group 2, RMON MIB group 3, RMON MIB group 9, RMON MIB2, and the 802.3 Ethernet MIB are supported. For RMON statistics and their corresponding counters, see the memory map.

15.6.3.7 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

15.6.3.7.1 Destination Address Recognition and Frame Filtering

The eTSEC can perform layer 2 frame filtering on the basis of destination Ethernet address (DA), as illustrated by the flowchart in Figure 15-146.

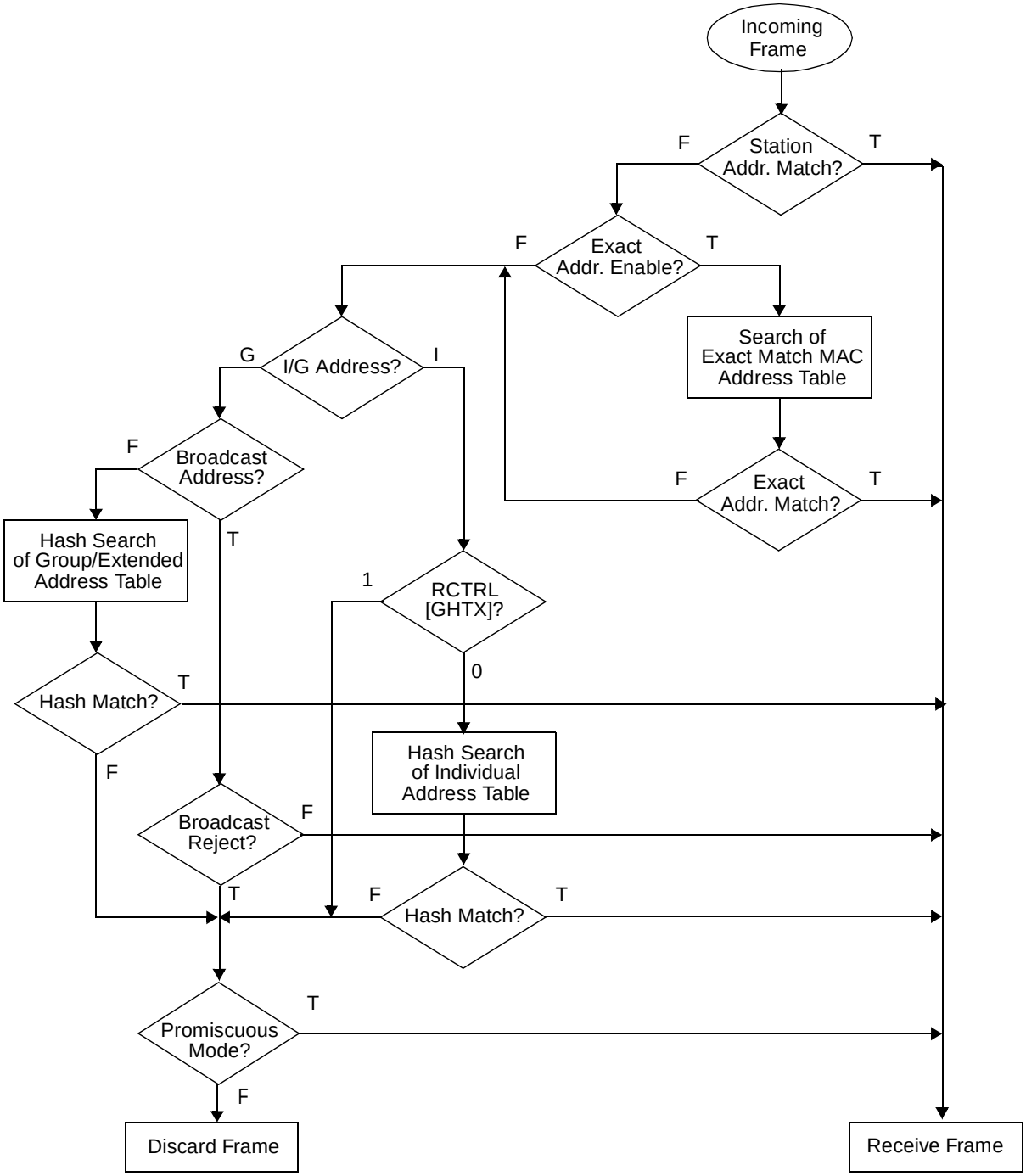


Figure 15-146. Ethernet Address Recognition Flowchart

In promiscuous mode, the eTSEC accepts all received frames regardless of DA. Note, however, that Ethernet frame filtering simply restricts the traffic seen by the receive queue filer. Therefore even in

promiscuous mode it remains possible to program the filer to reject frames based on their higher-layer header contents.

In the case of an individual address, the DA field of the received frame is compared with the physical address that the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, and exact MAC address matching is enabled via RCTRL[EMEN], the controller performs address recognition on the multiple MAC addresses written to the MAC*x*ADDR1 and MAC*x*ADDR2 registers. These virtual addresses give a particular eTSEC the ability to mirror other MACs on the network, which caters for router redundancy protocols, such as HSRP and VRRP.

If exact MAC address matching is not enabled, the eTSEC determines whether DA is a group or individual address. If DA is the standard broadcast address, and broadcast addresses are not rejected, the frame is accepted. If any other group address is received, the eTSEC looks-up the DA by means of the group hash table. The group hash table may be extended to 512 entries if RCTRL[GHTX] = 1. Otherwise, an individual address is hashed into the 256-entry individual hash table when RCTRL[GHTX] = 0.

15.6.3.7.2 Hash Table Algorithm

The hash table process used in the group hash filtering operates as follows. By default, the Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in IGADDR0–IGADDR7 for individual addresses, and the 256 bits in GADDR0–GADDR7 for group addresses. But in the case where RCTRL[GHTX] is set, both sets of registers are combined into an extended group-only hash table of 512 bits, where IGADDR0–IGADDR7 contain the first 256 bits and GADDR0–GADDR7 contain the last 256 bits. No individual-address table exists in extended mode.

The 48-bit destination address received by the MAC is passed through the Ethernet CRC-32 algorithm to produce a hash value. The CRC polynomial used is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The MAC initializes its CRC register to 0xFFFFFFFF before computing a CRC on the 6 bit-reversed octets of the DA. A non-optimized sample of C code for computing the DA hash is listed in Figure 15-147. The 9 most significant bits of the raw, uninverted CRC are used as the hash table index, H[8:0]. If RCTRL[GHTX] = 0, bits H[8:6] select one of the 8 IGADDR or GADDR registers, while bits H[5:1] select a bit within the 32-bit register. If RCTRL[GHTX] = 1, bits H[8:5] select one of the 16 registers in the {IGADDR, GADDR} set, while bits H[4:0] select a bit within the 32-bit register. For example, if H[8:5] = 7, IGADDR7 is selected, whereas H[8:5] = 9 selects GADDR1.

```
/* Wrapper macros for 256-bucket and 512-bucket hash tables:
   Pass 6-byte Ethernet MAC address as parameter. */
#define TSEC_HASH256(macaddr) ((crc32(macaddr) >> 24) & 0xff)
#define TSEC_HASH512(macaddr) ((crc32(macaddr) >> 23) & 0x1ff)

/* CRC constants.  Note: CRC-32 polynomial is bit-reversed. */
#define CRC_POLYNOMIAL 0xedb88320
#define CRC_INITIAL    0xffffffff
#define MAC_ADDRLEN    6
#define BITS_PER_BYTE  8

/* crc32() Takes the array of bytes, macaddr[], representing an
   Ethernet MAC address and returns the CRC-32 result over these bytes,
   where each byte is used in bit-reversed form (Ethernet bit order).
   Index 0 of macaddr[] is the first byte of the address on the wire.
   Test case: the result of crc32 on {0x00, 0x01, 0x02, 0x03, 0x04, 0x05}
   should be 0xad0c28f3.
 */
unsigned long crc32(unsigned char macaddr[MAC_ADDRLEN])
{
  unsigned long crc, result;
  int byte, i;

  /* CRC-32 algorithm starts by inverting first 4 bytes */
  crc = CRC_INITIAL;
  /* add each byte to running CRC accumulator */
  for (byte = 0; byte < MAC_ADDRLEN; ++byte) {
    crc ^= macaddr[byte];
    /* shift CRC right to perform but reversal on byte of address */
    for (i = 0; i < BITS_PER_BYTE; ++i)
      if (crc & 1)
     crc = (crc >> 1) ^ CRC_POLYNOMIAL;
      else
      crc >>= 1;
  }
  /* finally, reverse bits of result to get CRC in normal bit order */
  for (result = 0, i = 4*BITS_PER_BYTE-1; i >= 0; crc >>= 1, --i)
    result |= (crc & 1) << i;
  return result;
}
```

Figure 15-147. Sample C Code for Computing eTSEC Hash Table Indices

If the CRC hash table index selects a bit that is set in the hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the extended hash table prevents roughly 480/512 (93.8%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses. Alternatively, small multicast groups can be held in the exact match MAC address registers, which guarantees that only correct frames are admitted.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 512-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

NOTE

The hash table cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. The receive queue filer may be used to reject frames with unintended address hits in the hash table.

15.6.3.8 Magic Packet Mode

eTSEC implements the AMD Magic Packet™ specification for LAN-initiated power management. This mode is normally entered with the rest of the system in a low-power doze or nap mode. Note that sleep mode is not supported with Magic Packet. Software must enable normal receive function in the Ethernet MAC, and then finally set the MACCFG2[MPEN] bit to enable Magic Packet detection before the system enters a reduced mode. While the rest of the system is operating in low-power mode, the enabled eTSEC continues to receive Ethernet frames, but discards them immediately. Upon receipt of any frame whose contents contain the valid Magic Packet sequence, the eTSEC exits out of Magic Packet mode, thus clearing MACCFG2[MPEN], and raises an error/diagnostic interrupt via IEVENT[MAG], which causes the surrounding system to wake-up. Frames received after Magic Packet mode has exited are received into software buffers as usual. Software can abort Magic Packet mode by writing 0 to MACCFG2[MPEN] at any time.

AMD specify a Magic Packet™ to be any Ethernet frame containing a valid Ethernet header (Destination and Source Addresses) and valid FCS (CRC-32), and whose payload includes the specific Magic Packet byte sequence at any offset from the start of frame. The specific byte sequence comprises an unbroken stream of 102 bytes, the first 6 bytes of which are 0xFFFFFF_FFFFFF, followed by 16 copies of the MAC's unique IEEE station address in the normal byte order for Ethernet addresses. For example, if the station address were 0x112233_445566, then the MAC would have to receive 0xFFFFFF_FFFFFF, 0x112233_445566, ..., 0x112233_445566 in any payload to detect a Magic Packet. Only frames addressed specifically to the MAC's station address or a valid multicast or broadcast address can be examined for the Magic Packet sequence.

15.6.3.9 Flow Control

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. If the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value.

Table 15-155 lists the flow-control frame structure.

Table 15-155. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble	—	—
1	SFD	—	Start frame delimiter
6	Destination address	01-80-C2-00-00-01	Multicast address reserved for use in MAC frames (or MAC station address)
6	Source address	—	—

Table 15-155. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter	—	Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed independent constant of 512 bit-times (unlike slot time). The most-significant octet is transmitted first.
2	Extended MAC parameter	—	Pause time extended as defined by the PTV[PTE] field. The most significant octet is transmitted first.
40	Reserved	—	—
4	FCS	—	Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. Since the pause timer commences counting immediately upon receipt of a PAUSE frame, regardless of whether transmission is currently in progress, a sufficiently large pause time must be received to stop transmission past a frame of MTU size. During a pause, only a control frame can be sent (TCTRL[TFC_PAUSE] is set). Normal transmission resumes after the pause timer stops counting, or resumes immediately if a pause frame with a zero time-out is received. If another pause-control frame is received during the pause, the period changes to the new value received.

15.6.3.10 Interrupt Handling

The following describes what usually occurs within a eTSEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time. There are three kinds of interrupt:
  - Errors (all bits in IEVENT other than RXB, RXF, TXB, or TXF)
  - Receive interrupts, when bits RXB or RXF in IEVENT are set
  - Transmit interrupts, when bits TXB or TXF in IEVENT are set
- Process the TxBDs to reuse them if the IEVENT[TXB, TXF or TXE] were set. Consult register bits TSTAT[TXF0–TXF7] to determine which TxBD rings gave rise to the transmit interrupt in the case of TXF. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the eTSEC; thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set.
- Obtain data from RxBD rings if IEVENT[RXC, RXB or RXF] is set. Consult register bits RSTAT[RXF0–RXF7] to determine which RxBD rings gave rise to the receive interrupt in the case of RXF. If the receive speed is fast or the interrupt delay is long, the eTSEC may have received more than one RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the eTSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch, otherwise a BSY error occurs.

- Clear any set halt or frame interrupt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS] by writing 1s to these bits.
- Continue normal execution.

Table 15-156. Non-Error Transmit Interrupts

Interrupt	Description	Action Taken by the eTSEC
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, was updated in one of the enabled TxBD rings.	Programmable ‘write with response’ TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame from an enabled TxBD ring was transmitted and the last transmit buffer descriptor (TxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last TxBD before setting IEVENT[TXF].

Table 15-157. Non-Error Receive Interrupts

Interrupt	Description	Action Taken by the eTSEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated in one of the enabled RxBD rings.	Programmable ‘write with response’ RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received to an enabled RxBD ring and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last RxBD before setting IEVENT[RXF].

15.6.3.10.1 Interrupt Coalescing

Interrupt coalescing offers the user the ability to contour the behavior of the eTSEC with regard to frame interrupts. Separate but identical mechanisms exist for both transmitted frames and received frames. In either case, frame interrupts require that software set the I-bit in RxBDs or TxBDs, and disable buffer interrupts (IEVENT[RXB] or IEVENT[TXB]). Particular rings can remain free of interrupts by ensuring that the I-bit is consistently cleared in all BDs. While interrupt coalescing is enabled, a transmit or receive frame interrupt is raised either when a counter threshold-defined number of frames is received/transmitted or the timer threshold-defined period of time has elapsed, whichever occurs first. Disabling and then re-enabling interrupt coalescing forces reset of the coalescing timers and counters to reflect changes made to the threshold registers.

15.6.3.10.2 Interrupt Coalescing By Frame Count Threshold

To avoid interrupt bandwidth congestion due to frequent, consecutive interrupts, the user may enable and configure interrupt coalescing to deliberately group frame interrupts, reducing the total number of

interrupts raised. The number of frames received or transmitted prior to an interrupt being raised is determined by the frame threshold field (ICFT) in the appropriate interrupt coalescing configuration register (RXIC or TXIC). The frame threshold field may be assigned a value between 1 and 255. The internal transmit or receive frame counter decrements from this initial value each time a frame is transmitted or received. Upon reaching zero, an interrupt is raised, the appropriate threshold counter is reset to the value in the ICFT field, and then eTSEC continues counting frames while the interrupt is active. The appropriate threshold counter is also reset to the value in the ICFT field if an interrupt is raised subject to the corresponding threshold timer.

15.6.3.10.3 Interrupt Coalescing By Timer Threshold

To avoid stale frame interrupts, the user may also assign a timer threshold, beyond which any frame interrupts not yet raised are forced. The timer threshold fields of the receive and transmit interrupt coalescing configuration registers (RXIC[ICTT] and TXIC[ICTT]) are defined in units equivalent to 64 interface clocks or system clocks, depending on the setting of the ICCS field in RXIC and TXIC.

After transmitting a frame, the transmit interrupt coalescing threshold time begins counting down from the value in TXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful transmit stop completion before the coalescing timer expires, the eTSEC issues two interrupts, the first for GTS, the second for TXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GTS event, it is recommended that the user mask out the TXF event during execution of the service routine. After receiving a frame, the receive interrupt coalescing threshold time begins counting down from the value in RXIC[ICTT]. An interrupt is raised when the counter reaches zero. In the event of graceful receive stop completion before the coalescing timer expires, the eTSEC issues two interrupts, the first for GRS, the second for RXF (due to timer expiration of a pending event). To prevent the second interrupt from affecting servicing of the GRS event, it is recommended that the user mask out the RXF event during execution of the service routine.

The interrupt coalescing timer thresholds (transmit and receive, operating independently) may be values ranging from 0x0001 to 0xFFFF. Table 15-158 specifies the range of possible timing thresholds subject to timer clock source, the interface or system frequency, and the value of the RXIC[ICTT] or TXIC[ICTT] field.

Table 15-158. Interrupt Coalescing Timing Threshold Ranges

ICCS (Clock Source)	eTSEC Interface Format and Frequency or eTSEC System Frequency	Interrupt Coalescing Threshold Time	
		Minimum (ICTT = 0x0001)	Maximum (ICTT = 0xFFFF)
0 (I/F clock)	10Base-T at 2.5 MHz	25.6 µs	1.68 s
0 (I/F clock)	100Base-T at 25 MHz	2.56 µs	168 ms
0 (I/F clock)	1000Base-T at 125 MHz	0.51 µs	33.6 ms
1 (sys. clock)	eTSEC operating at 266 MHz	0.24 µs	15.7 ms
1 (sys. clock)	eTSEC operating at 333 MHz	0.19 µs	12.6 ms

The transmit timer threshold counter is reset to the value in TXIC[ICTT] and begins counting down on transmission of the frame following an interrupt.

The receive timer threshold counter is reset to the value in RXIC[ICTT] and begins counting down on receiving the frame following an interrupt.

15.6.3.11 Inter-Frame Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-frame gap, or IFG). The minimum inter-packet gap (IPG) time for back-to-back transmission is set by IPGIFG[Back-to-Back Inter-Packet-Gap]. The receiver receives back-to-back frames with the minimum interframe gap (IFG) as set in IPGIFG[Minimum IFG Enforcement]. If multiple frames are ready to transmit, the ethernet controller follows the minimum IPG as long as the following restrictions are met:

- The first TxBD pointer, TBPTR*n*, of any given frame is located at a 16-byte aligned address.
- Each TxBD[Data Length] is greater-than or equal to 64 bytes.

If the first TxBD alignment restriction is not met, the back-to-back IPG may be as many as 32 cycles. If the TxBD size restriction is not met, the back-to-back IPG may be significantly longer.

In half-duplex mode, after a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN (carrier sense negated) and then begins retransmission (retry) on the LAN. Retransmission begins 36 bit times after carrier sense is negated for at least 60 bit times. If the frame is not successfully sent within a specified number of retries, an error is indicated (collision retry limit exceeded).

15.6.3.12 Internal and External Loop Back

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default. Clearing this bit results in normal operation.

15.6.3.13 Error-Handling Procedure

The eTSEC reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in Table 15-159.

Table 15-159. Transmission Errors

Error	Response
Transmitter underrun	Transmitter underrun can occur either after frame transmission has commenced, or in response to an incomplete sequence of TxBDs. In the former case, the controller sends 32 bits that ensure a CRC error, and terminates buffer transmission. In the latter case, the relevant transmit queue is halted. In all cases, the eTSEC closes the buffer, sets TxBD[UN], IEVENT[XFUN], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, IEVENT[CRL], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).

Table 15-159. Transmission Errors (continued)

Error	Response
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, IEVENT[LC], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error, and therefore TxBD[UN] is set, but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Data parity error	Data in the transmit FIFO was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues transmission until halted explicitly.
Babbling transmit error	A frame is transmitted which exceeds the MAC's Maximum Frame Length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption.

Reception errors are described in Table 15-160.

Table 15-160. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, increments the discarded frame counter (RDRP), and sets IEVENT[RXF], The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY] and increments the discarded frame counter (RDRP). In addition, the RSTAT[QHLT*n*] bit is set. RDRP increments for each frame that is received while the receiver is halted due to a busy condition. The halted queue resumes reception once the RSTAT[QHLT*n*] bit is cleared.
Filed frame to invalid queue error	A frame is received and discarded as a result of the filer directing it to an RxBD ring that is currently not enabled. The controller sets IEVENT[FIQ] and increments the discarded frame counter (RDRP).
Parser error	If the receive frame parser is enabled, a parse error can be flagged as a result of inconsistencies discovered between fields of the embedded packet headers. For example, the L2 header may indicate an IPv4 header, but the IP version number fails to match. In the event of a parse error, parsing is terminated at the inconsistent header, and the RxFCB[PERR] field indicates at which layer of the protocol stack the error was discovered. Receiver function continues regardless of parse errors, but IEVENT[PERR] is set. The receive queue filer may operate with reduced or default information in some cases; therefore, filer rule sets should be constructed so as to be tolerant of misformed frames. **Note:** Any values in the length/type field between 1500 and 1536 are treated as a length, however, only illegal packets exist with this length/type since these are not valid lengths and not valid types. These are treated by the MAC logic as out of range. Software must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The eTSEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.

Table 15-160. Reception Errors (continued)

Error	Description
CRC error	If a CRC error occurs, the controller sets RxBD[CR], closes the buffer, and sets IEVENT[RXF]. This eTSEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame and increments the discarded frame counter (RDRP). In addition the RSTAT[QHLT*n*] bit is set. The halted queue resumes reception once the RSTAT[QHLT*n*] bit is cleared.
Data parity error	Data in the receive FIFO or filer table was potentially corrupted. The controller sets IEVENT[DPE], but otherwise continues reception until halted explicitly.
Babbling receive error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues.

15.6.4 TCP/IP Off-Load

Each eTSEC provides hardware support for accelerating the basic functions of TCP/IP packet transmission and reception. By default, these features are disabled and must be explicitly enabled via RCTRL and TCTRL. In this configuration, the eTSEC processes frames as vanilla Ethernet frames and none of the multi-ring QoS/CoS receive services or per-frame VLAN insertion and deletion are available. Operate eTSEC in this default configuration when using existing TCP/IP stack software that has not been modified to take advantage of TOE.

TOE can be enabled independently for Rx and Tx and at various levels. Receive TOE functions are controlled by RCTRL and transmit functions via a combination of TCTRL[TUCSEN] and the Tx frame control block.

On receive, according to RCTRL[PRSDEP], eTSEC can parse frames at layer 2 of the stack only (Ethernet headers and switching headers), layers 2 to 3 (including IPv4 or IPv6), or layers 2 to 4 (including TCP and UDP). TOE provides protocol header recognition, header verification (IPv4 header checksum verification), and TCP/UDP payload checksum verification including verification of associated pseudo-header checksums. For large frames off-load of checksum verification saves a significant fraction of the CPU cycles that would otherwise be spent by the TCP/IP stack. IP packet fragmentation and re-assembly, and TCP stream establishment and tear-down are not performed in hardware. The frame parser sets RQFPR[IPF] status flag encountering a fragmented frame. The frame parser in eTSEC searches a maximum of 512 bytes from the start of a received frame when attempting to locate headers; headers deeper than 512 bytes are assumed not to exist, and any associated receive status flags in the frame control block remain cleared.

On transmit, TOE provides IPv4 and TCP/UDP header checksum generation. Like receive TOE, checksum generation reduces CPU load significantly for TCP/IP stacks modified to exploit eTSEC TOE functions. The eTSEC does not checksum transmitted packets with IPv6 routing headers or calculate TCP/UDP checksums from IP fragments. If a transmitted TCP segment requires checksum generation but IPv6 extension headers would prevent eTSEC from calculating the pseudo-header checksum, software can calculate just the pseudo-header checksum in advance and supply it to the eTSEC as part of per-frame TOE configuration.

15.6.4.1 Frame Control Blocks

Frame control blocks (FCBs) are 8-byte blocks of TOE control and/or status data that are passed between software (driver and TCP/IP stack) and each eTSEC. A FCB always precedes the frame it applies to, and is present only when TOE functions are being used. As Figure 15-148 shows, the first BD of each frame points to the initial data buffer and the FCB. The initial data buffer must be at least 8 bytes long to contain the FCB without breaking it. Custom or received Ethernet preamble sequences also follow the FCB if preambles are visible.

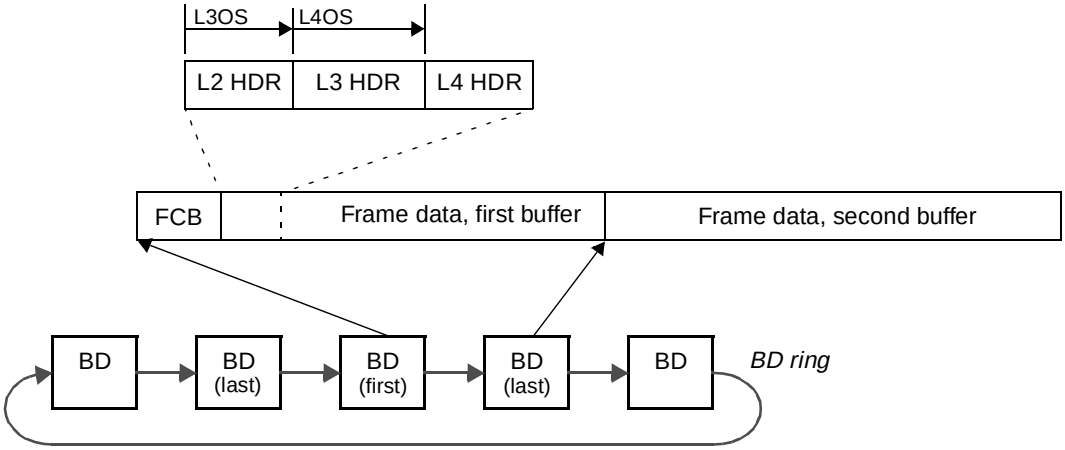


Figure 15-148. Location of Frame Control Blocks for TOE Parameters

For TxBD rings, FCBs are assumed present when the TxBD[TOE/UN] bit is set by user software. The eTSEC ignores the TxBD[TOE/UN] bit in all BDs other than those pointing to initial data buffers, therefore FCBs must not be inserted in second and subsequent data buffers. Since TxBD[TOE/UN] can be set under software discretion, TOE acceleration for transmit may be applied on a frame-by-frame basis.

In the case of RxBD rings, FCBs are inserted by the eTSEC whenever RCTRL[PRSDEP] is set to a non-zero value. Only one FCB is inserted per frame, in the buffer pointed to by the RxBD with bit F set. TOE acceleration for receive is enabled for all frames in this case.

15.6.4.2 Transmit Path Off-Load and Tx PTP Packet Parsing

TOE functions for transmit are defined by the contents of the Tx FCB. Figure 15-149 describes the definition for the Tx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	UDP	CIP	CTU	NPH								
Offset + 2	L4OS								L3OS							
Offset + 4	PHCS															
Offset + 6	VLCTL															

Figure 15-149. Transmit Frame Control Block

The user instructs the Tx packet to be timestamped via setting bit 15 in the TxFCB to mark a PTP packet. TxFCB[VLCTL] can be translated as the Tx PTP packet identification number. BD[TOE] has to be set to enable transmit PTP packet time stamping. TxFCB[PTP] bit takes precedence over TxFCB[VLN] bit. It disables per packet VLAN tag insertion. On a PTP packet, VLAN tag can be inserted from the DFVLAN register. A proposed TxFCB update for the PTP packet is shown in Figure 15-154.

The contents of the Tx FCB are defined in Table 15-161.

Table 15-161. Tx Frame Control Block Description

Bytes	Bits	Name	Description
0–1	0	VLN	VLAN control word valid. This bit is ignored when the PTP bit is set. VLAN tag is read from the DFVLAN register if PTP=1. 0 Ignore VLCTL field. 1 If VLAN tag insertion is enabled for eTSEC, use the VLCTL field as the VLAN control word.
	1	IP	Layer 3 header is an IP header. 0 Ignore layer 3 and higher headers. 1 Assume that the layer 3 header is an IPv4 or IPv6 header, and take L3OS field as valid.
	2	IP6	IP header is IP version 6. Valid only if IP = 1. 0 IP header version is 4. 1 IP header version is 6.
	3	TUP	Layer 4 header is a TCP or UDP header. 0 Do not process any layer 4 header. 1 Assume that the layer 4 header is either TCP or UDP (see UDP bit), and offload checksumming on the basis that the IP header has no extension headers.
	4	UDP	UDP protocol at layer 4. 0 Layer 4 protocol is either TCP (if TUP = 1) or undefined. 1 Layer 4 protocol is UDP if TUP = 1.
0–1	5	CIP	Checksum IP header enable. 0 Do not generate an IP header checksum. 1 Generate an IPv4 header checksum.
	6	CTU	Checksum TCP or UDP header enable. 0 Do not generate a TCP or UDP header checksum. RFC 768 advises that UDP packets not requiring checksum validation should have their checksum field set to zero. 1 Generate a TCP header checksum if IP = 1 and TUP = 1 and UDP = 0.
	7	NPH	Disable calculation of TCP or UDP pseudo-header checksum. This bit should be set if IP options need to be consulted in forming the pseudo-header checksum, as eTSEC does not examine IP options or extension headers for TCP/IP offload on transmit. 0 Calculate TCP or UDP pseudo-header checksum as normal, assuming that the IP header has no options. 1 Do not calculate a TCP or UDP pseudo-header checksum, but instead use the value in field PHCS when determining the overall TCP or UDP checksum.
	8–14	—	Reserved
	15	PTP	Indication to the transmitter that this is a PTP packet. Enabling PTP disables per packet VLAN tag insertion. Instead, VLAN tag is read from the DFVLAN when the PTP field is true. 0 Do not attempt to capture transmission event time 1 Valid PTP_ID field. When this packet is transmitted, capture the time of transmission. Must be clear if TMR_CTRL[TE] is clear.

Table 15-161. Tx Frame Control Block Description (continued)

Bytes	Bits	Name	Description
2–3	0–7	L4OS	Layer 4 header offset from start of layer 3 header. The layer 4 header starts L4OS octets after the layer 3 header if it is present. The maximum layer 3 header length supported is thus 255 bytes, which may prevent TCP/IP offload on particularly large IPv6 headers.
	8–15	L3OS	Layer 3 header offset from start of frame not including the 8 bytes for this FCB. The layer 3 header starts L3OS octets from the start of the frame including any custom preamble header that may be present. The maximum layer 2 header length supported is thus 255 bytes.
4–5	0–15	PHCS	Pseudo-header checksum (16-bit one's complement sum with carry wraparound, but without result inversion) for TCP or UDP packets, calculated by software. Valid only if NPH = 1.
6–7	0–15	VLCTL/ PTP_ID	VLAN control word for insertion in the transmitted VLAN tag. Valid only if VLN = 1.Tx PTP packet identification number. This number is copied into the Tx PTP packet time stamp identification field. PTP field takes precedence over VLN field.

15.6.4.3 Receive Path Off-Load

Upon receive, the Rx FCB returns the status of frame parse and TOE functions applied to the accompanying frame. Figure 15-150 describes the definition for the Rx FCB.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	VLN	IP	IP6	TUP	CIP	CTU	EIP	ETU					PERR			
Offset + 2			RQ						PRO							
Offset + 4																
Offset + 6	VLCTL															

Figure 15-150. Receive Frame Control Block

The contents of the Rx FCB are defined in Table 15-162.

Table 15-162. Rx Frame Control Block Descriptions

Bytes	Bits	Name	Description
0–1	0	VLN	VLAN tag recognized. This bit is set only if RCTRL[VLEX] is set. 0 No VLAN tag recognized. 1 IEEE 802.1Q VLAN tag found; VLAN control word in VLCTL is valid.
	1	IP	IP header found at layer 3.RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IP discovery. See also IP6 bit of FCB. 0 No layer 3 header recognized. 1 An IP header was recognized at layer 3; the IANA protocol identifier for the next header can be found in PRO; see PRO for more information. If S/W is relying on the RxFCB for the parse results, any RxFCB[IP] bits set with the corresponding RxFCB[PRO] = 0xFF indicates a fragmented packet (or that this packet had a back-to-back IPv6 routing extension header). Additionally, RQFPR[IPF] (see Section 15.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR)”) indicates that the packet was fragmented.
	2	IP6	IP version 6 header found at layer 3. 0 No IPv6 header was found. 1 The layer 3 header was an IPv6 header provided IP = 1.
	3	TUP	TCP or UDP header found at layer 4. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable TCP/UDP discovery. 0 No layer 4 header recognized. 1 The layer 4 header was recognized as either TCP (PRO = 0x06) or UDP (PRO = 0x11).
	4	CIP	IPv4 header checksum checked. RCTRL[PRSDEP] must be set to 10 or 11 in order to enable IPv4 checksum verification. 0 IPv4 header checksum not verified, either because verification was disabled or a valid IPv4 header could not be located. 1 IPv4 header checksum was verified by the eTSEC, and bit EIP indicates result.
	5	CTU	TCP or UDP header checksum checked. RCTRL[PRSDEP] must be set to 11 in order to enable layer 4 checksum verification. 0 TCP or UDP header checksum not verified, either because verification was disabled or a valid TCP or UDP header could not be located. If a UDP header with zero checksum was located, this bit is cleared in accordance with RFC 768. 1 TCP or UDP header checksum was verified by the eTSEC, and ETU indicates result.
	6	EIP	IPv4 header checksum verification error. Not valid unless CIP = 1. 0 No checksum error in IPv4 header. 1 Error in header checksum only if IP = 1 and IP6 = 0.
0–1	7	ETU	TCP or UDP header checksum verification error. Not valid unless CTU = 1. 0 No checksum error in TCP or UDP header. 1 Error in header checksum only if PRO = 0x06 or PRO = 0x11.
	8–11	—	Reserved
	12–13	PER	Parse error. 00 No error in L2 to L4 parse 01 Reserved 10 Inconsistent or unsupported L3 header sequence 11 Reserved
	14–15	—	Reserved

Table 15-162. Rx Frame Control Block Descriptions (continued)

Bytes	Bits	Name	Description
2–3	0–1	—	Reserved
	2–7	RQ	Receive queue index. This index was selected by the eTSEC Rx Filer (from a matching Filer rule's RQCTRL[Q] field) when it accepted the associated frame. If filing is not enabled, RQ is zero. Note that the 3 least significant bits of RQ correspond with the RxBD ring index whenever RCTRL[FSQEN] = 0.
	8–15	PRO	If IP = 1, PRO is set as follows: • PRO=0xFF for a fragment header or a back to back route header • PRO=0x*nn* for an unrecognized header, where *nn* is the next protocol field • PRO=(TCP/UDP header), as defined in the IANA specification, if TCP or UDP header is found If IP = 0, PRO is undefined. Note that the eTSEC parser logic stops further parsing when encountering an IP datagram that has indicated that it has fragmented the upper layer protocol. This in general means that there is likely no layer 4 header following the IP header and extension headers. eTSEC leaves the RxFCB[PRO] and RQFPR[L4P] fields 0xFF in this case, which usually means that there was no IP header seen. In this case RxFCB[IP] and optionally RxFCB[IP6] is set. IP header checksumming operate sand perform as intended. Most of the time, the eTSEC updates the RxFCB[PRO] field and RQFPR[L4P] fileds with whatever value was found in the protocol field of the IP header. See Section 15.5.3.3.8, "Receive Queue Filer Table Property Register (RQFPR)," for a description of RQFPR.
4–5	0–15	—	Reserved
6–7	0–15	VLCTL	VLAN control word as per IEEE 802.1Q. The lower 12 bits comprise the VLAN identifier. Valid only if VLN = 1.

15.6.5 Quality of Service (QoS) Provision

This section describes the quality of service support features of this device.

15.6.5.1 Receive Queue Filer

The receive queue filer receives protocol header properties extracted from the incoming frame by the eTSEC frame parse engine. A property is defined to be a field extracted from a packet header, such as a TCP port number or VLAN identifier. As soon as the last identifiable header has been recognized, the filer commences searching the receive queue filer table, comparing properties in the table against properties extracted from the frame. This table is illustrated in Figure 15-151. Software populates the table with property values, stored to the RQPROP field, and indicates how to match and interpret the properties by setting flags in the RQCTRL field. The eTSEC memory map provides access to these fields by way of an address register (RQFAR) and two porthole registers (RQFCR and RQFPR).

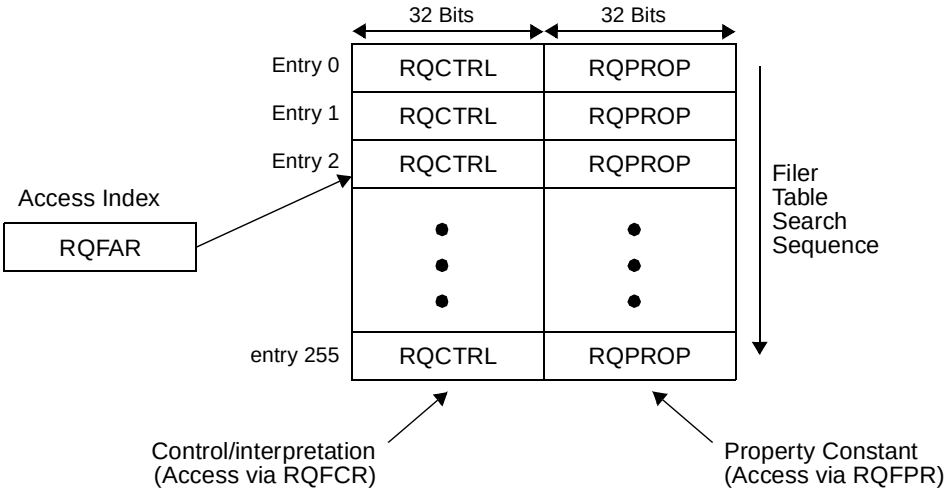


Figure 15-151. Structure of the Receive Queue Filer Table

15.6.5.1.1 Filing Rules

Unless the filer is disabled, every received frame from the Ethernet MAC or FIFO interface initiates a search of the receive queue filer table, starting at entry 0. The table search is terminated as soon as an entry is found whose contents match a property of the frame. Accordingly, software must guarantee that at least one entry results in a match—even if only to set a default receive queue index.

Since eTSEC searches the table at a rate of two entries every system clock cycle, all 256 entries can be searched in the time taken to receive a 64-byte Ethernet frame. However, for frames received through the 16-bit FIFO interface, fewer than 256 entries may be the maximum that can be searched while receiving a 40-byte packet. For example, with the 16-bit FIFO operating at 155 MHz, a 333-MHz eTSEC is limited to a maximum of 88 filing table entries.

Each entry of the receive queue filer table specifies a simple match rule for determining how to process the received frame. The elements of a filing rule, expressed in the RQCTRL and RQPROP fields, are summarized as follows:

- The PID field in RQCTRL identifies what property is being matched against RQPROP. The eTSEC supports 16 properties, some of which are different portions of the same header field. Reserved or unused bits in RQPROP are read as zero. See Section 15.5.3.3.8, “Receive Queue Filer Table Property Register (RQFPR),” on page 15-65 for a list of all properties and their associated PID values.
- The Q field in RQCTRL identifies which one of 64 virtual receive queues the frame should be filed to (sent via DMA) in the event of a filing rule match that accepts the frame. The physical RxBD ring this queue maps to is controlled by the RCTRL[FSQEN] bit. If RCTRL[FSQEN] = 0, the three least significant bits of the Q field indicate which physical RxBD ring hosts the queue. If RCTRL[FSQEN] = 1, RxBD ring 0 hosts all receive queues, but the RxFCB[RQ] field allows software to distinguish queues by ID. In all cases if Q maps to a RxBD ring that is not currently enabled, the frame is discarded with an IEVENT[FIQ] error.

- The REJ field in RQCTRL controls whether the frame is to be rejected (REJ = 1) or filed (REJ = 0) upon a filing rule match. Rejected frames occupy Rx FIFO space, but do not consume memory bus cycles.
- The CMP field in RQCTRL determines how property PID is compared against RQPROP. Equality, inequality, greater-or-equal, and less-than compares are available.
- The AND field in RQCTRL allows more than one comparison in a sequence to be chained together as a Boolean AND condition. Setting AND = 1 defers evaluation of the rule until the next entry has been matched, which may, in turn, have AND set. If any comparison involving AND = 1 fails, the entire chained sequence fails. A typical use for AND is to combine a pair of comparisons in a range match; the first such entry has AND = 1, the second has AND = 0 and its values of Q and REJ take effect.
- The CLE field in RQCTRL offers a way to bracket a set of consecutive—perhaps related—rules into a rule cluster. A cluster must be preceded by a guard rule, which simply determines whether the cluster rules can be evaluated. If the guard rule succeeds and its last entry has both CLE = 1 and AND = 1, the cluster rules that follow are enabled. The cluster ends at the first entry where CLE = 1 and AND = 0, which may also belong to a rule that files or rejects a frame. If the guard rule fails, all rules in the cluster are skipped, including mask_register assignments. Clusters must not be nested.
- The GPI field offers the user the ability to interrupt the core upon matching a rule that causes a frame to be filed to memory. Once the last RxBD corresponding to that frame is written to memory, the IEVENT[FGPI] event is asserted. This bit is set regardless of any interrupt coalescing that may be set.

15.6.5.1.2 Comparing Properties with Bit Masks

By default, extracted properties are compared arithmetically according to the CMP field in each RQCTRL word. This permits point value matches in each table entry, and range checks across a pair of table entries combined with the AND attribute in RQCTRL. However, inspection of the parse flags, Ethernet preamble, and IP addresses typically requires ‘don’t care’ bit fields in the properties to be cleared as part of the comparison. The eTSEC provides a dedicated 32-bit register, known as the mask_register, for performing such masking operations. At the start of each table search by the filer, mask_register is reset to 0xFFFF_FFFF, which ensures that no masking occurs.

Filer rules may be configured to assign specific bit patterns to mask_register. Such rules can be configured to either match always (useful for implementing a default rule and specifying an associated receive queue), or fail always (which prevents termination of the filer table search). Once mask_register has been assigned, it retains its value until it is reassigned or the table search terminates. All properties are non-destructively bit-wise ANDed with mask_register prior to comparison in subsequent rules, which allows an entire cluster of rules to make use of a common mask. Individual masks for specific rules can also be created simply by combining a mask_register assignment (match always form) with a regular rule using the AND attribute.

To create a mask_register assignment rule, it is necessary to select PID = 0 in RQCTRL, and choose CMP such that the rule either matches (CMP = 01) or fails (CMP = 11). In this entry, RQPROP is then considered to be the assigned bit vector.

15.6.5.1.3 Special-Case Rules

It is frequently useful to create rules that are guaranteed to succeed or fail, specifically to enforce a default filing decision or act as null entries. Suggested constructions for such rules are shown in Table 15-163.

Table 15-163. Special Filer Rules

Rule Description	RQCTRL Fields						RQPROP Word	RQCTRL Word[1]
	CLE	REJ	AND	Q	CMP	PID		
Default file—Always file frame to ring Q	0	0	0	Q	01	0000	0x0000_0000	0x0000_*qq*20
Default reject—Always discard frame	0	1	0	000_000	01	0000	0x0000_0000	0x0000_0120
Empty rule in AND—Always matches	0/1[2]	0	1	000_000	01	0000	0xFFFF_FFFF	0x0000_00A0
Empty rule in rule set—Always fails	0/1[3]	0	0	000_000	11	0000	0xFFFF_FFFF	0x0000_0060

1 Hexadecimal digits *qq* denotes field Q shifted left 2 bits.

2 Set CLE = 1 if the empty rule guards a cluster.

3 Set CLE = 1 if the empty rule occurs at the end of a cluster.

15.6.5.1.4 Filer Interrupt Events

The filer can produce three interrupt events in IEVENT. Event FIR indicates an error condition where the filer was unable to provide a definite result, either because no rule in the table succeeded, or because frames arrived too rapidly to complete searching of the table. Event FIQ indicates that the filer accepted a frame to a RxBD ring that was not enabled in RQCTRL (this can also occur if the filer is disabled, but RxBD ring 0—default queue or FSQEN mode queue—is not enabled). FIQ is also asserted in the case where no rule in the entire table succeeded. The various combinations of these interrupt events and their interpretation appear in Table 15-164.

Table 15-164. Receive Queue Filer Interrupt Events

IEVENT[FIR]	IEVENT[FIQ]	Description
0	0	No error. The filer successfully rejected or filed a frame.
0	1	Illegal queue error. The filer accepted a frame to a RxBD ring that is disabled (including ring 0 if filing is disabled).
1	0	Partial search error. The filer did not have sufficient time to complete its search of the filer table.
1	1	No matching rule error. The filer searched all 256 entries of the filer table without finding a rule that succeeds.

A functional interrupt is provided via use of the general purpose interrupt (GPI) bit in the filer table. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the filer sets IEVENT[FGPI] when the corresponding receive frame is written to memory. This allows the user to set up a filer rule where the core is interrupted upon the reception of ‘special’ frames.

If the timer is enabled (TMR_CTRL[TE] = 1), then the interrupt dedicated for timer events (in addition to the usual receive, transmit and error interrupts) is asserted.

15.6.5.1.5 Setting Up the Receive Queue Filer Table

The eTSEC frame parser always provides values for all properties, even where the relevant headers are not available. In the latter case, the filer is given default properties that can be used to avoid conflict with normal, defined property values. Accordingly, the rules in the filer table can be partitioned into rule sets such that if all rules in a given set fail (due to headers being unavailable), lower priority rule sets can be subsequently searched until either a rule set provides a match or a single default—catch-all—rule specifies a definite receive queue. For example, an 802.1p priority rule set may be followed by an IP TOS rule set, followed by a default rule; thus, if no VLAN tag appears in the received frame, the TOS rules are checked, or the default is activated should no IP header be present.

The rule cluster feature is used to conditionalize evaluation of rule sets. Typically, this avoids evaluating rules based on properties that may not be valid or relevant to the filing or filtering decision. For example, TCP-related rules might be clustered behind a guard rule that checks that a TCP header has appeared and the IP address matches our home address. Property 1—the parse flags property—is provided specifically to check the characteristics of the received frame and the parser error status. The mask_register is typically assigned beforehand to extract specific flags, in which case care should be taken that mask_register be reassigned an appropriate mask vector for following comparisons.

In many cases it is possible to write the entire filer table before using eTSEC, as the rule set is static. However, dynamic rule updates can be supported by pre-allocating partially instantiated rule sets, which software rewrites as necessary. Rules that are not instantiated should be composed of empty entries, as indicated in Table 15-163. In many cases empty entries can be overwritten by software without stopping eTSEC's receive function.

15.6.5.1.6 Filer Example—802.1p Priority Filing

This example, shown in Table 15-165, illustrates how to file frames according to layer 2 802.1p priority. This matches against property 1001, comparing each specific priority level in order to associate them with a RxBD ring index. Note that if a VLAN tag does not appear in the frame, the parser passes priority 0 to the filer, which always matches the rule at entry 7 and terminate the table search.

Table 15-165. Filer Table Example—802.1p Priority Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	000_000	00	1001	0x0000_0007	File priority 7 to ring 0	0x0000_0009
1	0	0	0	000_001	00	1001	0x0000_0006	File priority 6 to ring 1	0x0000_0409
2	0	0	0	000_010	00	1001	0x0000_0005	File priority 5 to ring 2	0x0000_0809
3	0	0	0	000_011	00	1001	0x0000_0004	File priority 4 to ring 3	0x0000_0C09
4	0	0	0	000_100	00	1001	0x0000_0003	File priority 3 to ring 4	0x0000_1009
5	0	0	0	000_101	00	1001	0x0000_0002	File priority 2 to ring 5	0x0000_1409

Table 15-165. Filer Table Example—802.1p Priority Filing (continued)

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
6	0	0	0	000_110	00	1001	0x0000_0001	File priority 1 to ring 6	0x0000_1809
7	0	0	0	000_111	00	1001	0x0000_0000	File undefined 802.1p or priority 0 to ring 7—Default always matches	0x0000_1C09

15.6.5.1.7 Filer Example—IP Diff-Serv Code Points Filing

This example demonstrates use of rule priority for determining class selector codepoints (RFC 2474) from the IP TOS property. An example filer table is shown in Table 15-166. The example relies on the fact that the first rule matched terminates the search, hence successively lower Diff-Serv codepoint ranges can be compared in each step until the default (zero or greater) range is reached. By default, property 1010 (IP TOS) takes the value 0x00 if no IP headers were recognized, therefore the table search always terminates.

Table 15-166. Filer Table Example—IP Diff-Serv Code Points Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	0	0	0	001_000	01	1010	0x0000_00E0	File class 7 to queue 8 (TOS >= 0xE0)	0x0000_202A
1	0	0	0	001_001	01	1010	0x0000_00C0	File class 6 to queue 9 (TOS >= 0xC0)	0x0000_242A
2	0	0	0	001_010	01	1010	0x0000_00A0	File class 5 to queue 10 (TOS >= 0xA0)	0x0000_282A
3	0	0	0	001_011	01	1010	0x0000_0080	File class 4 to queue 11 (TOS >= 0x80)	0x0000_2C2A
4	0	0	0	000_100	01	1010	0x0000_0060	File class 3 to queue 4 (TOS >= 0x60)	0x0000_102A
5	0	0	0	001_100	01	1010	0x0000_0040	File class 2 to queue 12 (TOS >= 0x40)	0x0000_302A
6	0	0	0	010_100	01	1010	0x0000_0020	File class 1 to queue 20 (TOS >= 0x20)	0x0000_502A
7	0	0	0	011_100	01	1010	0x0000_0000	File class 0 to queue 28 (TOS >= 0x00) or file to ring 4 by default	0x0000_702A

15.6.5.1.8 Filer Example—TCP and UDP Port Filing

This example demonstrates rule clusters and AND-combined entries for filing packets based on transport protocol and well-known port numbers in a termination application. An example filer table is shown in Table 15-167. The example contains two clusters; the first is entered only for TCP packets, the second is entered only for UDP packets. A default filing rule catches the case where neither TCP nor UDP headers are found. Each cluster compares source port number (property 1111) against a list of server ports, and files the packets accordingly. Note that entries 1 and 2 form an AND rule for checking that the port number >= 20 and port number < 22. Entries 4 and 5 are initially set up to always fail (zero port number), and thus comprise empty entries that can be used at a later time.

Table 15-167. Filer Table Example—TCP and UDP Port Filing

Table Entry	RQCTRL Fields						RQPROP	Comment	RQCTRL Word
	CLE	REJ	AND	Q	CMP	PID			
0	1	0	1	000_000	00	1011	0x0000_0006	Enter cluster if layer 4 is TCP	0x0000_028B
1	0	0	1	000_000	01	1111	0x0000_0014	AND rule—FTP from TCP ports 20 and 21: file to ring 2	0x0000_00AF
2	0	0	0	000_010	11	1111	0x0000_0016		0x0000_086F
3	0	0	0	000_011	00	1111	0x0000_0017	telnet from TCP port 23: file to ring 3	0x0000_0C0F
4	0	0	0	000_000	00	1111	0x0000_0000	*empty entry reserved for future use*	0x0000_000F
5	0	0	0	000_000	00	1111	0x0000_0000	*empty entry reserved for future use*	0x0000_000F
6	1	0	0	000_001	01	0000	0x0000_0000	end cluster; default TCP: file to ring 1	0x0000_0620
7	1	0	1	000_000	00	1011	0x0000_0011	Enter cluster if layer 4 is UDP	0x0000_028B
8	0	0	0	000_101	00	1111	0x0000_0801	NFS from UDP port 2049	0x0000_140F
9	0	0	0	000_111	00	1111	0x0000_0208	Route from UDP port 520	0x0000_000F
10	0	0	0	000_110	00	1111	0x0000_0045	TFTP from UDP port 69	0x0000_180F
11	1	0	0	000_100	01	0000	0x0000_0000	End cluster; default UDP: file to ring 4	0x0000_1220
12	0	0	0	000_000	01	0000	0x0000_0000	By default, file to ring 0	0x0000_0020

15.6.5.2 Transmission Scheduling

Each eTSEC can maintain multiple TxBD rings (or transmission queues) to satisfy QoS requirements. The ability to choose from a number of transmission streams dynamically is especially important during periods of network congestion. Certain application such as voice and video streaming are delay sensitive, but loss insensitive. For instance, VoIP applications require little bandwidth, but are highly sensitive to latency. Conversely, FTP or SMTP protocols are delay insensitive, but loss sensitive.

eTSEC has a transmission scheduler that implements a programmable QoS regime. The scheduler is responsible for choosing which of the prefetched TxBDs shall be processed next, and accordingly issuing DMA requests to service the data stream described by the chosen BD(s). The scheduler cycle is one of:

1. decide on a TxBD queue,
2. transmit exactly one frame from that queue, and
3. return to deciding on another queue, in step 1.

If TCTRL[TXSCHED] is set to 00, no transmission scheduling occurs, and only TxBD ring 0 is polled for new data to transmit, with DMACTRL controlling waiting or polling. TCTRL[TXSCHED], if not zero, can be programmed to invoke one of two scheduling algorithms, namely priority-based queuing (PBQ), and modified weighted round-robin queuing (MWRR). In all cases where TCTRL[TXSCHED] is not zero, the scheduler can choose from among 1 to 8 TxBD rings per eTSEC, with individual rings being enabled by the setting of TQUEUE[EN0–EN7] bits. For example, TxBD rings 3, 4, and 7 may be enabled for scheduling by setting EN3, EN4, and EN7, and clearing all other EN bits.

15.6.5.2.1 Priority-Based Queuing (PBQ)

PBQ is the simplest scheduler decision policy. The enabled TxBD rings are assigned a priority value based on their index. Rings with a lower index have precedence over rings with higher indices. For example, TxBD ring 0 has higher priority than TxBD ring 1, and TxBD ring 1 has higher priority than TxBD ring 2, and so on.

The scheduling decision is then achieved as follows:

```
loop
      priority_ring = null;
      ring = 0;
      while priority_ring == null and ring <= 7 loop
       if enabled(ring) and not ring_empty(ring) then
              priority_ring = ring;
       endif
       ring = ring + 1;
      endloop
      if priority_ring >= 0 then
       while not ring_empty(priority_ring) loop
              transmit_frame(priority_ring);
       endloop
      endif
endloop
```

In practice a protocol stack or device driver can abuse PBQ by attempting to queue too much traffic onto high priority rings. It is recommended that the highest priority ring should normally not be used at all except for frames requiring the utmost urgent transmission. This allows emergency traffic to overtake backlogged queues out of sequence.

15.6.5.2.2 Modified Weighted Round-Robin Queuing (MWRR)

eTSEC implements a modified weighted round-robin scheduling algorithm across all enabled TxBD rings when TCTRL[TXSCHED] = 10. In MWRR, the weights in the TR03WT and TR47WT registers determine the ideal size of each transmit slot, as measured in multiples of 64 bytes. Thus, to set a transmit slot of 512 bytes, a weight of 512/64 or 8 needs to be set for the ring. In this mode TxBD rings 1–7 are selected in round-robin fashion, whereas TxBD ring 0, if enabled with ready data for transmission, is always selected in between other rings so as to expedite transmission from ring 0.

The scheduling decision is then achieved as follows:

```
for ring = 1..7 and enabled(ring) loop
      credit[ring] = 0;
endloop
for ring = 1..7 and enabled(ring) loop
      if not ring_empty(0) then
       credit[0] = credit[0] + weight[0];
       while credit[0] > 0 loop
              transmit_frame(0);
              credit[0] = credit[0] - frame_size;
              if ring_empty(0) then
               credit[0] = 0;
              endif
       endloop
      endif
```

```
        if not ring_empty(ring) then
         credit[ring] = credit[ring] + weight[ring];
        endif
        while credit[ring] > 0 loop
         transmit_frame(ring);
         credit[ring] = credit[ring] - frame_size;
         if ring_empty(ring) then
                credit[ring] = 0;
         endif
        endloop
endloop
```

The algorithm checks registers TQUEUE[EN0–EN7] for `enabled()`, TSTAT[THLT0–THLT7] for `ring_empty()`, and TR*x*WT for `weight()`. For TxBD ring *k*, having a weight WT*k*, the long term average throughput for that ring is:

rate of queue[k] (K = 1 to 7) = (available bandwidth) * WT*k*/(sum(WT*i*) + 6WT0)

rate of queue(0) = (available bandwidth) * 7 * WT0/(sum(WT*i*) + 6WT0)

where *i* = 0 to 7

15.6.6 Lossless Flow Control

The eTSEC DMA subsystem is designed to be able to support simultaneous receive and transmit traffic at gigabit line rates. If the host memory has sufficient bandwidth to support such line rates, then the principle cause of overflow on receive traffic is due to a lack of Rx BDs. Thus, the long term receive throughput is determined by the rate at which software can process receive traffic. If a user desires to prevent dropped packets, they can inform the far-end link to stop transmission while the software processing catches up with the backlog.

To avoid overflow in the latter case, back pressure must be applied to the far-end transmitter before the Rx descriptor controller encounters a non-empty BD and halts with a BSY error. As there is lag between application of back-pressure and response of the far-end, the pause request must be issued while there are still BDs free in the ring. In the traditional eTSEC descriptor ring programming model, there is no way for hardware to know how many free BDs are available, so software must initiate any pause requests required during operation. If software is backlogged, the request may be not be issued in time to prevent BSY errors. To allow the eTSEC to generate the pause request automatically, additional information (a pointer the last free BD and ring length) is required.

15.6.6.1 Back Pressure Determination via Free Buffers

Ultimately, the rate of data reception is determined by how quickly software can release buffers back into the receive ring(s). Each time a buffer is freed, the associated BD has its empty bit set and hardware is free to consume both. Thus the number of free BDs in a given Rx ring indicates how close hardware is to the end of that ring. To prevent data loss, back pressure should be applied when the number of free BDs drops below some critical level. The number of BDs that can be consumed by an incoming packet stream while back-pressure takes effect is determined by several factors, such as: receive traffic profile, transmit traffic profile, Rx buffer size, physical transmission time between eTSEC and far-end device and intra-device latency. Theoretically, the worst case is:

$$\text{FreeBDsRequired} = \frac{\text{MaxFrameSize}}{\text{MinFrameSize} + \text{IFG}} + \frac{\text{MaxFrameSize}}{\text{RxBufferSize}} + \text{LinkDelay}$$

This case comes about when:

- The eTSEC has just started transmitting a large frame and thus cannot send out a pause frame
- Upon reception of the pause request the far-end has just started transmission of a large frame
- The eTSEC receives a burst of short frames with minimum inter-frame-gap (96bit times for ethernet)

Once the user has determined the worst case scenario for their application, they program the required free BD threshold into the eTSEC (via RQPRM[PBTHR]). Since different BD rings may have different sizes and expected packet arrival rates, a separate threshold is provided for each active ring. It is recommended that a threshold of at least fourBDs is the practical minimum for gigabit ethernet links.

For the Rx descriptor controller to determine the number of free BDs remaining in the ring, it needs to know the following:

1. The location of the current BD being used by hardware
2. The location of the last BD that was released (freed) by software
3. The length of the Rx BD ring.

For each active ring, the current BD pointer (RBPTR*n*) is maintained by the eTSEC. Software knows both the size of the Rx ring and the location of the last freed BD. By providing the eTSEC with those values (via RQPRM[LEN] and RFBPTR respectively) the eTSEC always knows how many receive buffers are available to be consumed by incoming data.

The number of guaranteed free BDs in the ring is then determined by:

When RFBPTRn < RBPTR*n*

$$\text{FreeBDs} = \text{RQPRM}n[\text{LEN}] - \text{RBPTR}n + \text{RFBPTR}n$$

When RFBPTR*n* > RBPTR*n*

$$\text{FreeBDs} = \text{RFBPTR}n - \text{RBPTR}n$$

When RBPTR*n* = RFBPTR*n* the number of free BDs in the ring is either one (since RFBPTR*n* points to a free BD) or equal to the ring length. Since the BD pointed to by RBPTR*n* may be either in use or about to be used, it is not considered in the free BD count. To resolve the case where the two pointers collide, the following logic applies:

If RBASE*n* was updated and thus initializes both RBPTR*n* and RFBPTR*n*, the ring is deemed empty.

If RFBPTR*n* is updated by a software write and matches RBPTR*n*, the ring is deemed empty.

If HW updates RBPTR*n* and the result matches RFBPTR*n*, the ring is deemed to have one BD remaining. Upon writing this BD back to memory (indicating the buffer is occupied) the ring is deemed to be full.

Important. There is a possibility that if software is severely backlogged in updating RFBPTR*n*, the hardware could wrap around the ring entirely, consume exactly the remaining number of BDs and not halt with a BSY error. If software then increments RFBPTR*n* to the next address (thereby equalling RBPTR*n*), the hardware assumes the ring is now empty (when in fact there is only a single BD freed up). This results in the hardware failing to maintain back pressure on the far end. Upon software incrementing RFBPTR*n* a subsequent time, the wrap condition is successfully detected and hardware recognizes a nearly full ring (rather than a nearly empty one). Since software can increment RFBPTR*n* by any amount, it is not possible for hardware to determine in this case whether the user has cleared the entire ring or just one BD. Users can eliminate the possibility of this condition occurring by ensuring that RFBPTR*n* is incremented by at least two BDs each time (for example, clear at least two buffers whenever the RxBD unload routine is called).

Once the eTSEC determines that this threshold has been reached, back pressure is applied accordingly. The type of back pressure that is applied varies according to the physical interface that is used.

- **Half duplex Ethernet:** No support in this mode
- **Full duplex Ethernet:** An IEEE 802.3 PAUSE frame (see Section 15.6.3.9, “Flow Control”) is issued as if the TCTRL[TFC_PAUSE] bit was set. An internal counter tracks the time the far end controller is expected to remain in pause (based on the setting of PTV[PT]). When that counter reaches half the value of PTV[PT], the eTSEC reissues a pause frame if the free BD calculation for any ring is below the threshold for that ring. For example, if PTV[PT] is set to 10 quanta, a pause frame is re-issued when five quanta have elapsed if the free BD threshold is still not met. A practical minimum for PTV[PT] of 4 quanta is recommended.
- **FIFO packet interface:** Link layer flow control is asserted via use of the RFC signal (CRS pin). Flow control is asserted for the entire time that free BD threshold is not met. The same mechanism is used for both GMII-style and encoded packet modes.

15.6.6.2 Software Use of Hardware-Initiated Back Pressure

15.6.6.2.1 Initialization

Software configures RBASE*n* and RQPRM*n*[LEN] according to the parameters for that ring. Then the number of free BDs that are required to prevent the eTSEC from automatically asserting flow control are programmed in RQPRM[FBTHR]. The receiver is then enabled.

NOTE

The act of programming RBASE*n* initializes RFBPTR*n* to the start of the of the ring. When the ring is in this initial empty state, there is no concept of a last freed BD. In this case, the calculated number of free BDs is the size of the ring. Since the BD that the hardware is currently pointing to is to be considered in-use, the free BD count is actually one higher than the total available. As soon as the hardware consumes a BD (by writing it back to memory), RBPTR*n* advances and the free BD count reflects the correct number of available free BDs.

15.6.6.2.2 Operation

As software frees BDs from the ring, it writes the physical address of the BD just freed to RFBPTR*n*. The eTSEC asserts flow control if the distance (using modulo arithmetic) between RBPTR*n* and RFBPTR*n* is < RQPRM*n*[FBTHR]. In multi-ring operation, if the free BD count of **any** active ring drops below the threshold for that ring, flow control is asserted. Once enough BDs are freed for **all** active rings to meet their respective free BD thresholds, application of back pressure ceases.

Note: The eTSEC does not issue an exit pause frame (such as, pause frame with PTV of 0x0000) once all active rings have sufficient BDs. Instead, it waits for the far-end pause timer to expire and start re-transmission.

15.6.7 Hardware Assist for IEEE 1588 Compatible Timestamping

There is a push in industrial control applications to use Ethernet as the principal link layer for communications. This requires Ethernet to be used for both data transfer and real-time control. For real-time systems, each node is required to be synchronized to a master clock. The precision of this clock is dictated by the application, but generally needs to be of the order of <1uSec for high-speed machinery (for example, printing presses).

IEEE 1588 specifies a mechanism for synchronizing multiple nodes to a master clock. Support for IEEE 1588 can be done entirely in software running on a host CPU, but applications that require sub 10uSec accuracy need hardware support for accurate timestamping of incoming packets.

The eTSEC includes a new timer clock module to support the IEEE 1588 timer standard. The following sections describe the features, programming model, and implementation information.

15.6.7.1 Features

- 64-bit free running timer running from an external oscillator or internal clock
- Programmable timer oscillator clock selection
- Self-correcting precision timer with nano-second resolution
- Time stamp all incoming packets inline
  - Maskable interrupts on received PTP packet’s filer rule match
- Time stamp transmit packets when instructed in the TxFCB
  - Maskable interrupts on transmit timestamp capture
- Two Tx time stamp registers per eTSEC with 16-bit tag for each of them to support burst mode.
- Time stamp capture on two general-purpose external triggers
  - Maskable interrupts on GPIO timestamp trigger
  - Programmable polarity of external trigger (GPIO) edge
- Two 64-bit alarm (future time) registers for future time comparison
  - Maskable interrupts on alarm

- Three programmable timer output pulse period phase aligned with 1588 timer clock
  - Maskable interrupts associated with each pulse
- Separate maskable timer interrupt event register
- Recognition of incoming PTP packet through filer rule match
- Phase aligned adjustable (divide by N) clock output
- Supports all Ethernet modes supported by the eTSEC, including full- and half-duplex modes
- Supports both master and slave modes
- Supports timestamp of nano-second resolution

15.6.7.2 Timer Logic Overview

The 1588 timer module can be partitioned into four different sub-modules as shown in Figure 15-152.

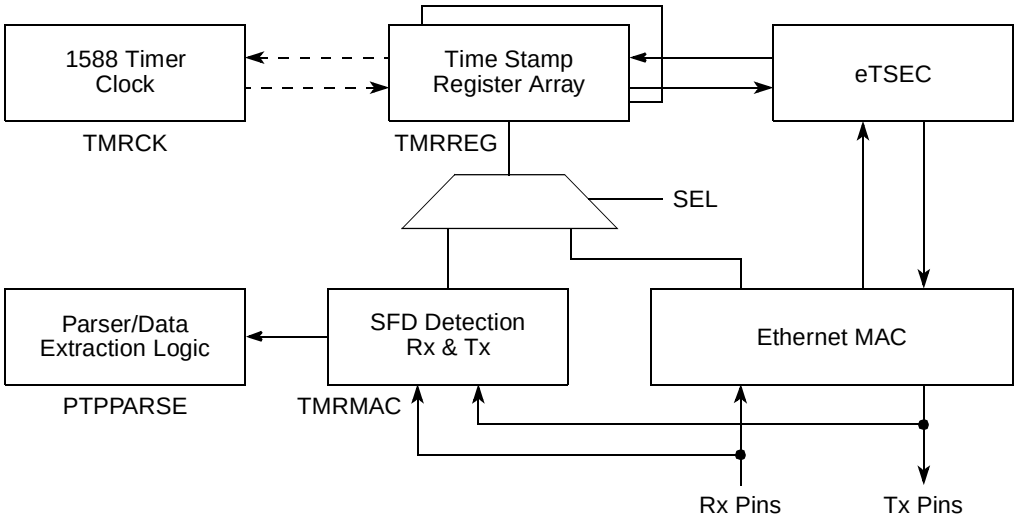


Figure 15-152. 1588 Timer Design Partition

15.6.7.3 Time-Stamp Insertion on the Received Packets

Every incoming packet's 8-byte time stamp is inserted into the packet data buffer as padding alignment bytes. Time-stamp insertion into the data buffer requires RCTRL[PAL] to be set to a value greater than or equal to 8 and the control bit RCTRL[TS] bit to be set.

15.6.7.3.1 Timestamp Point

The required timestamp point as specified in 1588 is shown in Figure 15-153. From this, it is clear that the end of the SFD is the critical point in the MII data stream.

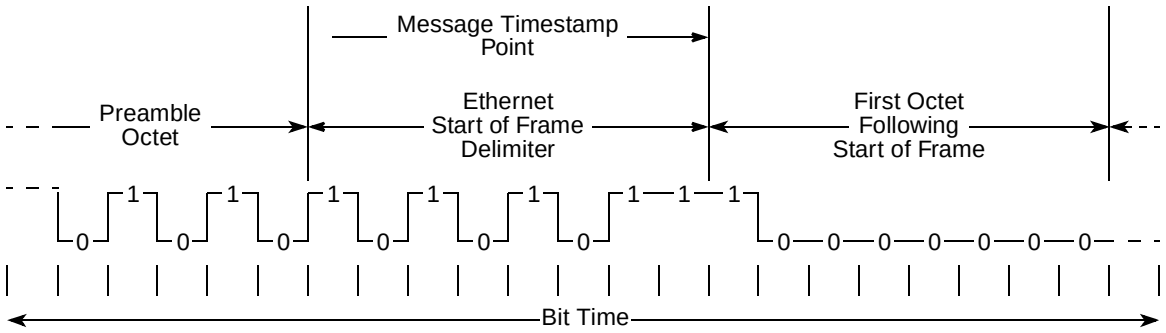


Figure 15-153. Ethernet Sampling Points for 1588

The sample point coincides with the cycle after the SFD detection by the MAC. For received frames, this is at least 4 bit times (MII) or 8 bit times (GMII) after the message timestamp point specified in [1588]. For transmission, the eTSEC sample point precedes the sample point specified in [1588] by at least 4-bit times (MII) or 8-bit times (GMII). For a particular mode, the eTSEC sample point is a consistent number of bit times relative to the SFD detection. Thus, the offset from the [1558] specified sample point can be accounted for in the PTP software implementation.

The MAC and the timer is running from the different asynchronous clock sources. Sampling of the time stamp point is synchronized in the eTSEC system clock domain. Minimum specified eTSEC system clock speed is 133 MHz.

15.6.7.4 PTP Packet Parsing

PTP packets are typically embedded within a UDP payload with special IP source and destination address and special source and destination ports numbers. Special fields of interest of a PTP packet are listed in Table 15-168.

Table 15-168. PTP Payload Special Fields

Layer	Octet (Offset from the SFD)	Field	Value	eTSEC filer PID	Comments
Ethernet	12-13	Length/Packet	0x0800	ETY-RQPFR[PID=0111]	IPv4
IP header	22	Time to live	0x00	RBIFX-choose an arbitrary extraction byte	Must be 0
IP header	23	IP Protocol	0x11	L4P-RQPFR[PID=1011]	UDP
IP header	26-29	Source IP Address IANA defines 4 multicast address for the PTP packet	—	SIA-RQPFR[PID=1101]	—

Table 15-168. PTP Payload Special Fields (continued)

Layer	Octet (Offset from the SFD)	Field	Value	eTSEC filer PID	Comments
IP header	30-33	Destination IP Address IANA defines 4 multicast address for the PTP packet	224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132	DIA-RQPFR[PID=1100]	DefaultPTPdomain AlternatePTPdomain1 AlternatePTPdomain2 AlternatePTPdomain3
UDP header	34-35	Source port number		SPT-RQPFR[PID=1011]	—
UDP header	36-37	Destination port number	319 320	DPT-RQPFR[PID=1011]	EventPort GeneralPort
UDP data	74	Control	0x0 0x1 0x2 0x3 0x4	RBIFX-choose an arbitrary extraction byte	Sync Delay_req Follow_up Delay_resp Management

A representation of the PTP packet is shown in Figure 15-154.

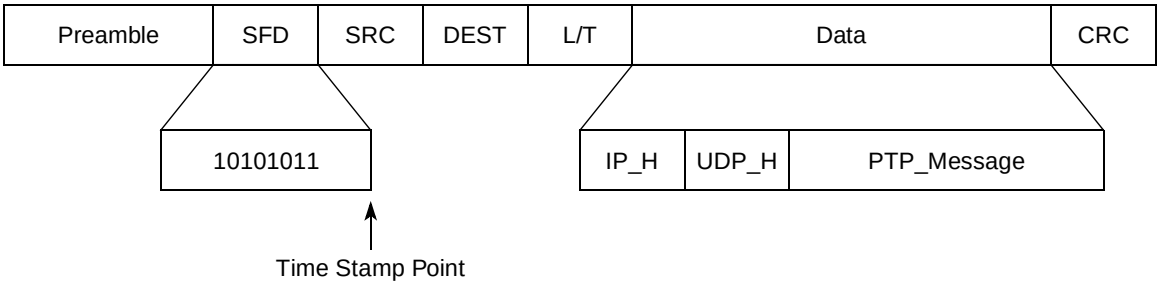


Figure 15-154. PTP Packet Format

15.6.7.4.1 General Purpose Filer Rule

The eTSEC receive filer has been enhanced with the addition of a general-purpose event bit. This event bit can be used in conjunction with filing table rules to identify 1588 packets and indicate these packets by setting special timer status register bits (TMR_STAT). Additionally, 1588 packets can be easily identified by upper-layer software by using the filer to queue all PTP packets to one or more predefined virtual queues. See Section 15.6.5.1.1, “Filing Rules for further information.

15.6.8 Buffer Descriptors

The eTSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse across the PowerQUICC network processor family. Drawing from the MPC8260 FEC BD programming model, the eTSEC descriptor base registers point to the beginning of BD rings. The eTSEC BD also expands upon the MPC8260 BD model to accommodate the eTSEC’s unique features. However, the 8-byte data BD format is designed to be compatible with the existing MPC8260 BD model.

The eTSEC is capable of duplicating—or extracting—data directly into the L2 cache memory. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory, which holds the master copy. This results in substantial improvement in throughput and hence reduction in latency.

15.6.8.1 Data Buffer Descriptors

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 15-155). Data BDs encapsulate all information necessary for the eTSEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of pre-fetching, a minimum of four buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings. Transmit rings are limited to a maximum size of 65536 BDs due to BD and frame data prefetching. Software also must have the data pointer pointing to a legal memory location. Within the status field, there exists an ownership bit which defines the current state of the buffer (pointed to by the data pointer). Other bits in the status field of the buffer descriptor are used to communicate status/control information between the eTSEC and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see Figure 15-156), all BDs must reside sequentially in memory. The eTSEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the eTSEC to loop back to the beginning of the BD chain. Software must initialize the TBASE and RBASE registers that point to the beginning transmit and receive BDs for eTSEC.

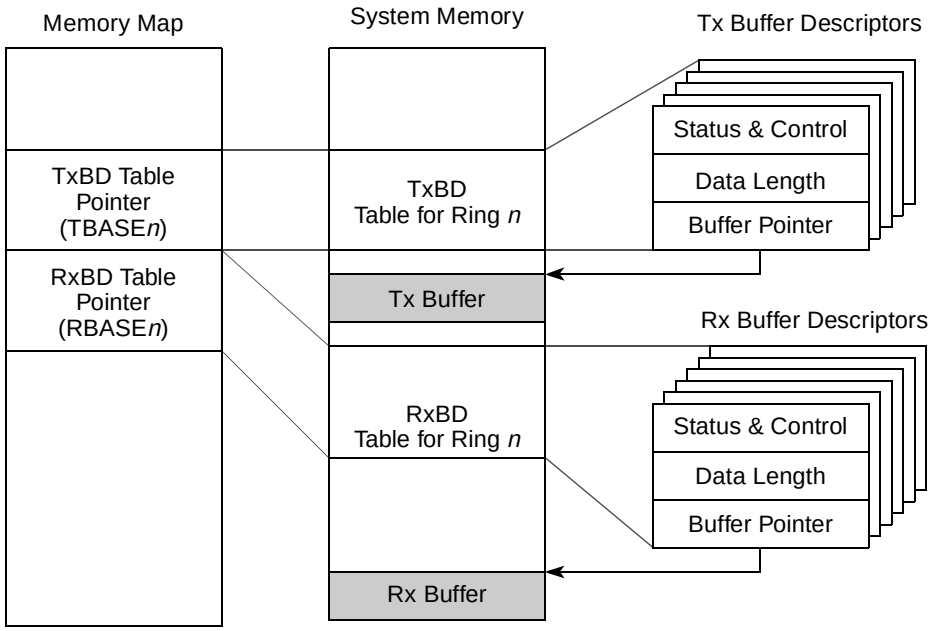


Figure 15-155. Example of eTSEC Memory Structure for BDs

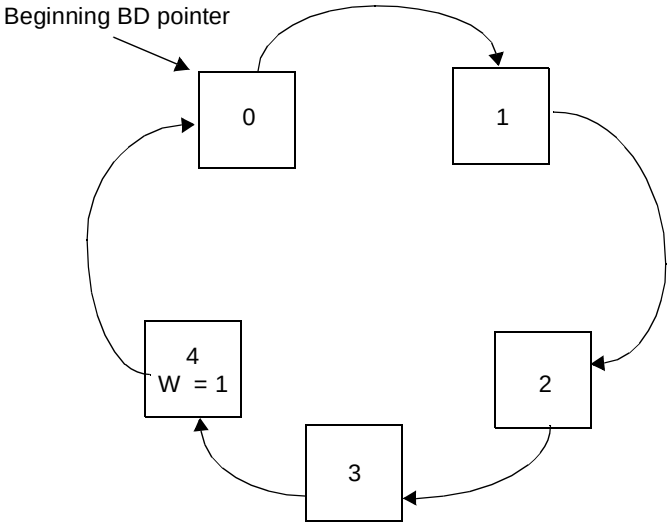


Figure 15-156. Buffer Descriptor Ring

15.6.8.2 Transmit Data Buffer Descriptors (TxBD)

Data is presented to the eTSEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, I, L, TC, PRE, HFE, CF, and TOE bits and the length (in bytes) in the first word, and the buffer pointer in the second word. Unused fields or fields written by the eTSEC must be initialized to zero. For transmission over the FIFO interface the Ethernet specific bits (PRE, DEF, HFE, LC, CF, RL and RC) have no meaning.

The eTSEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block.

Software must expect eTSEC to prefetch multiple TxBDs, and for TCP/IP checksumming an entire frame must be read from memory before a checksum can be computed. Accordingly, the R bit of the first TxBD in a frame must not be set until at least one entire frame can be fetched from this TxBD onwards. If eTSEC prefetches TxBDs and fails to reach a last TxBD (with bit L set), it halts further transmission from the current TxBD ring and report an underrun error as IEVENT[XFUN]; this indicates that an incomplete frame was fetched, but remained unprocessed. The relevant TBPTR register points to the next unread TxBD following the error.

Figure 15-157 defines the TxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	PAD/CRC	W	I	L	TC	PRE/DEF	0	HFE/LC	CF/RL	RC				TOE/UN	TR
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

Figure 15-157. Transmit Buffer Descriptor

The TxBD definition is interpreted by eTSEC hardware as if TxBDs mapped to C data structures in the manner illustrated by Figure 15-158.

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct txbd_struct {
       uint_16 flags;
       uint_16 length;
       uint_32 bufptr;
} txbd;
```

Figure 15-158. Mapping of TxBDs to a C Data Structure

The TxBD fields are detailed in Table 15-169.

Table 15-169. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
0–1	0	R	Ready, written by eTSEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The eTSEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
	1	PAD/CRC	Padding for frames. (Valid only while it is set in the first BD and MACCFG2[PAD enable] is cleared). If MACCFG2[PAD enable] is sector if the eTSEC is operating in FIFO mode (ECTRL[FIFM] is set), this bit is ignored. 0 Do not add padding to short frames. 1 Add PAD to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, the eTSEC PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
	4	L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.

Table 15-169. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	5	TC	Tx CRC. Written by user. (Valid only while it is set in first BD and TxBD[PAD/CRC] is cleared and MACCFG2[PAD/CRC enable] is cleared and MACCFG2[CRC enable] is cleared.) If MACCFG2[PAD/CRC enable] is set or MACCFG2[CRC enable] is set, this bit is ignored in ethernet modes. If FIFOCFG[CRCAPP] is set, this bit is ignored in FIFO modes 0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
	6	PRE	Transmit user-defined Ethernet preamble. Written by user. Valid only if set in the first BD of a frame, and MACCFG2[PreAm TxEN] is set. 0 This frame does not contain Ethernet preamble bytes for transmission. 1 This frame includes a user-defined Ethernet preamble sequence prior to the destination address in the data buffer.
		DEF	Defer indication. The eTSEC updates this bit after transmitting a frame (TxBD[L] is set) 0 This frame was not deferred. 1 This frame did not have a collision before it was sent but it was sent late because of deferring
	7	—	Reserved
	8	HFE	Huge frame enable. Written by user. Valid only if set in the first BD of a frame and MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's maximum frame length. 1 Allow large frames to be transmitted without truncation.
		LC	Late collision. Written by the eTSEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The eTSEC terminates the transmission and updates LC.
	9	CF	Control Frame. Written by user. Valid only if set in the first BD of a frame. 0 Regular frame; transmission is deferred when eTSEC is in PAUSE. 1 Control frame; transmission starts even if eTSEC is in PAUSE.
		RL	Retransmission Limit. Written by the eTSEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The eTSEC terminates the transmission and updates RL.
	10–13	RC	Retry Count. Written by the eTSEC. 0 The frame is sent correctly the first time. x One or more attempts where needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.

Table 15-169. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	14	UN	Underrun. Written by the eTSEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. This could also have occurred in relation to a bus error causing IEVENT[EBERR]. The eTSEC terminates the transmission and updates UN.
		TOE	TCP/IP off-load enable. Written by user. Valid only if set in the first BD of a frame. 0 No TCP/IP off-load acceleration is applied to the frame prior to transmission. 1 eTSEC looks for a TOE Frame Control Block preceding the frame, and applies TCP/IP off-load acceleration as controlled by the FCB.
	15	TR	Truncation. Written by the eTSEC. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for a frame (a frame length greater than or equal to the value set in the maximum frame length register is encountered, the HFE bit in the BD is cleared, and MACCFG2[Huge Frame] is cleared). The frame is sent truncated.
2–3	0–15	Data Length	Data length is the number of octets the eTSEC should transmit from this BD's data buffer. It is never modified by the eTSEC. This field must be greater than zero, as zero indicates a BD not ready.
4–7	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. The data buffer pointer for the first BD of a TxPAL-enabled frame must be aligned on an 8-byte boundary. There are no alignment restrictions for the data buffer pointers of the second or subsequent BDs of a TxPAL-enabled frame, or for non-TxPAL frames.

15.6.8.3 Receive Buffer Descriptors (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the eTSEC modifies the E, L, F, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first word of the buffer descriptor are only modified by the eTSEC if the L (last BD in frame) bit is set. The first word of the RxBD contains control and status bits. For packets received over the FIFO interface, those bits which are ethernet specific (M, BC, MC, LG, NO, and TR) should be ignored. Its formats are detailed below.

The number of buffer descriptors in a ring is set by using the W bit to indicate that the next buffer wraps back to the beginning of the ring. See Section 15.5.3.5.5, "Maximum Frame Length Register (MAXFRM)," for information on setting the size of the buffer ring.

Figure 15-159 defines the RxBD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	I	L	F	0	M	BC	MC	LG	NO	SH	CR	OV	TR
Offset + 2	DATA LENGTH															
Offset + 4	RX DATA BUFFER POINTER															
Offset + 6																

Figure 15-159. Receive Buffer Descriptor

The RxBD definition is interpreted by eTSEC hardware as if RxBDs mapped to C data structures in the manner illustrated by Figure 15-160.

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct rxbd_struct {
       uint_16 flags;
       uint_16 length;
       uint_32 bufptr;
} rxbd;
```

Figure 15-160. Mapping of RxBDs to a C Data Structure

Table 15-170 describes the fields of the RxBD.

Table 15-170. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
0–1	0	E	Empty, written by the eTSEC (when cleared) and by the user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
	1	RO1	Receive software ownership bit. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
	2	W	Wrap, written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
	3	I	Interrupt, written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] or IMASK[RXFEN]). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
	4	L	Last in frame, written by the eTSEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
	5	F	First in frame, written by the eTSEC. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	—	Reserved
	7	M	Miss, written by the eTSEC. (This bit is valid only if the L-bit is set and eTSEC is in promiscuous mode.) This bit is set by the eTSEC for frames that were accepted in promiscuous mode, but were flagged as a “miss” by the internal address recognition; thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.

Table 15-170. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
0–1	8	BC	Broadcast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
	9	MC	Multicast. Written by the eTSEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.
	10	LG	Rx frame length violation, written by the eTSEC (only valid if L is set). A frame length greater than or equal to the maximum frame length was recognized; in this case LG is set regardless of the setting of MACCFG2[Huge Frame]. If MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the maximum frame length register. This bit is valid only if the L bit is set. This bit is not set when in FIFO mode as truncation cannot occur.
	11	NO	Rx non-octet aligned frame, written by the eTSEC (only valid if L is set). A frame that contained a number of bits not divisible by eight was received. This bit cannot be set in FIFO mode.
	12	SH	Short frame, written by the eTSEC (only valid if L is set). A frame length less than the minimum 64B that is defined for ethernet. was recognized, provided RCTRL[RSF] is set. This bit should be disregarded in FIFO mode.
	13	CR	Rx CRC error, written by the eTSEC (only valid if L is set). This frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected.
	14	OV	Overrun, written by the eTSEC (only valid if L is set). A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR and TR lose their normal meaning and are zero.
	15	TR	Truncation, written by the eTSEC (only valid if L is set). Set if the receive frame is truncated. This can happen if a frame length greater than the maximum frame length is received and MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect. This bit is not set when in FIFO mode as truncation cannot occur.
2–3	0–15	Data Length	Data length, written by the eTSEC. Data length is the number of octets written by the eTSEC into this BD's data buffer if L is cleared (the value is equal to MRBLR), or, if L is set, the length of the frame including CRC, FCB (if RCTRL[PRSDEP > 00) and any padding (RCTRL[PAL]).
4–7	0–31	RX Data Buffer Pointer	Receive buffer pointer, written by the user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 8-byte aligned. For best performance, use 64-byte aligned receive buffer pointer addresses. The buffer must reside in memory external to the eTSEC.

15.7 Initialization/Application Information

15.7.1 Interface Mode Configuration

This section describes how to configure the eTSEC in different supported interface modes. These include:

- MII
- RMII
- GMII

- RGMII
- SGMII
- TBI
- RTBI
- 8-bit FIFO
- 16-bit FIFO

The pinout, the data registers that must be initialized, as well as speed selection options are described.

15.7.1.1 MII Interface Mode

Table 15-171 describes the signal configurations required for MII interface mode.

Table 15-171. MII Interface Mode Signal Configuration

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		

Table 15-171. MII Interface Mode Signal Configuration (continued)

eTSEC Signals			MII Interface		
			Frequency [MHz] 25		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
RxD[7]	I	1	not used		
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	COL	I	1
CRS	I	1	CRS	I	1
Sum		25	**Sum**		16

Table 15-172 describes the shared signals of the MII interface.

Table 15-172. Shared MII Signals

eTSEC Signals	I/O	No. of Signals	MII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
ECGTX_CLK125	I	1	not used	I	0	Reference clock
Sum		3	**Sum**		2	—

Table 15-173 describes the register initializations required to configure the eTSEC in MII mode.

Table 15-173. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half duplex operation. Set I/F Mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.

Table 15-173. MII Mode Register Initialization Steps (continued)

Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY Writing to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00.

Table 15-173. MII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done and Link is up) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.2 GMII Interface Mode

Table 15-174 describes the signal configurations required for GMII interface mode.

Table 15-174. GMII Interface Mode Signal Configuration

eTSEC Signal s			GMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	TxD[4]	O	1
TxD[5]	O	1	TxD[5]	O	1
TxD[6]	O	1	TxD[6]	O	1
TxD[7]	O	1	TxD[7]	O	1
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	RxD[4]	I	1
RxD[5]	I	1	RxD[5]	I	1
RxD[6]	I	1	RxD[6]	I	1
RxD[7]	I	1	RxD[7]	I	1
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		23

Table 15-175 describes the shared signals of the GMII interface.

Table 15-175. Shared GMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
ECGTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	—

Table 15-176 describes the register initializations required to configure the eTSEC in GMII mode.

Table 15-176. GMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for GMII, Full duplex operation. Set I/F Mode bit. MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (This example has Full Duplex = 1, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example.
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example.
Reset the management interface, MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0111]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY), MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]

Table 15-176. GMII Mode Register Initialization Steps (continued)

Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register #1 to set up the interface mode selection MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection, MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_000u_00u1_0100_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, Read the MIIMSTAT register and check bit 10 (AN Done and Link is up), MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]

Table 15-176. GMII Mode Register Initialization Steps (continued)

Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.3 TBI Interface Mode

Table 15-177 describes the signal configurations required for TBI interface mode.

Table 15-177. TBI Interface Mode Signal Configuration

eTSEC Signals			TBI Interface		
			Frequency [MHz] 62.5		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TCG[4]	O	1
TxD[5]	O	1	TCG[5]	O	1
TxD[6]	O	1	TCG[6]	O	1
TxD[7]	O	1	TCG[7]	O	1
TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RCG[4]	I	1
RxD[5]	I	1	RCG[5]	I	1
RxD[6]	I	1	RCG[6]	I	1
RxD[7]	I	1	RCG[7]	I	1
RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RCG[9]	I	1
COL	I	1	not used		
CRS	I	1	SDET	I	1
Sum		25	**Sum**		24

Table 15-178 describes the shared signals for the TBI interface.

Table 15-178. Shared TBI Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
ECGTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	—

Table 15-179 describes the register initializations required to configure the eTSEC in TBI mode.

Table 15-179. TBI Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.

Table 15-179. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle to verify state of TBI Control Register(0ptional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] the Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 15-179. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.4 RGMII Interface Mode

Table 15-180 shows the signals configurations required for RGMII interface mode.

Table 15-180. RGMII Interface Mode Signal Configuration

eTSEC Signals			RGMII Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		12

Table 15-181 describes the shared signals for the RGMII interface.

Table 15-181. Shared RGMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
GTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
Sum		3	**Sum**		3	—

Table 15-182 describes the register initializations required to configure the eTSEC in RGMII mode.

Table 15-182. RGMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has RGMII 10Mbps mode, Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that TSEC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)

Table 15-182. RGMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_1x10_0000]

Table 15-182. RGMII Mode Register Initialization Steps (continued)

Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.5 RMII Interface Mode

Table 15-183 shows the signals configurations required for RMII interface mode.

Table 15-183. RMII Interface Mode Signal Configuration

eTSEC Signals			RMII Interface		
			Frequency [MHz] 50		
			Voltage [V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	leave unconnected		
TX_CLK	I	1	REF_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	leave unconnected		
TxD[3]	O	1	leave unconnected		
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	leave unconnected		
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	not used		
RxD[3]	I	1	not used		
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	CRS_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**Sum**		8

Table 15-184 describes the shared signals for the RMII interface.

Table 15-184. Shared RMII Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
TX_CLK	I	1	REF_CLK	I	1	Reference clock
Sum		3	**Sum**		3	—

Table 15-185 describes the register initializations required to configure the eTSEC in RMII mode.

Table 15-185. RMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0001_0000] (Used to setup Reduced-Pin mode = 1, and TBIM = 0,statistics enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543 for example
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543 for example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_1101] set system clock divide by 14 for example to insure that MDC clock speed = 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.

Table 15-185. RMII Mode Register Initialization Steps (continued)

Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (1) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Setting up the MII Mgmt for a write cycle to TBI MII Mgmt register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_1011] the TBI control register is at offset address 0x11 from TBIPA
Perform an MII Mgmt write cycle Writing to MII Mgmt Control with 16-bit data intended for TBI's MII Mgmt control register (TBI control), MIIMCON[0000_0000_0000_0000_0000_0010_0001_0000] This configures the TBI control to GMII mode and AN sense
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicate that the write cycle was completed

Table 15-185. RMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_0001_0000]
Check to see if PHY has completed Auto-Negotiation Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010] the PHY Status control register is at address 0x2 and lets say the PHY Address is 0x2
Perform an MII Mgmt read cycle of Status Register Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] other information about the link is also returned (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register MIIMADD[0000_0000_0000_0000_0000_0010_0000_0110] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (6) placed in MIIMADD register), read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register (Optional) MIIMADD[0000_0000_0000_0000_0000_0010_0000_0101] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (5) placed in MIIMADD register), read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10 (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_00X_1110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.6 RTBI Interface Mode

Table 15-186 describes the signal configurations required for RTBI interface mode.

Table 15-186. RTBI Interface Mode Signal Configuration

eTSEC Signal s			RTBI Interface		
			Frequency [MHz] 125		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	not used		
TxD[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1	leave unconnected		
TxD[5]	O	1	leave unconnected		
TxD[6]	O	1	leave unconnected		
TxD[7]	O	1	leave unconnected		
TX_EN	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	1	leave unconnected		
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RCG[0]/RCG[5]	I	1
RxD[1]	I	1	RCG[1]/RCG[6]	I	1
RxD[2]	I	1	RCG[2]/RCG[7]	I	1
RxD[3]	I	1	RCG[3]/RCG[8]	I	1
RxD[4]	I	1	not used		
RxD[5]	I	1	not used		
RxD[6]	I	1	not used		
RxD[7]	I	1	not used		
RX_DV	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1	not used		
COL	I	1	not used		
CRS	I	1	not used		
Sum		25	**sum**		12

Table 15-187 describes the shared signals for the RTBI interface.

Table 15-187. Shared RTBI Signals

eTSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
MDIO	I/O	1	MDIO	I/O	1	Management interface I/O
MDC	O	1	MDC	O	1	Management interface clock
ECGTX_CLK125	I	1	GTX_CLK125	I	1	Reference clock
Sum		3	Sum		3	—

Table 15-188 describes the register initializations required to configure the eTSEC in RTBI mode.

Table 15-188. RTBI Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that TSEC_MDC clock speed is not greater than 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.

Table 15-188. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle to verify state of TBI Control Register(0ptional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Write to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 15-188. RTBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_00x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

15.7.1.7 8-Bit FIFO Mode

Table 15-189 describes the signal configurations required for 8-bit FIFO interface mode on eTSEC1 and/or eTSEC2.

Table 15-189. 8-Bit FIFO Interface Mode Signal Configurations, eTSEC1/2

eTSEC Signals			8-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC*n*_GTX_CLK	O	1	FIFO*n*_GTX_CLK	O	1
TSEC*n*_TX_CLK	I	1	FIFO*n*_TX_CLK	I	1
TSEC*n*_TxD[0]	O	1	FIFO*n*_TxD[0]	O	1
TSEC*n*_TxD[1]	O	1	FIFO*n*_TxD[1]	O	1
TSEC*n*_TxD[2]	O	1	FIFO*n*_TxD[2]	O	1
TSEC*n*_TxD[3]	O	1	FIFO*n*_TxD[3]	O	1
TSEC*n*_TxD[4]	O	1	FIFO*n*_TxD[4]	O	1
TSEC*n*_TxD[5]	O	1	FIFO*n*_TxD[5]	O	1
TSEC*n*_TxD[6]	O	1	FIFO*n*_TxD[6]	O	1
TSEC*n*_TxD[7]	O	1	FIFO*n*_TxD[7]	O	1
TSEC*n*_TX_EN	O	1	FIFO*n*_TX_EN	O	1
TSEC*n*_TX_ER	O	1	FIFO*n*_TX_ER	O	1
TSEC*n*_RX_CLK	I	1	FIFO*n*_RX_CLK	I	1
TSEC*n*_RxD[0]	I	1	FIFO*n*_RxD[0]	I	1
TSEC*n*_RxD[1]	I	1	FIFO*n*_RxD[1]	I	1
TSEC*n*_RxD[2]	I	1	FIFO*n*_RxD[2]	I	1
TSEC*n*_RxD[3]	I	1	FIFO*n*_RxD[3]	I	1
TSEC*n*_RxD[4]	I	1	FIFO*n*_RxD[4]	I	1
TSEC*n*_RxD[5]	I	1	FIFO*n*_RxD[5]	I	1
TSEC*n*_RxD[6]	I	1	FIFO*n*_RxD[6]	I	1
TSEC*n*_RxD[7]	I	1	FIFO*n*_RxD[7]	I	1
TSEC*n*_RX_DV	I	1	FIFO*n*_RX_DV	I	1
TSEC*n*_RX_ER	I	1	FIFO*n*_RX_ER	I	1
TSEC*n*_COL	I	1	FIFO*n*_TX_FC	I	1
TSEC*n*_CRS	I/O	1	FIFO*n*_RX_FC	O	1
MDIO	I/O	1	leave unconnected		
MDC	O	1	leave unconnected		
Sum		27	**Sum**		25

Table 15-190 describes the signal configurations required for 8-bit FIFO interface mode on eTSEC3 sharing signals with eTSEC4.

Table 15-190. 8-Bit FIFO Interface Mode Signal Configurations, eTSEC3/4

eTSEC Signals			8-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC3_GTX_CLK	O	1	FIFO3_GTX_CLK	O	1
TSEC3_TX_CLK	I	1	FIFO3_TX_CLK	I	1
TSEC3_TxD[0]	O	1	FIFO3_TxD[0]	O	1
TSEC3_TxD[1]	O	1	FIFO3_TxD[1]	O	1
TSEC3_TxD[2]	O	1	FIFO3_TxD[2]	O	1
TSEC3_TxD[3]	O	1	FIFO3_TxD[3]	O	1
TSEC3_TX_EN	O	1	FIFO3_TX_EN	O	1
TSEC3_TX_ER	O	1	FIFO3_TX_ER	O	1
TSEC3_RX_CLK	I	1	FIFO3_RX_CLK	I	1
TSEC3_RxD[0]	I	1	FIFO3_RxD[0]	I	1
TSEC3_RxD[1]	I	1	FIFO3_RxD[1]	I	1
TSEC3_RxD[2]	I	1	FIFO3_RxD[2]	I	1
TSEC3_RxD[3]	I	1	FIFO3_RxD[3]	I	1
TSEC3_RX_DV	I	1	FIFO3_RX_DV	I	1
TSEC3_RX_ER	I	1	FIFO3_RX_ER	I	1
TSEC3_COL	I	1	FIFO3_TX_FC	I	1
TSEC3_CRS	I/O	1	FIFO3_RX_FC	O	1
TSEC4_GTX_CLK	O	1	leave unconnected		
TSEC4_TX_CLK	I	1	not used		
TSEC4_TxD[0]	O	1	FIFO3_TxD[4]	O	1
TSEC4_TxD[1]	O	1	FIFO3_TxD[5]	O	1
TSEC4_TxD[2]	O	1	FIFO3_TxD[6]	O	1
TSEC4_TxD[3]	O	1	FIFO3_TxD[7]	O	1
TSEC4_TX_EN	O	1	leave unconnected		
TSEC4_TX_ER	O	1	leave unconnected		
TSEC4_RX_CLK	I	1	not used		
TSEC4_RxD[0]	I	1	FIFO3_RxD[4]	I	1

Table 15-190. 8-Bit FIFO Interface Mode Signal Configurations, eTSEC3/4 (continued)

eTSEC Signals			8-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC4_RxD[1]	I	1	FIFO3_RxD[5]	I	1
TSEC4_RxD[2]	I	1	FIFO3_RxD[6]	I	1
TSEC4_RxD[3]	I	1	FIFO3_RxD[7]	I	1
TSEC4_RX_DV	I	1	not used		
TSEC4_COL	I	1	not used		
TSEC4_CRS	I/O	1	not used		
MDIO	I/O	1	leave unconnected		
MDC	O	1	leave unconnected		
Sum		35	**Sum**		25

Table 15-191 describes the register initializations required to configure the eTSEC in 8-bit FIFO mode.

Table 15-191. 8-Bit FIFO Mode Register Initialization Steps

Set FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_1100_0000_0000_0000] (Reset RX = 1, reset Tx = 1, Rx enable = 0, Tx enable = 0)
Clear FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_0000_0000_0000_1000] (Reset RX = 0, reset Tx = 0, Rx enable = 0, Tx enable = 0)
Ensure MACCFG2 is set to default values. MACCFG2[0000_0000_0000_0000_0111_0000_0000_0000]
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_1000_0000_0000_0000] (Used to set up FIFO mode = 1, and statistics enable = 0)
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) transmit descriptor ring and fill buffers with data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]

Table 15-191. 8-Bit FIFO Mode Register Initialization Steps (continued)

Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx over FIFO, FIFOCFG[0000_0000_0000_0000_0011_0000_1101_1000] (Rx enable = 1, Tx enable = 1, enable flow control and CRC, 8-bit mode)

15.7.1.8 16-Bit FIFO Mode

Table 15-192 describes the signal configurations required when eTSECs 1 and 2 are placed into 16-bit FIFO interface mode.

Table 15-192. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs1 and 2)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TSEC1_GTX_CLK	O	1	FIFO1_GTX_CLK	O	1
TSEC1_TX_CLK	I	1	FIFO1_TX_CLK	I	1
TSEC1_TxD[0]	O	1	FIFO1_TxD[0]	O	1
TSEC1_TxD[1]	O	1	FIFO1_TxD[1]	O	1
TSEC1_TxD[2]	O	1	FIFO1_TxD[2]	O	1
TSEC1_TxD[3]	O	1	FIFO1_TxD[3]	O	1
TSEC1_TxD[4]	O	1	FIFO1_TxD[4]	O	1
TSEC1_TxD[5]	O	1	FIFO1_TxD[5]	O	1
TSEC1_TxD[6]	O	1	FIFO1_TxD[6]	O	1
TSEC1_TxD[7]	O	1	FIFO1_TxD[7]	O	1
TSEC1_TX_EN	O	1	FIFO1_TXC[0]	O	1
TSEC1_TX_ER	O	1	FIFO1_TXC[1]	O	1
TSEC1_RX_CLK	I	1	FIFO1_RX_CLK	I	1
TSEC1_RxD[0]	I	1	FIFO1_RxD[0]	I	1
TSEC1_RxD[1]	I	1	FIFO1_RxD[1]	I	1

Table 15-192. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs1 and 2) (continued)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
TSEC1_RxD[2]	I	1	FIFO1_RxD[2]	I	1
TSEC1_RxD[3]	I	1	FIFO1_RxD[3]	I	1
TSEC1_RxD[4]	I	1	FIFO1_RxD[4]	I	1
TSEC1_RxD[5]	I	1	FIFO1_RxD[5]	I	1
TSEC1_RxD[6]	I	1	FIFO1_RxD[6]	I	1
TSEC1_RxD[7]	I	1	FIFO1_RxD[7]	I	1
TSEC1_RX_DV	I	1	FIFO1_RXC[0]	I	1
TSEC1_RX_ER	I	1	FIFO1_RXC[1]	I	1
TSEC1_COL	I	1	FIFO1_TX_FC	I	1
TSEC1_CRS	I/O	1	FIFO1_RX_FC	O	1
MDIO	I/O	1	leave unconnected		
MDC	O	1	leave unconnected		
TSEC2_GTX_CLK	O	1	leave unconnected		
TSEC2_TX_CLK	I	1	not used		
TSEC2_TxD[0]	O	1	FIFO1_TxD[8]	O	1
TSEC2_TxD[1]	O	1	FIFO1_TxD[9]	O	1
TSEC2_TxD[2]	O	1	FIFO1_TxD[10]	O	1
TSEC2_TxD[3]	O	1	FIFO1_TxD[11]	O	1
TSEC2_TxD[4]	O	1	FIFO1_TxD[12]	O	1
TSEC2_TxD[5]	O	1	FIFO1_TxD[13]	O	1
TSEC2_TxD[6]	O	1	FIFO1_TxD[14]	O	1
TSEC2_TxD[7]	O	1	FIFO1_TxD[15]	O	1
TSEC2_TX_EN	O	1	FIFO1_TXC[2]	O	1
TSEC2_TX_ER	O	1	leave unconnected		
TSEC2_RX_CLK	I	1	not used		
TSEC2_RxD[0]	I	1	FIFO1_RxD[8]	I	1
TSEC2_RxD[1]	I	1	FIFO1_RxD[9]	I	1
TSEC2_RxD[2]	I	1	FIFO1_RxD[10]	I	1
TSEC2_RxD[3]	I	1	FIFO1_RxD[11]	I	1
TSEC2_RxD[4]	I	1	FIFO1_RxD[12]	I	1

Table 15-192. 16-Bit FIFO Interface Mode Signal Configuration (eTSECs1 and 2) (continued)

eTSEC Signals			16-Bit FIFO Interface		
			Frequency [MHz] 155		
			Voltage [V] 2.5		
TSEC2_RxD[5]	I	1	FIFO1_RxD[13]	I	1
TSEC2_RxD[6]	I	1	FIFO1_RxD[14]	I	1
TSEC2_RxD[7]	I	1	FIFO1_RxD[15]	I	1
TSEC2_RX_DV	I	1	FIFO1_RXC[2]	I	1
TSEC2_RX_ER	I	1	not used		
TSEC2_COL	I	1	not used		
TSEC2_CRS	I	1	not used		
Sum		52	**Sum**		43

Table 15-193 describes the register initializations required to configure the controlling the eTSEC (eTSEC1) in 16-bit FIFO mode. The disabled eTSEC (eTSEC2) must be held in soft reset, and have its transmit and receive functions disabled.

Table 15-193. 16-Bit FIFO Mode Register Initialization Steps

Set FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_1100_0000_0000_0000] (Reset RX = 1, reset Tx = 1, Rx enable = 0, Tx enable = 0)
Clear FIFO Soft_Reset, FIFOCFG[0000_0000_0000_0000_0000_0000_0000_1000] (Reset RX = 0, reset Tx = 0, Rx enable = 0, Tx enable = 0)
Ensure MACCFG2 is set to default values. MACCFG2[0000_0000_0000_0000_0111_0000_0000_0000]
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_1000_0000_0000_0000] (Used to setup FIFO mode = 1, and statistics enable = 0)
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]

Table 15-193. 16-Bit FIFO Mode Register Initialization Steps (continued)

Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx over FIFO, FIFOCFG[0000_0000_0000_0000_0011_0000_1101_1100] (Rx enable = 1, Tx enable = 1, enable flow control and CRC, 16-bit mode)

15.7.1.9 SGMII Interface Support

Table 15-194. SGMII Interface Signal Configuration (4-Wire)

SerDes Signals			**SGMII Interface**		
Frequency [MHz] 1250			**Frequency [MHz] 1250**		
Voltage [V] LVDS			**Voltage [V] LVDS**		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
SD2_TX*n*/$\overline{\text{SD2\_TX}n}$	O	2	TXD	O	2
SD2_RX*n*/$\overline{\text{SD2\_RX}n}$	I	2	RXD	I	2
Sum		4	**Sum**		4

SGMII mode initialization sequence is very similar to TBI mode initialization. Additional initialization is required for the SerDes. An example of SGMII mode initialization sequence is shown in Table 15-195.

Table 15-195. SGMII Mode Register Initialization Steps

Initialize SerDes to select SGMII. The initialization sequence should be prepended with SerDes initialization.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (I/F Mode = 2, Full Duplex = 1) (Set I/F mode = 1 in SGMII 10/100 Mbps speed)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0010_0010] (This example has Statistics Enable = 1, TBIM = 1, SGMIIM = 1) (Set R100M = 1 in SGMII 100 Mbps speed)

Table 15-195. SGMII Mode Register Initialization Steps (continued)

Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example.
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example.
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example.
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0101] set source clock divide by 14 for example to insure that MDC clock speed is not greater than 2.5 MHz
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the eTSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register (CR) is at offset address 0x0 from the TBI's address.
Perform an MII Mgmt read cycle to verify state of TBI Control Register (optional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBICON register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0001_0001] The TBICON register is at offset address 0x11 from the TBI's address.
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBICON register, MIIMCON[0000_0000_0000_0000_0000_0000_0010_0000] This sets TBI in single clock mode and MII Mode off to enable communication with SerDes.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address.
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.

Table 15-195. SGMII Mode Register Initialization Steps (continued)

Additional SerDes setup as required
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] the control register (CR) is at offset address 0x00 from the TBI's address.
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0011_0100_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The PHY Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MIIMSTAT register and check bit 10 (AN Done) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY] = 0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_1110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACnADDR1/2 (Optional) MACnADDR1/2[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]

Table 15-195. SGMII Mode Register Initialization Steps (continued)

Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE0–TBASE7, TBASE0–TBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Initialize (Empty) Receive Descriptor ring and fill with empty buffers Initialize RBASE0–RBASE7, RBASE0–RBASE7[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Transmit Queues Initialize TQUEUE
Enable Receive Queues Initialize RQUEUE
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

Chapter 16
10/100 Fast Ethernet Controller

This chapter describes the 10/100 fast Ethernet controller of the MPC8572E.

16.1 Introduction

The Ethernet IEEE 802.3 protocol is widely used on LANs that are based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. 10/100 Ethernet provides increased Ethernet speed from 10 to 100 megabits per second (Mbps) and provides a simple, cost-effective option for backbone and server connectivity.

The Ethernet protocol implements the bottom two layers of the open systems interconnection (OSI) 7-layer model, that is, the data link and physical sublayers. Figure 16-1 shows the typical Ethernet protocol stack and the relationship to the OSI model.

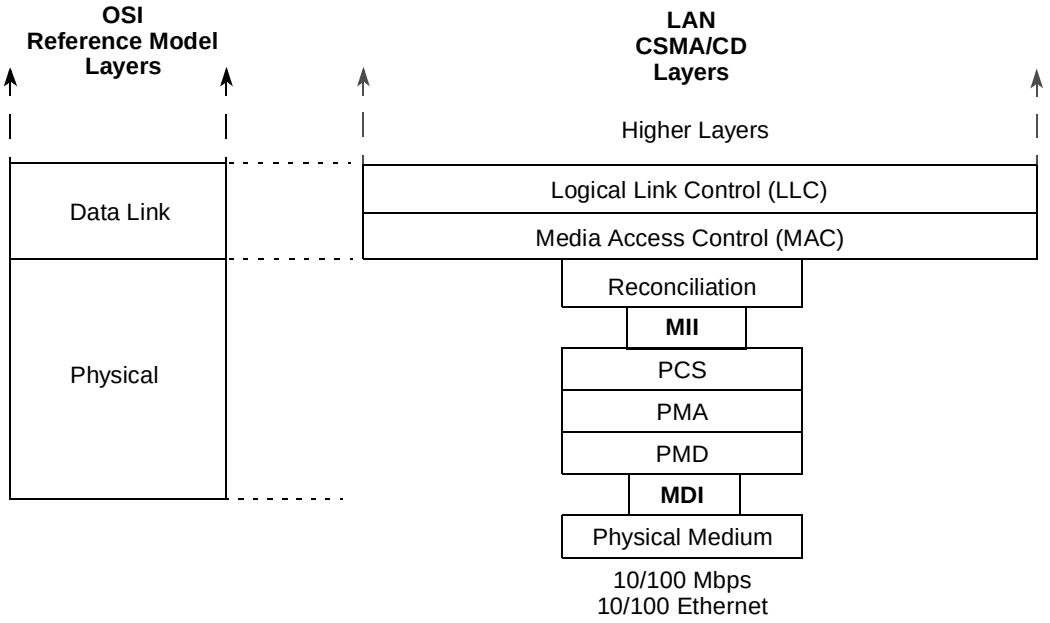


Figure 16-1. Ethernet Protocol in Relation to the OSI Protocol Stack

Ethernet provides the following sublayers:

- Media access control (MAC) sublayer—The MAC sublayer provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.

- Reconciliation sublayer—The reconciliation sublayer acts as a command translator. It maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.
- MII (media-independent interface) sublayer—The MII sublayer provides a standard interface between the MAC layer and the physical layer for 10/100 Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.
- PCS (physical coding sublayer)—The PCS sublayer is responsible for encoding and decoding data stream to and from the MAC sublayer.
- PMA (physical medium attachment) sublayer—The PMA sublayer is responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SerDes) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers.
- PMD (physical medium dependent) sublayer—The PMD sublayer is responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.
- MDI (medium-dependent interface) sublayer—MDI is a connector. It defines different connector types for different physical media and PMD devices.

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 16-2. The term ‘packet’ is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

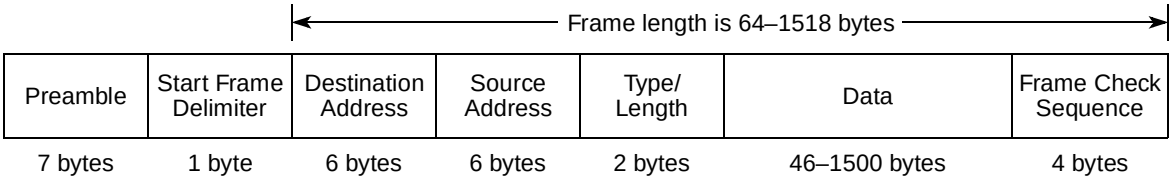


Figure 16-2. Ethernet/IEEE 802.3 Frame Structure

The elements of an Ethernet frame are as follows:

- Preamble— The 7-byte preamble of alternating ones and zeros used for receiver timing synchronization (each byte containing the value 0x55).
- Start frame delimiter (SFD)—A sequence of 0xD5 (10101011 because the bit ordering is lsb first) indicates the beginning of the frame.
- 48-bit destination address (DA)—The first bit identifies the address as an individual address (0) or a group address (1). The second bit is used to indicate whether the address is locally-defined (1) or globally-defined (0).
- 48-bit source address (SA)— (Original versions of the IEEE 802.3 specification allowed 16-bit addressing, which has never been used widely.)
- Ethernet type field/IEEE 802.3 length field—The type field signifies the protocol (for example, TCP/IP) used in the rest of the frame. The length field specifies the length of the data portion of the frame. For both Ethernet and IEEE 802.3 frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This limitation requires that a type field be identified by a decimal number equal to or greater than 1536 (0x0600) but less than 65535

(0xFFFF). If the number, however, is between 0 and 1,500 (0x0000 through 0x05DC) then this field indicates the length of the MAC client data. The range from 1,501 to 1,535 (0x5DD through 0x5FF) was intentionally left undefined.

- Data and padding—Padding is optional. It is only needed if the data is smaller than 46 octets (one octet = one byte) to ensure the minimum frame size of 64 octets as specified in the IEEE 802.3 standard. In 802.3x the first two octets of the data field are used as opcode (OP) (pause = 0x0001) and the second two octets are used to transmit a pause time (PT) parameter (pausetime = 0x0000 for on and 0xFFFF for off). In addition, a third two-octet field can be used for an extended pause control parameter (PTE). Because the use of these fields varies with the protocol used, the ability to examine them and report their content can significantly accelerate Ethernet frame processing.
- Frame-check sequence (FCS)—Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD and CRC.

Figure 16-3 provides additional details of the Ethernet/IEEE 802.3 frame structure.

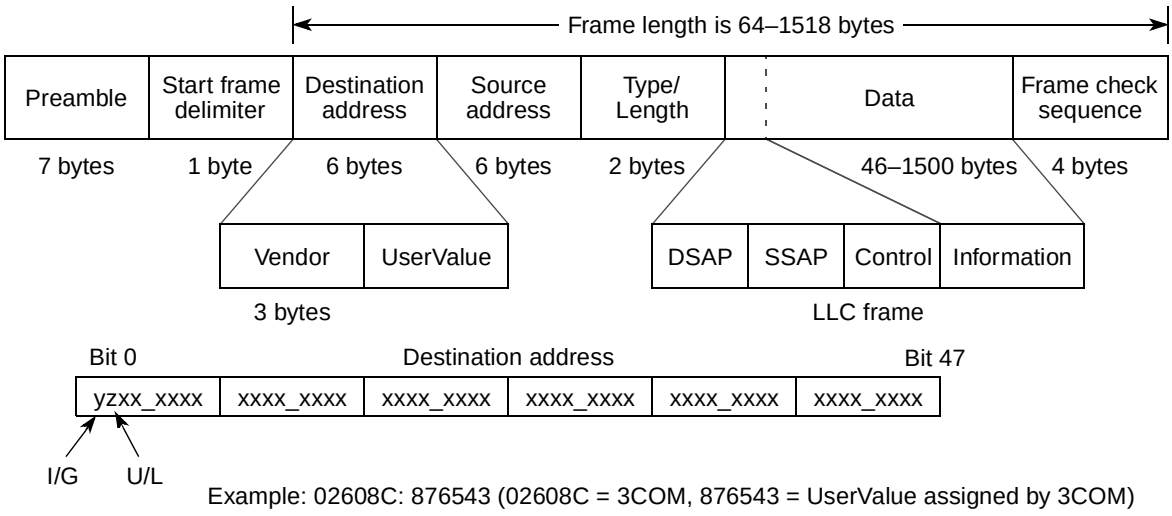


Figure 16-3. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 16-3, the IEEE 802.3 section 3.11 (MAC frame format) defines the frame format such that the octets of a frame are transmitted from left to right (preamble first, the FCS last), and the bits of each octet are transmitted least-significant bit (lsb) first. The destination address example shown in Figure 16-3 (02608C:876543) that would normally be written as

0000 0010 0110 0000 1000 1100 1000 0111 0110 0101 0100 0011

is transmitted bit by bit as

0100 0000 0000 0110 0011 0001 1110 0001 1010 0110 1100 0010

which is an individual address (because the lsb is cleared) but locally-defined (because the second least-significant bit is set).

When first originated, a type field was used for protocol identification. The IEEE 802.3 specification eliminated the type field, replacing it with the length field. The length field is used to identify the length, in bytes, of the data field. The protocol type in 802.3 frames are held within the data portion of the packet.

The logical link control (LLC) is responsible for providing services to the network layer regardless of media type, such as FDDI, Ethernet, token ring, and others. The LLC layer makes use of LLC protocol data units (PDUs) in order to communicate between the media access control (MAC) layer and the upper layers of the protocol stack. Three variables determine access into the upper layers through the LLC-PDU.

The variables include the destination service access point (DSAP), the source service access point (SSAP), and a control variable. The DSAP address specifies a unique identifier within the station providing protocol information for the upper layer. The SSAP provides the same information for the source address.

The LLC defines service access for protocols that conform to the open system interconnection (OSI) model for network protocols. However, many protocols do not obey the rules for those layers and additional information must be added to the LLC in order to provide information regarding those protocols. Protocols that fall into this category include IP and IPX. The method used to provide this additional protocol information is called a subnetwork access protocol (SNAP) frame. A SNAP encapsulation is indicated by the DSAP and SSAP addresses being set to 0xAA and the LLC control field being set to 0x03. If that address is seen, a SNAP header follows. The SNAP header is five bytes long. The first three bytes consist of the organization code (SNAP OUI), which is assigned by the IEEE. The last two bytes become the type value set from the original Ethernet specifications if SNAP OUI = 0, or they become a SNAP protocol identifier if SNAP OUI is non zero.

The fast Ethernet controller (FEC) allows the flexibility to accelerate the identification and retrieval of all the standard and non-standard protocols mentioned above. Figure 16-4 shows the block diagram of the FEC.

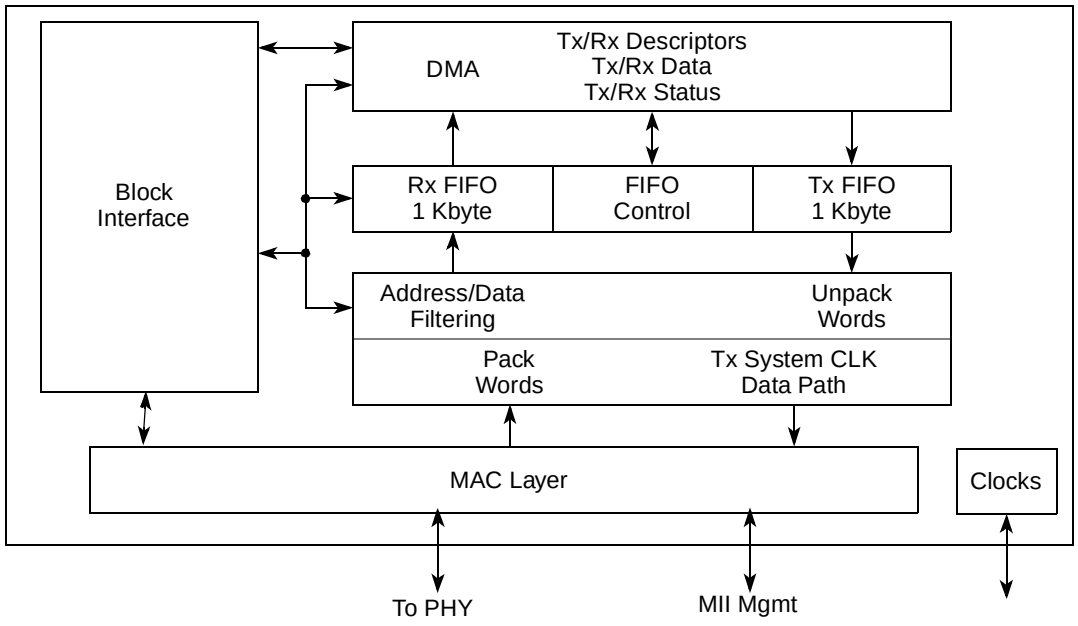


Figure 16-4. FEC Block Diagram

16.1.1 Fast Ethernet Controller Overview

The FEC is designed to support 10 and 100 Mbps Ethernet/802.3 networks and contains the following components:

- Ethernet media access controller (MAC)
- First-in first-out (FIFO) controller
- Direct memory access (DMA) controller

The most-significant byte of data in a receive or transmit data buffer corresponds to the most-significant byte of a frame, respectively.

The complete FEC is designed for single MAC applications. The FEC supports the 10/100 Mbps MII interface to connect to an external Ethernet transceiver.

The FEC software programming model is similar to the FEC offered by the MPC8540 device. It enables customers to leverage already implemented Ethernet drivers, reducing the software development cycle.

16.2 Features

The FEC includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3ac compliant
- Support for 10/100 Mbps IEEE 802.3 MII Ethernet physical interface
- Full- and half-duplex support
- IEEE 802.3 full-duplex flow control (software programmed PAUSE frame generation and recognition)
- Support for out-of-sequence transmit queue (for initiating flow-control)
- Retransmission from transmit FIFO following a collision
- Support for CRC generation and verification of inbound/outbound packets
- Address recognition
  - Broadcast address (accept/reject)
  - Exact match 48-bit individual (unicast) address
  - Hash (256-bit hash) check of individual (unicast) addresses
  - Hash (256-bit hash) check of group (multicast) addresses
  - Promiscuous mode
- Extraction data and its associated buffer descriptors can be directed to the processor’s L2 cache to reduce access latency

16.3 Modes of Operation

The primary FEC operational modes are the following:

- Full- and half-duplex operation

  This is determined by MACCFG2 register's full-duplex bit. Full-duplex mode is intended for use on point to point links between switches or end node to switch. Half-duplex mode is used in connections between an end node and a repeater or between repeaters.

  If configured in half-duplex mode (MACCFG2 register's full-duplex bit is cleared), the MAC complies with the IEEE CSMA/CD access method.

  If configured in full-duplex mode (MACCFG2 register's full-duplex bit is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.
- 10- and 100-Mbps MII interface operation

  The MAC-PHY interface operates in MII mode by configuring bits MACCFG2[22:23] = 01. The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the FEC_TX_CLK and FEC_RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed or it may be controlled by software through the serial management interface (EC5_MDC/EC5_MDIO signals) to the transceiver.
- Address recognition options

  The options supported are promiscuous, broadcast, exact individual address hash or exact match, and multicast hash match. For detailed descriptions refer to Section 16.5.3.7.1, "Individual Address Registers 0–7 (IADDRn)," Section 16.5.3.7.2, "Group Address Registers 0–7 (GADDRn)," Section 16.5.3.4.1, "Receive Control Register (RCTRL)," and Section 16.6.2.5, "Frame Recognition."
- Internal loop back

  Internal loop back mode is selected through the loop back bit in the MACCFG1 register. See Section 16.6.2.9, "Internal and External Loop Back," for details.

16.4 External Signals Description

This section defines the FEC signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention (that is, TxD[3:0] means 0 is the lsb). Notice that except for external physical interfaces the buses and registers follow a big-endian format.

The FEC network interface requires 18 I/O signals (including the EC5_MDIO and EC5_MDC MII management interface) and supports both a data and a management interface to the PHY (transceiver) device. The MII option supports both 10- and 100-Mbps Ethernet rates.

16.4.1 Detailed Signal Descriptions

Table 16-1 contains detailed descriptions of the FEC interface signals. All modes follow the IEEE 802.3 standard, 2000 Edition. Input signals not used are internally disabled. Output signals not used are driven low.

Table 16-1. FEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
FEC_COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.	
		State Meaning	Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.
FEC_CRS	I	Carrier sense input.	
		State Meaning	Asserted/Negated—In MII mode, FEC_TX_CLK is asserted while the transmit or receive medium is not idle. In the event of a collision, FEC_CRS must remain asserted for the duration of the collision.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.
EC5_MDC	O	Management data clock. In MII mode this signal is a clock (typically 2.5 MHz) supplied by the MAC (IEEE–set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed). This clock is generated by dividing the core complex bus (CCB) clock by eight. The ratio can be modified further by writing to MIIMCFG[29:31].	
EC5_MDIO	I/O	Management data input/output, BIDI	
		State Meaning	Asserted/Negated—In MII mode EC5_MDIO is a bi-directional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles.
		Timing	Asserted/Negated—This signal is required to be synchronous with the EC5_MDC signal.
FEC_RX_CLK	I	Receive clock. In MII mode, the receive clock FEC_RX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for FEC_RX_DV, FEC_RXD, and FEC_RX_ER.	
FEC_RX_DV	I	Receive data valid. In MII mode, if FEC_RX_DV is asserted, the PHY is indicating that valid data is present on the MII interface.	
FEC_RXD[3:0]	I	Receive data in. FEC_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when FEC_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While FEC_RX_DV is not asserted, FEC_RXD has no meaning.	
FEC_RX_ER	I	Receive error	
		State Meaning	Asserted/Negated—In MII mode, if FEC_RX_ER and FEC_RX_DV are asserted, the PHY has detected an error in the current frame.
FEC_TX_CLK	I	Transmit clock in. In MII mode, FEC_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the FEC_TX_EN, FEC_TXD, and FEC_TX_ER signals.	
FEC_TXD[3:0]	O	Transmit data out. In MII mode, FEC_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when FEC_TX_EN is asserted and have no meaning while FEC_TX_EN is negated.	

Table 16-1. FEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
FEC_TX_EN	O	Transmit data valid. In MII mode, if FEC_TX_EN is asserted, the MAC is indicating that valid data is present on the MII's FEC_TXD signals.
FEC_TX_ER	O	Transmit error. In MII mode, assertion of FEC_TX_ER for one or more clock cycles while FEC_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting FEC_TX_ER has no effect while operating at 10 Mbps or while FEC_TX_EN is negated. This signal transitions synchronously with respect to FEC_TX_CLK.

16.5 Memory Map/Register Definition

The FEC uses a software model similar to that employed by the fast Ethernet function supported on the MPC8540 FEC.

The FEC device is programmed by a combination of control/status registers (CSR) and buffer descriptors. The CSRs are used for mode control, interrupts, and to access status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in Section 16.6.3, "Buffer Descriptors."

The MII registers are described in this section.

16.5.1 Top-Level Module Memory Map

The FEC implementation requires 4 Kbytes of memory-mapped space, of which more than 1 Kbyte is reserved for future expansion. The space is divided into the following sections:

- General control/status registers
- Transmit-specific control/status registers
- Receive-specific control/status registers
- MAC registers
- Hash function registers

Table 16-2 defines the top-level memory map.

Table 16-2. Module Memory Map Summary

Address	Function
000–0FF	FEC general control/status registers
100–2FF	FEC transmit control/status registers
300–4FF	FEC receive control/status registers
500–5FF	FEC MAC registers
600–7FF	Reserved

Table 16-2. Module Memory Map Summary (continued)

800–8FF	FEC HASH function registers
900–AFF	Reserved
B00–BFF	FEC attribute registers
C00–FFF	Future expansion space

16.5.2 Detailed Memory Map—Control/Status Registers

Table 16-3 lists the address, name, and a cross-reference to the complete description of each register. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 16-3. Module Memory Map

Offset	Name	Access[1]	Reset	Section/Page
Fast Ethernet Controller (FEC) (base address offset: 0x2_8000)				
FEC General Control and Status Registers				
0x000–0x00C	Reserved	R	0x0000_0000	—
0x010	IEVENT—Interrupt event register	R/W	0x0000_0000	16.5.3.1.1/16-12
0x014	IMASK—Interrupt mask register	R/W	0x0000_0000	16.5.3.1.2/16-15
0x018	EDIS—Error disabled register	R/W	0x0000_0000	16.5.3.1.3/16-17
0x01C–0x020	Reserved	R	0x0000_0000	—
0x024	MINFLR—Minimum frame length register	R/W	0x0000_0040	16.5.3.1.4/16-17
0x028	PTV—Pause time value register	R/W	0x0000_0000	16.5.3.1.5/16-18
0x02C	DMACTRL—DMA control register	R/W	0x0000_0000	16.5.3.1.6/16-19
0x030–0x088	Reserved	R	0x0000_0000	—
FEC FIFO Control and Status Registers				
0x04C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	16.5.3.2.1/16-21
0x08C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	16.5.3.2.2/16-21
0x090–0x094	Reserved	R	0x0000_0000	—
0x098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	16.5.3.2.3/16-22
0x09C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	16.5.3.2.4/16-22
0x0A0–0x0FC	Reserved	R	0x0000_0000	—
FEC Transmit Control and Status Registers				
0x100	TCTRL—Transmit control register	R/W	0x0000_0000	16.5.3.3.1/16-23

Table 16-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x104	TSTAT—Transmit status register	R/W	0x0000_0000	16.5.3.3.2/16-24
0x108	Reserved	R	0x0000_0000	—
0x10C	TBDLEN—TxBD data length register	R	0x0000_0000	16.5.3.3.3/16-25
0x110–0x120	Reserved	R	0x0000_0000	—
0x124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	16.5.3.3.4/16-26
0x128–0x180	Reserved	R	0x0000_0000	—
0x184	TBPTR—TxBD pointer register	R/W	0x0000_0000	16.5.3.3.5/16-26
0x188–0x200	Reserved	R	0x0000_0000	—
0x204	TBASE—TxBD base address register	R/W	0x0000_0000	16.5.3.3.6/16-27
0x208–0x2AC	Reserved	R	0x0000_0000	—
0x2B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	16.5.3.3.7/16-27
0x2B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	16.5.3.3.8/16-29
0x2B8–0x2FC	Reserved	R	0x0000_0000	—
FEC Receive Control and Status Registers				
0x300	RCTRL—Receive control register	R/W	0x0000_0000	16.5.3.4.1/16-30
0x304	RSTAT—Receive status register	R/W	0x0000_0000	16.5.3.4.2/16-30
0x308	Reserved	R	0x0000_0000	—
0x30C	RBDLEN—RxBD data length register	R	0x0000_0000	16.5.3.4.3/16-31
0x310–0x320	Reserved	R	0x0000_0000	—
0x324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	16.5.3.4.4/16-32
0x328–0x33C	Reserved	R	0x0000_0000	—
0x340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	16.5.3.4.5/16-32
0x344–0x380	Reserved	R	0x0000_0000	—
0x384	RBPTR—RxBD pointer register	R/W	0x0000_0000	16.5.3.4.6/16-33
0x388–0x400	Reserved	R	0x0000_0000	—

Table 16-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x404	RBASE—RxBD base address register	R/W	0x0000_0000	16.5.3.4.7/16-33
0x408–0x4FC	Reserved	R	0x0000_0000	—
FEC MAC Registers				
0x500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	16.5.3.6.1/16-36
0x504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	16.5.3.6.2/16-38
0x508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	16.5.3.6.3/16-39
0x50C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	16.5.3.6.4/16-40
0x510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	16.5.3.6.5/16-41
0x514–0x51C	Reserved	R	0x0000_0000	—
0x520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	16.5.3.6.6/16-41
0x524	MIIMCOM—MII management command register	R/W	0x0000_0000	16.5.3.6.7/16-42
0x528	MIIMADD—MII management address register	R/W	0x0000_0000	16.5.3.6.8/16-43
0x52C	MIIMCON—MII management control register	W	0x0000_0000	16.5.3.6.9/16-44
0x530	MIIMSTAT—MII management status register	R	0x0000_0000	16.5.3.6.10/16-44
0x534	MIIMIND—MII management indicator register	R	0x0000_0000	16.5.3.6.11/16-45
0x538	Reserved	R	0x0000_0000	—
0x53C	IFSTAT—Interface status register	R/W	0x0000_0000	16.5.3.6.13/16-47
0x540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	16.5.3.6.14/16-47
0x544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	16.5.3.6.15/16-48
0x548–0x67C	Reserved	R	0x0000_0000	—
FEC Hash Function Registers				
0x800	IADDR0—Individual address register 0	R/W	0x0000_0000	16.5.3.7.1/16-49
0x804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x80C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x81C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x820–0x87C	Reserved	R	0x0000_0000	—

Table 16-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x880	GADDR0—Group address register 0	R/W	0x0000_0000	16.5.3.7.2/16-49
0x884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x88C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x89C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x8A0–0xAFF	Reserved	R	0x0000_0000	—
FEC Attribute Registers				
0xB00–0xBF4	Reserved	R	0x0000_0000	—
0xBF8	ATTR—Attribute register	R/W	0x0000_0000	16.5.3.8.1/16-50
0xBFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	16.5.3.8.2/16-51
FEC Future Expansion Space				
0xC00–0xFFC	Reserved	R	0x0000_0000	—

[1] R = means read-only, W = write only, R/W = read and write, LH = latches high, SC = self-clearing.

16.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the FEC registers. Because all of the FEC registers are 32 bits wide, only 32-bit register accesses are supported. Note that the address offsets listed are relative to the block base address of 0x2_8000.

16.5.3.1 FEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

16.5.3.1.1 Interrupt Event Register (IEVENT)

If an event occurs that sets a bit in the interrupt event (IEVENT) register, shown in Figure 16-5, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. Clearing the IEVENT bit clears the interrupt signal. The bit in the IEVENT register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are:

- GTSC, GRSC, TXF, TXB, TXC, RXF, RXB, RXC, and MSRO

Interrupts resulting from errors/problems detected in the network or transceiver are:

- BABR, BABT, LC, and CRL/XDA

Interrupts resulting from internal errors are:

- EBERR, XFUN, and BSY

Some of the error interrupts are independently counted in the management information base (MIB) block counters. Software may choose to mask off these interrupts because these errors are visible to network management through the MIB counters.

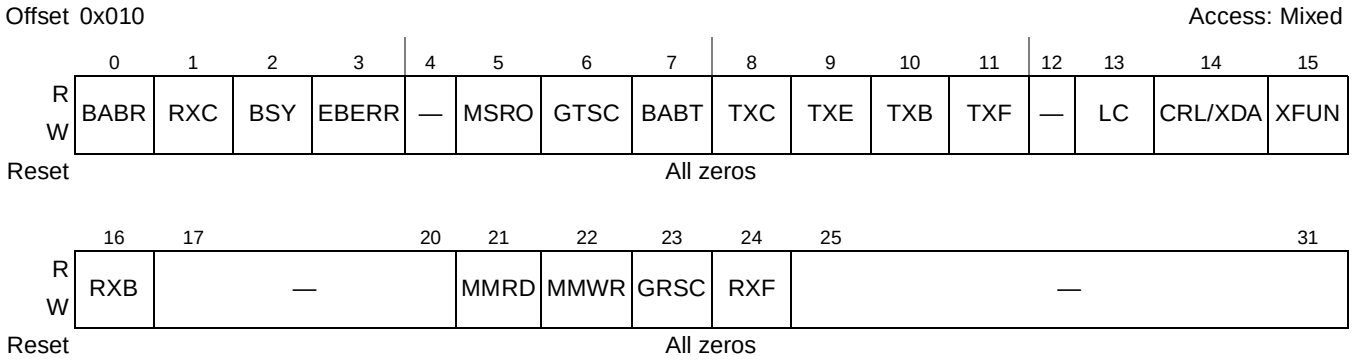


Figure 16-5. IEVENT Register Definition

Table 16-4 describes the fields of the IEVENT register.

Table 16-4. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received 1 Excessive frame received
1	RXC	Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0 Control frame not received 1 Control frame received
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. When IEVENT[BSY] is set RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit and RSTAT[QHLT] are set whenever the FEC reads an RxBD with its Empty field cleared. 0 No frame received and discarded 1 Frame received and discarded

Table 16-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
3	EBERR	Ethernet bus error. This bit indicates that a system bus error for a memory read occurred while a DMA transaction was underway. If the EBERR is set while transmission is in progress, the DMA stops sending data to the Tx FIFO which eventually causes an underrun error (XFUN) and TSTAT[THLT] is set. If the EBERR is set while receiving a frame, the DMA discards the frame and RSTAT[QHLT] is set. 0 No system bus error occurred. 1 System bus error occurred.
4	—	Reserved
5	MSRO	MSTAT register overflow. This interrupt is asserted if the count for one of the MSTAT registers has exceeded the size of the register. 0 MSTAT count not exceeding register size 1 MSTAT count exceeds register size
6	GTSC	Graceful transmit stop complete. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. This interrupt is asserted for one of two reasons. • A graceful stop that was initiated by setting DMACTRL[GTS] is now complete. • A graceful stop that was initiated by setting TCTRL[TFC_PAUSE] is now complete. 0 Graceful transmit stop not complete or not initiated 1 Graceful transmit stop completed
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's maximum frame length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame. 0 Transmitted frame length not exceeding maximum frame length 1 Transmitted frame length exceeding maximum frame length
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted 1 Control frame transmitted
9	TXE	Transmit error. This bit indicates that an error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by FEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL/XDA, XFUN). It is not set if DMACTRL[WOP] is set and FEC runs out of TxBDs to process. In order to begin transmitting packets again, the user must clear TSTAT[THLT]. 0 No transmit channel error occurred 1 Transmit channel error occurred
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (Interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated 1 Transmit buffer descriptor updated
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (Interrupt) bit in the status word of the buffer descriptor is set. 0 No frame transmitted/TxBD not updated 1 Frame transmitted/TxBD updated
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred

Table 16-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
14	CRL/ XDA	Collision retry limit/excessive defer abort. This bit indicates either one of two conditions occurred while attempting to transmit a frame: 1) the number of successive transmission collisions has exceeded the MAC's HAFDUP[Retransmission Maximum] count or 2) an excessive defer abort condition has occurred. An excessive defer abort condition occurs when the FEC waits more than 3036 bytes while attempting to send a frame and HAFDUP[EXCESS_DEFER] is 0. The TxBD[DEF] or OSTBD[DEF] is also set. In either case the frame is discarded without being transmitted and the FEC is halted (TSTAT[THLT] is set). The CRL or XDA condition can only occur while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum and/or no excessive defer abort condition has occurred. 1 Successive transmission collisions exceed maximum or an excessive defer abort condition has occurred.
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun 1 Transmit FIFO underrun
16	RXB	Receive buffer. These bits indicate that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated 1 Receiver buffer descriptor updated
17–22	—	Reserved
21	MMRD	MII management read completion 0 MII management read not issued or in process 1 MII management read completed that was initiated by a user through the MII scan or read cycle command.
22	MMWR	MII management write completion 0 MII management write not issued or in process 1 MII management write completed that was initiated by a user write to the MIIMCON register
23	GRSC	Graceful receive stop complete. This interrupt is asserted if a graceful receive stop is completed. It allows the user to know if the system has completed the stop and it is safe to write to receive registers (status, control or configuration registers) that are used by the system during normal operation. 0 Graceful stop not completed 1 Graceful stop complete.
24	RXF	Receive frame interrupt. The last receive buffer descriptor (RxBD) of a frame was updated. This occurs only if the I (interrupt) bit in the buffer descriptor status word is set. 0 Frame not received 1 Frame received
25–31	—	Reserved

16.5.3.1.2 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events are allowed to generate an actual interrupt. All implemented bits in this CSR are R/W. This register is cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, an interrupt is generated. The interrupt signal can be cleared by clearing the corresponding IEVENT bit.

Figure 16-6 shows the IMASK register.

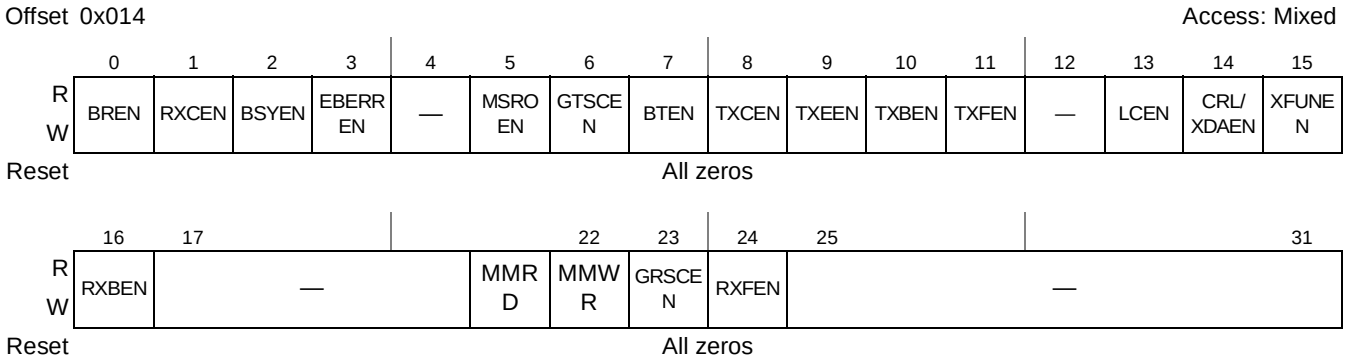


Figure 16-6. IMASK Register Definition

Table 16-5 describes the fields of the IMASK register.

Table 16-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MSTAT register overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRL/XDAEN	Collision retry limit/excessive defer enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–20	—	Reserved
21	MMRD	MII management read completion enable
22	MMWR	MII management write completion enable
23	GRSCEN	Graceful receive stop complete interrupt enable

Table 16-5. IMASK Field Descriptions (continued)

Bits	Name	Description
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

16.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register, shown in Figure 16-7, controls error reporting by the FEC. The IEVENT bit corresponding to an error is not set if the error is disabled in EDIS.

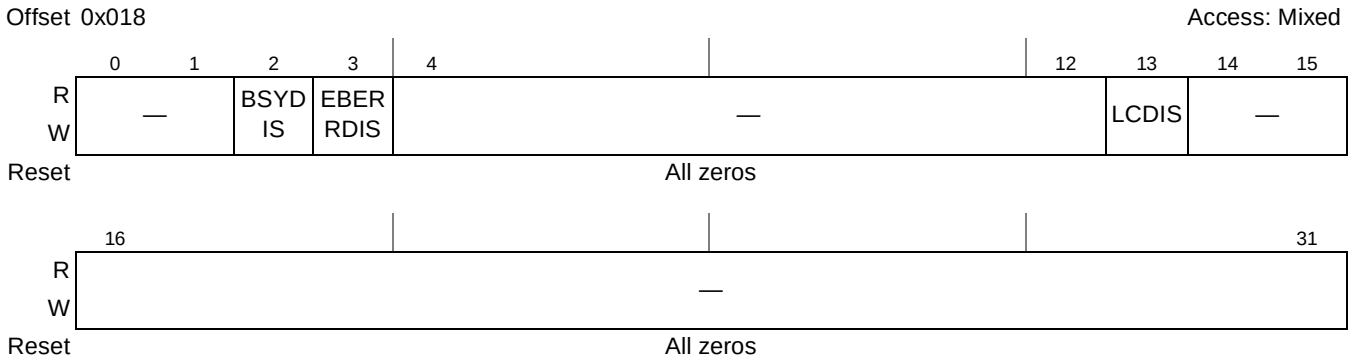


Figure 16-7. Error Disabled Register (EDIS)

Table 16-6 describes the fields of the EDIS register.

Table 16-6. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable 0 Allow FEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable 0 Allow FEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–12	—	Reserved
13	LCDIS	Late collision disable 0 Allow FEC to report IEVENT[LC] status, set the buffer descriptor LC field, and halt buffer descriptor queue if LC condition occurs. 1 Do not set IEVENT[LC] nor the buffer Descriptor LC field, and do not halt buffer descriptor queue if LC condition occurs.
14–31	—	Reserved

16.5.3.1.4 Minimum Frame Length Register (MINFLR)

MINFLR is written by the user. It tells FEC the smallest packet to accept and place in a receive buffer pointed to by the RxBD. Figure 16-8 shows the MINFLR register.

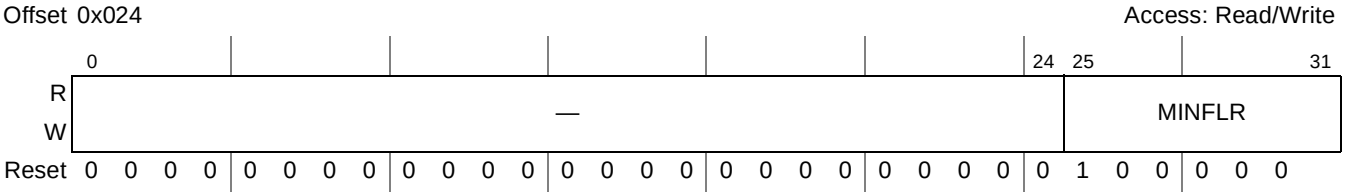


Figure 16-8. MINFLR Register Definition

Table 16-7 describes the fields of the MINFLR register.

Table 16-7. MINFLR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	MINFLR	Minimum receive frame length (typically 64 decimal). If the Ethernet receives an incoming frame shorter than MINFLR, it discards that frame unless RCTRL[RSF] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The largest allowable value for MINFLR is 64. If padding is requested, FEC always PADS transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

16.5.3.1.5 Pause Time Value Register (PTV)

PTV, shown in Figure 16-9, is written by the user to store the pause duration used when the FEC initiates a pause frame through TCTRL[TFC_PAUSE]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of pause_quanta, equal to 512 bit times. The pause time can range from 0 to 65,535 pause_quanta, or 0 to 33,553,920 bit times. See Section 16.6.2.6, “Flow Control for additional details.

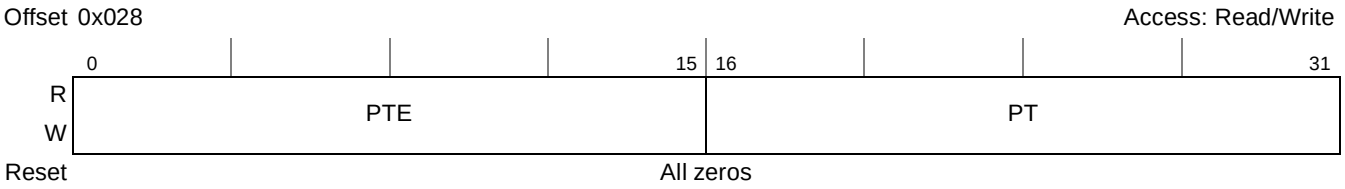


Figure 16-9. PTV Register Definition

Table 16-8 describes the fields of the PTV register.

Table 16-8. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the pause frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 pause frame format requires this parameter to be set to 0.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (that is, 512 bit times). This pause value is used as part of the pause frame to be sent when TCTRL[TFC_PAUSE] is set. See “Section 16.6.2.6, Flow Control” on page 62 for more information.

16.5.3.1.6 DMA Control Register (DMACTRL)

DMACTRL, shown in Figure 16-10, is used to configure the DMA block. register.

Figure 16-10. DMACTRL Register Definition

Table 16-9 describes the fields of the DMACTRL register.

Table 16-9. DMACTRL Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TDSEN	Tx Data snoop enable 0 Disables snooping of all transmit frames from memory 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable 0 Disables snooping of all transmit BD memory accesses 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received. (That is, after a valid end of frame was received). The buffer of the receive frame associated with the EOF is closed and the IEVENT[GRSC] is set. Because the MAC's receive enable bit (MACCFG[Rx_EN]) may still be set, the MAC may continue to receive but the FEC ignores the receive data until GRS is cleared. If this bit is cleared, the FEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the FEC hardware without fear of conflict. 0 FEC scans input data stream for valid frame. 1 FEC stops receiving frames following completion of current frame.
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after the completion of any frame that is currently being transmitted and the GTSC interrupt in the IEVENT register is asserted. If frame transmission is not currently underway, the GTSC interrupt is asserted immediately. Once transmission has completed, a 'restart' can be accomplished by clearing GTS. 0 Controller continues 1 Controller stops transmission after completion of current frame
29	—	Reserved

Table 16-9. DMACTRL Field Descriptions (continued)

Bits	Name	Description
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, IE, XFUN, LC, CRL/XDA, RXB, RXF, the FEC waits for acknowledgement from system that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll. This bit, which is applicable only to the transmitter, provides the user the option for the FEC to periodically poll TxBD or to wait for software to tell FEC to fetch a buffer descriptor. While operating in the “Wait” mode, the FEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven. (IEVENT[TXE] is clear.) To resume transmission, software must clear TSTAT[THLT]. Note that if this bit is set, the user must ensure that all TxBDs involved with sending a frame have their ready bits set before any transmission begins. Otherwise, FEC behaves in a boundedly undefined fashion. A buffer descriptor is considered not ready when its TxBD[Ready] field is clear or its TxBD[Data Length] field is zero. It is considered ready when both TxBD[Ready] is set and TxBD[Data Length] is non-zero. In Wait mode (DMACTRL[WOP] is set) when FEC is processing a frame and an intermediate TxBD’s ready bit is cleared, FEC does not halt, but instead continuously polls the same TxBD until the TxBD becomes ready or an Ethernet interface error or a memory error is encountered. If the TxBD becomes ready, FEC continues to process the frame. If an error occurs before the TxBD becomes ready, FEC reports the error in both the IEVENT register as well as the TxBD for Ethernet interface errors, or the IEVENT[EBERR] for a memory error and sets the TSTAT[THLT] bit. Note that software must eventually set all of its TxBDs for a frame, because FEC continuously reads an intermediate TxBD until it becomes ready if insufficient data has been read to surpass the FIFO Transmit Threshold (FIFO_TX_THR) register value. Note, the polling in Wait mode differs from the polling in Poll mode. In Poll mode polling is performed every 512 Ethernet bus Tx clocks. In Wait mode polling is performed as fast as possible after the TxBD[Ready]=0 is read. If software is slow in setting all TxBD Ready bits of a frame to a one, then the system performance may decrease. 0 Poll TxBD every 512 clocks. 1 Do not poll, but wait for a write to TSTAT[THLT].

16.5.3.2 FEC FIFO Control and Status Registers

The following registers allow the user to change some of the default settings in the FIFO that can be used to optimize operation for performance or for safety. They must be set carefully in order to avoid an underrun condition. Underrun is an error condition in which data is not retrieved from external memory quickly enough, leaving the TX FIFO empty before the complete frame is transmitted. Because different combinations of events, several of which are determined by the user, can lead to underrun, the FEC provides several FIFO registers that allow the user to select the proper setting to be able to tune the system and obtain the maximum performance with minimal chance of underrun. The principal causes for underrun in the FEC are:

- Misaligned data buffer addresses
- Small data buffer sizes
- Combinations of the above

It is recommended that the minimum size data buffers be 64 bytes and that data buffers be 64-byte aligned. The user can deviate from these recommended values to try to increase performance or to use less memory, but unless the default values of some of the FIFO registers are adjusted, the probability of an underrun may also increase. The FIFO_TX_THR (default is 128 entries or 512 bytes) indicates the amount of data required to be in the FIFO before starting the transmission of a frame. The FIFO_TX_STARVE (default is

64 entries or 256 bytes) is used to indicate that the amount of data in the FIFO is so low that the risk of underrun is extremely high. The FIFO_TX_STARVE_SHUTOFF (default is 128 entries or 512 bytes) contains the watermark level to be used for exiting the starve state. These registers are intended to allow the user to make the proper trade-off. If triggered, the starve mode, for instance, automatically raises the priority of FEC fetches from memory.

16.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 16-11, is writable by the user to configure the properties of the FEC FIFO.

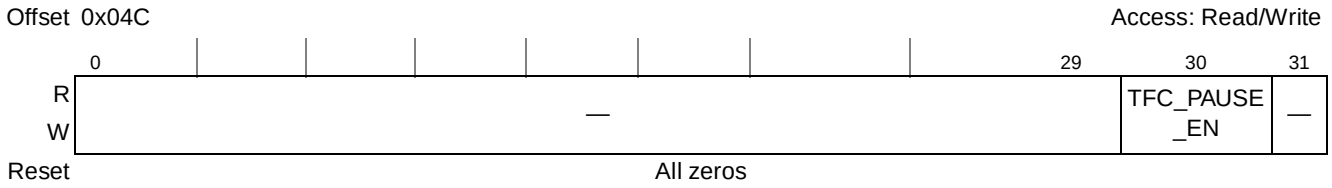


Figure 16-11. FIFO_PAUSE_CTRL Register Definition

Table 16-10 describes the fields of the FIFO_PAUSE_CTRL register.

Table 16-10. FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset. 0 Pause control frame transmission disabled. 1 Pause control frame transmission enabled.
31	—	Reserved

16.5.3.2.2 FIFO Transmit Threshold Register (FIFO_TX_THR)

The main purpose of the threshold register is to trigger the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0–255 for 1-Kbyte FIFO) to trigger the threshold function. If the number of valid entries in the FIFO is equal to or greater than the threshold register, transmission can begin. This register is read/write by software and is initialized to 0000_0000_0000_0000_0000_0000_1000_0000 at system reset. Figure 16-12 shows the FIFO_TX_THR register.

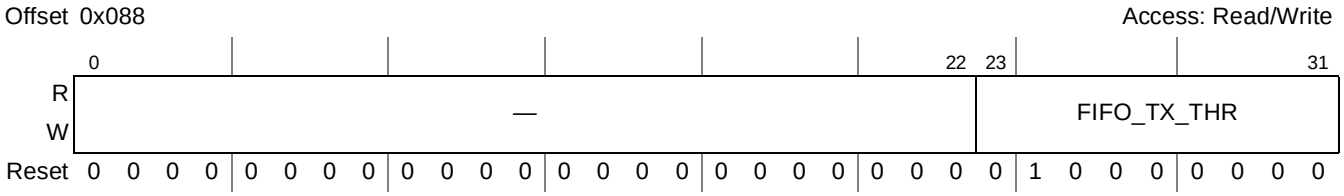


Figure 16-12. FIFO_TX_THR Register Definition

Table 16-11 describes the fields of the FIFO_TX_THR register.

Table 16-11. FIFO_TX_THR Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_THR	FIFO transmit threshold. Marks the number of entries in the transmit FIFO that, when reached, trigger the unloading of frame data into the MAC.

16.5.3.2.3 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register, shown in Figure 16-13, is to inform the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0–255 for 1-Kbyte FIFO) to trigger the starve function. If the number of valid entries in the FIFO is less than or equal to the starve register, a starve alert is triggered.

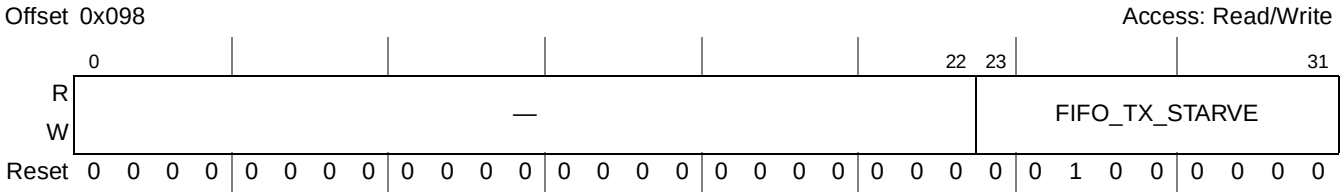


Figure 16-13. FIFO_TX_STARVE Register Definition

Table 16-12 describes the fields of the FIFO_TX_STARVE register.

Table 16-12. FIFO_TX_STARVE Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE	FIFO transmit starve. These bits indicate the value to trigger the transmit starve function. It triggers once the number of valid entries in the FIFO is less than or equal to the FIFO Tx starve. The starve state turns off when the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

16.5.3.2.4 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff register, shown in Figure 16-14, contains the watermark level to be used for exiting the starve state. If the starve state is in effect and the number of valid entries in the FIFO becomes greater than or equal to the value in the FIFO transmit starve shutoff register, the starve condition ends. This register is read/write by software.

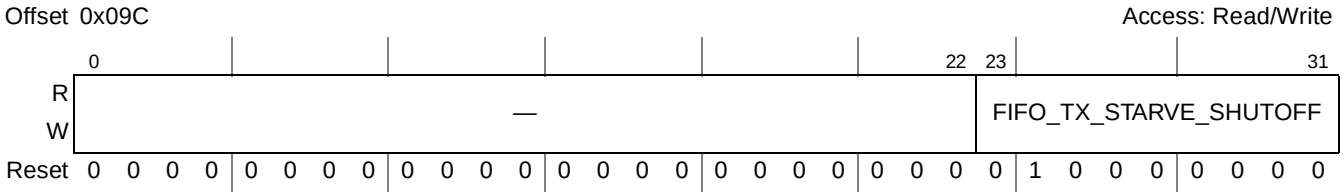


Figure 16-14. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 16-13 describes the fields of the FIFO transmit starve shutoff register.

Table 16-13. FIFO_TX_STARVE_SHUTOFF Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE_SHUTOFF	FIFO transmit starve shutoff. Indicates the value beyond which to exit the starve state. The starve state turns off if the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

16.5.3.3 FEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

16.5.3.3.1 Transmit Control Register (TCTRL)

TCTRL, shown in Figure 16-15, is writable by the user to configure the transmit block.

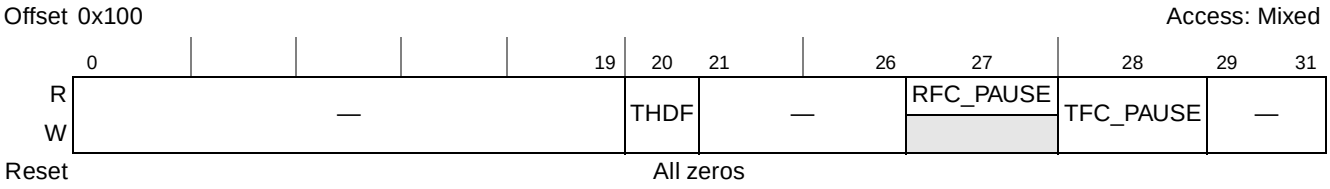


Figure 16-15. TCTRL Register Definition

Table 16-14 describes the fields of the TCTRL register.

Table 16-14. TCTRL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20	THDF	Transmit half-duplex flow control. Written by user. This bit is not self-resetting. 0 Disable back pressure. 1 Back pressure is applied to media.
21–26	—	Reserved
27	RFC_PAUSE	Receive flow control pause frame. Written by FEC. This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete 1 Pause duration in progress
28	TFC_PAUSE	Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. Wait for the GTSC bit in the IEVENT register to be set, then set TFC_PAUSE bit. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame with the duration value obtained from the PTV register. The TXC interrupt occurs after sending the control pause frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. Note that if the transmitter is paused due to user assertion of GTS or reception of a PAUSE frame, the MAC may still transmit a MAC control PAUSE frame. 0 No outstanding pause frame transmission request. 1 Pause frame transmission requested.
29–31	—	Reserved

16.5.3.3.2 Transmit Status Register (TSTAT)

TSTAT, shown in Figure 16-16, is written by FEC to convey DMA status information.

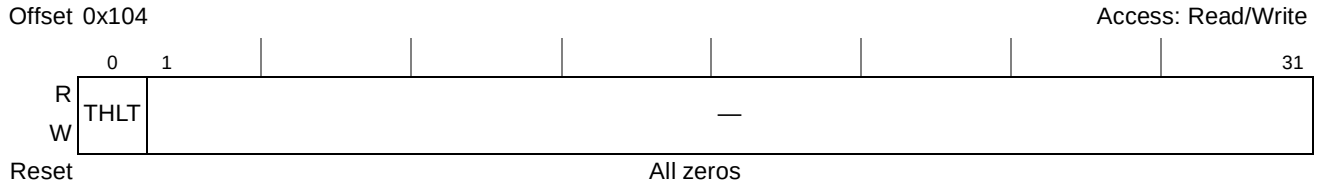


Figure 16-16. TSTAT Register Definition

Table 16-15 describes the fields of the TSTAT register.

Table 16-15. TSTAT Field Descriptions

Bits	Name	Description
0	THLT	Transmit halt 0 No hardware-initiated transmission halt 1 FEC transmission function halted. This bit is written by FEC to inform the user that it is no longer processing transmit frames and that the transmit DMA function is disabled by hardware. To re-start the transmission function, the user must clear this bit by writing a one.
1–31	—	Reserved

16.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 16-17 shows the TBDLEN register.

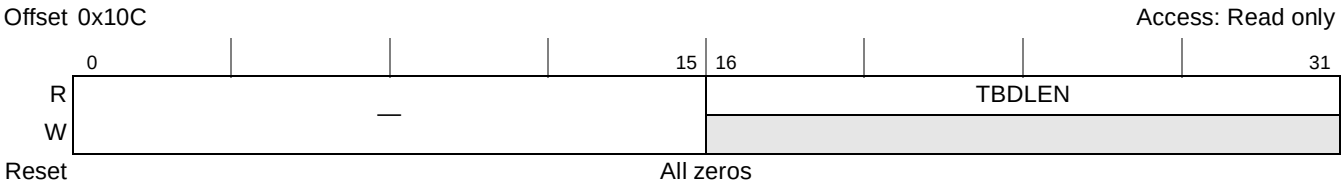


Figure 16-17. TBDLEN Register Definition

Table 16-16 describes the fields of the TBDLEN register.

Table 16-16. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	Internally written by the DMA module. The transmit channel remains active until TBDLEN is 0.

16.5.3.3.4 Current Transmit Buffer Descriptor Pointer Register (CTBPTR)

CTBPTR contains the address of the transmit buffer descriptor either currently being processed, or processed most recently. Figure 16-18 shows the CTBPTR register.

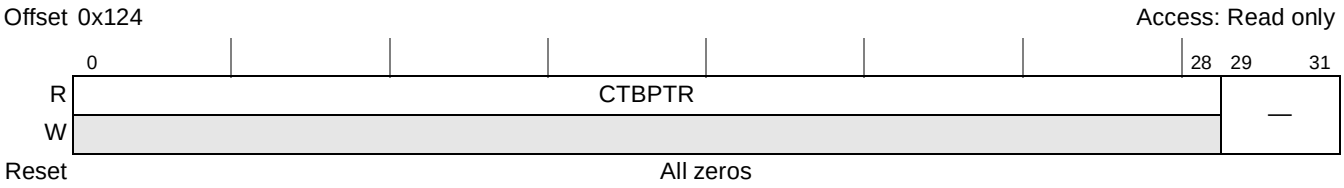


Figure 16-18. CTBPTR Register Definition

Table 16-17 describes the fields of the CTBPTR register.

Table 16-17. CTBPTR Field Descriptions

Bits	Name	Description
0–28	CTBPTR	The CTBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

16.5.3.3.5 Transmit Buffer Descriptor Pointer Register (TBPTR)

TBPTR, shown in Figure 16-19, contains the low-order 32 bits of the next transmit buffer descriptor address. TBPTR takes on the TBASE value when TBASE is written by software. Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by IEVENT[GTSC] or TSTAT[THLT].

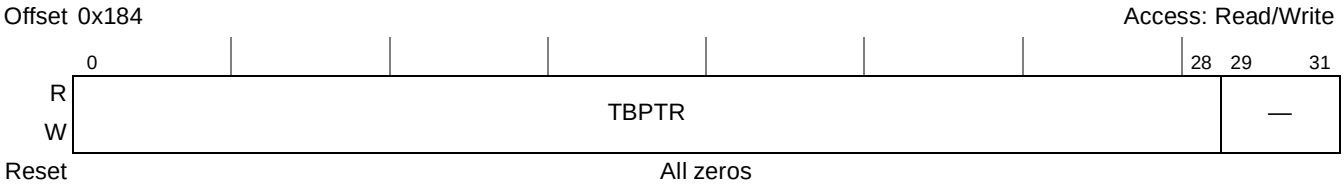


Figure 16-19. TBPTR Register Definition

Table 16-18 describes the fields of the TBPTR register.

Table 16-18. TBPTR Field Descriptions

Bits	Name	Description
0–28	TBPTR	The TBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

16.5.3.3.6 Transmit Descriptor Base Address Register (TBASE)

The TBASE register, shown in Figure 16-20, is written by the user with the TxBD base address. The value must be divisible by eight for the 8-byte data buffer descriptors.

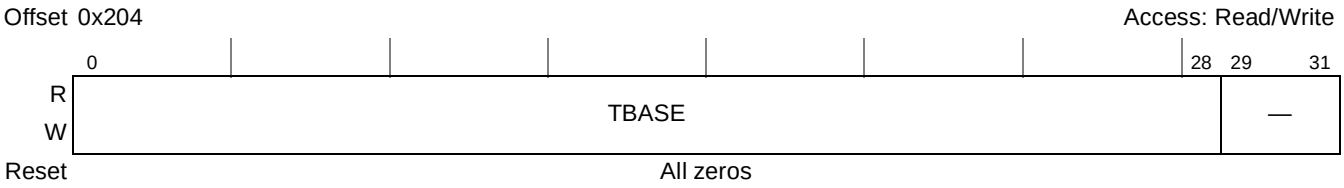


Figure 16-20. TBASE Register Definition

Table 16-19 describes the fields of the TBASE register.

Table 16-19. TBASE Field Descriptions

Bits	Name	Description
0–28	TBASE	Transmit base. TBASE defines the starting location in the memory map for the FEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the FEC transmit function.
29–31	—	Reserved

16.5.3.3.7 Out-of-Sequence TxBD Register (OSTBD)

The out-of-sequence TxBD register, OSTBD, shown in Figure 16-21, includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. If a flow control frame is sent and OSTBD[I] is set, a TXC event is generated after frame transmission. This area must be cleared while not in use. Once the FEC is in paused mode the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC regards it as a regular TxBD.

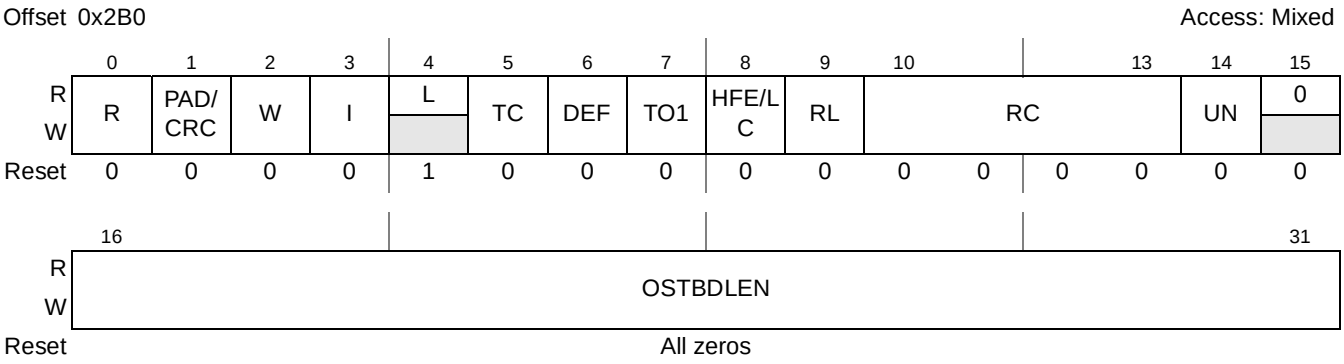


Figure 16-21. OSTBD Register Definition

Table 16-20 describes the fields of the OSTBD register.

Table 16-20. OSTBD Field Descriptions

Bits	Name	Description
0	R	Ready. Written by FEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which was prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
1	PAD/CRC	Padding and CRC attachment for short frames. (Valid only when MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] bit is cleared.) If MACCFG2[PAD/CRC] is set, pads are added to all short frames; however, this bit is ignored. 0 Do not add PADs to short frames unless OSTBD[TC] is set. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. FEC always PADs up to the IEEE minimum frame length of 64 bytes.
2	W	Wrap. Written by user. This bit is ignored by FEC.
3	I	Interrupt. Written by user 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXF] is set after this buffer is serviced. This bit can cause an interrupt if IMASK[TXFEN] is enabled.
4	L	Last in frame. The OSTBD is always the last in the frame, so L is always set. (Hardwired to a value of 1.)
5	TC	Tx CRC. Written by user. (Valid only while it is set in the first BD and OSTBD[PAD/CRC] is cleared, MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, a CRC is added to all frames and this bit is ignored. 0 End transmission immediately after the last data byte, unless OSTBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. Written by FEC. Hardware updates this bit after transmitting a frame if used as a ‘Defer indicator.’ Software/user updates this bit while building a transmit buffer descriptor if used as a ‘Hardware Event indicator.’ 0 This frame was not deferred. 1 This frame did not have a collision before it was sent but it was sent late because of deferring.
7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.
8	HFE/LC	Huge frame enable (written by user)/Late collision (written by FEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC’s Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by FEC 0 No late collision 1 A collision occurred after 64 bytes are sent. The FEC terminates the transmission and updates LC.
9	RL	Retransmission limit. Written by FEC 0 Transmission before maximum retry limit is hit 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The FEC terminates the transmission and updates RL.

Table 16-20. OSTBD Field Descriptions (continued)

Bits	Name	Description
10–13	RC	Retry count. Written by FEC 0 The frame is sent correctly the first time. 1 More than zero attempts were needed to send the transmit frame. If this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. Written by FEC 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The FEC terminates the transmission and updates UN.
15	—	Reserved
16–31	OSTBDLEN	Out-of-sequence TxBD data length. Written by user. Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. This field must be greater than zero in order for a transfer to take place.

16.5.3.3.8 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

The out-of-sequence Tx data buffer pointer register (OSTBDP), shown in Figure 16-22, contains the data buffer pointer fields in the same format as a regular TxBD, With OSTBD, it provides the complete 8-byte descriptor. This area must be cleared while not in use.

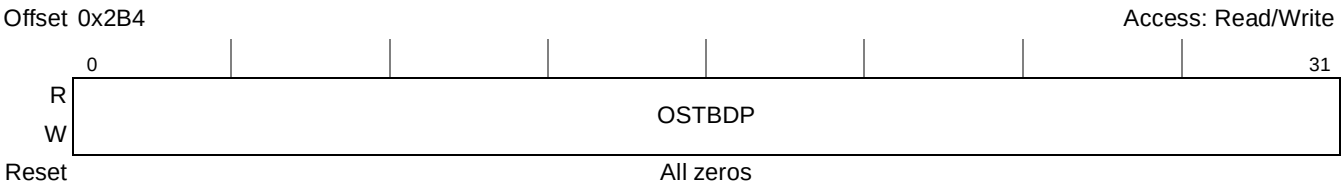


Figure 16-22. OSTBDP Register Definition

Table 16-21 describes OSTBDP.

Table 16-21. OSTBDP Field Descriptions

Bits	Name	Description
0–31	OSTDBP	Out-of-sequence Tx data buffer pointer. Written by user. The transmit data buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

16.5.3.4 FEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

16.5.3.4.1 Receive Control Register (RCTRL)

RCTRL, shown in Figure 16-23, controls the operational mode of the receive block. It must be written only after a system reset (at initialization) or if DMACTRL[GRS] is cleared.

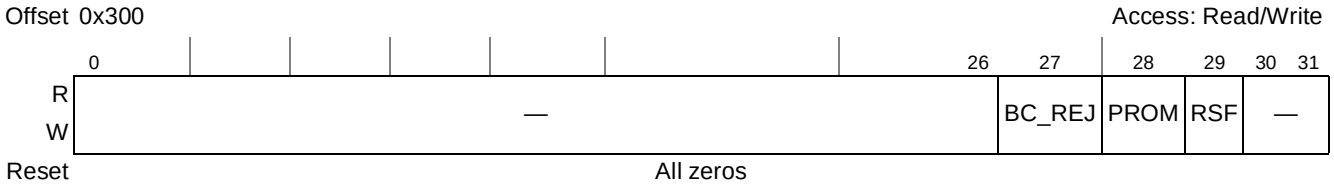


Figure 16-23. RCTRL Register Definition

Table 16-22 describes the fields of the RCTRL register.

Table 16-22. RCTRL Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD. 0 Broadcast frame reject is disabled. 1 Broadcast frame reject is enabled.
28	PROM	Promiscuous mode. When set, all frames except pause frames are accepted. 0 Promiscuous mode is disabled. 1 Promiscuous mode is enabled.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than MINFLR bytes. Note that in order for short frames to be received when RSF is set, a DA hit must occur. When RSF is cleared, all frames shorter than MINFLR are automatically rejected. 0 Receive short frame mode is disabled. 1 Receive short frame mode is enabled.
30–31	—	Reserved

16.5.3.4.2 Receive Status Register (RSTAT)

FEC writes to RSTAT, shown in Figure 16-24, under the following conditions:

- The receiver runs out of descriptors.
- The receiver is halted because an error condition was encountered while receiving a frame.

Software must clear the QHLT bit to take the FEC receiver function out of a halt state.

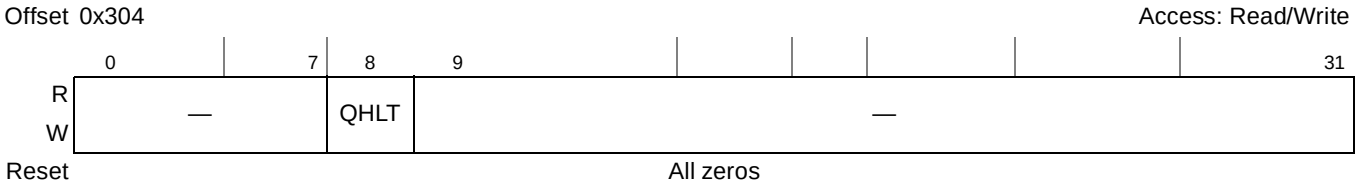


Figure 16-24. RSTAT Register Definition

Table 16-23 describes the fields of the RSTAT register.

Table 16-23. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the FEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication. (DMA_CTRL[GRS] being set by the user does not cause this bit to be set.) The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.) 1 All Ethernet controller receive activity to RxBD queue is halted.
9–31	—	Reserved

16.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer. Figure 16-25 shows the RBDLEN register.

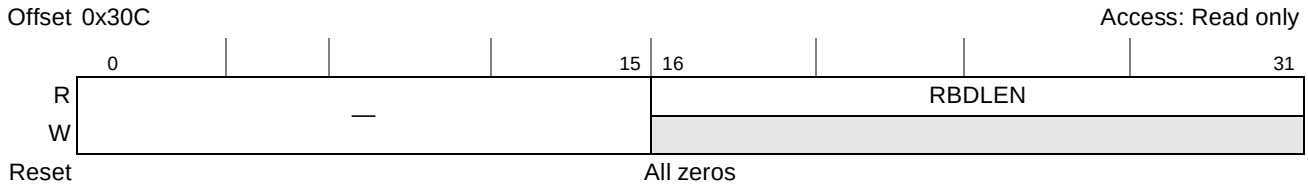


Figure 16-25. RBDLEN Register Definition

Table 16-24 describes the fields of the RBDLEN register.

Table 16-24. RBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RBDLEN	The RBDLEN is internally written by the DMA module. If RBDLEN is cleared, all activity in the receive channel stops.

16.5.3.4.4 Current Receive Buffer Descriptor Pointer Register (CRBPTR)

CRBPTR contains the address of the receive buffer descriptor either currently being processed, or processed most recently. Figure 16-26 shows the CRBPTR register.

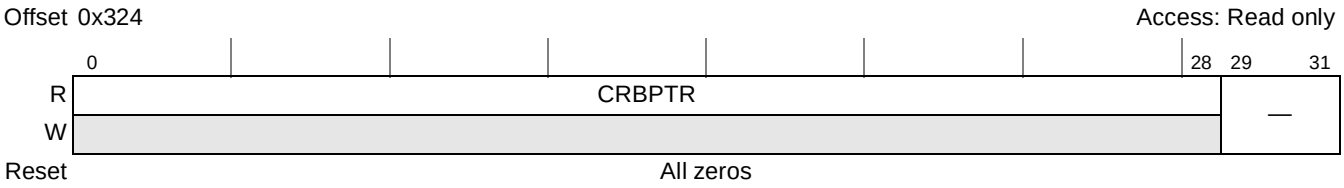


Figure 16-26. CRBPTR Register Definition

Table 16-25 describes the fields of the CRBPTR register.

Table 16-25. CRBPTR Field Descriptions

Bits	Name	Description
0–28	CRBPTR	The CRBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

16.5.3.4.5 Maximum Receive Buffer Length Register (MRBLR)

The MRBL register is written by the user. It informs the FEC how much space is in the receive buffer pointed to by the RxBD. Figure 16-27 shows the MRBLR.

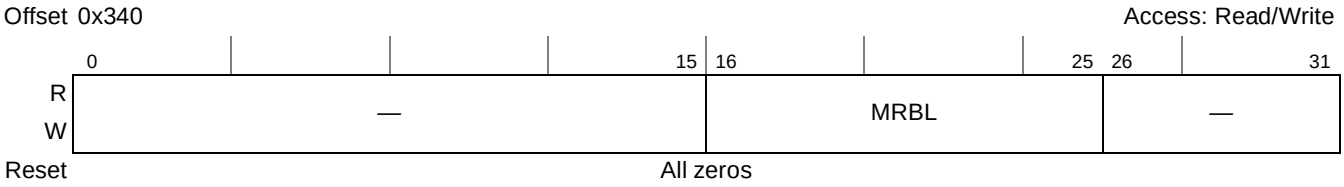


Figure 16-27. MRBL Register Definition

Table 16-26 describes the fields of the MRBL register.

Table 16-26. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	To ensure that MRBL is a multiple of 64, these bits are reserved and must be cleared.
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the FEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. FEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL.
26–31	—	To ensure that MRBL is a multiple of 64, these bits are reserved and must be cleared.

16.5.3.4.6 Receive Buffer Descriptor Pointer Register (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 16-28 shows the RBPTR register. This register takes on the value of RBASE when the RBASE register is written by software. Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by TSTAT[THLT].

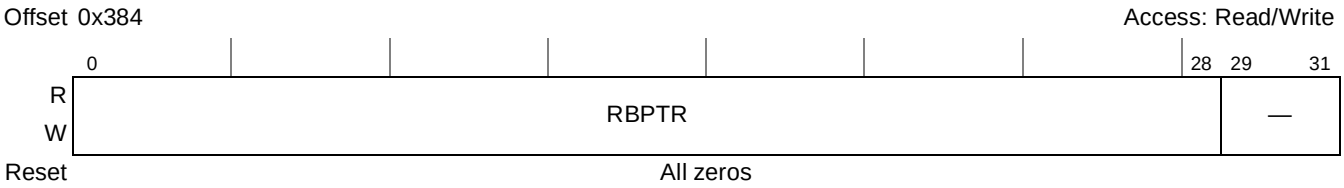


Figure 16-28. RBPTR Register Definition

Table 16-27 describes the fields of the RBPTR register.

Table 16-27. RBPTR Field Descriptions

Bits	Name	Description
0–28	RBPTR	The RBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

16.5.3.4.7 Receive Descriptor Base Address Register (RBASE)

The RBASE register is written by the user with the RxBD base address and must be divisible by eight for the 8-byte buffer descriptors. Figure 16-29 shows the RBASE register.

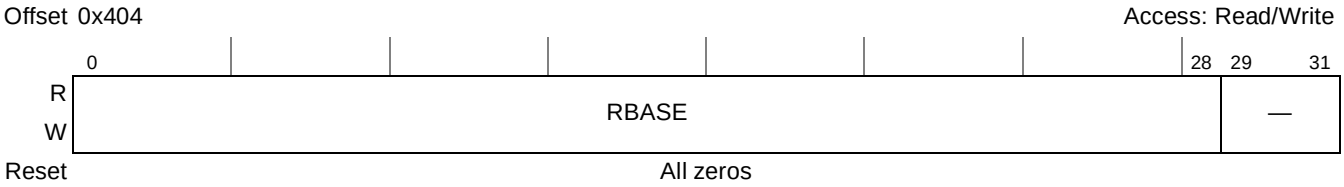


Figure 16-29. RBASE Register Definition

Table 16-28 describes the fields of the RBASE register.

Table 16-28. RBASE Field Descriptions

Bits	Name	Description
0–28	RBASE	Receive base. RBASE defines the starting location in the memory map for the FEC RxBD.
29–31	—	Reserved. These bits must be set to zero, to cause the value of RBASE to be a multiple of eight.

16.5.3.5 MAC Functionality

This section describes the MAC registers and provides a brief overview of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

16.5.3.5.1 Configuring the MAC

MAC configuration registers 1 and 2 provide a way to configure the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value.
- Varying pad/CRC combinations—Three pad/CRC combinations are provided to handle a variety of system requirements. The most simple are frames that already have a valid FCS field. In this case, the CRC is checked and reported through the transmit statistics vector (TSV[51:0]). The other two options include appending a valid CRC, or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

16.5.3.5.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the FEC. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

In full-duplex mode both the carrier sense (CRS) and collision (COL) indications from the PHY are ignored, but in half-duplex mode the FEC defers to CRS and, following a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if CRS is sensed. During the final one-third of the IPG, CRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

16.5.3.5.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the FEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating that a collision occurred and that the start of the frame is needed for retransmission. The FEC then backs off of the medium for a time determined by the truncated binary exponential back-off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured through the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system must flush the frame and move to the next one in line. If the system requests to send a packet while the FEC is deferring to a carrier, the FEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

If packet transmission attempts experience collisions, the FEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly-distributed random integer *r* in the range:

$0 \leq r \leq 2^k$, where $k = \min(n,10)$.

So, after the first collision, FEC backs-off either 0 or 1 slot times. After the fifth collision, FEC backs-off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back-off is 1024. This can be adjusted through the half-duplex register. An alternate truncation point, such as seven for instance, can be programmed. On average, the MAC is more aggressive after seven collisions than other stations on the network.

16.5.3.5.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within the FEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of the packet, the end occurs around the 56th byte of the frame data.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The FEC implements the optional back pressure mechanism using the raise carrier method. If the system receive logic wishes to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is set (TCTRL[THDF]). If the medium is idle, the FEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, the FEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the configuration bit, back pressure no back-off (half-duplex) [BPNB]. These transmitting-preamble-for-back pressure collisions are not counted. If BPNB is set, the FEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If cleared, the FEC adheres to the truncated BEB algorithm that allows the possibility of packets being received. This also can be detrimental in that packets can now experience excessive collisions, causing them to be dropped in the stations from which they originate. To reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, BPNB must be set.

The FEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the FEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. BPNB applies for any collision that occurs during the sending of this packet. The FEC does not defer while attempting to send packets while in back pressure. Again, back pressure is non-standard, yet it can be effective in reducing the flow of receive packets.

16.5.3.5.5 Controlling PHY Links

Control and status to and from the PHY is provided through the two-wire MII management interface described in IEEE 802.3u. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) are used to exercise this interface between a host processor and one or more PHY devices.

The FEC MII’s registers provide the ability to perform continuous read cycles, called a scan cycle. Scan cycles, however, are not explicitly defined in the standard. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link

characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY Status].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the FEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

16.5.3.6 MAC Registers

The following sections describe the MAC registers.

16.5.3.6.1 MAC Configuration Register 1 (MACCFG1)

The MACCFG1 register is written by the user. Figure 16-30 shows the MACCFG1 register.

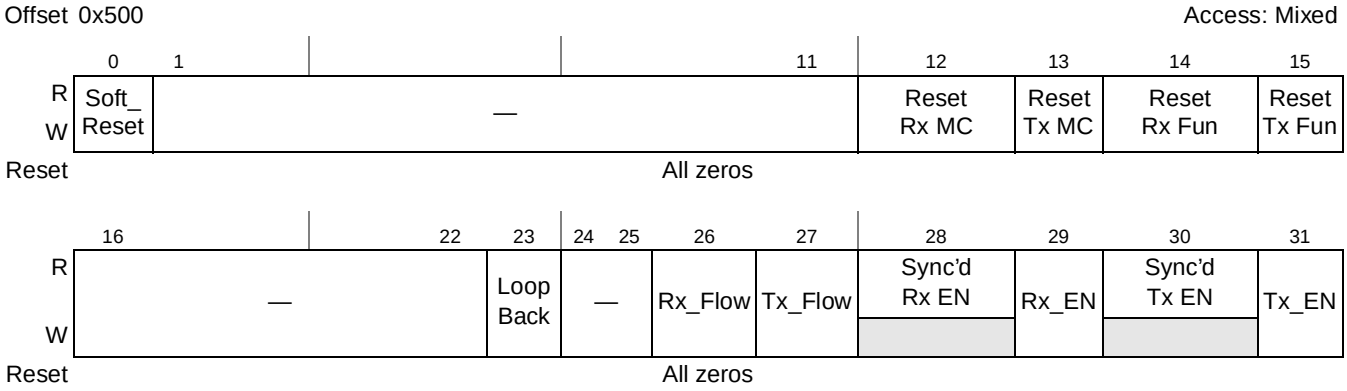


Figure 16-30. MACCFG1 Register Definition

Table 16-29 describes the fields of the MACCFG1 register.

Table 16-29. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 16.6.2.2, "Soft Reset and Reconfiguring Procedure," for more information on setting this bit. 0 Normal operation 1 Place all modules within the MAC in reset.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the receive MAC control block in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation. 1 Place the PETMC transmit MAC control block in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.

Table 16-29. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation. 1 Place the receive function block in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation. 1 Place the transmit function block in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop Back	Loop back. This bit is cleared by default. Software must also set MACCFG2[FULL DUPLEX] if this bit is set. 0 Normal operation. 1 Loop back the MAC transmit outputs to the MAC receive inputs.
24–25	—	Reserved
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful transmit stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

16.5.3.6.2 MAC Configuration Register 2 (MACCFG2)

The MACCFG2 register is written by the user. Figure 16-31 shows the MACCFG2 register.

Offset 0x504 Access: Read/Write

	0–15
R	—
W	
Reset	All zeros

	16–19	20–21	22–23	24–25	26	27	28	29	30	31
R	Preamble Length	—	I/F Mode	—	Huge Frame	Length check	—	PAD/ CRC	CRC EN	Full Duplex
W										
Reset	0 1 1 1	0 0	0 0	0 0	0	0	0	0	0	0

Figure 16-31. MACCFG2 Register Definition

Table 16-30 describes the fields of the MACCFG2 register.

Table 16-30. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble Length	Determines the length in bytes of the preamble field of the packet. Its default is 0x7. A preamble length of 0 is not supported.
20–21	—	Reserved
22–23	I/F Mode	Determines the type of interface to which the MAC is connected. Its default is 00. 00 Reserved 01 Nibble mode (MII) 10–11Reserved
24–25	—	Reserved
26	Huge Frame	Huge frame enable. Cleared by default. 0 Limit the length of frames to the MAXIMUM FRAME LENGTH value. 1 Frames longer than the MAXIMUM FRAME LENGTH may be transmitted and received.
27	Length check	Length check. This bit is cleared by default. 0 No length field checking is performed. 1 The MAC checks the frame's length field to ensure it matches the actual data field length.
28	—	Reserved
29	PAD/CRC	Pad and append CRC. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.

Table 16-30. MACCFG2 Field Descriptions (continued)

Bits	Name	Description
30	CRC EN	CRC enable. If the configuration bit PAD/CRC or the per-packet PAD/CRC bit is set, CRC EN is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	FULL DUPLEX	Full duplex configure. This bit is cleared by default. Software must set this bit if in internal loopback mode (MACCFG1[LOOP BACK] = 1). 0 The MAC to operate in half-duplex mode only. 1 The MAC operates in full-duplex mode.

16.5.3.6.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. Figure 16-32 shows the IPGIFG register.

Offset 0x508 Access: Read/Write

	0	1						7	8	9						15
R W	—	Non-Back-to-Back Inter-Packet-Gap, Part 1							—	Non-Back-to-Back Inter-Packet-Gap, Part 2						
Reset	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0

	16							23	24	25						31
R W	Minimum IFG Enforcement								—	Back-to-Back Inter-Packet-Gap						
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0

Figure 16-32. IPGIFG Register Definition

Table 16-31 describes the fields of the IPGIFG register.

Table 16-31. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	Programmable field representing the optional carrier Sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision; thus, ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	Programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	Programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.

Table 16-31. IPGIFG Field Descriptions (continued)

Bits	Name	Description
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	Programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

16.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 16-33 shows the HAFDUP register.

Offset 0x50C Access: Read/Write

	0							7	8			11	12	13	14	15
R W	—								Alternate BEB Truncation				Alt BEB	BP No Back Off	No Back Off	Excess Defer
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1

	16			19	20			25	26			31
R W	Retransmission Maximum				—				Collision Window			
Reset	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1

Figure 16-33. Half-Duplex Register Definition

Table 16-32 describes the fields of the HAFDUP register.

Table 16-32. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while alternate binary exponential back-off enable is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential back-off. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back-off rule. 1 The Tx MAC uses the alternate binary exponential back-off truncation setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 210 as the maximum back-off time.
13	BP No BackOff	Back pressure no back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.
14	No BackOff	No back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.

Table 16-32. HAFDUP Field Descriptions (continued)

Bits	Name	Description
16–19	Retransmission Maximum	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The standard specifies the attempt limit to be 0xF (15d). Its default value is 0xF.
20–25	—	Reserved
26–31	Collision Window	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Because the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

16.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 16-34 shows the MAXFRM register.

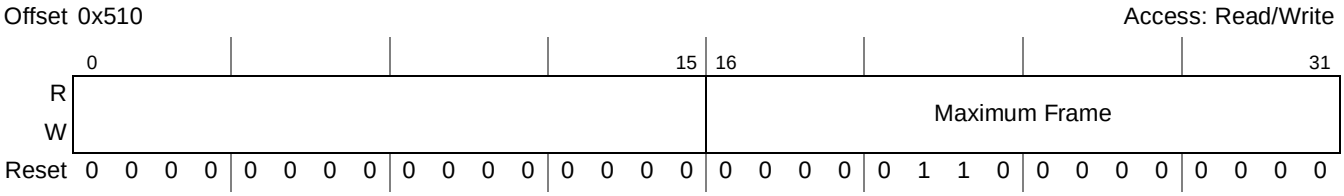


Figure 16-34. Maximum Frame Length Register Definition

Table 16-33 describes the fields of the MAXFRM register.

Table 16-33. MAXFRM Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536d). It sets the maximum frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].)

16.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 16-35 shows the MIIMCFG register.

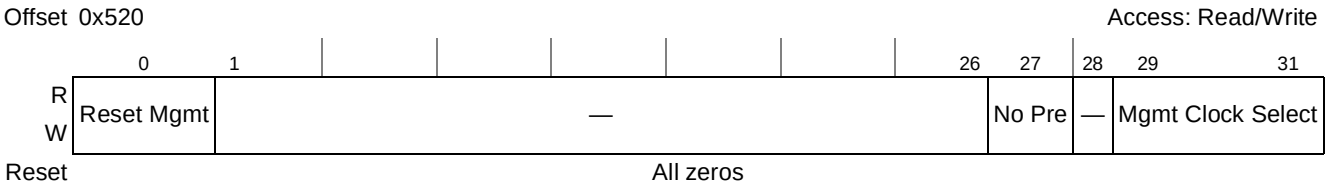


Figure 16-35. MII Management Configuration Register Definition

Table 16-34 describes the fields of the MIIMCFG register.

Table 16-34. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII mgmt to perform management read/write cycles if requested. 1 Reset the MII mgmt.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII mgmt performs management read/write cycles with 32 clocks of preamble. 1 The MII mgmt suppresses preamble generation and reduces the management cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (EC5_MDC). Its default value is 000. The source clock frequency is equal to the core complex bus clock divided first by eight and then further divided by the following value: 000 Source clock divided by 4 001 Source clock divided by 4 010 Source clock divided by 6 011 Source clock divided by 8 100 Source clock divided by 10 101 Source clock divided by 14 110 Source clock divided by 20 111 Source clock divided by 28

16.5.3.6.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 16-36 shows the MIIMCOM register.

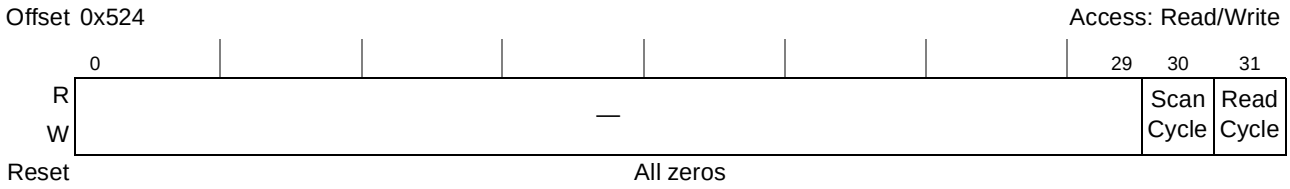


Figure 16-36. MIIMCOM Register Definition

Table 16-35 describes the fields of the MIIMCOM register.

Table 16-35. MIIMCOM Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation 1 The MII management continuously performs read cycles. This is useful for polling a PHY register, for example, monitoring link fails. Data is returned in register MIIMSTAT[PHY Status]. IEVENT[MMRD] is set once a read has been completed.
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0 to 1 transition of this bit also causes the MIIMIND[BUSY] bit to be set. The read is complete when the MIIMIND[BUSY] bit clears. Data is returned in register MIIMSTAT[PHY Status]. IEVENT[MMRD] is also set once the read has been completed.

16.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 16-37 shows the MIIMADD register.

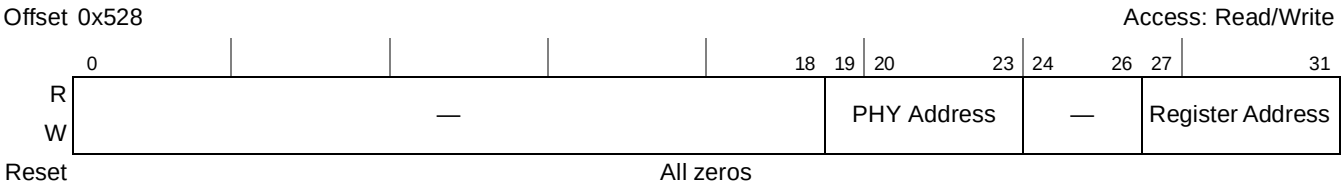


Figure 16-37. MIIMADD Register Definition

Table 16-36 describes the fields of the MIIMADD register.

Table 16-36. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of management cycles. Up to 31 PHYs can be addressed (0 is reserved).
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of management cycles. Up to 32 registers can be accessed. Its default value is 0x00.

16.5.3.6.9 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 16-38 shows the MIIMCON register.

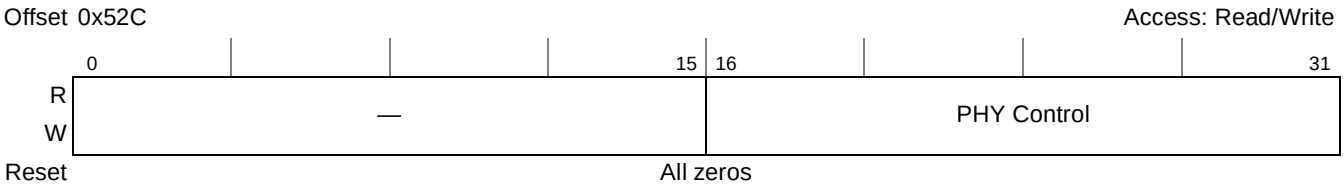


Figure 16-38. MII Management Control Register Definition

Table 16-37 describes the fields of the MIIMCON register.

Table 16-37. MIIMCON Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Control	If written, an MII mgmt write cycle is performed using this 16-bit data, the pre-configured PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). Its default value is 0x0000. IEVENT[MMWR] is set once the write has been completed.

16.5.3.6.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT, shown in Figure 16-39, is read-only by the user.

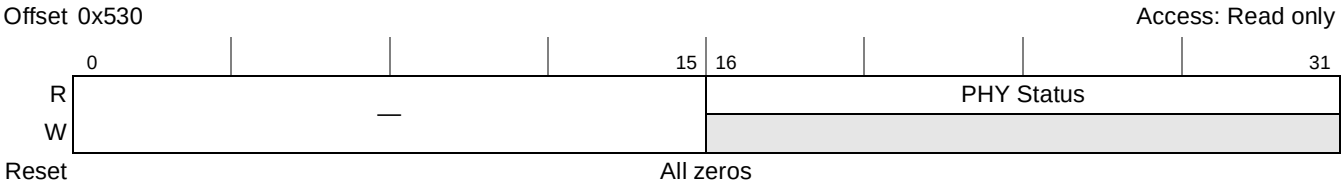


Figure 16-39. MIIMSTAT Register Definition

Table 16-38 describes the fields of the MIIMSTAT register.

Table 16-38. MIIMSTAT Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII mgmt read cycle, the 16-bit data can be read from this location. Its default value is 0x0000.

16.5.3.6.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 16-40 shows the MIIMIND register.

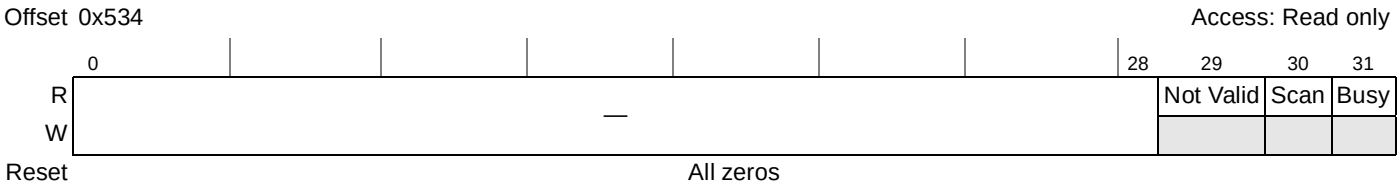


Figure 16-40. MII Management Indicator Register Definition

Table 16-39 describes the fields of the MIIMIND register.

Table 16-39. MIIMIND Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	Not Valid	Not valid 0 MII mgmt read cycle has completed and the read data is valid. 1 MII mgmt read cycle has not completed and the read data is not yet valid.
30	Scan	Scan in progress 0 A scan operation (continuous MII mgmt read cycles) is not in progress. 1 A scan operation (continuous MII mgmt read cycles) is in progress.
31	Busy	Busy 0 MII mgmt block is not currently performing an MII mgmt read or write cycle. 1 MII mgmt block is currently performing an MII mgmt read or write cycle. IEVENT[MMRD] or IEVENT[MMWR] is set once the read or write, respectively, has been completed.

16.5.3.6.12 Interface Control (Interface Control)—User

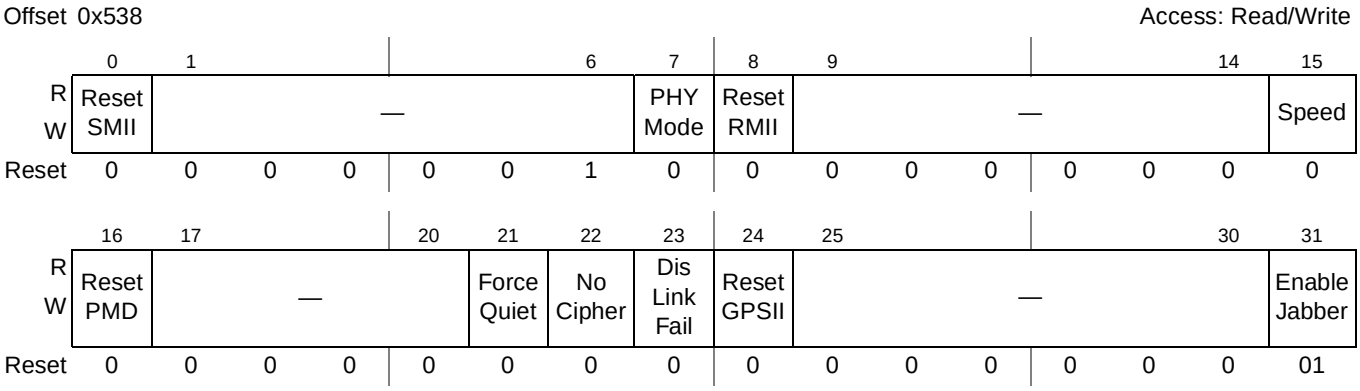


Figure 16-41. Interface Control Register Definition

Table 16-40. Interface Control Register Field Descriptions

Bits	Name	Description
0	Reset SMII	Reset SMII. This bit is cleared by default. 0 Normal operation. 1 The PESMII serial MII module is reset.
1–6	—	Reserved
7	PHY Mode	PHY mode configuration. This bit is set by default. 0 The PESMII configured to be in MAC to MAC mode. In this configuration, the serial MII module reverts to pre-defined settings of 100 Mbps, full-duplex. 1 The PESMII serial MII module configured to be in PHY mode. Link characteristics are taken directly from the Rx segments supplied by the PHY.
8	Reset RMII	Reset RMII. This bit is cleared by default. 0 Normal operation. 1 The PERMII reduced MII module is reset.
9–14	—	Reserved
15	Speed	Operating speed. This bit configures the PERMII reduced MII module with the current operating speed. This bit is cleared by default. 0 10 Mbps mode is selected. 1 100 Mbps mode is selected.
16	Reset PMD	PMD reset. This bit is cleared by default. 0 Normal operation. 1 The PE100X module, which contains the 4B/5B symbol encipher/decipher logic, is reset.
17–20	—	Reserved
21	Force Quiet	Force quiet enable. Affects PE100X module only. This bit is cleared by default. 0 Normal operation. 1 Transmit data is quieted, which allows the contents of the cipher to be output.
22	No Cipher	Cipher inhibit. Affects PE100X module only. This bit is cleared by default. 0 Normal ciphering. 1 The raw transmit 5B symbols are transmitted without ciphering.
23	Dis Link Fail	Link fail timer disable. Affects PE100X module only. This bit is cleared by default. 0 Normal operation 1 The 330ms link fail timer is disabled, allowing for shorter simulations. Removes the 330 mS link-up time before reception of streams is allowed.
24	Reset GPSII	GPSII reset. Affects PE10T module only. This bit is cleared by default. 0 Normal operation 1 The PE10T module, which converts MII nibble streams to the serial bit stream of ENDEC PHYs, is reset.
25–30	—	Reserved
31	Enable Jabber	Jabber protection enable. Jabber is the condition where a transmitter is stuck on for longer than 50 ms preventing other stations from transmitting. Affects PE10T module only. This bit is cleared by default. 0 Jabber protection logic is disabled within the PE10T in ENDEC mode. 1 Jabber protection logic is enabled within the PE10T in ENDEC mode.

16.5.3.6.13 Interface Status Register (IFSTAT)

The IFSTAT register is written by the user. Figure 16-42 shows the IFSTAT register.

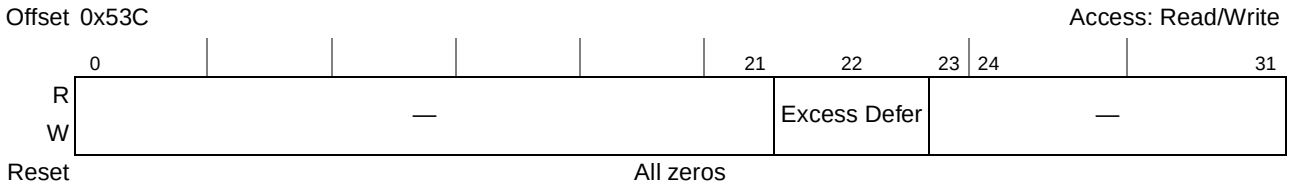


Figure 16-42. Interface Status Register Definition

Table 16-41 describes the fields of the IFSTAT register.

Table 16-41. IFSTAT Field Descriptions

Bits	Name	Description
0–21	—	Reserved
22	Excess Defer	Excessive transmission defer. This bit latches high and is cleared when read. This bit is cleared by default. 0 Normal operation 1 The MAC excessively defers a transmission.
23–31	—	Reserved

16.5.3.6.14 Station Address Register Part 1 (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 16-43 shows the MACSTNADDR1 register. The value of the station address written by the user into MACSTNADDR1 and MACSTNADDR2 is byte-reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, perform a write to MACSTNADDR1 of 0xCDAB7856, and to MACSTNADDR2 of 0x34120000. When the user reads MACSTNADDR1, 0xCDAB7856 is returned. A read of MACSTNADDR2 returns a value of 0x34120000. Note, the I/G and U/L bits of the frame's DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

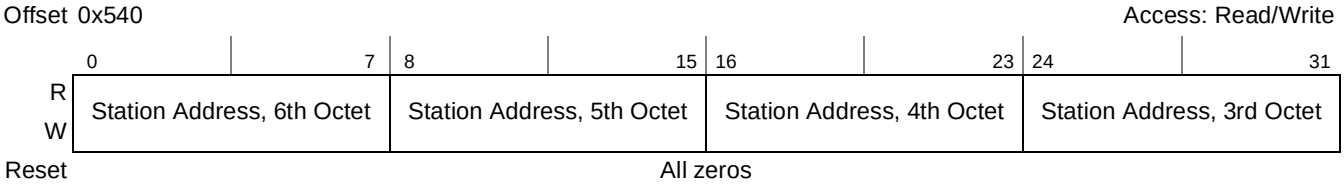


Figure 16-43. Station Address Part 1 Register Definition

Table 16-42 describes the fields of the MACSTNADDR1 register.

Table 16-42. MACSTNADDR1 Field Descriptions

Bits	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x00.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x00.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x00.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x00.

16.5.3.6.15 Station Address Register Part 2 (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 16-44 shows the MACSTNADDR2 register. Note, the I/G and U/L bits of the frame's DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

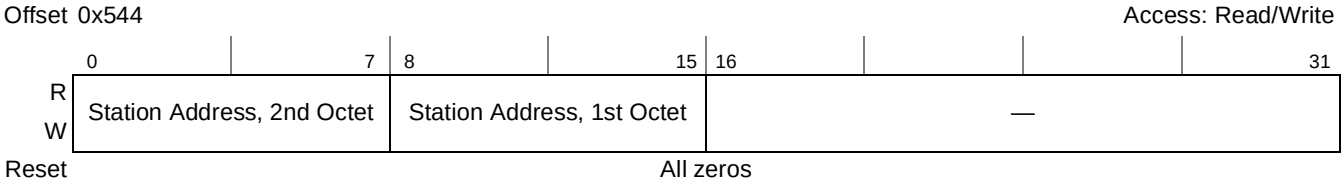


Figure 16-44. Station Address Part 2 Register Definition

Table 16-43 describes the fields of the MACSTNADDR2 register.

Table 16-43. MACSTNADDR2 Field Descriptions

Bits	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x00.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x00.
16–31	—	Reserved

16.5.3.7 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. If the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to a hash table entry. The user can enable a hash entry by setting the appropriate bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. The user must further filter the address. See Section 16.6.2.5.2, "Hash Table Algorithm," for more information on the hash algorithm.

16.5.3.7.1 Individual Address Registers 0–7 (IADDR*n*)

The IADDR*n* registers, shown in Figure 16-45, are written by the user. These registers represent 256 entries of the individual (unicast) address hash table used in the address recognition process. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 high-order bits (0–7) of the CRC remainder are mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

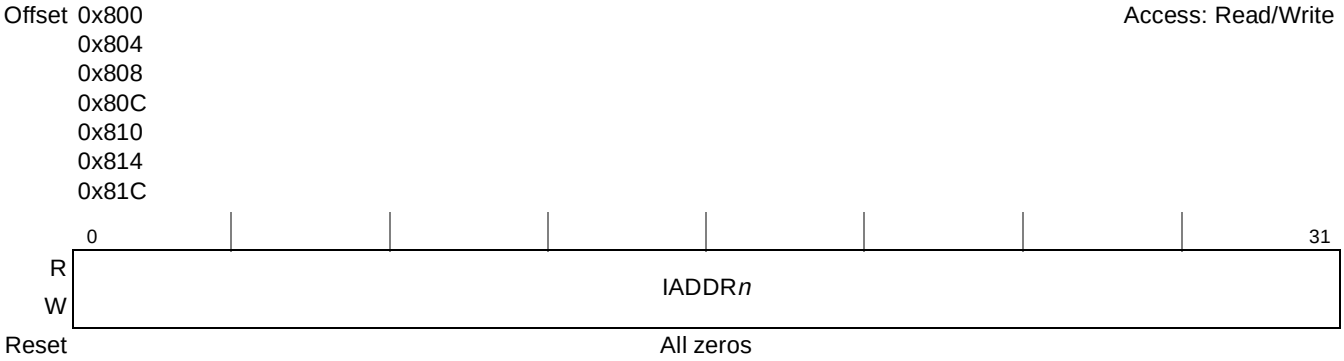


Figure 16-45. IADDR*n* Register Definition

Table 16-44 describes the field of the IADDR*n* registers.

Table 16-44. IADDR*n* Field Description

Bits	Name	Description
0–31	IADDR*n*	Represents the 32-bit value associated with the corresponding register. IADDR0 contains the high-order 32 bits of the 256-entry hash table and IADDR7 represents the low-order 32 bits. For instance, the MSB of IADDR0 correlates to entry 0 and the LSB of IADDR7 correlates to entry 255.

16.5.3.7.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers are written by the user. Together these registers represent 256 entries of the group (multicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 16-46 shows the GADDR*n* registers.

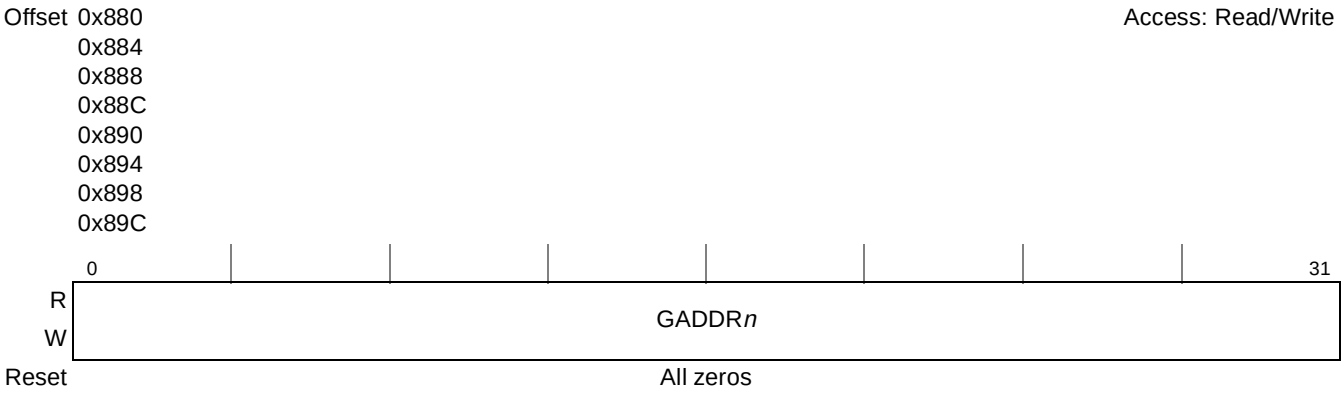


Figure 16-46. GADDR*n* Register Definition

Table 16-45 describes GADDR*n*.

Table 16-45. GADDR*n* Field Descriptions

Bits	Name	Description
0–31	GADDR*n*	Represents the 32-bit value associated with the corresponding register. GADDR0 contains the high-order 32 bits of the 256 entry group hash table and GADDR7 represents the low-order 32 bits. For instance, the MSB of GADDR0 correlates to entry 0 and the LSB of GADDR7 correlates to entry 255.

16.5.3.8 Attribute Registers

This section describes the two FEC frame attribute registers.

16.5.3.8.1 Attribute Register (ATTR)

The attribute register, shown in Figure 16-47, defines attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data. Buffer descriptors may be written with attributes that cause allocation into the L2 cache with or without line locking. Similarly, sections of a receive frame header may have attributes attached that cause allocation and locking in the L2 cache. This process of specifying a region of each frame to stash into the L2 cache is referred to as extraction, which is specified in conjunction with ATTRELI. ATTR[ELCWT] only has meaning if ATTRELI[EL] is non-zero.

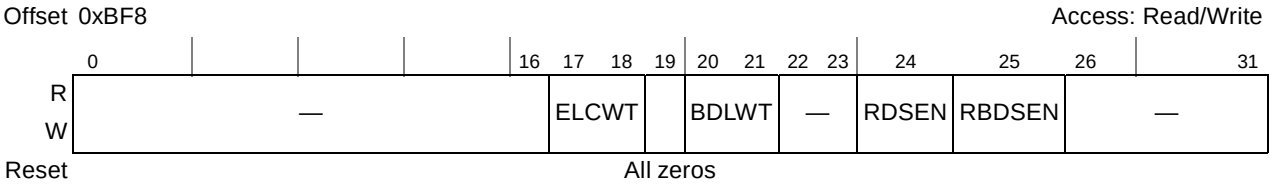


Figure 16-47. ATTR Register Definition

Table 16-46 describes the fields of the ATTR register.

Table 16-46. ATTR Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Specifies the write transaction type to perform for the extracted data. This occurs only if ATTRELI[EL] is non-zero. For maximum performance, it is recommended that if ELCWT is set to allocate, BDLWT must also be set to allocate. Writes to cache are always performed with snoop. This setting overrides the RDSEN bit setting. 00 No allocation performed 01 Reserved, no extraction occurs 10 Allocate L2 cache line 11 Reserved
19	—	Reserved
20–21	BDLWT	Buffer descriptor L2 cache write type. Specifies the write transaction type to perform for the buffer descriptor for a receive frame. Writes to cache are always performed with snoop. 00 No allocation performed. This setting overrides the RBDSEN bit setting. 01 Reserved 10 Allocate L2 cache line 11 Reserved
22–23	—	Reserved
24	RDSEN	Rx data snoop enable. This bit is superseded by the ELCWT settings. 0 Disables snooping of all receive frames data to memory unless ELCWT specifies L2 allocation 1 Enables snooping of all receive frames data to memory
25	RBDSEN	RxBD snoop enable. This bit is superseded by the BDLWT settings. 0 Disables snooping of all receive BD memory accesses unless BDLWT specifies L2 allocation 1 Enables snooping of all receive BD memory accesses
26–31	—	Reserved

16.5.3.8.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length. Figure 16-48 shows the ATTRELI register.

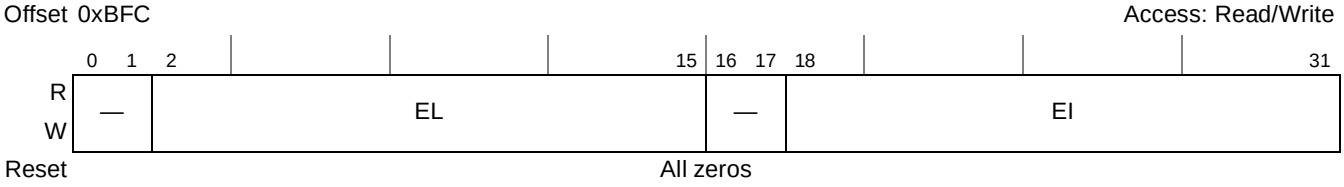


Figure 16-48. ATTRELI Register Definition

Table 16-47 describes the fields of the ATTRELI register.

Table 16-47. ATTRELI Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–15	EL	Extracted length. Specifies the number of bytes to extract from the receive frame. The DMA controller uses this field to perform extraction. If set to zero, no extraction is performed.
16–17	—	Reserved
18–31	EI	Extracted index. Points to the first byte within the receive frame from which to begin extracting data

16.6 Functional Description

This section describes many of the functions of the FEC controller.

16.6.1 Connecting to Physical Interfaces

This section describes how to connect the FEC to the MII interface. The bus follows the bus conventions used in the IEEE 802.3 specification because each PHY follows the same convention. (For instance, in the bus FEC_TXD[3:0], bit 3 is the MSB and bit 0 is the LSB.)

16.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the FEC. Figure 16-49 shows the basic components of the MII, including the signals required to establish FEC module connection with a PHY.

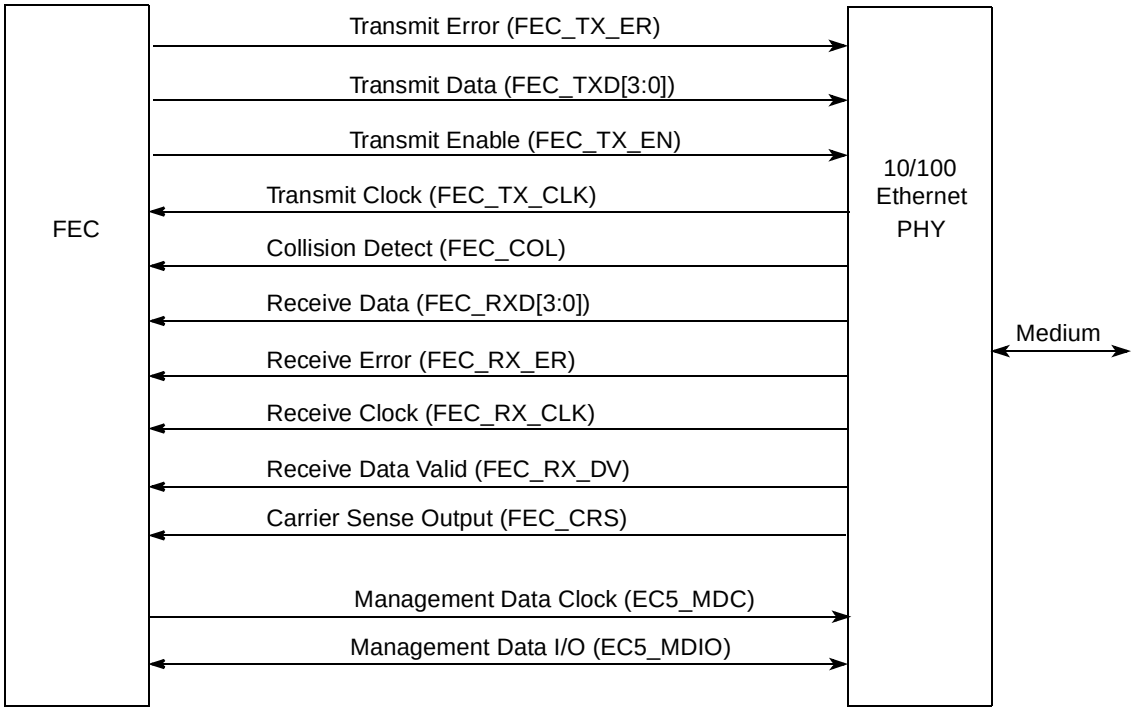


Figure 16-49. FEC-MII Connection

An MII interface has 18 signals (including the EC5_MDC and EC5_MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

Table 16-48 describes the MII interface signals.

Table 16-48. MII Interface Signals

Frequency [MHz] 25 Voltage [V] 3.3					
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
FEC_TX_CLK	I	1	FEC_RxD[0]	I	1
FEC_TxD[0]	O	1	FEC_RxD[1]	I	1
FEC_TxD[1]	O	1	FEC_RxD[2]	I	1
FEC_TxD[2]	O	1	FEC_RxD[3]	I	1
FEC_TxD[3]	O	1	FEC_RX_DV	I	1
FEC_TX_EN	O	1	FEC_RX_ER	I	1
FEC_TX_ER	O	1	FEC_COL	I	1
FEC_RX_CLK	I	1	FEC_CRS	I	1
EC5_MDIO	I/O	1	EC5_MDC	O	1

16.6.2 FEC Channel Operation

This section describes the operation of the FEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Address recognition and hash table algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

16.6.2.1 Initialization Sequence

This section describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the FEC.

16.6.2.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All FEC registers and control logic are reset to their default states after a hard reset has occurred.

16.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic FEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. The module memory map in Table 16-3 lists all the FEC registers. Table 16-49 describes the minimum steps for register initialization.

Table 16-49. Steps of Minimum Register Initialization

Description
1. Set, then clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII mgmt Interface
5. Clear IEVENT
6. Initialize IMASK
7. Initialize IADDR*n*
8. Initialize GADDR*n*
9. Initialize RCTRL
10. Initialize DMACTRL

After the registers are initialized, the user must execute the following steps in the order described below to bring the FEC into a functional state (out of reset):

1. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE register. A minimum of two buffer descriptors per ring is required. Setting the ring to a size of one causes the same frame to be transmitted twice.
2. Likewise, for the reception of Ethernet frames, the receive queue must be ready, with its RxBD pointed to by the RBASE register. Both transmit and receive can be gracefully stopped after transmission and reception begins.
3. Write to MACCFG1 and set the appropriate bits. These need to include Rx_EN and Tx_EN. To enable flow control, Rx_Flow and Tx_Flow must also be set.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. DMACTRL[GRS] must be cleared if the receiver had been previously stopped. See Section 16.5.3.1.6, “DMA Control Register (DMACTRL),” and Section 16.6.3.1, “Transmit Data Buffer Descriptor (TxBD),” for more information.

16.6.2.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft reset to and/or reconfiguring the MAC with new parameters, user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors are used, the user must load the pointers into the TBASE register. Likewise if a new set of Rx buffer descriptors are used, the RBASE register must be written with the new pointer.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GTS bit in DMACTRL register
2. Poll GTSC bit in IEVENT register until detected as set
3. Clear both Rx_EN and Tx_EN bits in MACCFG1
4. Wait for a period of 1.5 Kbytes worth of data on the interface.
5. Set GRS bit in DMACTRL register
6. Poll GRSC bit in IEVENT register until detected as set
7. Set Soft_Reset bit in MACCFG1 register
8. Clear Soft_Reset bit in MACCFG1 register
9. Load TBASE with new TxBD pointer
10. Load RBASE with new RxBD pointer
11. Set up other MAC registers (MACCFG2, MAXFRM, etc.)
12. Set WWR and WOP bits in DMACTRL register
13. Clear THLT bit in TSTAT register and QHLT bit in RSTAT register by writing 1 to these bits.
14. Clear GRS/GTS bits in DMACTRL (do not change other bits)
15. Enable Tx_EN/Rx_EN in MACCFG1 register

16.6.2.3 FEC Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the FEC begins to poll the first transmit buffer descriptor (TxBD) in the TxBD ring every 512 transmit clocks. If TxBD[R] is set, FEC begins moving transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the MII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again.

There is one internal buffer for out-of-sequence flow control frames. While the FEC is between frames, this buffer is polled. The buffer must contain the whole frame. Once the FEC is in paused mode, the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC regards it as a regular TxBD.

In half-duplex mode (MACCFG2[Full Duplex] is cleared) the MAC defers transmission if the line is busy (CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after CRS originally became negated). If CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is re-transmitted in case of a collision. This improves bus usage and latency.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap must be enforced.

The transmit block also implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter's request to send data until the pause duration is over. If the MAC is currently sending data when a pause frame is received, the MAC finishes sending the current frame, then suspends subsequent frames (except a pause frame) until the pause duration is over. The pause duration is defined by the received pause control frame and begins when the frame was first received. In addition, the transmitter supports transmission of flow control frames through TCTRL[TFC_PAUSE]. The transmit pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC EN] is set

Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes.

To pause transmission or rearrange the transmit queue, set DMACTRL[GTS]. This can help in transmitting expedited data ahead of previously linked buffers or for error situations. If GTS is set, the FEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with an error. The IEVENT[GTSC] interrupt occurs once the graceful transmit stop operation is completed. After GTS is cleared, the FEC resumes transmission with the next frame.

The FEC sends bytes least-significant nibble first and each nibble is sent lsb first.

16.6.2.4 FEC Frame Reception

The FEC Ethernet receiver is designed to work with little core intervention and can perform address recognition, CRC checking, short frame checking, and maximum frame-length checking. The receiver can also force frame headers and buffer descriptors to be allocated into the L2 cache. See Section 16.6.4, "Data Extraction to the L2 Cache," for additional information.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. If FEC_RX_DV is asserted

and FEC_COL remains negated, the MAC strips a valid preamble/SFD (start of frame delimiter) header and begins processing the frame. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the FEC controller begins to perform the frame recognition function through destination address (DA) recognition (See Section 16.6.2.5, “Frame Recognition,” for additional information.). Based on this match the frame can be accepted or rejected. Once accepted, the FEC processes the frame based on user-defined attributes.

The receiver can also filter frames based on physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame recognition algorithm is complete, system bus usage is not wasted on frames unwanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from the queue. If RxBD is not being used by software (RxBD[E] is set), the FEC starts transferring the incoming frame. RxBD[F] is set for the first RxBD used for any particular receive frame.

If the buffer is filled, the FEC clears RxBD[E] and, if RxBD[I] is set, generates an interrupt. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table. If it is empty, the controller continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicate LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBL). The smallest valid value is 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

Receive frames are not truncated if they exceed maximum frame bytes in the MAC’s maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the FEC receiver performs a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signalled after the graceful receive stop operation is completed. While in this mode the user can then clear IEVENT[GRSC] and can write to registers that are accessible to both the user and the FEC hardware without fear of conflict. After DMACTRL[GRS] is cleared, the FEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

16.6.2.5 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

16.6.2.5.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods.

Figure 16-50 is a flowchart for address recognition on received frames that is used to explain the concept. In the actual implementation most of the decision points shown in the figure actually occur simultaneously.

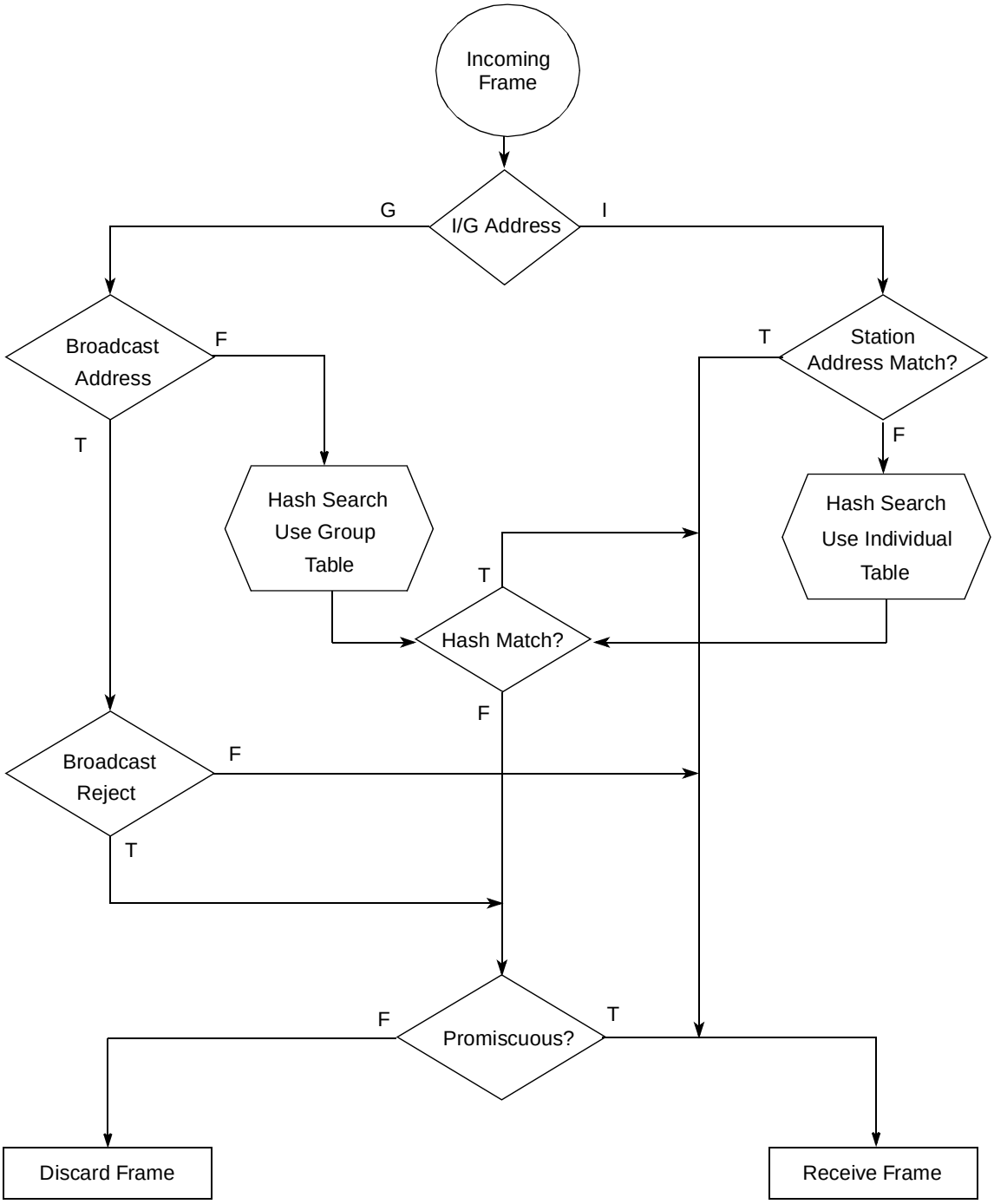


Figure 16-50. Ethernet Address Recognition Flowchart

The Ethernet controller compares the destination address field of the received frame with the physical address the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, then the controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. The user must write zeros to the hash in order to avoid

a hash match, and ones to station address in order to avoid individual address match, or the user can turn on promiscuous mode. (See Section 16.5.3.4.1, “Receive Control Register (RCTRL)”).

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast, and broadcast addresses are enabled, the frame is accepted. If the group address is not a broadcast address, the user can perform address recognition on multiple group addresses using the GADDR*n* hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address.

16.6.2.5.2 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in GADDR0–7 or IADDR0–7. The eight high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table. The high-order three bits of this 8-bit field are used to select one of the eight registers in either the individual or group hash table. The low-order five bits select a bit within the 32-bit register. A value of 0 in the high-order three bits selects IADDR0/GADDR0.

The same process is used if the Ethernet controller receives a frame. If the CRC checksum selects a bit that is set in the group/individual hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 224/256 (87.5%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if 32 group and 32 physical addresses are stored in their respective hash tables, because 87.5% of all group addresses and 87.5% of all individual address are rejected, then 87.5% of all frames are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 256-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

16.6.2.5.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of FEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value: x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+x0 or 0x04c11db7.

Given a destination MAC address of DA=01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

16.6.2.6 Flow Control

Because collisions cannot occur in full-duplex mode, the FEC can operate at the maximum rate. If the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 16-50 lists the flow-control frame structure.

Table 16-50. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble	—	—
1	SFD	—	Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address	—	—
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter	—	Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed-independent constant of 512 bit-times (unlike slot time). The most-significant octet is sent first.

Table 16-50. Flow Control Frame Structure (continued)

Size [Octets]	Description	Value	Comment
2	Extended MAC parameter	—	Extended pause control parameter as defined by the PTV[PTE] field. The most-significant octet is sent first.
40	Reserved	—	—
4	FCS	—	Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. During this pause, only a control frame can be sent (TCTRL[TFC_PAUSE] is set). Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

16.6.2.7 Interrupt Handling

The following describes what usually occurs within a FEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time.
- Process the TxBDs to reuse them if the IEVENT[TXB or TXF] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the FEC; thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set. See Table 16-51.
- Obtain data from the RxBD if IEVENT[RXC,RXB or RXF] is set. If the receive speed is fast or the interrupt delay is long, the FEC may have received more than one RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the FEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 16-52.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 16-51. Non-Error Transmit Interrupts

Interrupt	Description	Action taken by FEC
GTSC	Graceful transmit stop complete: transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None

Table 16-51. Non-Error Transmit Interrupts (continued)

Interrupt	Description	Action taken by FEC
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, was updated.	Programmable ‘write with response’ TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame was transmitted and the last transmit buffer descriptor (TxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last TxBD before setting IEVENT[TXF].

Table 16-52. Non-Error Receive Interrupts

Interrupt	Description	Action taken by FEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was first received.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated.	Programmable ‘write with response’ RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable ‘write with response’ to memory on the last RxBD before setting IEVENT[RXF].

16.6.2.8 Inter-Packet Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). After a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the packet is not successfully sent within a specified number of retries, an error is indicated.

The minimum inter-packet gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back packets with this minimum spacing. In addition, after waiting a required number of clocks (based on the back-off algorithm), the transmitter waits for carrier sense to be negated before retransmitting the packet. Retransmission begins 36 serial clocks after carrier sense is negated for at least 60 serial clocks.

16.6.2.9 Internal and External Loop Back

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default.

16.6.2.10 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Programming note: When the FEC encounters a halt condition (TSTAT[THLT] is set), it stops processing the frame at the current TxBD. The FEC relies on the user to manage the buffer descriptor pointer, TBPTR, or the buffer descriptor queue before resuming transmissions. Once the FEC resumes, it fetches the TxBD pointed to by TBPTR.

Transmission errors are described in Table 16-53.

Table 16-53. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, sets TxBD[UN], closes the buffer, sets IEVENT[XFUN] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, sets IEVENT[CRL/XDA] and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Excessive defer abort	The controller terminates buffer transmission, sets TxBD[DEF], closes the buffer, sets IEVENT[CRC/XDA], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared.
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, sets IEVENT[LC] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Babbling transmit error	A frame is transmitted which exceeds the MAC's maximum frame length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame.

Reception errors are described in Table 16-54.

Table 16-54. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, and sets IEVENT[RXF], The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY]. In addition, the RSTAT[QHLT] bit is set. The halted queue resumes reception once the user clears the RSTAT[QHLT] bit.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The FEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.

Table 16-54. Reception Errors (continued)

Error	Description
CRC error	If a CRC error occurs, the controller sets RxBD[CR], closes the buffer, and sets IEVENT[RXF]. This FEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory read error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR] and discards the frame. In addition the RSTAT[QHLT] bit is set. The halted queue resumes reception once the RSTAT[QHLT] bit is cleared.
Babbling receive error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues

16.6.3 Buffer Descriptors

The FEC buffer descriptor (BD) is modeled after the MPC8540 Fast Ethernet controller BD for ease of reuse. Drawing from the MPC8540 FEC BD programming model, the FEC descriptor base registers point to the beginning of BD rings. The 8-byte data BD format is similar to the MPC8540 BD model.

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 16-51). Data BDs encapsulate all information necessary for the FEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of pre-fetching, a minimum of two buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings. Software also must have the data pointer pointing to memory. Within the status field, there exists an ownership bit which defines the current state of the buffer (pointed to by the data pointer). Other bits in the status field of the buffer descriptor are used to communicate status/control information between the FEC and the software driver.

The status field of the BD is 16-bit field, as is the length field. The data buffer pointer is a 32-bit field. Therefore, the BDs should be accessed with the following C structure:

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct bd_struct {
         uint_16 flags;
         uint_16 length;
         uint_32 bufptr;
};
```

Because there is no next BD pointer in the transmit/receive BD (see Figure 16-52), all BDs must reside sequentially in memory. The FEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the FEC to loop back to the beginning of the BD chain. Software must initialize TBASE and RBASE that point to the beginning transmit and receive BDs for FEC.

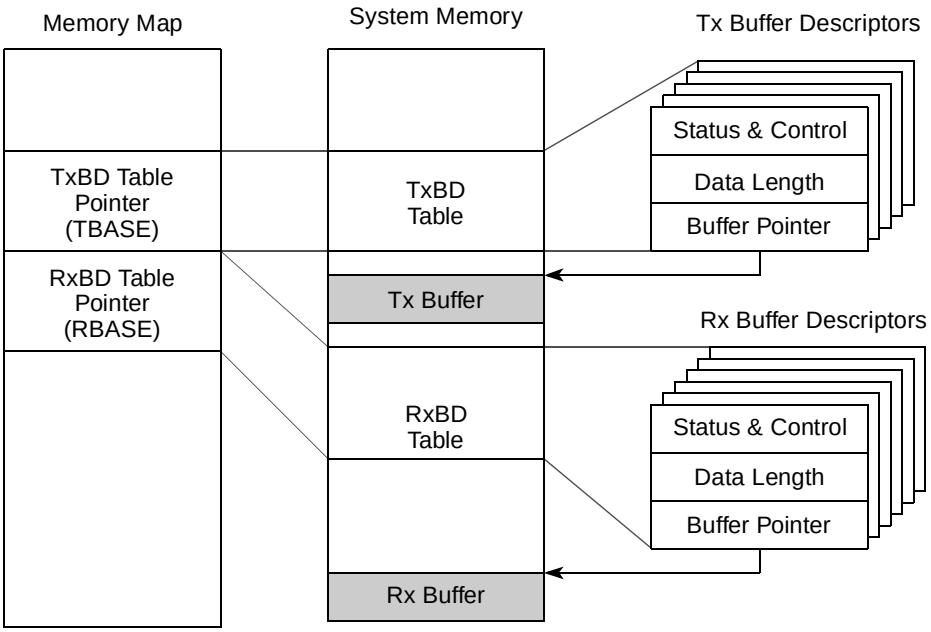


Figure 16-51. Example of FEC Memory Structure for BD

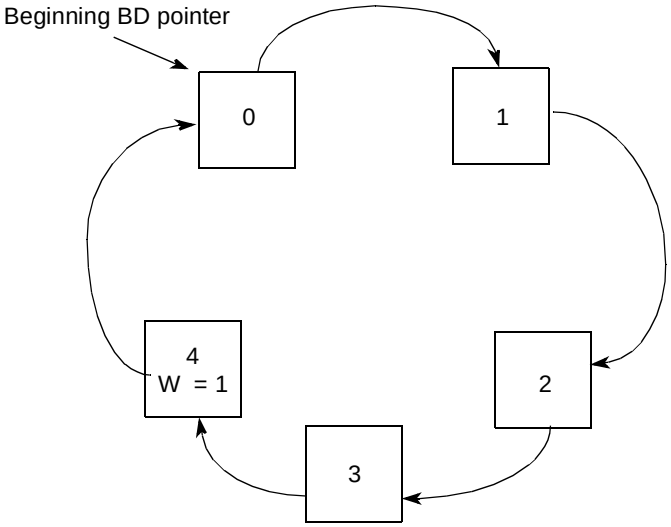


Figure 16-52. Buffer Descriptor Ring

16.6.3.1 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD/CRC, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The FEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block. Figure 16-53 shows the TxBD.

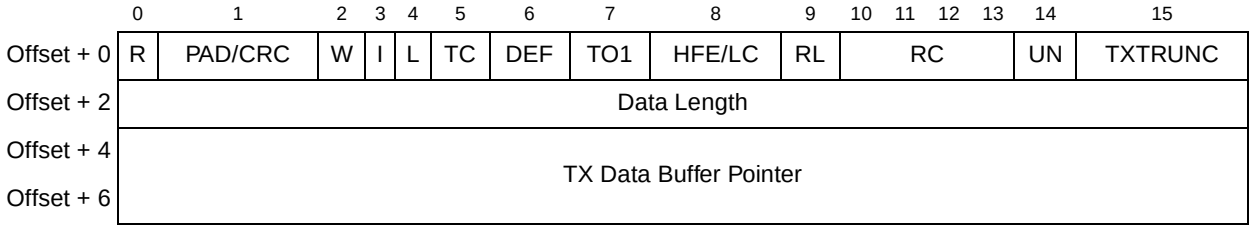


Figure 16-53. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 16-55.

Table 16-55. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	R	Ready. Written by FEC and user 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
Offset + 0	1	PAD/ CRC	PAD/CRC. Padding and CRC attachment for frames. (Valid only while it is set in the first BD and MACCFG2[PAD/CRC] is cleared.) If MACCFG2[PAD/CRC] is set, this bit is ignored. 0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. FEC PADs always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
Offset + 0	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
Offset + 0	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).
Offset + 0	4	L	Last in frame. Written by user 0 The buffer is not the last in the transmit frame. 1 The buffer is the last in the transmit frame.
Offset + 0	5	TC	Tx CRC. Written by user. (Valid only while it is set in first BD and TxBD[PAD/CRC] is cleared and MACCFG2[PAD/CRC] is cleared and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, this bit is ignored. 0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.

Table 16-55. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	6	DEF	Defer indication. Hardware updates this bit if an excessive defer condition occurs. 0 This frame was not deferred. 1 If HAFDUP[EXCESS_DEFER]=1, this frame did not have a collision before it was sent but it was sent late because of deferring. If HAFDUP[EXCESS_DEFER]=0, this frame was aborted and not sent.
Offset + 0	7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.
Offset + 0	8	HFE/LC	Huge frame enable (written by user)/Late collision (written by FEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's maximum frame length register. 1 Do not truncate the transmit frame. Late collision. Written by FEC 0 No late collision 1 A collision occurred after 64 bytes are sent. The FEC terminates the transmission and updates LC.
Offset + 0	9	RL	Retransmission Limit. Written by FEC 0 Transmission before maximum retry limit is hit 1 The transmitter failed (max. retry limit + 1) attempts to successfully send a message due to repeated collisions. The FEC terminates the transmission and updates RL.
Offset + 0	10–13	RC	Retry Count. Written by FEC 0 The frame is sent correctly the first time or if RL is set then the retry limit has been reached x One or more attempts were needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
Offset + 0	14	UN	Underrun. Written by FEC 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The FEC terminates the transmission and updates UN.
Offset + 0	15	TXTRUNC	TX truncation. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for the frame
Offset + 2	0–15	Data Length	Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. This field must be greater than zero.
Offset + 4	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

16.6.3.2 Receive Buffer Descriptor (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the FEC modifies the E, L, F, M, BC, MC, LG, NO, SH, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, SH, CR, OV, and TR bits in the first word of the buffer descriptor are only modified by the FEC if the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed in Figure 16-54 below.

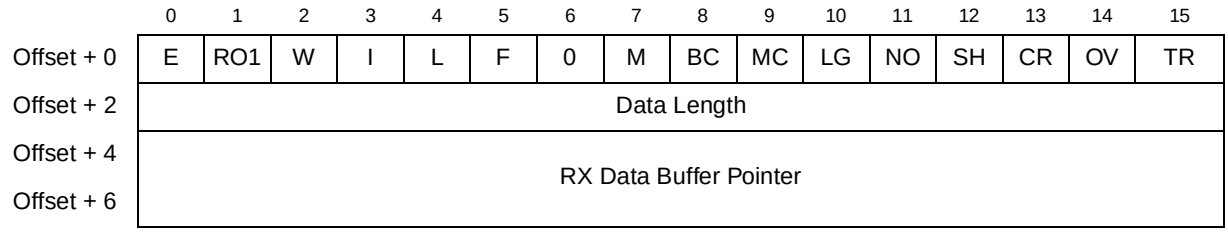


Figure 16-54. Receive Buffer Descriptor

Table 16-56 describes the fields of the RxBD.

Table 16-56. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty. Written by FEC (when cleared) and by user (when set) 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1	RO1	Receive software ownership bit. Reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	2	W	Wrap. Written by user 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
Offset + 0	3	I	Interrupt. Written by user 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] is set or IMASK[RXFEN] is set). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
Offset + 0	4	L	Last in frame. Written by FEC 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5	F	First in frame. Written by FEC 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
Offset + 0	6	—	Reserved
Offset + 0	7	M	Miss. Written by FEC. (This bit is valid only if the L-bit is set and FEC is in promiscuous mode.) This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a ‘miss’ by the internal address recognition; thus, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	8	BC	Broadcast. Written by FEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
Offset + 0	9	MC	Multicast. Written by FEC. (Only valid if L is set.) Set if the DA is multicast and not BC

Table 16-56. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	10	LG	Rx frame length violation. Written by FEC. (Only valid if L is set.) A frame length greater than maximum frame length was recognized while MACCFG2[Huge Frame] was set. Note, if MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the Maximum Frame Length register.
Offset + 0	11	NO	Rx non-octet aligned frame. Written by FEC. (Only valid if L is set.) A frame that contained a number of bits not divisible by eight was received.
Offset + 0	12	SH	Short frame. Written by FEC. (only valid if L is set.) A frame length that was less than the minimum length defined for this channel (MINFLR) was recognized, provided RCTRL[RSF] is set.
Offset + 0	13	CR	Rx CRC error. Written by FEC. (Only valid if L is set.) This frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected.
Offset + 0	14	OV	Overrun. Written by FEC. (Only valid if L is set.) A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, and CR lose their normal meaning and are zero.
Offset + 0	15	TR	Truncation. Written by FEC. (Only valid if L is set.) Is set if the receive frame is truncated. This can happen if a frame length greater than maximum frame length was received and the MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	0–15	Data Length	Data length. Written by FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L is cleared (the value is equal to MRBL), or the length of the frame including CRC, if L is set.
Offset + 4	0–31	Rx Data Buffer Pointer	Receive buffer pointer. Written by user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 64-byte aligned. The buffer must reside in memory external to the FEC.

16.6.4 Data Extraction to the L2 Cache

Some applications require the ability to identify selected portions of data within a frame's data payload. This process is called extraction, although the data is not truly removed. Rather than literarily extracting a section of the data and copying it into a new memory location, the data is placed in the L2 cache. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory. This results in substantial improvement in throughput and hence reduction in latency.

Extraction functionality is controlled and configured with ATTR and ATTRELI. See Section 16.5.3.8.1, "Attribute Register (ATTR)," and Section 16.5.3.8.2, "Attribute Extract Length and Extract Index Register (ATTRELI)," for specific register information.

16.7 Initialization/Application Information

16.7.1 Interface Mode Configuration

This section describes how to configure the FEC in the MII mode. The pinout, the data registers that must be initialized, as well as speed selection options are described. See Section 4.4.3.10, “FEC Configuration,” for reset configuration information.

16.7.1.1 MII Interface Mode

Table 16-57 describes the signal configurations required for MII interface mode.

Table 16-57. MII Interface Mode Signal Configuration

MII Interface Frequency [MHz] 25 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**
TX_CLK	I	1
TxD[0]	O	1
TxD[1]	O	1
TxD[2]	O	1
TxD[3]	O	1
TX_EN	O	1
TX_ER	O	1
RX_CLK	I	1
RxD[0]	I	1
RxD[1]	I	1
RxD[2]	I	1
RxD[3]	I	1
RX_DV	I	1
RX_ER	I	1
COL	I	1
CRS	I	1
EC5_MDIO	I/O	1
EC5_MDC	I	1
Sum		**18**

Table 16-58 describes the register initializations required to reconfigure the PHY to MII mode following initial auto-negotiation.

Table 16-58. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half-duplex operation. Set I/F mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC station address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC station address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0000]
Setup the MII mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0101] set source clock divide by 14, for example, to insure that EC5_MDC clock speed is approximately 2.5 MHz.
Read MII MII mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the FEC MII mgmt bus is idle.
Set up the MII mgmt for a write cycle to the external PHY auxiliary control and status register to configure the PHY through the management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII mgmt write cycle to the external PHY Writing to MII mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII mgmt write is complete Read MII mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII mgmt for a write cycle to the external PHY extended PHY control register 1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII mgmt write cycle to the external PHY. Write to MII mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]

Table 16-58. MII Mode Register Initialization Steps (continued)

Check to see if MII mgmt write is complete. Read MII mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII mgmt for a write cycle to the external PHY mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII mgmt write cycle to the external PHY. Write to MII mgmt control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII mgmt write is complete Read MII mgmt indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed auto-negotiation. Set up the MII mgmt for a read cycle to PHY MII mgmt register (write the PHY address and register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY status register is at address 0x1 and in this case the PHY address is 0x00.
Perform an MII mgmt read cycle of status register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done and Link is up) MIIMSTAT ---> [0000_0000_0000_0000_0000_000_0010_0100] Other information about the link is also returned (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability).
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK, (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n,* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n,* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL, (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL, (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (empty) transmit descriptor ring and fill buffers with data. Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]

Table 16-58. MII Mode Register Initialization Steps (continued)

Initialize (empty) receive descriptor ring and fill with empty buffers. Initialize RBASE, RBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
Enable Rx and Tx, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]

Chapter 17
Pattern Matcher (PM) 1.1

17.1 Preface

17.1.1 Conventions

Table 17-1 provides a key for register figures and tables.

Table 17-1. Register Conventions

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

17.1.1.1 Glossary

Table 17-2. Glossary

Term	Definition
Buffer	A temporary physically contiguous memory area for holding data.
Deflate	Lossless compressed data format that compresses data using a combination of the LZ77 algorithm and Huffman coding. Deflate is specified in RFC 1951 - DEFLATE Compressed Data Format Specification version 1.3.
Flow	An instance of uni-directional communication between two end points.
Pattern	An arrangement of symbols to be matched against an input string. The input string is referred in this document as SUI.
Pattern Set	An exclusive grouping (no set overlap) of patterns that are to be searched simultaneously by the Pattern Matcher. The Pattern Matcher supports 256 (mutually exclusive) pattern sets. When scanning for patterns in the data, the search is restricted to a particular pattern set.
Pattern Subset	An non-exclusive grouping (subset overlap permitted) of patterns within a given pattern set that are to be searched simultaneously by the Pattern Matcher with or without other subsets. Unlike a pattern set, patterns may be assigned to multiple pattern subsets.The Pattern Matcher supports 16 subsets per pattern set. When scanning for patterns, the search may use a list of subsets.
Regular Expression (regex)	A special text string for describing a pattern to be matched against an input string (referred in this document as SUI). It contains literal characters and meta characters representing alternation, grouping and quantification. Examples of regular expression syntax: Perl-compatible regular expressions, POSIX regular expressions. The Pattern Matcher is capable of matching an SUI against patterns constructed from regular expressions. The Pattern Matcher also provides the ability to express patterns with capabilities beyond that provided by regex languages.
Residue	Previously scanned data appended to the next data “unit” within a given stream in order to allow pattern matching across data “units” (for example, can match patterns split across TCP segments).
Session	A logical grouping of one or more input streams, typically a pair of flows representing a bi-directional network conversation. Stateful rules are applied on a per session basis.
Stateful Rule	Pattern Matcher capability for expressing complex relationships between matched patterns.
Stream	A stream represents input/output data to/from the Pattern Matcher. The input data typically corresponds to the data (ordered set of bytes or set of packets) of a flow to be acted on by the Pattern Matcher. The output data corresponds to the pattern matching reports and decompressed output data.
Stream Context	A collection of information associated with a stream. For the Patter Matcher, a stream context is maintained only for input streams where it contains necessary information for the Pattern Matcher to process any data belonging to the stream.
String Under Inspection (SUI)	The string of bytes to be searched by the Pattern Matcher. An SUI is composed of the contents of a work unit input data, prepended with the residue from the previous SUI on this data stream (if residue is enabled).
Work Unit	Represents atomic work request operation to the Pattern Matcher.

17.2 Introduction

The Pattern Matcher 1.1 IP block (henceforth referred to as Pattern Matcher) includes both pattern matching and decompression functionality in a single integrated block.

The pattern matching functionality includes the following:

- Ability to search input data against patterns constructed from regular expressions
- Support for patterns with significant capabilities beyond that provided by regex languages
- Ability to search input data for a large number of patterns simultaneously
- Ability to track stateful relationships between patterns
- DMA based “look-a-side” interface

The decompress functionality provides the ability to decompress data which has been compressed using the Deflate compression format before sending it for pattern matching.

17.2.1 Overview

The Pattern Matcher IP block contains the following functional units, shown in Figure 17-1.

- Memory Interface Arbiter—Provides access to system memory for the functional units within the Pattern Matcher allowing them to initiate read and write transactions over the system bus.
- Internal Register Interface—Provides access to all registers within the Pattern Matcher.
- DMA Engine—Offers four independent DMA channels which driver software uses to command pattern matching/decompression work and to receive notifications of completed pattern matching/decompression work.
- Free Buffer Manager—Manages pools of free buffers that are used by the DMA Engine for generating pattern matching reports and decompressed data outputs.
- Deflate Engine—Implements data decompression using the Deflate algorithm. Handles raw Deflate, as well as gzip and zlib format headers containing Deflate payloads. Allows input data from memory to be decompressed before being sent for pattern match scanning. Optionally, decompressed data can also be written out to memory (useful for forensics when a pattern match is found in the decompressed data).
- Key Element Scanner—Implements a multi-stage, hash based, proprietary algorithm to determine the likelihood of the data (here after referred as a String Under Inspection, or SUI) being a match with a pattern. Matches found at this stage are delivered to the Data Examination Engine which performs a more stringent comparison to determine if the match actually exists or not.
- Data Examination Engine—Implements a Non-deterministic Finite Automaton (NFA) (state machine) capable of implementing a significant subset of the regular expression (regex) pattern definition language. It also provides the ability to express patterns with capabilities beyond that provided by regex languages.
- Stateful Rule Engine—Provides a means to link multiple pattern matches together and maintains state between matches including information extracted from the data under inspection.

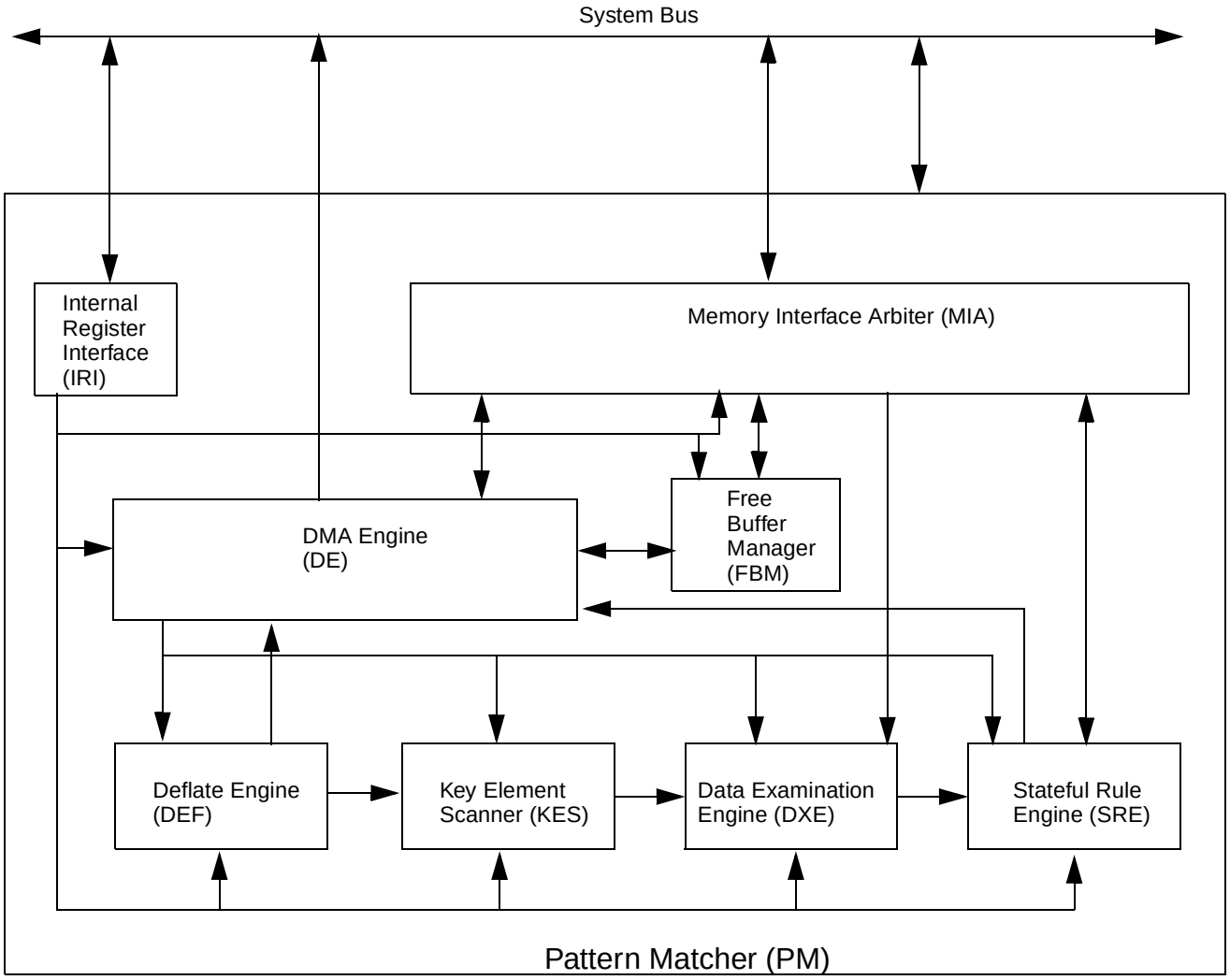


Figure 17-1. High Level Block Diagram of the PM

17.2.2 Features

The PM includes these distinctive features:

- High performance pattern matching for up to 16,000 patterns simultaneously.
- Improvements over other pattern matching technologies:
  - No pattern explosion to support wildcarding.
  - Allows for fast compilation of patterns
  - Fast incremental pattern additions
  - Patterns stored in main DDR DRAM, not SRAM or FCRAM
  - On-chip hash tables for low system memory utilization.
- Pattern matching capabilities:

  - Support for patterns expressed in regex with capabilities beyond that provided by the regex language.
  - The 16,000 patterns can be divided into 256 non-overlapping sets where each set is of variable size.
  - Within a set, patterns can be placed in one or more subsets (up to 16). A search is performed against one or more subsets.
  - Support for patterns with matched lengths between 1 and 128 bytes (for example, a pattern can’t match against more that 128 bytes in the SUI).
  - Pattern matching across data units (for example, can match patterns split across TCP segments)
- Means (called stateful rule) to link multiple pattern matches together and to maintain state between matches including information extracted from the data under inspection.
  - A maximum of 8192 stateful rules is supported.
  - A maximum of 128-Mbyte sessions is supported (with session context size of 32 bytes). The maximum number of sessions are lower than 128M if the session context size is higher than 32 bytes. Stateful rules are applied on a per session basis where a session represents a logical grouping of one or more flows.
- Implements data decompression using the Deflate algorithm.
  - Ability to perform data decompression before pattern matching without writing the decompressed data into system memory. Optionally, decompressed data can also be written out to system memory.
  - Provides the option to send decompressed data to system memory without sending the data for pattern matching.
  - Supports the standard 32KB deflate history memory size.
  - Supports the following formats:
    - Deflate format as specified in RFC 1951
    - gzip format as specified in RFC 1952
    - zlib compressed data format as specified in RFC 1950
- DMA based “look-a-side” interface
  - Offers four independent DMA channels which driver software uses to command pattern matching/decompression work and to receive notifications of completed pattern matching/decompression work.
  - Supports “gather buffer lists” for specifying input data and “scatter buffer lists” for specifying the location of output data.
  - In addition to scatter/gather lists it supports a linked list mechanism for connecting multiple buffers.
  - Provides up to eight hardware managed free pools of buffers which can be used for responses/reports/output data when no output buffers are specified with the input data.
- Incorporates a Memory Interface Arbiter that buffers and arbitrates memory access requests from the internal PM engines to system memory.
- 333-MHz input clock.

- Maximum data scan rate of 1 byte per cycle (2.66 Gbps @ 333 MHz).
- 5 separate interrupt outputs, one for each DMA channel and one interrupt output used for common control (non-channel related) interrupts.
- Pattern Matcher provides 32 events that can be counted by the central Performance Monitor.

17.3 Memory Map and Register Definition

Table 17-3 shows the overall register memory map of the PM. The PM occupies 20 Kbytes in the configuration, control, and status register (CCSR) space of the system. CCSR space is a fixed 1 Mbyte in size, and all CCSR registers in the system are accessed using a 20-bit address offset from CCSRBAR. The address offsets shown in Table 17-3 refer to the full 20-bit address offset from CCSRBAR, the lower 15 bits of which are passed to PM and used to address the individual blocks and registers within the PM. The PM is always mapped onto a 32-Kbyte aligned boundary in CCSR space.

On an e500 platform the minimum MMU page size is 4 Kbytes, therefore each set of per-channel control registers is mapped into a separate 4-Kbyte space, such that each channel's registers can be individually protected. The remaining registers which are common to the entire PM block (for example, not associated with a channel) are mapped in 128-byte increments divided between the various internal blocks of the PM, and placed in the first 4-Kbyte block of PM's register space.

Table 17-3. Pattern Matcher Overall Register Memory Map

Address Offset	Size	Module	Description
0x1_0000 to 0x1_007F	128 B	DMA Engine	DMA Engine common control registers
0x1_0080 to 0x1_00FF	128 B	Key Element Scanner	Control registers
0x1_0100 to 0x1_017F	128 B	Data Examination Engine	Control registers
0x1_0180 to 0x1_01FF	128 B	Stateful Rule Engine	Control registers
0x1_0200 to 0x1_027F	128 B	Deflate Engine	Control registers
0x1_0280 to 0x1_02FF	128 B	Free Block Manager	Free Buffer Manager common control registers
0x1_0300 to 0x1_0B7F	2176 B	—	Not used
0x1_0B80 to 0x1_0BFF	128 B	Memory Interface Arbiter	Control registers
0x1_0C00 to 0x1_0FFF	1 KB	—	Not used
0x1_1000 to 0x1_13FF	1 KB	DMA Engine	Channel 0 control registers
0x1_1400 to 0x1_17FF	1KB	Free Block Manager	Channel 0 control registers
0x1_1800 to 0x1_1FFF	2 KB	—	Not used
0x1_2000 to 0x1_23FF	1 KB	DMA Engine	Channel 1 control registers
0x1_2400 to 0x1_27FF	1KB	Free Block Manager	Channel 1 control registers
0x1_2800 to 0x1_2FFF	2 KB	—	Not used

Table 17-3. Pattern Matcher Overall Register Memory Map (continued)

Address Offset	Size	Module	Description
0x1_3000 to 0x1_33FF	1 KB	DMA Engine	Channel 2 control registers
0x1_3400 to 0x1_37FF	1KB	Free Block Manager	Channel 2 control registers
0x1_3800 to 0x1_3FFF	2 KB	—	Not used
0x1_4000 to 0x1_43FF	1 KB	DMA Engine	Channel 3 control registers
0x1_4400 to 0x1_47FF	1KB	Free Block Manager	Channel 3 control registers
0x1_4800 to 0x1_4FFF	2 KB	—	Not used

Table 17-4 provides the memory map for each register within the PM.

Table 17-4. PM Individual Register Memory Map

Offset	Register	Access	Reset Value	Section/Page
DMA Engine—Common Control Registers				
0x1_0000	ISRCC—Interrupt Status Register Common Control	w1c	0x0000_0000	17.3.1.1/17-14
0x1_0004	IERCC—Interrupt Enable Register Common Control	R/W	0x0000_0000	17.3.1.2/17-16
0x1_0008	ISDRCC—Interrupt Status Disable Register Common Control	R/W	0x0000_0000	17.3.1.3/17-17
0x1_000C	IIRCC—Interrupt Inhibit Register Common Control	R/W	0x0000_0000	17.3.1.4/17-18
0x1_0010	Reserved	—	—	—
0x1_0014	SCOS—Statistics Counters Overflow Status Register	w1c	0x0000_0000	17.3.1.5/17-19
0x1_0018	Reserved	—	—	—
0x1_001C	PERFT—Performance Monitor Threshold Register	R/W	0x0000_0000	17.3.1.6/17-20
0x1_0020	Reserved	—	—	—
0x1_0024	CSCR—Channel Scheduling Control Register	R/W	0x0000_0000	17.3.1.7/17-21
0x1_0028-0x1_007C	Reserved	—	—	—
KES Control Registers				
0x1_0080	STNIB—Statistic: Number of Input Bytes Scanned for Pattern Matching	RR	0x0000_0000	17.3.2.1/17-22
0x1_0084	STNIS—Statistic: Number of Input SUIs	RR	0x0000_0000	17.3.2.2/17-23
0x1_0088	STNTH1—Statistic: Number of Trigger Hits Against 1-Byte Trigger	RR	0x0000_0000	17.3.2.3/17-23
0x1_008C	STNTH2—Statistic: Number of Trigger Hits Against 2-Byte Trigger	RR	0x0000_0000	17.3.2.4/17-24
0x1_0090	STNTHL—Statistic: Number of Trigger Hits Against Variable Length Trigger	RR	0x0000	17.3.2.5/17-24
0x1_0094	STNTHS—Statistic: Number of Trigger Hits Against Special Triggers	RR	0x0000	17.3.2.7/17-25
0x1_0098	STNCH—Statistic: Number of Confidence Stage Hits	RR	0x0000	17.3.2.7/17-25

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_009C	SWDB—Software Database Version Reg/Scratch Pad	R/W	0x0000_0000	17.3.2.8/17-26
0x1_00A0	KVLTS—KES Variable Length Trigger Size	R/W	0x0000_0007	17.3.2.9/17-26
0x1_00A4	KEC—KES Error Configuration	R/W	0x0000_0000	17.3.2.10/17-27
0x1_00A8	PEHE—PME Error Handling Enables	R/W	0x0000_0000	17.3.2.11/17-27
DXE Control Registers				
0x1_0100	STNPM—Statistic: Number of Pattern Matches	RR	0x0000	17.3.3.1/17-29
0x1_0104	STNS1M—Statistic: Number of SUI with at Least 1 Pattern Match	RR	0x0000	17.3.3.2/17-29
0x1_0108	DRCC—DXE Pattern Range Counter Configuration	R/W	0x0000_0000	17.3.3.3/17-30
0x1_010C	STNPMR—Statistic: Number of Pattern Matches Within Range of DRCC	RR	0x0000	17.3.3.4/17-31
0x1_0120	PDSRBAH—Pattern Description and Stateful Rule Base Address Register High	R/W	0x0000_0000	17.3.3.5/17-31
0x1_0124	PDSRBAL—Pattern Description and Stateful Rule Base Address Register Low	R/W	0x0000_0000	17.3.3.5/17-31
0x1_0128	DMCR—DXE Memory Control Register	R/W	0x0000_0000	17.3.3.6/17-32
0x1_012C	DEC—DXE Error Configuration	R/W	0x0000_0000	17.3.3.7/17-33
SRE Control Registers				
0x1_0180	STNDSR-—Statistic: Number of DXE Generated SR Executions	RR	0x0000	17.3.4.1/17-34
0x1_0184	STNESR-—Statistic: Number of End of SUI generated SR executions	RR	0x0000	17.3.4.2/17-34
0x1_0188	STNS1R-—Statistic: Number of SUI with at Least 1 Report	RR	0x0000	17.3.4.3/17-35
0x1_018C	STNOB-—Statistic: Number of Output Bytes Produced in Reports	RR	0x0000_0000	17.3.4.4/17-35
0x1_01A8	SCBARH—SRE Context Base Address Register High	R/W	0x0000_0000	17.3.4.5/17-36
0x1_01AC	SCBARL—SRE Context Base Address Register Low	R/W	0x0000_0000	17.3.4.5/17-36
0x1_01B0	SMCR—SRE Memory Control Register	R/W	0x0000_0000	17.3.4.6/17-37
0x1_01B4	SREC—SRE Configuration	R/W	0x0000_0000	17.3.4.7/17-38
0x1_01B8	SRRV0—SRE Rule Reset Vector 0	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01BC	SRRV1—SRE Rule Reset Vector 1	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C0	SRRV2—SRE Rule Reset Vector 2	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C4	SRRV3—SRE Rule Reset Vector 3	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01C8	SRRV4—SRE Rule Reset Vector 4	R/W	0x0000_0000	17.3.4.8/17-39

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_01CC	SRRV5—SRE Rule Reset Vector 5	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D0	SRRV6—SRE Rule Reset Vector 6	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D4	SRRV7—SRE Rule Reset Vector 7	R/W	0x0000_0000	17.3.4.8/17-39
0x1_01D8	SRRFI—SRE Rule Reset First Index	R/W	0x0000_0000	17.3.4.9/17-39
0x1_01DC	SRRI—SRE Rule Reset 32-byte Increment	R/W	0x0000_0000	17.3.4.10/17-40
0x1_01E0	SRRR—SRE Rule Reset Repetitions	R/W	0x0000_0000	17.3.4.11/17-40
0x1_01E4	SRRWC—SRE Rule Reset Work Configuration	R/W	0x0000_0000	17.3.4.12/17-41
0x1_01E8	SFRCC—SRE Free Running Counter Configuration	R/W	0x0000_0000	17.3.4.13/17-42
0x1_01EC	SEC1—SRE Error Configuration 1	R/W	0x0000_0000	17.3.4.14/17-43
0x1_01F0	SEC2—SRE Error Configuration 2	R/W	0x0000_0000	17.3.4.15/17-43
0x1_01F4	SEC3—SRE Error Configuration 3	R/W	0x0000_0000	17.3.4.16/17-44
Deflate Control Registers				
0x1_0200	STDBC—Statistic: Deflate Bytes Consumed	RR	0x0000_0000	17.3.5.1/17-44
0x1_0204	STDBP—Statistic: Deflate Bytes Produced	RR	0x0000_0000	17.3.5.2/17-45
0x1_0208	STDWC—Statistic: Deflate Work Units Consumed	RR	0x0000	17.3.5.3/17-45
0x1_020C-0x1_0220	Reserved	—	—	—
0x1_0224	DBPPWL— Configuration: Deflate Bytes Produced Per Work Unit Limit	R/W	0x0000_0000	17.3.5.4/17-46
FBM Common Control Registers				
0x1_0280-0x1_029C	Reserved	—	—	—
0x1_02A0	FBL0SIZE—Free Buffer List 0 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02A4	FBL1SIZE—Free Buffer List 1 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02A8	FBL2SIZE—Free Buffer List 2 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02AC	FBL3SIZE—Free Buffer List 3 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B0	FBL4SIZE—Free Buffer List 4 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B4	FBL5SIZE—Free Buffer List 5 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02B8	FBL6SIZE—Free Buffer List 6 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02BC	FBL7SIZE—Free Buffer List 7 Size Register	R/W	0x0000_0000	17.3.6.1/17-46
0x1_02C0	FBLAAR—Free Buffer List Assignment A Register	R/W	0x0000_0000	17.3.6.2/17-47

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_02C4	FBLABR—Free Buffer List Assignment B Register	R/W	0x0000_0000	17.3.6.3/17-48
0x1_02C8-0x1_02DC	Reserved	—	—	—
0x1_02E0	FBM_CR	R/W	0x0000_0000	17.3.6.4/17-50
0x1_02E4-0x1_02FF	Reserved	—	—	—
Memory Interface Arbiter Registers				
0x1_0B80	MIA_BYC—MIA Byte Count Register	RR	0x0000_0000	17.3.7.1/17-51
0x1_0B84	MIA_BLC—MIA Block Count Register	RR	0x0000_0000	17.3.7.2/17-51
0x1_0B88	MIA_CE—MIA Count Enable Register	R/W	0x0000_0000	17.3.7.3/17-52
0x1_0B8C-0x1_0B9C	Reserved	—	—	—
0x1_0BA0	MIA_CR—MIA Control Register	R/W	0x0000_0000	17.3.7.4/17-55
0x1_0BA4	MIA_ACR—MIA Arbitration Control Register	R/W	0x0000_0000	17.3.7.5/17-56
0x1_0BA8-0x1_0BAC	Reserved	—	—	—
0x1_0BB0	MIA_LSHC0—MIA Local Snoop Hit Count 0 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BB4	MIA_LSHC1—MIA Local Snoop Hit Count 1 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BB8	MIA_LSHC2—MIA Local Snoop Hit Count 2 Register	RR	0x0000_0000	17.3.7.6/17-58
0x1_0BBC	Reserved	—	—	—
0x1_0BC0	MIA_CWC0—MIA Cancelled Write Count 0 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BC4	MIA_CWC1—MIA Cancelled Write Count 1 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BC8	MIA_CWC2—MIA Cancelled Write Count 2 Register	RR	0x0000_0000	17.3.7.7/17-58
0x1_0BCC 0x1_0BF4	Reserved	—	—	—
0x1_0BF8	PM_IP_REV1—PM IP Block Revision 1 Register	R	0x0010_0000	17.3.7.8/17-59
0x1_0BFC	PM_IP_REV2—PM IP Block Revision 2 Register	R	0x0000_0000	17.3.7.9/17-59
DMA Engine—Channel 0 Control Registers				
0x1_1000	ISR0—Interrupt Status Register 0	w1c	0x0000_0000	17.3.8.1/17-60
0x1_1004	IER0—Interrupt Enable Register 0	R/W	0x0000_0000	17.3.8.2/17-62
0x1_1008	ISDR0—Interrupt Status Disable Register 0	R/W	0x0000_0000	17.3.8.3/17-63
0x1_100C	IIR0—Interrupt Inhibit Register 0	R/W	0x0000_0000	17.3.8.4/17-64
0x1_1010	Reserved	—	—	—

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_1014-0x1_101C	Reserved	—	—	—
0x1_1020	IICR0—Interrupt Interval Control Register 0	R/W	0x0000_0000	17.3.8.5/17-65
0x1_1024-0x1_105C	Reserved	—	—	—
0x1_1060	NPIR0—Notifications Produced Index Register Channel 0	R	0x0000_0000	17.3.8.6/17-65
0x1_1064	CCIR0—Commands Consumed Index Register Channel 0	R	0x0000_0000	17.3.8.7/17-66
0x1_1068-0x1_106C	Reserved	—	—	—
0x1_1070	FBCIR0—Free Buffer Deallocate FIFO Consumer Index Channel 0	R	0x0000_0000	17.3.8.8/17-67
0x1_1074	CHERR0—Channel Error Status Channel 0	R	0x0000_0000	17.3.8.9/17-68
0x1_1078-0x1_107C	Reserved	—	—	—
0x1_1080	NCIR0—Notifications Consumed Index Register Channel 0	R/W	0x0000_0000	17.3.8.10/17-68
0x1_1084	CPIR0—Commands Produced Index Register Channel 0	R/W	0x0000_0000	17.3.8.11/17-69
0x1_1088	CEIR0—Commands Expired Index Register Channel 0	R/W	0x0000_0000	17.3.8.12/17-70
0x1_108C	Reserved	—	—	—
0x1_1090	FBPIR0—Free Buffer Deallocate FIFO Producer Index Channel 0	R/W	0x0000_0000	17.3.8.13/17-70
0x1_1094-0x1_109C	Reserved	—	—	—
0x1_10A0	TRUNCIC0—Output Truncated Incidence Counter Channel 0	RR	0x0000_0000	17.3.8.14/17-71
0x1_10A4	RBC0—Read Byte Counter Channel 0	RR	0x0000_0000	17.3.8.15/17-72
0x1_10A8	SCOS0—Statistics Counters Overflow Status Register Channel 0	w1c	0x0000_0000	17.3.8.16/17-72
0x1_10AC-0x1_10FC	Reserved	—	—	—
0x1_1100	CRCR0—Channel Reset and Control Register Channel 0	R/W	0x0000_0000	17.3.8.17/17-73
0x1_1104-0x1_111C	Reserved	—	—	—
0x1_1120	CFIFO_BH0—Command FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.18/17-74
0x1_1124	CFIFO_BL0—Command FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.18/17-74
0x1_1128	CFIFO_DEPTH0—Command FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.19/17-75
0x1_112C	CFIFO_THRES0—Command FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.20/17-76
0x1_1130-0x1_113C	Reserved	—	—	—
0x1_1140	NFIFO_BH0—Notification FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.21/17-76

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_1144	NFIFO_BL0—Notification FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.21/17-76
0x1_1148	NFIFO_DEPTH0—Notification FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.22/17-78
0x1_114C	NFIFO_THRES0—Notification FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.23/17-78
0x1_1150	NDLL0—Notification Deflate Length Limit Register Channel 0	R/W	0x0000_0000	17.3.8.24/17-79
0x1_1154	NRLL0—Notification Result Length Limit Register Channel 0	R/W	0x0000_0000	17.3.8.25/17-79
0x1_1154-0x1_115C	Reserved	—	—	—
0x1_1160	FBFIFO_BH0—Free Buffer Deallocate FIFO Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.26/17-80
0x1_1164	FBFIFO_BL0—Free Buffer Deallocate FIFO Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.26/17-80
0x1_1168	FBFIFO_DEPTH0—Free Buffer Deallocate FIFO Depth Register Channel 0	R/W	0x0000_0000	17.3.8.27/17-81
0x1_116C	FBFIFO_THRES0—Free Buffer Deallocate FIFO Threshold Register Channel 0	R/W	0x0000_0000	17.3.8.28/17-82
0x1_1170-0x1_117C	Reserved	—	—	—
0x1_1180	SC_CONFIG0—Stream Context Configuration Register Channel 0	R/W	0x0000_0000	17.3.8.29/17-82
0x1_1184	SC_BH0—Stream Context Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.30/17-83
0x1_1188	SC_BL0—Stream Context Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.30/17-83
0x1_118C-0x1_119C	Reserved	—	—	—
0x1_11A0	RES_CONFIG0—Residue Configuration Register Channel 0	R/W	0x0000_0000	17.3.8.31/17-84
0x1_11A4	RES_BH0—Residue Base Address Hi Register Channel 0	R/W	0x0000_0000	17.3.8.32/17-85
0x1_11A8	RES_BL0—Residue Base Address Low Register Channel 0	R/W	0x0000_0000	17.3.8.32/17-85
0x1_11AC-0x1_11BC	Reserved	—	—	—
0x1_11C0	CACR0—Cache Awareness Control Register Channel 0	R/W	0x0000_0000	17.3.8.33/17-86
0x1_11C4	MTPCR0—Memory Transaction Priority Control Register Channel 0	R/W	0x0000_0000	17.3.8.34/17-88
0x1_11C8-0x1_13FC	Reserved	—	—	—
FBM - Channel 0 Control Registers				
0x1_1400	FBL_THRESH_A0—Free Buffer List Threshold A Register Channel 0	R/W	0x0000_0000	17.3.9.1/17-89
0x1_1404	FBL_HLWM_A0—Free Buffer List High/Low Watermark A Register Channel 0	WR	0x0000_0000	17.3.9.2/17-90

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_1408	FBL_AD_CA0—Free Buffer List Allocation and Deallocation Counter Register A Channel 0	RR	0x0000_0000	17.3.9.3/17-91
0x1_140C	FBL_ODD_A0—Free Buffer List On Deck Disable A Register Channel 0	R/W	0x0000_0000	17.3.9.4/17-91
0x1_1410	FBL_PH_AH0—Free Buffer List Physical Head Address A High Register Channel 0	R	0x0000_0000	17.3.9.5/17-92
0x1_1414	FBL_PH_AL0—Free Buffer List Physical Head Address A Low Register Channel 0	R	0x0000_0000	17.3.9.5/17-92
0x1_1418	FBL_VH_AH0—Free Buffer List Virtual Head Address A High Register Channel 0	R	0x0000_0000	17.3.9.6/17-93
0x1_141C	FBL_VH_AL0—Free Buffer List Virtual Head Address A Low Register Channel 0	R	0x0000_0000	17.3.9.6/17-93
0x1_1420	FBL_PT_AH0—Free Buffer List Physical Tail Address A High Register Channel 0	R	0x0000_0000	17.3.9.7/17-94
0x1_1424	FBL_PT_AL0—Free Buffer List Physical Tail Address A Low Register Channel 0	R	0x0000_0000	17.3.9.7/17-94
0x1_1428-0x1_142C	Reserved	—	—	—
0x1_1430	FBL_NB_A0—Free Buffer List Number of Buffers A Register Channel 0	R	0x0000_0000	17.3.9.8/17-95
0x1_1434	FBL_BS_A0—Free Buffer List Buffer Size A Register Channel 0	R	0x0000_0000	17.3.9.9/17-96
0x1_1438	FBL_VD_AH0—Free Buffer List Virtual On Deck Pointer A High Register Channel 0	R	0x0000_0000	17.3.9.10/17-96
0x1_143C	FBL_VD_AL0—Free Buffer List Virtual On Deck Pointer A Low Register Channel 0	R	0x0000_0000	17.3.9.10/17-96
0x1_1440	FBL_THRESH_B0—Free Buffer List Threshold B Register Channel 0	R/W	0x0000_0000	17.3.9.11/17-97
0x1_1444	FBL_HLWM_B0—Free Buffer List High/Low Watermark B Register Channel 0	WR	0x0000_0000	17.3.9.12/17-97
0x1_1448	FBL_AD_CB0—Free Buffer List Allocation and Deallocation Counter Register B Channel 0	RR	0x0000_0000	17.3.9.13/17-98
0x1_144C	FBL_ODD_B0—Free Buffer List On Deck Disable B Register Channel 0	R/W	0x0000_0000	17.3.9.14/17-98
0x1_1450	FBL_PH_BH0—Free Buffer List Physical Head Address B High Register Channel 0	R	0x0000_0000	17.3.9.15/17-98
0x1_1454	FBL_PH_BL0—Free Buffer List Physical Head Address B Low Register Channel 0	R	0x0000_0000	17.3.9.15/17-98
0x1_1458	FBL_VH_BH0—Free Buffer List Virtual Head Address B High Register Channel 0	R	0x0000_0000	17.3.9.16/17-98
0x1_145C	FBL_VH_BL0—Free Buffer List Virtual Head Address B Low Register Channel 0	R	0x0000_0000	17.3.9.16/17-98
0x1_1460	FBL_PT_BH0—Free Buffer List Physical Tail Address B High Register Channel 0	R	0x0000_0000	17.3.9.17/17-98

Table 17-4. PM Individual Register Memory Map (continued)

Offset	Register	Access	Reset Value	Section/Page
0x1_1464	FBL_PT_BL0—Free Buffer List Physical Tail Address B Low Register Channel 0	R	0x0000_0000	17.3.9.17/17-98
0x1_1468-0x1_146C	Reserved	—	—	—
0x1_1470	FBL_NB_B0—Free Buffer List Number of Buffers B Register Channel 0	R	0x0000_0000	17.3.9.18/17-98
0x1_1474	FBL_BS_B0—Free Buffer List Buffer Size B Register Channel 0	R	0x0000_0000	17.3.9.19/17-99
0x1_1478	FBL_PD_BH0—Free Buffer List Virtual On Deck Pointer B High Register Channel 0	R	0x0000_0000	17.3.9.20/17-99
0x1_147C	FBL_PD_BL0—Free Buffer List Virtual On Deck Pointer B Low Register Channel 0	R	0x0000_0000	17.3.9.20/17-99
DMA Engine—Channel 1 Control Registers				
0x1_2000-0x1_27FF	DMA Engine and FBM Channel 1 has the same memory mapped registers that are described for channel 0.			
DMA Engine—Channel 2 Control Registers				
0x1_3000-0x1_37FF	DMA Engine and FBM Channel 2 has the same memory mapped registers that are described for channel 0.			
DMA Engine—Channel 3 Control Registers				
0x1_4000-0x1_47FF	DMA Engine and FBM Channel 3 has the same memory mapped registers that are described for channel 0.			

17.3.1 DMA Engine Common Control Registers

The DMA Engine control registers are divided into four per-channel register groups and one common DMA Engine control register group. This section describes the common control register group.

17.3.1.1 Interrupt Status Register Common Control (ISRCC)

If an event or condition occurs that sets a bit in the ISRCC register the Pattern Matcher Common Control interrupt is generated if the corresponding bit in the IERCC register is also set and the corresponding bit in the ISDRCC register is cleared. Clearing the ISRCC bit clears the Pattern Matcher Common Control interrupt signal (as long as no other ISRCC/IERCC bit pairs are set). The bit in the ISRCC register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

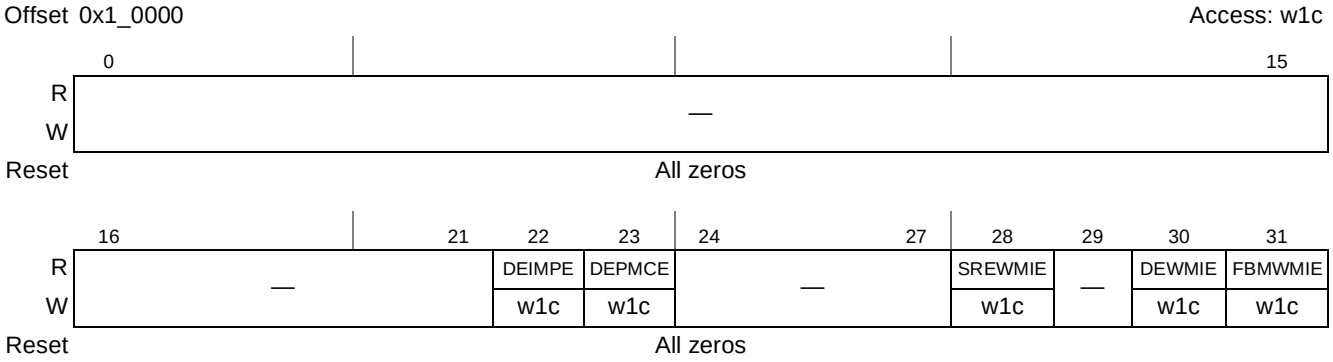


Figure 17-2. Interrupt Status Register Common Control Format

Table 17-5. Interrupt Status Register Common Control Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	DEIMPE	DMA Engine Internal Memory Parity Error. This bit is asserted when a parity error is detected by the DMA Engine during a read from its internal context memory. When this error is detected the DMA engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. Any work units that encounter this error has the “DE Internal Memory Parity Error” code (see Table 17-156) set in their Notification FIFO entry. 0 No error detected. 1 An error was detected.
23	DEPMCE	DMA Engine Pattern Matching Control Error. This bit is asserted when any of the DMA Engine channels encounter an unrecognizable Pattern Matcher Control command or when the Key Element Scanner, Data Examination Engine or Stateful Rule Engine encounter an error as a result of an ill-formed pattern matching data structure. When this error is detected the DMA Engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. Any work units that were in flight at the time the error was detected, and were affected by the error, has the appropriate error code (see Table 17-156) set in their Notification FIFO entry. 0 No error detected. 1 An error was detected.
24–27	—	Reserved, should be cleared.
28	SREWMIE	Stateful Rule Engine Write Memory Interface Error. This bit is asserted when a memory write transaction performed by the Stateful Rule Engine has resulted in a system memory bus error. When this error is detected the DMA Engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. Any work units that were in flight at the time the error was detected, and were affected by the error, has the “SRE Context System Memory Write Error” code (see Table 17-156) set in their Notification FIFO entry. 0 No error detected. 1 A system memory bus error (data error or transfer error) was detected.
29	—	Reserved, should be cleared.

Table 17-5. Interrupt Status Register Common Control Field Descriptions (continued)

Bits	Name	Description
30	DEWMIE	DMA Engine Write Memory Interface Error. This bit is asserted when a memory write transaction performed by the DMA Engine has resulted in a system memory bus error. When this error is detected the DMA Engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. Any work units that were in flight at the time the error was detected, and were affected by the error, has the “DE System Memory Write Error” code (see Table 17-156) set in their Notification FIFO entry. 0 No error detected. 1 A system memory bus error (data error or transfer error) was detected.
31	FBMWMIE	Free Buffer Manager Write Memory Interface Error. This bit is asserted when a memory write transaction performed by the Free Buffer Manager has resulted in a system memory bus error. When this error is detected the DMA Engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. Any work units that were in flight at the time the error was detected, and were affected by the error, has the “FBM System Memory Write Error” code (see Table 17-156) set in their Notification FIFO entry. 0 No error detected. 1 A system memory bus error (data error or transfer error) was detected.

17.3.1.2 Interrupt Enable Register Common Control (IERCC)

The IERCC register provides control over which possible interrupt events or conditions are allowed to generate the actual Pattern Matcher Common Control interrupt. All implemented bits in this register are R/W. This register is cleared upon hardware reset. If the corresponding bits in both ISRCC and IERCC registers are set, then the Pattern Matcher Common Control interrupt is generated. The interrupt signal can be cleared by clearing the corresponding ISRCC bit. The state of an IERCC bit does not affect setting of the corresponding ISRCC bit, it controls whether or not setting of the ISRCC bit causes an interrupt.

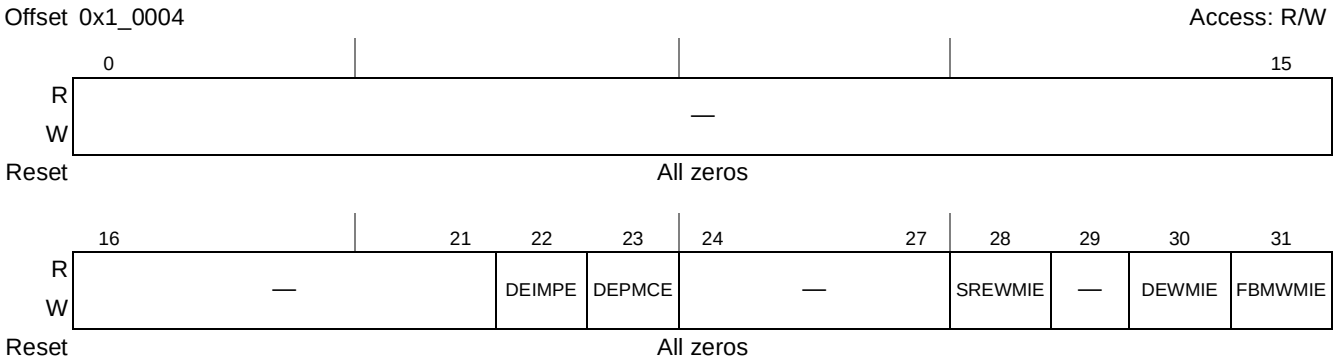


Figure 17-3. Interrupt Enable Register Common Control Format

Table 17-6. Interrupt Enable Register Common Control Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	DEIMPE	DMA Engine Internal Memory Parity Error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt

Table 17-6. Interrupt Enable Register Common Control Field Descriptions (continued)

Bits	Name	Description
23	DEPMCE	DMA Engine Pattern Matcher Control Error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
24-27	—	Reserved, should be cleared.
28	SRE WMIE	Stateful Rule Engine Write Memory Interface Error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
29	—	Reserved, should be cleared.
30	DEWMIE	DMA Engine Write Memory Interface Error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
31	FBMWMIE	Free Buffer Manager Write Memory Interface Error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt

17.3.1.3 Interrupt Status Disable Register Common Control (ISDRCC)

The ISDRCC register controls ISRCC reporting. An ISRCC bit is not set if the corresponding bit in the ISDRCC register is set (interrupt status is disabled).

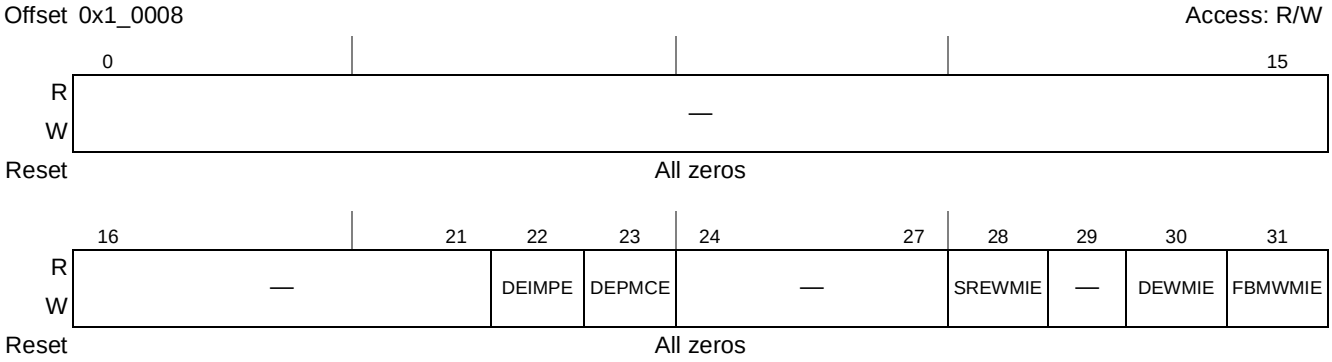


Figure 17-4. Interrupt Status Disable Register Common Control Format

Table 17-7. Interrupt Status Disable Register Common Control Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	DEIMPE	DMA Engine Internal Memory Parity Error Interrupt Status Disable. 0 Enable status 1 Disable status
23	DEPMCE	DMA Engine Pattern Matcher Control Error Interrupt Status Disable. 0 Enable status 1 Disable status
24–27	—	Reserved, should be cleared.

Table 17-7. Interrupt Status Disable Register Common Control Field Descriptions (continued)

Bits	Name	Description
28	SREWMIE	Stateful Rule Engine Write Memory Interface Error Interrupt Status Disable. 0 Enable status 1 Disable status
29	—	Reserved, should be cleared.
30	DEWMIE	DMA Engine Write Memory Interface Error Interrupt Status Disable. 0 Enable status 1 Disable status
31	FBMWMIE	Free Buffer Manager Write Memory Interface Error Interrupt Status Disable. 0 Enable status 1 Disable status

17.3.1.4 Interrupt Inhibit Register Common Control (IIRCC)

The IIRCC register allows the Pattern Matcher Common Control interrupt signal to be inhibited without modifying the enable or status disable registers.

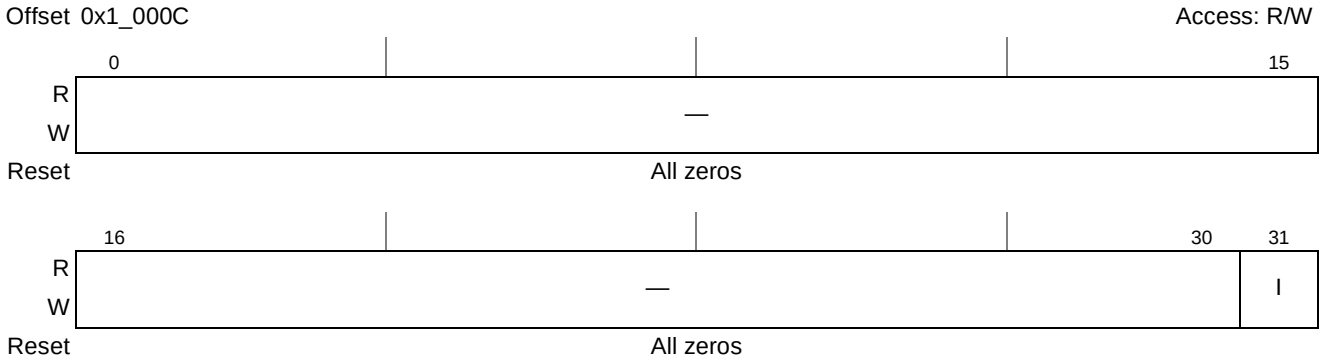


Figure 17-5. Interrupt Inhibit Register Common Control Format

Table 17-8. Interrupt Inhibit Register Common Control Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	I	Inhibit. 0 Interrupt not inhibited. Pattern Matcher Common Control interrupt signal may assert. 1 Interrupt inhibited. Pattern Matcher Common Control interrupt signal does not assert.

17.3.1.5 Statistics Counters Overflow Status (SCOS)

The SCOS register contains overflow bits for the various statistics counters in the Pattern Matcher. If a statistics counter rolls over, the corresponding bit in SCOS is set. Each overflow status bit is individually cleared by writing a 1 in that bit location.

Offset 0x1_0014 Access: w1c

	0	6	7	8	9	10	11	12	13	14	15
R	—		STNTHS-OS	MIA_CWC2-OS	MIA_CWC1-OS	MIA_CWC0-OS	MIA_LSHC2-OS	MIA_LSHC1-OS	MIA_LSHC0-OS	MIA_BLC-OS	MIA_BYC-OS
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	STDWC-OS	STDBP-OS	STDBC-OS	STNOB-OS	STNS1R-OS	STNESR-OS	STNDSR-OS	STNPMR-OS	STNS1M-OS	STNPM-OS	STNCH-OS	STNTHL-OS	STNTH2-OS	STNTH1-OS	STNIS-OS	STNIB-OS
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros															

Figure 17-6. Statistics Counters Overflow Status Register (SCOS)

Table 17-9. SCOS Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	STNTHS-OS	Overflow Status for the STNTHS counter.
8	MIA_CWC2-OS	Overflow Status for the MIA_CWC2 counter.
9	MIA_CWC1-OS	Overflow Status for the MIA_CWC1 counter.
10	MIA_CWC0-OS	Overflow Status for the MIA_CWC0 counter.
11	MIA_LSHC2-OS	Overflow Status for the MIA_LSHC2 counter.
12	MIA_LSHC1-OS	Overflow Status for the MIA_LSHC1 counter.
13	MIA_LSHC0-OS	Overflow Status for the MIA_LSHC0 counter.
14	MIA_BLC-OS	Overflow Status for the MIA_BLC counter.
15	MIA_BYC-OS	Overflow Status for the MIA_BYC counter.
16	STDWC-OS	Overflow Status for the STDWC counter.
17	STDBP-OS	Overflow Status for the STDBP counter.
18	STDBC-OS	Overflow Status for the STDBC counter.
19	STNOB-OS	Overflow Status for the STNOB counter.
20	STNS1R-OS	Overflow Status for the STNS1R counter.
21S	STNESR-O	Overflow Status for the STNESR counter.
22	STNDSR-OS	Overflow Status for the STNDSR counter.
23	STNPMR-OS	Overflow Status for the STNPMR counter.

Table 17-9. SCOS Field Descriptions (continued)

Bits	Name	Description
24	STNS1M-OS	Overflow Status for the STNS1M counter.
25	STNPM-OS	Overflow Status for the STNPM counter.
26	STNCH-OS	Overflow Status for the STNCH counter.
27	STNTHL-OS	Overflow Status for the STNTHL counter.
28	STNTH2-OS	Overflow Status for the STNTH2 counter.
29	STNTH1-OS	Overflow Status for the STNTH1 counter.
30	STNIS-OS	Overflow Status for the STNIS counter.
31	STNIB-OS	Overflow Status for the STNIB counter.

17.3.1.6 Performance Monitor Threshold (PERFT)

The PERFT register allows programming of a time based latency threshold such that memory read transactions whose latency exceeds the threshold can be measured using the integrated performance monitor. There are multiple state machines who's memory latency can be measured and they all reference this single threshold register.

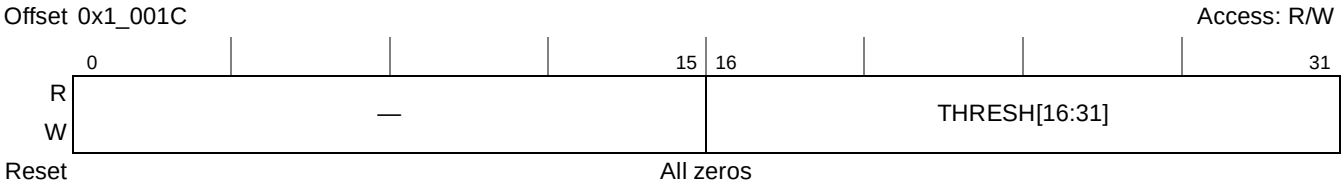


Figure 17-7. Performance Monitor Threshold Format

Table 17-10. Performance Monitor Threshold Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	THRESH	Threshold. The count, in 333 MHz clock cycles, above which an outstanding memory read transaction creates a signal to the integrated performance monitor indicating that a read is in-flight. The signal remains asserted so long as the read is outstanding such that a performance monitor counter can be used to analyze excess memory latency.

17.3.1.7 Channel Scheduling Control Register (CSCR)

The CSCR register is used to specify and control the channel scheduling algorithm that the DMA Engine uses when servicing the DMA channels.

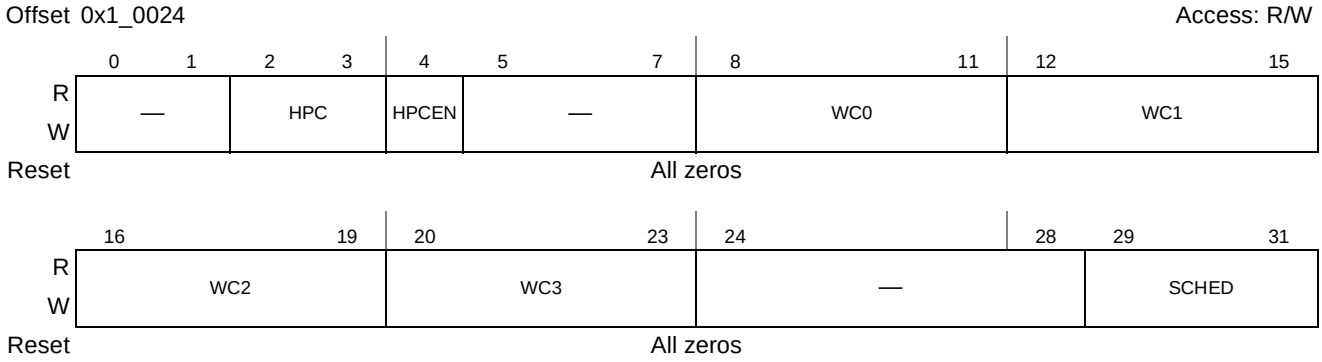


Figure 17-8. Channel Scheduling Control Register Format

Table 17-11. Channel Scheduling Control Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–3	HPC	High Priority Channel. This field, when the High Priority Channel Enable bit is set, represents the DMA Engine channel designated as a high priority channel. The channel designated as high priority is serviced before all others regardless of the programmed channel scheduling algorithm.
4	HPCEN	High Priority Channel Enable. When set, this bit enables designation of a single DMA Engine channel as having absolute highest service priority.
5–7	—	Reserved, should be cleared.
8–11	WC0	Weight Channel 0. This field controls the weight given to channel 0 when weighted scheduling schemes are being used. Weights from 0–15 are supported.
12–15	WC1	Weight Channel 1. This field controls the weight given to channel 1 when weighted scheduling schemes are being used. Weights from 0–15 are supported.
16–19	WC2	Weight Channel 2. This field controls the weight given to channel 2 when weighted scheduling schemes are being used. Weights from 0–15 are supported.
20–23	WC3	Weight Channel 3. This field controls the weight given to channel 3 when weighted scheduling schemes are being used. Weights from 0–15 are supported.
24–25	—	Reserved, should be cleared.
26	—	Reserved, should be cleared.
27	—	Reserved, should be cleared.

Table 17-11. Channel Scheduling Control Register Field Descriptions (continued)

Bits	Name	Description
28	—	Reserved, should be cleared.
29–31	SCHED	Scheduling Algorithm. For a general description of this field, see Section 17.4.5.7, “Channel Scheduling”. 000 Encoding 0. Weighted Round Robin By 4. 001 Encoding 1. Reserved 010 Encoding 2. Reserved 011 Encoding 3. Reserved 100 Encoding 4. Reserved 101 Encoding 5. Reserved 110 Encoding 6. Reserved 111 Encoding 7. Weighted Round Robin 2 BY 2.

17.3.2 Key Element Scanner Control Registers

17.3.2.1 Statistic—Number of Input Bytes Scanned for Pattern Matching (STNIB)

The STNIB register counts the number of input bytes scanned by the Pattern Matcher. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

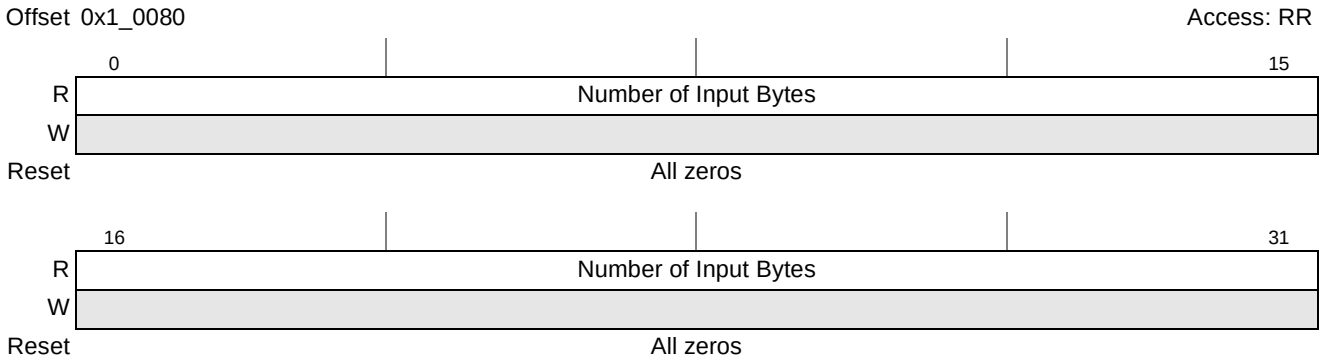


Figure 17-9. STNIB Register Format

Table 17-12. STNIB Register Field Descriptions

Bits	Name	Description
0–31	—	Number of input bytes scanned.

17.3.2.2 Statistic—Number of Input SUIs (STNIS)

The STNIS register counts the number of SUIs scanned by the Pattern Matcher. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

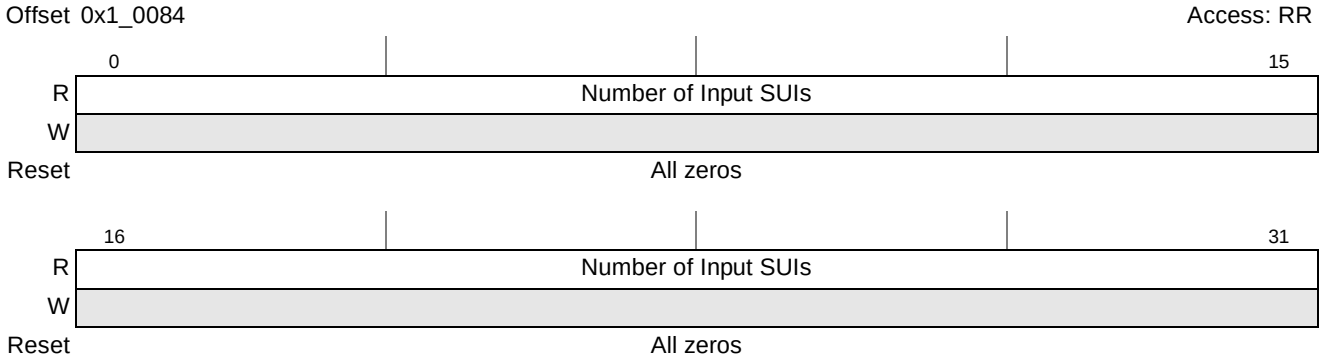


Figure 17-10. STNIS Register Format

Table 17-13. STNIS Register Field Descriptions

Bits	Name	Description
0–31	—	Number of input SUIs scanned.

17.3.2.3 Statistic—Number of Trigger Hits Against 1-Byte Triggers (STNTH1)

The STNTH1 register counts the number of triggers found by the 1-byte Trigger mechanism. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

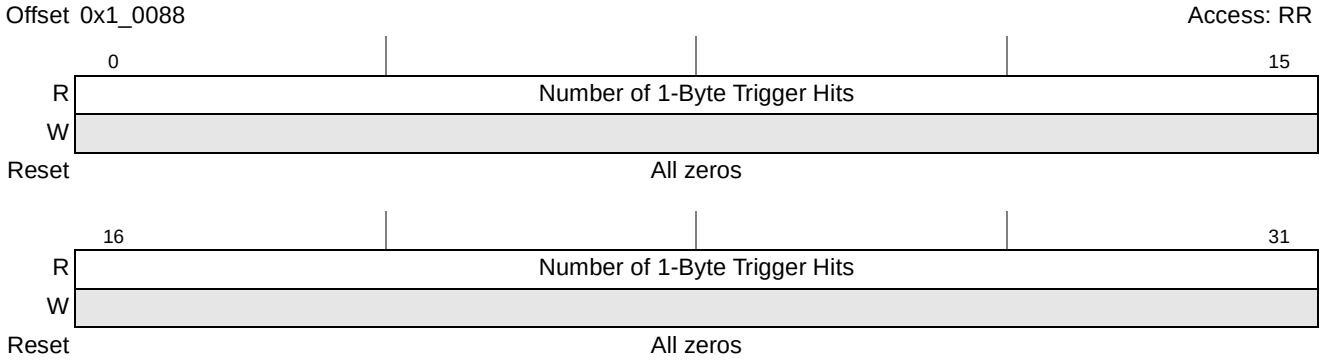


Figure 17-11. STNTH1 Register Format

Table 17-14. STNTH1 Register Field Descriptions

Bits	Name	Description
0–31	—	Number of 1-byte trigger hits.

17.3.2.4 Statistic—Number of Trigger Hits Against 2-Byte Triggers (STNTH2)

The STNTH2 register counts the number of triggers found by the 2-byte Trigger mechanism. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

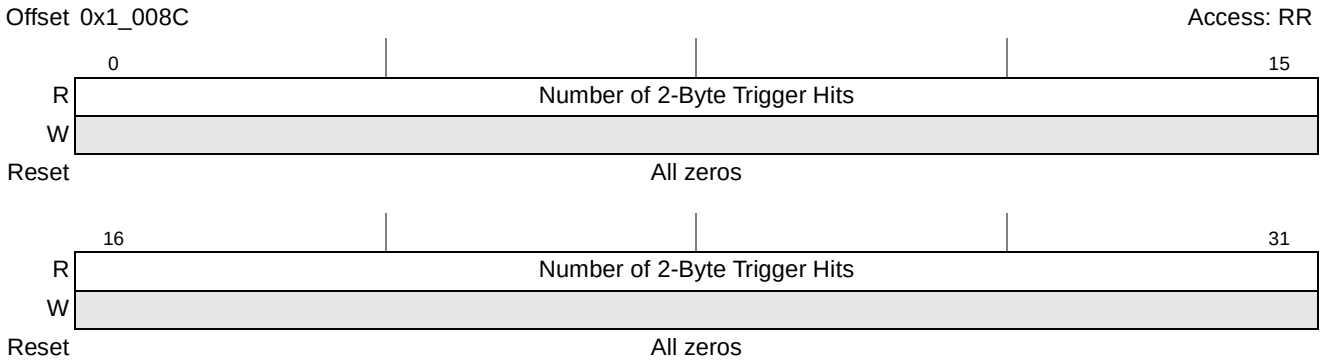


Figure 17-12. STNTH2 Register Format

Table 17-15. STNTH2 Register Field Descriptions

Bits	Name	Description
0–31	—	Number of 2-byte trigger hits.

17.3.2.5 Statistic—Number of Trigger Hits Against Variable Length Triggers (STNTHL)

The STNTHL register counts the number of triggers found by the Variable Length Trigger mechanism. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

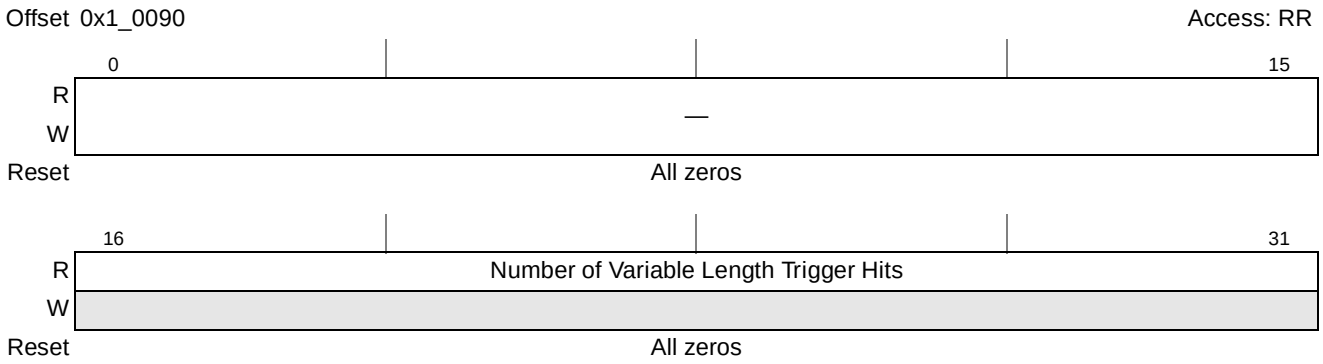


Figure 17-13. STNTHL Register Format

Table 17-16. STNTHL Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of Variable Length trigger hits.

17.3.2.6 Statistic—Number of Trigger Hits against special triggers (STNTHS)

The STNTHS register counts the number of triggers found by the Special Triggers mechanism. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

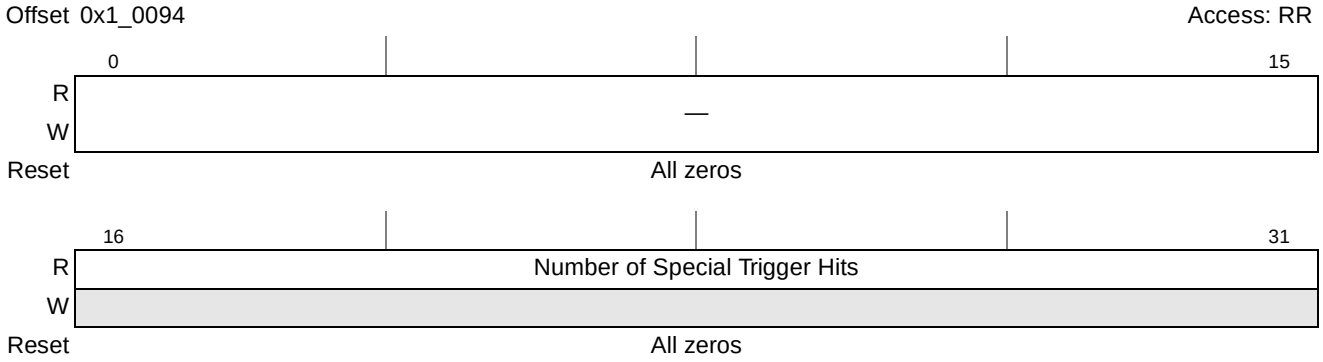


Figure 17-14. STNTHS Register Format

Table 17-17. STNTHS Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of Special trigger hits.

17.3.2.7 Statistic—Number of Confidence Stage Hits (STNCH)

The STNCH register counts the number of confidence check passed by the confidence mechanism. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

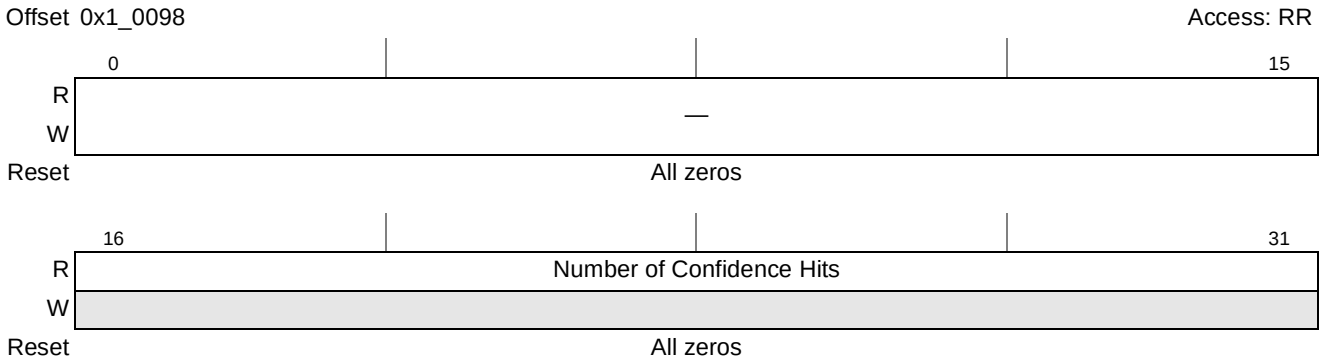


Figure 17-15. STNCH Register Format

Table 17-18. STNCH Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of confidence hits.

17.3.2.8 Software Database Version Register / Scratch Pad (SWDB)

Software can use the SWDB register to hold a version number for the pattern match database. It can be used as a general purpose scratch pad otherwise.

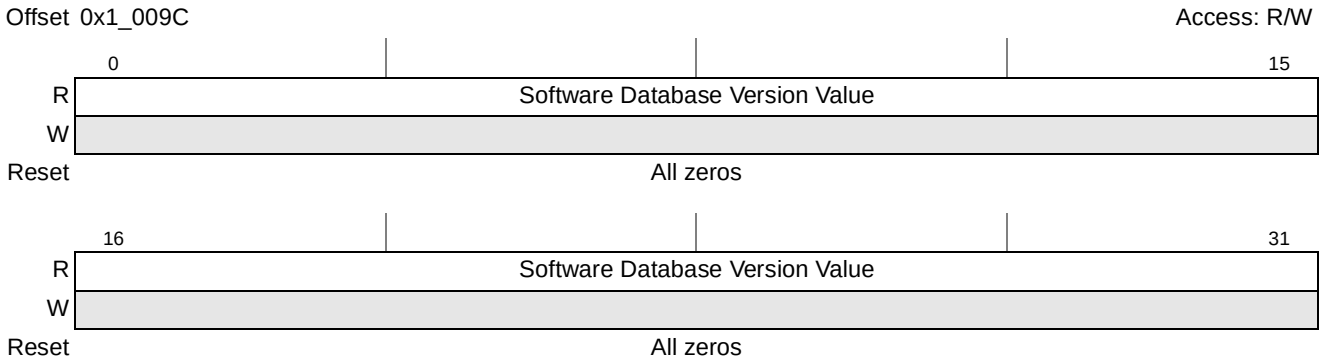


Figure 17-16. SWDB Register Format

Table 17-19. SWDB Register Field Descriptions

Bits	Name	Description
0–31	—	Software Database Version Value.

17.3.2.9 KES Variable Length Trigger Size (KVLTS)

The KVLTS register sets the size of the KES Variable Length Trigger from 3 to 16 bytes.

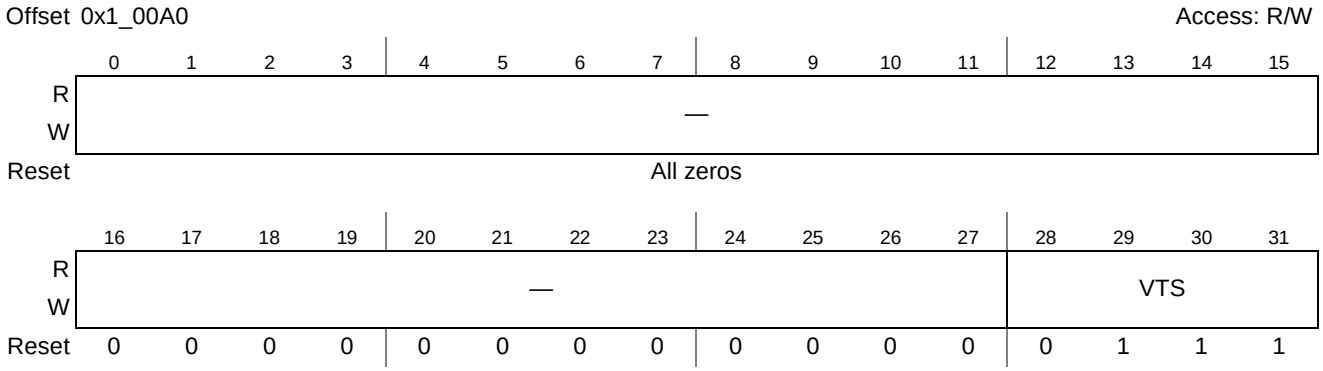


Figure 17-17. KES Variable Length Trigger Size Register Format

Table 17-20. KES Variable Length Trigger Size Register Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	VTS	Variable Trigger Size value of 0 is invalid value of 1–15 means trigger size of 2–16

17.3.2.10 KES Error Configuration (KEC)

The KEC register configures various parameters relating to error handling in the KES.

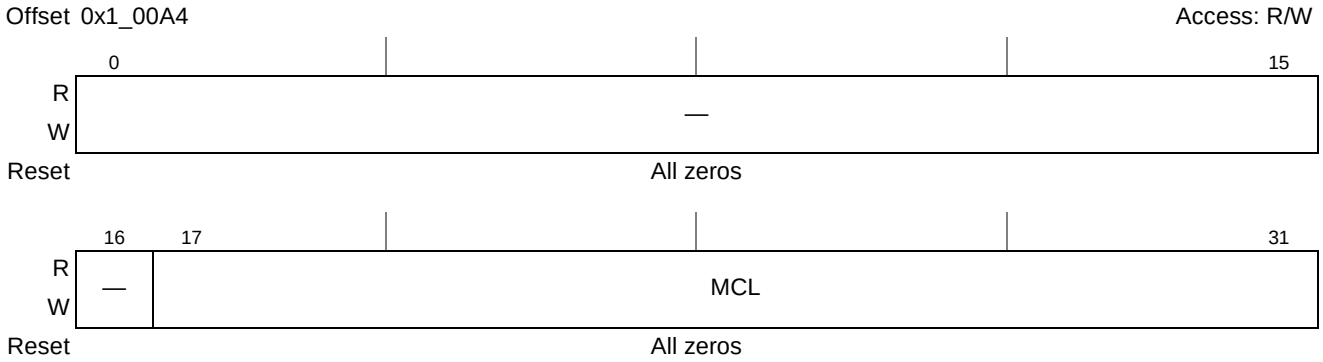


Figure 17-18. KES Error Configuration Register Format

Table 17-21. KES Error Configuration Register Field Descriptions

Bits	Name	Description
0–16	—	Reserved, should be cleared.
17–31	MCL	Specifies the maximum allowed entries in Confidence Collision chains. Allowable range is 0 to 32767. When set to 0, the ‘KES Confidence Collision Limit’ error handler is disabled.

17.3.2.11 PME Error Handling Disables (PEHD)

The PEHD register disables selected pattern matching engines (KES, DXE or SRE) error handlers. When the pattern matching engines (KES, DXE or SRE) encounters any of these errors and the corresponding error handler is enabled (bit set to 0), it ceases operation on the current work unit (no attempt is made to complete the processing on that work unit), report the error with the appropriate Notification FIFO exception code in the Notification FIFO entry produced for that work unit and then proceed to halt processing on all channels in a graceful manner. For each channel, no new commands are consumed and in-flight work (work in the pipeline) are allowed to complete to ensure software is able to unwind. Once the entire pipeline for all channels has been flushed, the DMA Engine Pattern Matching Control Error bit in the ISRCC register is set and the Common Control Error bit in every channel Interrupt Status registers also are set.

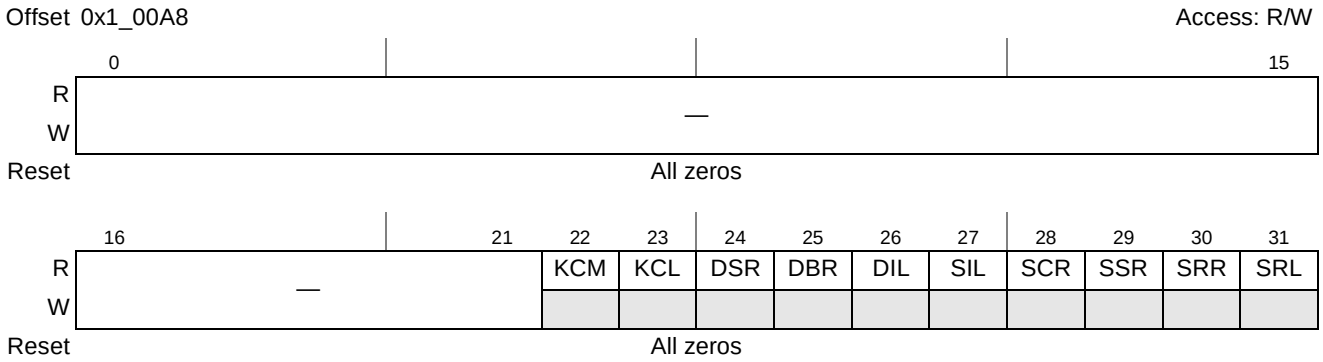


Figure 17-19. PME Error Handling Disables Register Format

Table 17-22. PME Error Handling Disables Register Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	KCM	Disables ‘KES Confidence Mask’ error handler. 0 Enable error handler 1 Disable error handler
23	KCL	Disables ‘KES Confidence Collision Limit’ error handler. 0 Enable error handler 1 Disable error handler
24	DSR	Disables ‘DXE Subsequent Link Reverse’ error handler. 0 Enable error handler 1 Disable error handler
25	DBR	Disables ‘DXE Pattern Description and Stateful Rule Space Outrange’ error handler. 0 Enable error handler 1 Disable error handler
26	DIL	Disables ‘DXE Instruction Limit’ error handler. 0 Enable error handler 1 Disable error handler
27	SIL	Disables ‘SRE Instruction Limit’ error handler. 0 Enable error handler 1 Disable error handler
28	SCR	Disables ‘SRE Context Table Outrange’ error handler. 0 Enable error handler 1 Disable error handler
29	SSR	Disables ‘SRE Session Context Outrange’ error handler. 0 Enable error handler 1 Disable error handler

Table 17-22. PME Error Handling Disables Register Field Descriptions (continued)

Bits	Name	Description
30	SRR	Disables ‘SRE Pattern Description and Stateful Rule Space Outrange’ error handler. 0 Enable error handler 1 Disable error handler
31	SRL	Disables ‘SRE Reaction Limit’ error handler. 0 Enable error handler 1 Disable error handler

17.3.3 Data Examination Engine Control Registers

17.3.3.1 Statistic—Number of Pattern Matches (STNPM)

The STNPM register counts the number of pattern matches found by the Data Examination Engine. This includes 100% matches and inconclusive (partial) matches. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

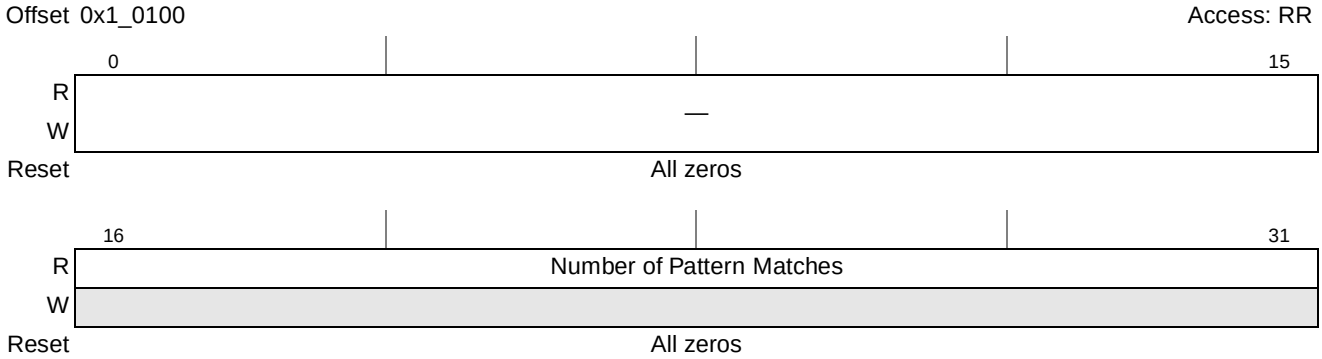


Figure 17-20. STNPM Register Format

Table 17-23. STNPM Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of pattern matches.

17.3.3.2 Statistic—Number of SUIs with at Least 1 Pattern Match (STNS1M)

The STNS1M register counts the number of SUIs with at least one pattern match found within it. This includes 100% matches and inconclusive (partial) matches. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

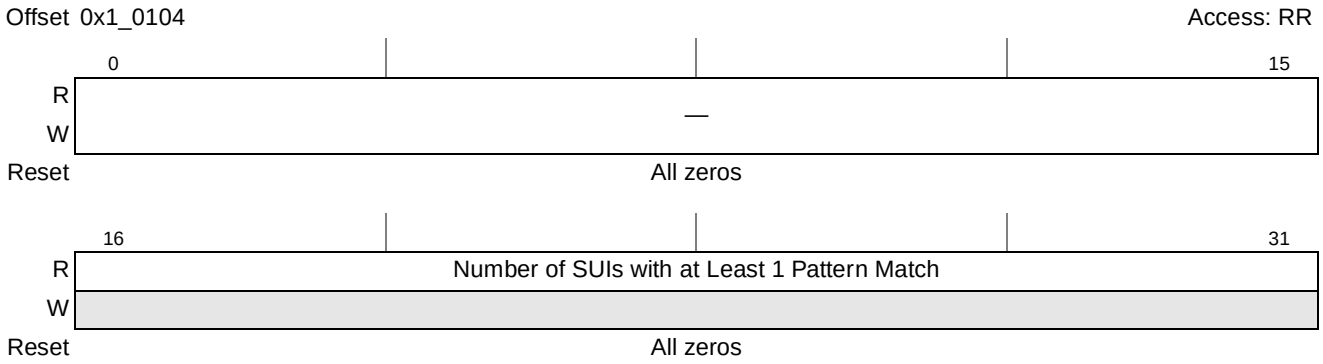


Figure 17-21. STNS1M Register Format

Table 17-24. STNS1M Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of SUIs with at least 1 pattern match.

17.3.3.3 DXE Pattern Range Counter Configuration (DRCC)

The DRCC register configures the pattern index (for example, an index into the Pattern Description and Stateful Rule table) and mask used by the STNPMR register to count pattern match attempts within a certain range of pattern indexes.

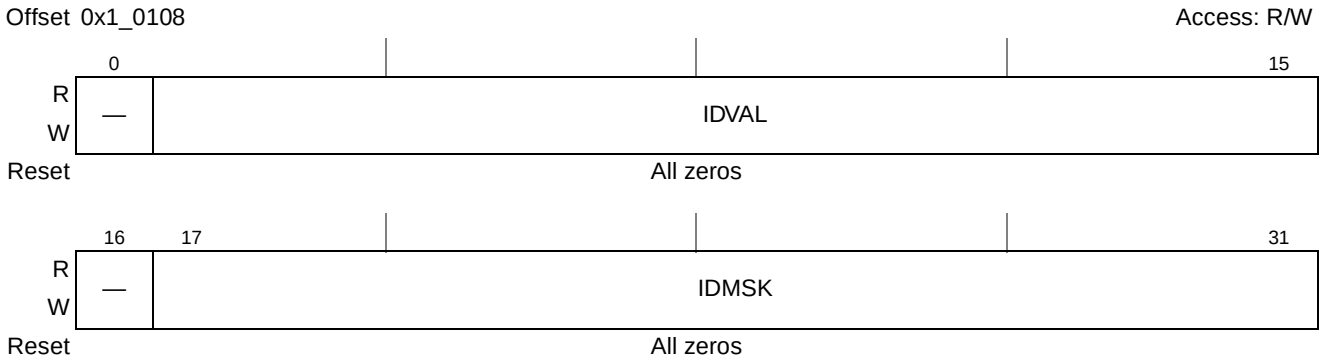


Figure 17-22. DXE Pattern Range Counter Configuration Register Format

Table 17-25. DXE pattern range counter configuration Register Field Descriptions

Bits	Name	Description
0	—	Reserved
1–15	IDVAL	Index value targeted for counting pattern matches.
16	—	Reserved
17–31	IDMSK	Index mask for IDVAL, allowing ranges of addresses to be targeted. A 0 in any bit means a ‘don’t care’ in the corresponding IDVAL bit.

17.3.3.4 Statistic—Number of Pattern Match Attempts Within a Range of Indexes (STNPMR)

The STNPMR register counts the number of pattern match attempts within the index range defined by the DRCC register. This register counts regardless of whether the pattern matches or not. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

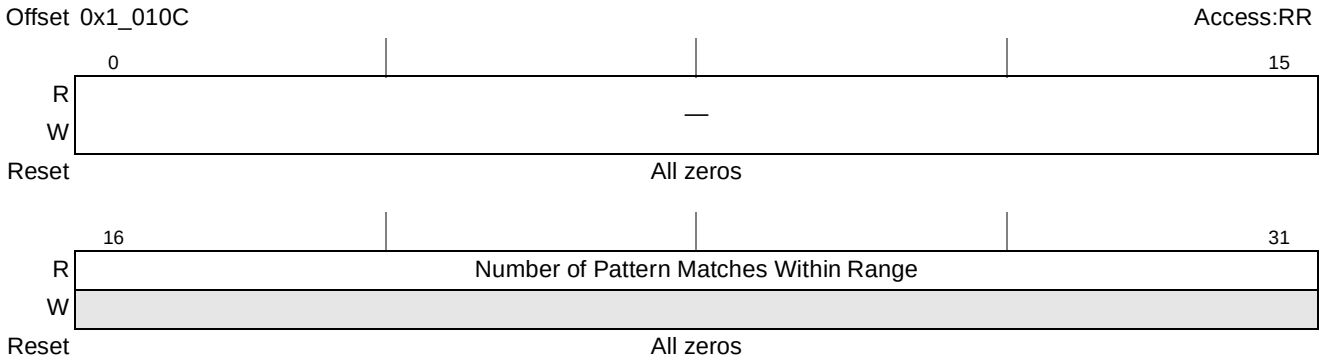


Figure 17-23. STNPMR Register Format

Table 17-26. STNPMR Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of pattern matches within range defined by DRCC.

17.3.3.5 Pattern Description and Stateful Rule Base Address High and Low Registers (PDSRBAH and PDSRBAL)

The PDSRBAH and PDSRBAL registers are used to provision the start location of the Pattern Description and Stateful Rule table space in system memory. The table base address must be aligned to a natural 128 byte address boundary. Maximum size for the Pattern Description and Stateful Rule table is 32 Mbytes.

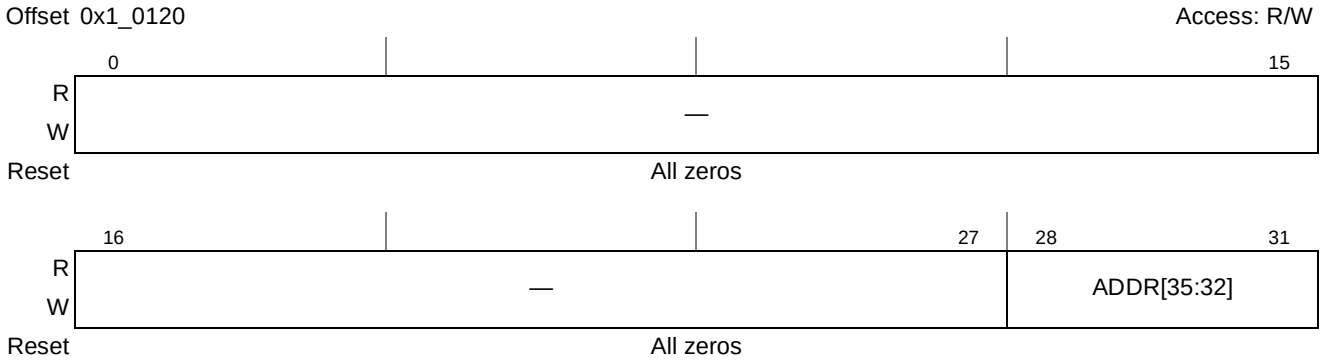


Figure 17-24. Pattern Description and Stateful Rule Base Address High Register Format

Table 17-27. Pattern Description and Stateful Rule Base Address High Register Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Pattern Description and Stateful Rule space in system memory.

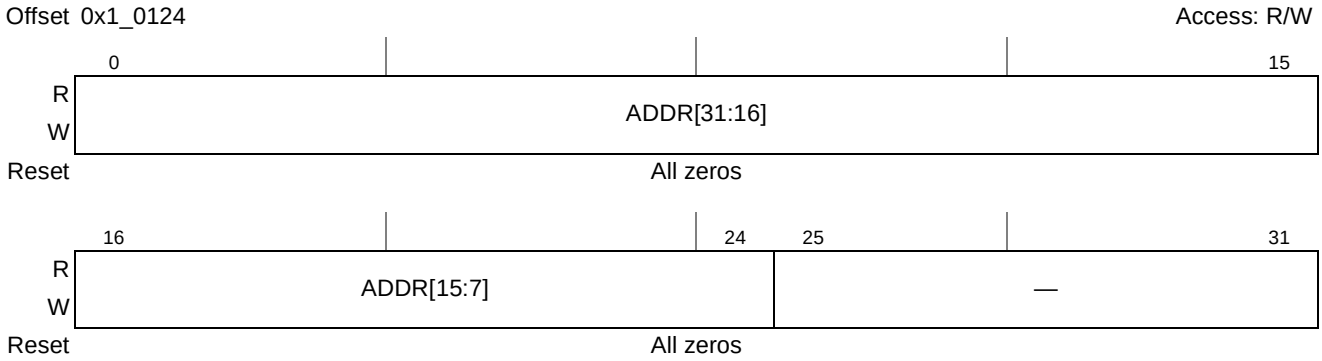


Figure 17-25. Pattern Description and Stateful Rule Base Address Low Register Format

Table 17-28. Pattern Description and Stateful Rule Base Address Low Register Field Descriptions

Bits	Name	Description
0–24	ADDR	The base address of the Pattern Description and Stateful Rule space in system memory.
25–31	—	Reserved, should be cleared.

17.3.3.6 DXE Memory Control Register (DMCR)

The DMCR is used to provides control over cache aware attributes and system memory transaction priority for the DXE.

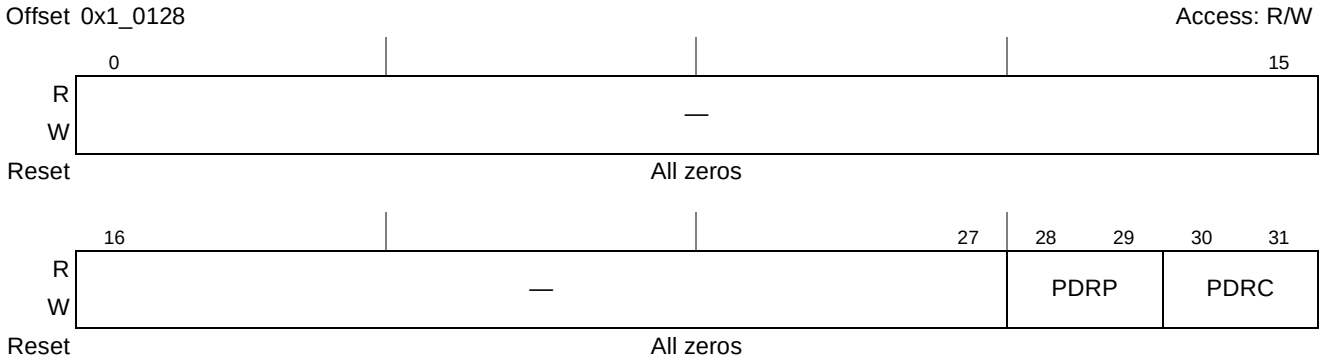


Figure 17-26. DXE Memory Control Register Format

Table 17-29. DXE Memory Control Register Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27	—	Reserved, should be cleared.
28–29	PDRP	DXE Pattern Description Read Priority. Specifies the transaction priority applied to DXE Pattern Description reads. Valid codings are 0 (lowest) to 3 (highest).
30–31	PDRC	DXE Pattern Description Read Cache Aware Controls cache aware attributes during DXE Pattern Description read transactions. For example, it may be desirable to lock certain high-runner patterns in cache, in which case snooping becomes necessary. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved

17.3.3.7 DXE Error Configuration (DEC)

The DXE Error Configuration (DEC) register configures several parameters for the error handling capabilities in the DXE.

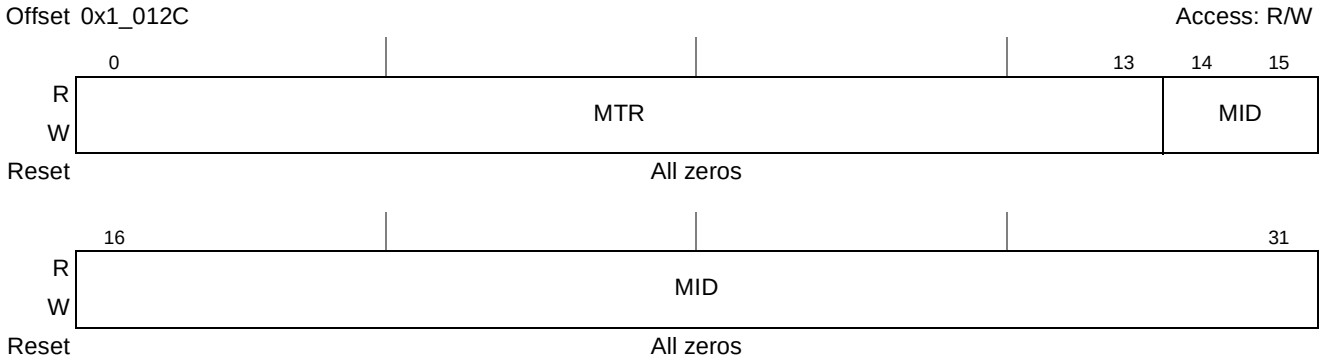


Figure 17-27. DXE Error Configuration Register Format

Table 17-30. DXE Error Configuration Register Field Descriptions

Bits	Name	Description
0–13	MTR	Maximum allowed Test Line executions per pattern (scaled by a factor of 8; the actual maximum allowed test line executions is equal to MTR*8). The maximum allowed value for this field is 0x3fff which translates into a maximum allowed test line execution value of 131064.
14–31	MID	Maximum allocated index into the Pattern Matcher and Stateful Rule table available to the DXE.

17.3.4 Stateful Rule Engine Control Registers

17.3.4.1 Statistic—Number of DXE Generated Stateful Rule Executions (STNDSR)

The STNDSR register counts the number of Stateful Rule executions caused by DXE generated events. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

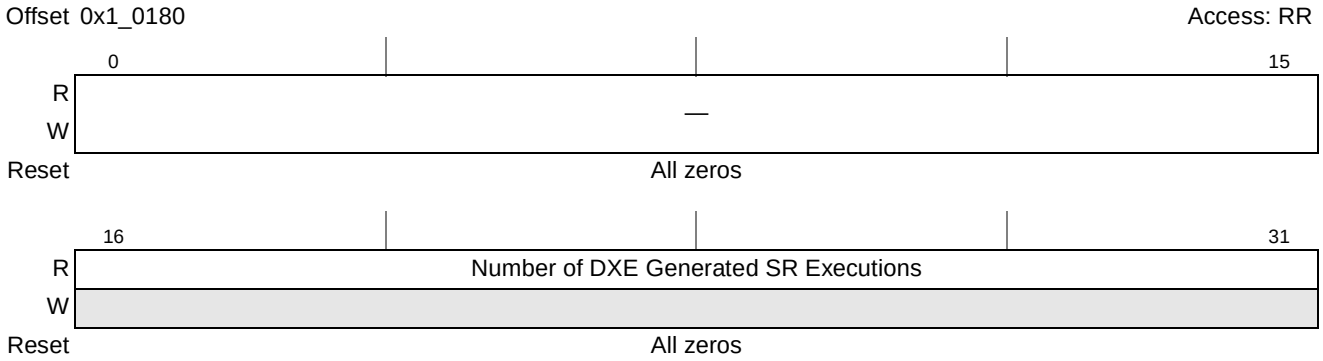


Figure 17-28. STNDSR Register Format

Table 17-31. STNDSR Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of DXE generated Stateful Rule executions.

17.3.4.2 Statistic—Number of End of SUI Generated Stateful Rule Executions (STNESR)

The STNESR register counts the number of Stateful Rule executions caused by End of SUI events. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

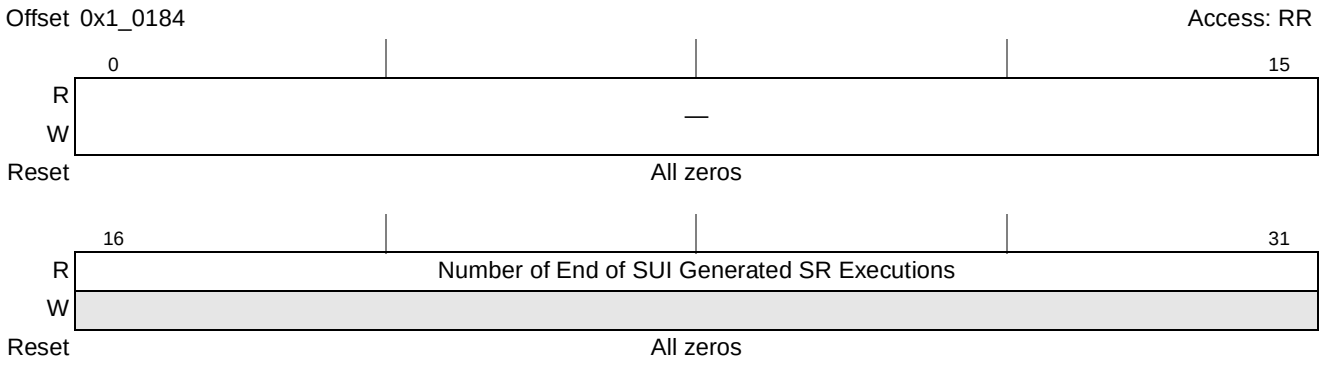


Figure 17-29. STNESR Register Format

Table 17-32. STNESR Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of End of SUI generated SR executions.

17.3.4.3 Statistic—Number of SUIs with at Least 1 Report (STNS1R)

The STNS1R register counts the number of SUIs scanned that produce at least one report. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

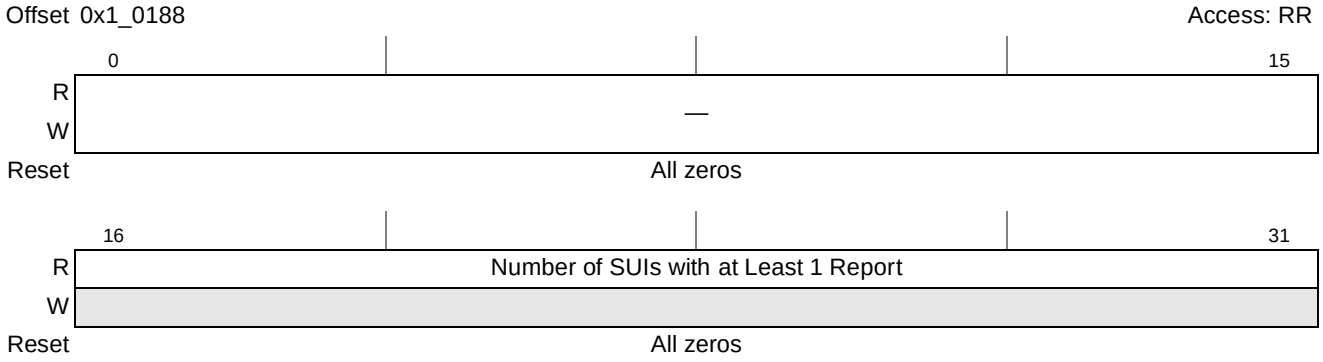


Figure 17-30. STNS1R Register Format

Table 17-33. STNS1R Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of SUIs with at least 1 report.

17.3.4.4 Statistic—Number of Output Bytes Produced in Reports (STNOB)

The STNOB register counts the number of output bytes produced in reports by the Pattern Matcher. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

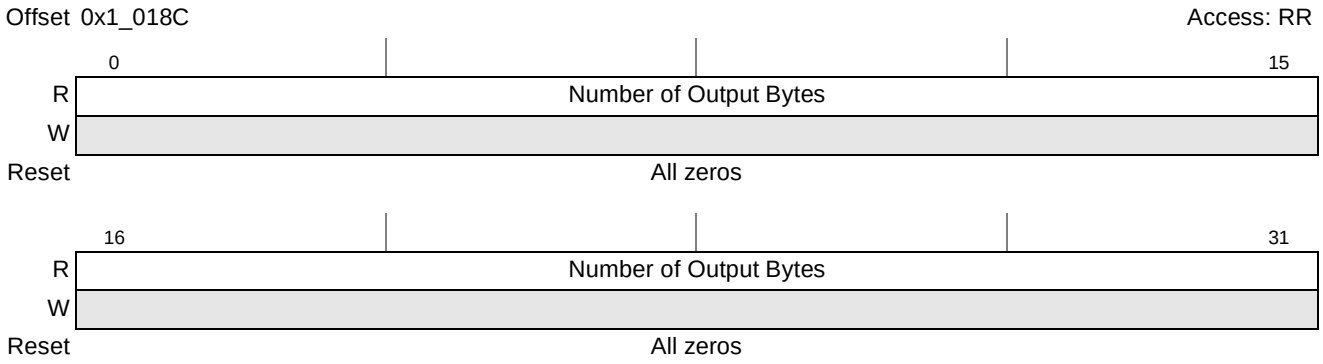


Figure 17-31. STNOB Register Format

Table 17-34. STNOB Register Field Descriptions

Bits	Name	Description
0–31	—	Number of output bytes reported

17.3.4.5 SRE Context Base Address High and Low Registers (SCBARH and SCBARL)

The SCBARH and SCBARL registers are used to configure the location of the SRE Context table in system memory. The table base address must be aligned to a natural 32 byte address boundary. Maximum size for the SRE Context table is 4 Gbytes.

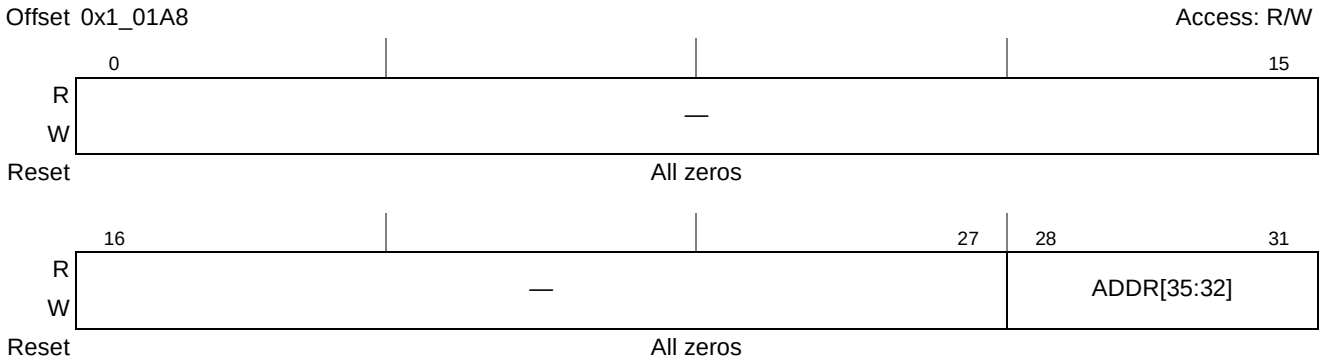


Figure 17-32. SRE Context Base Address High Register Format

Table 17-35. SRE Context Base Address High Register Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the SRE meta-state context region in system memory.

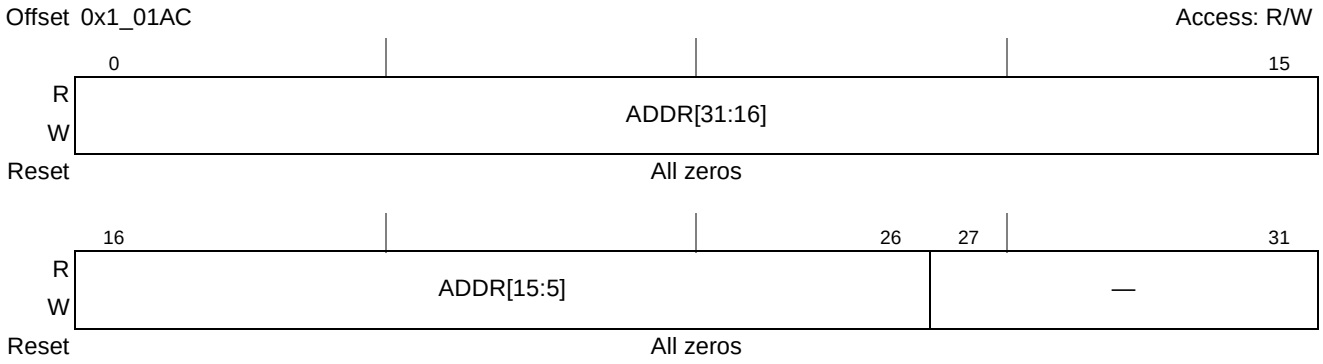


Figure 17-33. SRE Context Base Address Low Register Format

Table 17-36. SRE Context Base Address Low Register Field Descriptions

Bits	Name	Description
0–26	ADDR	The base address of the SRE meta-state context region in system memory.
27–31	—	Reserved, should be cleared.

17.3.4.6 SRE Memory Control Register (SMCR)

The SMCR is used to provide software with control over cache aware attributes and system memory transaction priority for SRE.

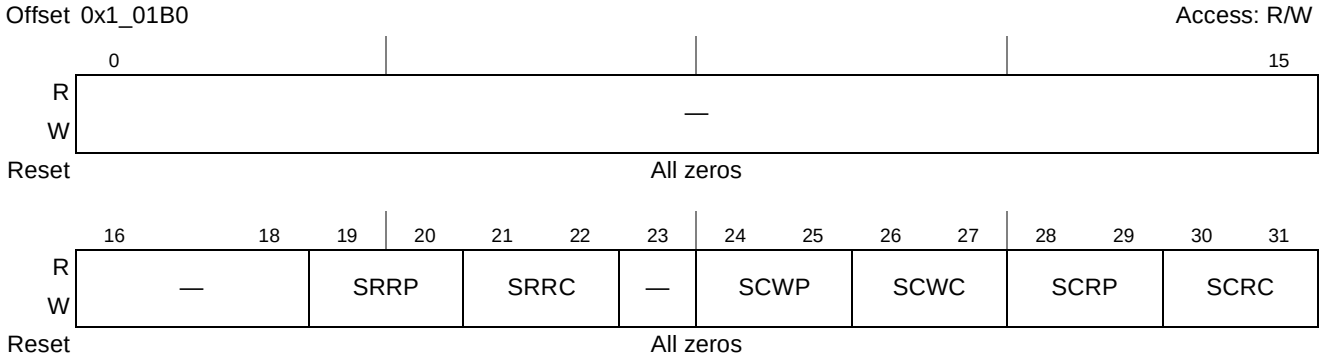


Figure 17-34. SRE Memory Control Register Format

Table 17-37. SRE Memory Control Register Field Descriptions

Bits	Name	Description
0–22	—	Reserved, should be cleared.
18	—	Reserved, should be cleared.
19–20	SRRP	SRE Stateful Rule Read Priority. Specifies the transaction priority applied to SRE Stateful Rule reads. Valid codings are 0 (lowest) to 3 (highest).
21–22	SRRC	SRE Stateful Rule Read Cache Aware. Controls cache aware attributes during SRE Stateful Rule read transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved
23	—	Reserved, should be cleared.
24–25	SCWP	SRE Context Write Priority. Specifies the transaction priority applied to SRE Context writes. Valid codings are 0 (lowest) to 3 (highest).
27–26	SCWC	SRE Context Write Cache Aware. Controls cache aware attributes during SRE Context write transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved

Table 17-37. SRE Memory Control Register Field Descriptions (continued)

Bits	Name	Description
28–29	SCRP	SRE Context Read Priority. Specifies the transaction priority applied to SRE Context reads. Valid codings are 0 (lowest) to 3 (highest).
30–31	SCRC	SRE Context Read Cache Aware. Controls cache aware attributes during SRE Context read transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved

17.3.4.7 SRE Configuration Register (SREC)

The SREC is used to configure various functions within the Stateful Rule Engine.

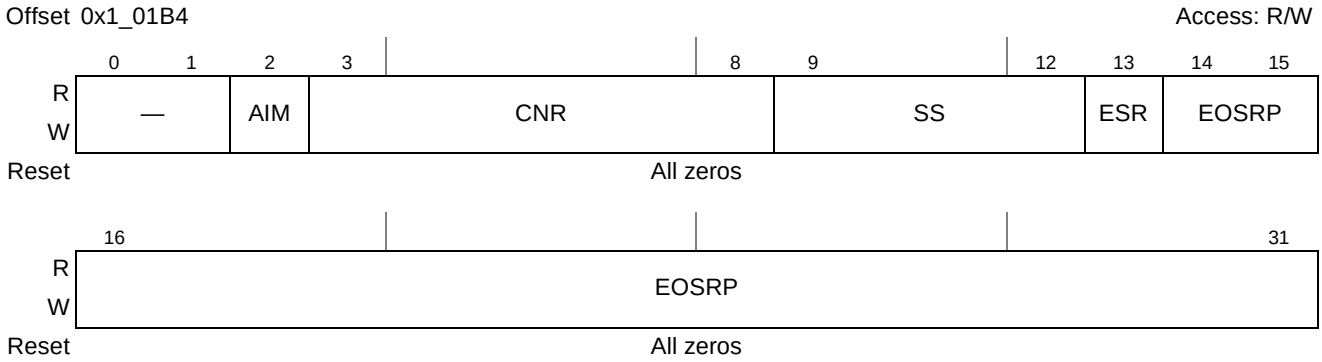


Figure 17-35. SRE Configuration Register Format

Table 17-38. SRE Configuration Register Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	AIM	Alternate Inconclusive Mode. This field specifies the inconclusive match mode treatment. When this field is set 1, the “alternate” inconclusive mode is used whereby when this field is set to 0 the “default” inconclusive mode is used. The inconclusive treatment modes are described in Section 17.4.2.1.5, “Inconclusive Match.
3–8	CNR	Configured Number of Stateful Rules (in multiples of 256 rules). Valid range: 0–32 (= 0–8192 rules)
9–12	SS	Context size per session: 0:reserved 1:32B 2:64B 3:128B 4:256B 5:512B 6:1KB 7:2KB 8:4KB 9:8KB 10:16KB 11:32KB 12–15:invalid
13	ESR	1 End of SUI Simple Report Enabled 0 End of SUI Simple Report Disabled
14–31	EOSRP	End of SUI Reaction Linked List Pointer. This pointer represents the head of the reaction linked list that is to be traversed by the SRE for every End of SUI event. The pointer is an index of a 128-byte-block relative to the start of the Pattern Descriptor and Stateful Rule table. When this field is set to zero, it indicates that there is no reaction linked-list to be traversed for End of SUI events.

17.3.4.8 SRE Rule Reset Vector 0–7 (SRRV0–7)

The SRRV0–7 registers are used to specify the reset mask for a set of 256 Stateful rules. SSRV0 specifies rules 0–31, SSRV1 specifies rules 32–63, ...etc, SSRV7 specifies rules 224–255 within a 32-byte entry in the Stateful Rules Summary area of a session context entry.

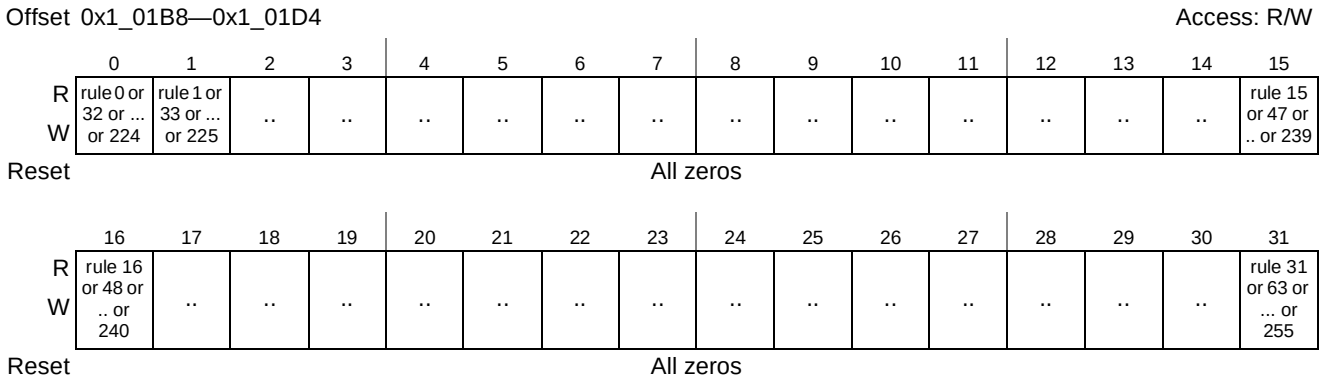


Figure 17-36. SRE Rule Reset Vector 0–7 Register Format

Table 17-39. SRE Rule Reset Vector 0–7 Register Field Descriptions

Bits	Name	Description
0–31	—	Rule reset mask bits 0: do not reset rule 1: reset rule

17.3.4.9 SRE Rule Reset First Index (SRRFI)

The SRRFI register is used to specify the first 32 byte entry in the SRE Context table to apply the SRE Rule Reset Vectors (see Section 17.3.4.8, “SRE Rule Reset Vector 0–7 (SRRV0–7)). This index should point to an entry within the Stateful Rules Summary area (first entry of Stateful Rules Summary area corresponds to Stateful rules number 0–255, second entry corresponds to Stateful rules number 255–511 and so on) of the first session context entry to be reset. This index is specified in 32 byte quantity relative to the SCBAR address.

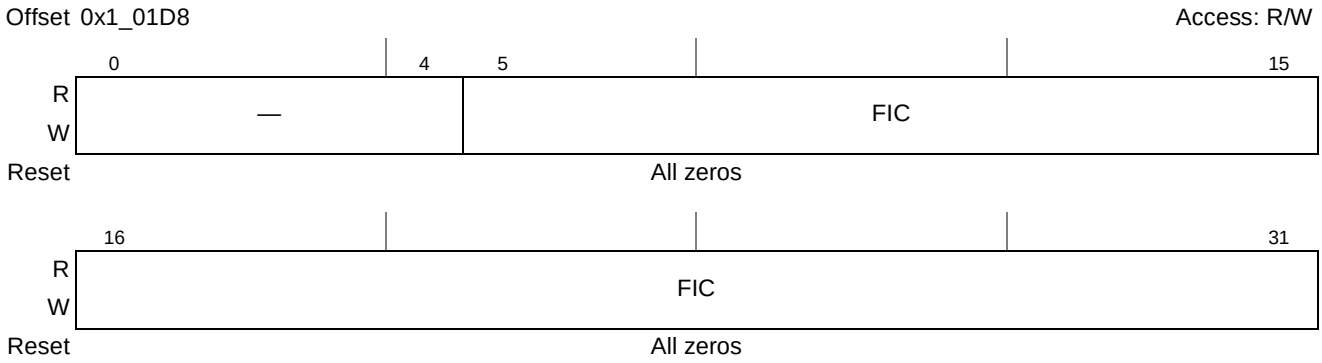


Figure 17-37. SRE Rule Reset First Index Register Format

Table 17-40. SRE Rule Reset First Index Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–31	FIC	First Index in the SRE Context table to apply the SRE Rule Reset Vectors (must be an index relative to SCBAR in 32-byte quantity).

17.3.4.10 SRE Rule Reset 32-Byte Increment (SRRI)

The SRRI register is used to specify the number of 32 byte to increment after each step of the reset operation. Typically this value should be the size of the session context.

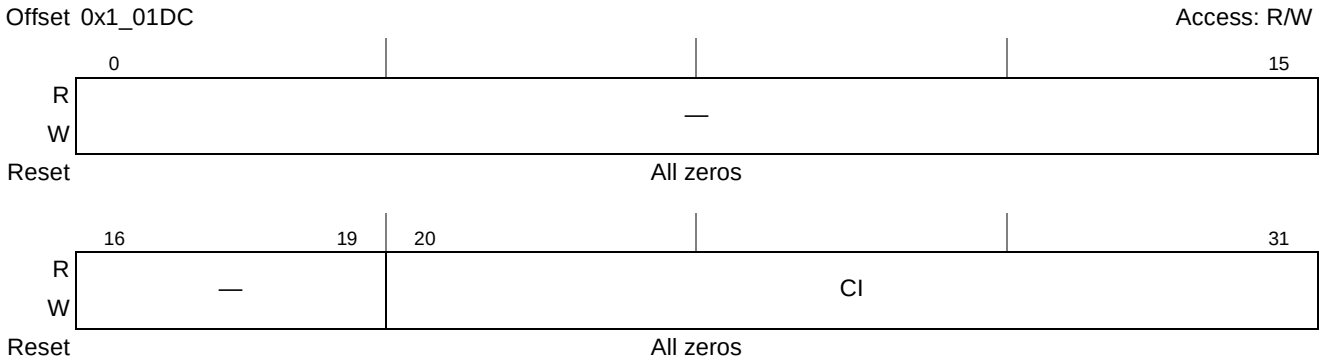


Figure 17-38. SRE Rule Reset 32-Byte Increment Register Format

Table 17-41. SRE Rule Reset 32-Byte Increment Register Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	CI	Specifies the number of 32-byte increments after each step of the reset operation.

17.3.4.11 SRE Rule Reset Repetitions (SRRR)

The SRRR register is used to specify the number of times that the reset operation should cycle. Starting at the SRRFI, the reset operation applies the SRRV0–7 registers, then increment the index by SRRI, SRRR times. Typically this value should be the number of sessions in the system. This register must be written last to start the reset operation. Subsequently reading a value of 0 from this register indicates that the reset operation is finished. The SRRR register also includes an error bit that indicates if an error in the rule reset operation has occurred. The error bit is read-only, and clear on read.

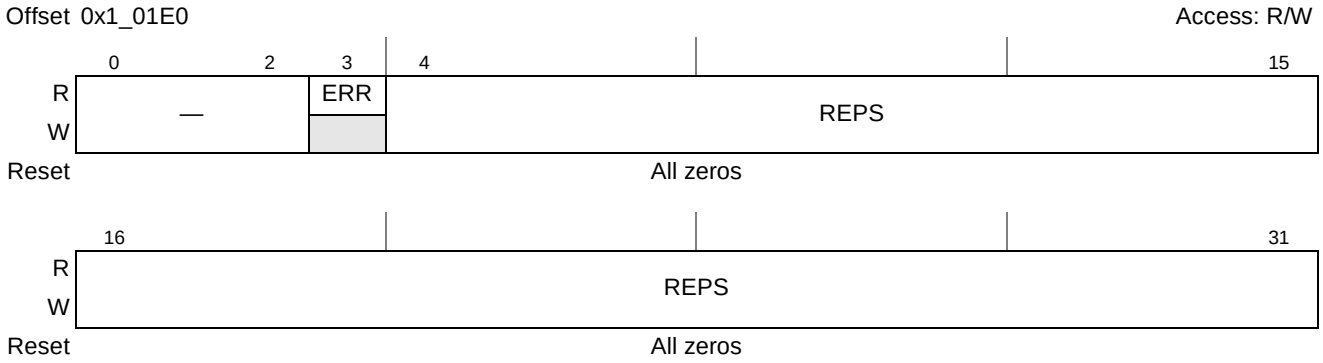


Figure 17-39. SRE Rule Reset Repetitions Register Format

Table 17-42. SRE Rule Reset Repetitions Register Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	ERR	Set if an error has occurred in the rule reset operation. This bit clears on a read of this register.
4–31	REPS	Number of repetitions for the reset rule operation.

17.3.4.12 SRE Rule Reset Work Configuration (SRRWC)

The SRRWC register is used to specify the priority of the reset operation within the Pattern Matcher block.

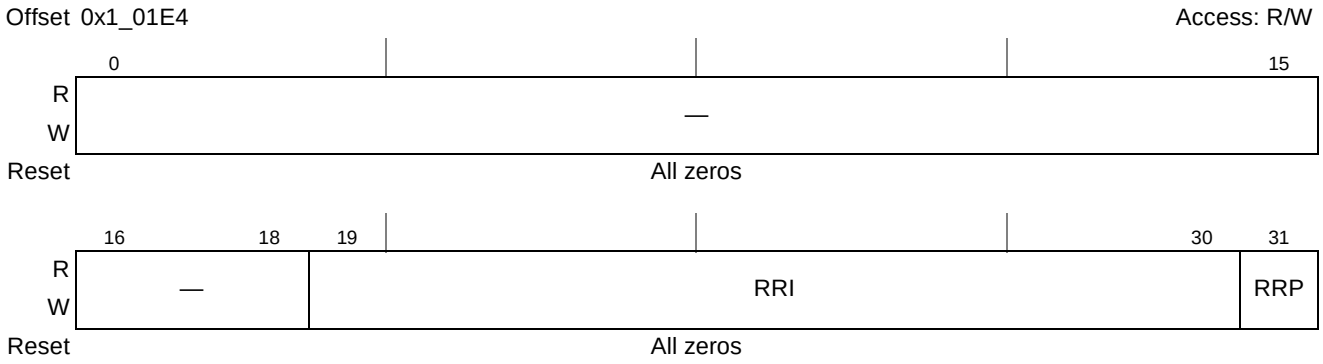


Figure 17-40. SRE Rule Reset Work Configuration Register Format

Table 17-43. SRE Rule Reset Work Configuration Register Field Descriptions

Bits	Name	Description
0–18	—	Reserved

Table 17-43. SRE Rule Reset Work Configuration Register Field Descriptions (continued)

Bits	Name	Description
19–30	RRI	Reset Rule Interval—A countdown cycle timer value. This many cycles must elapse before the next reset rule cycle is permitted. A low value means the reset process finishes quickly, but may use a high amount of system memory bandwidth. A high value means the reset process takes longer, but uses a low amount of system memory bandwidth.
31	RRP	Reset Rule Priority 0: incoming scanned data reaction requests have priority over the reset process 1: the reset process has priority over incoming scanned data reaction requests

17.3.4.13 SRE Free Running Counter Configuration (SFRCC)

The SRRWC register is used to set the prescaler value for the SRE Instruction Operand register $C free running counter.

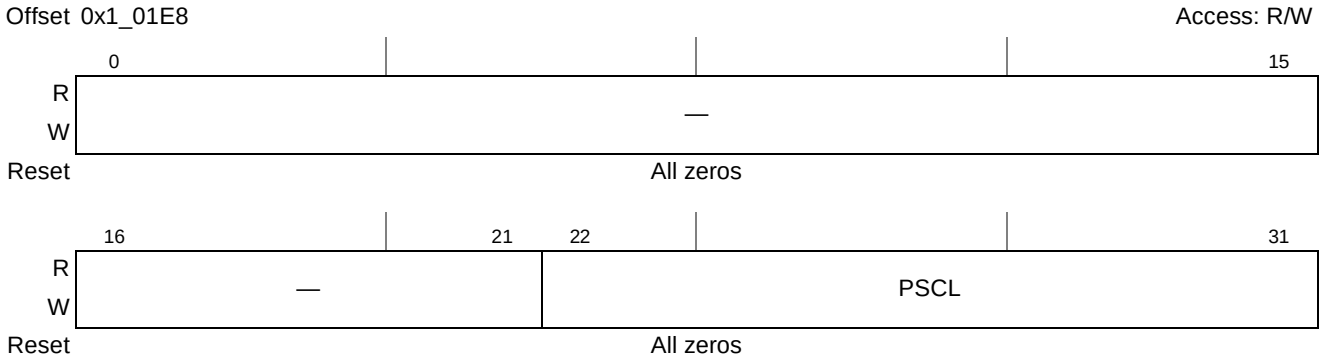


Figure 17-41. SRE Free Running Counter Configuration Register Format

Table 17-44. SRE Free Running Counter Configuration Register Field Descriptions

Bits	Name	Description
0–18	—	Reserved
22–31	PSCL	Prescaler Value—must be chosen such that the SRE Instruction Operand register $C results in a 1 usec counter. For example, if clock frequency is 333Mhz, PSCL must be set to 333.

17.3.4.14 SRE Error Configuration 1 (SEC1)

The SEC1 register configures various parameters relating to error handling in the SRE.

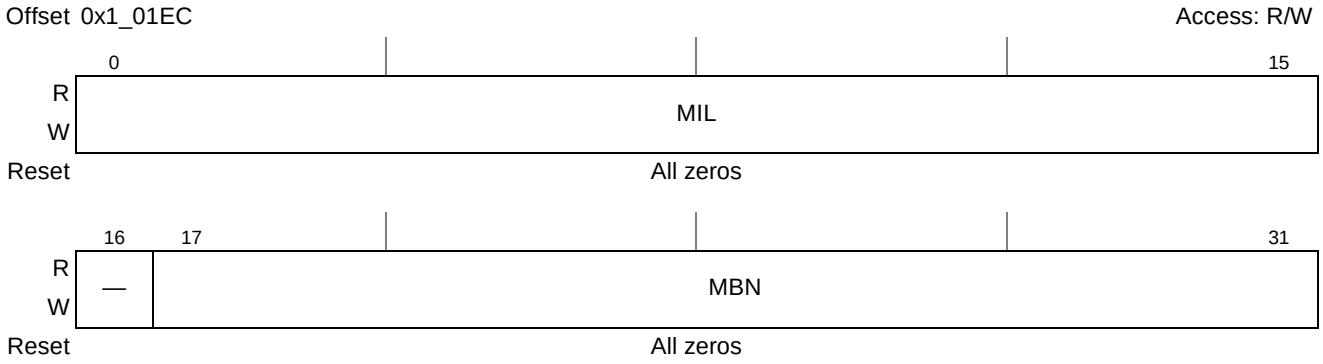


Figure 17-42. SRE Error Configuration 1 Register Format

Table 17-45. SRE Error Configuration 1 Register Field Descriptions

Bits	Name	Description
0–15	MIL	Maximum number of SRE instructions to be executed per reaction.
16	—	Reserved
17–31	MBN	Maximum number of Reaction Head blocks to be traversed per pattern match event (for example, matched pattern or End of SUI event).

17.3.4.15 SRE Error Configuration 2 (SEC2)

The SEC2 register configures various parameters relating to error handling in the SRE.

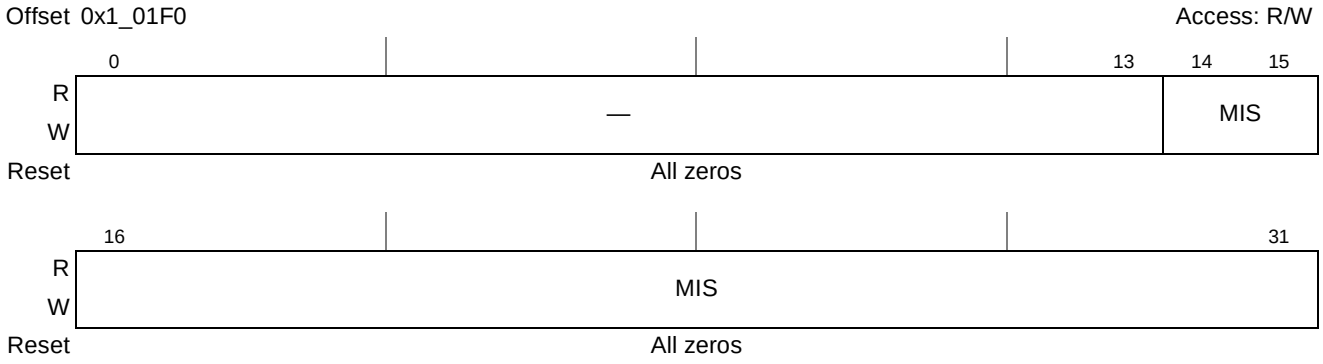


Figure 17-43. SRE Error Configuration 2 Register Format

Table 17-46. SRE Error Configuration 2 Register Field Descriptions

Bits	Name	Description
0–13	—	Reserved
14–31	MIS	Maximum allocated index into the Pattern Matcher and Stateful Rule table available to the SRE.

17.3.4.16 SRE Error Configuration 3 (SEC3)

The SEC3 register configures various parameters relating to error handling in the SRE.

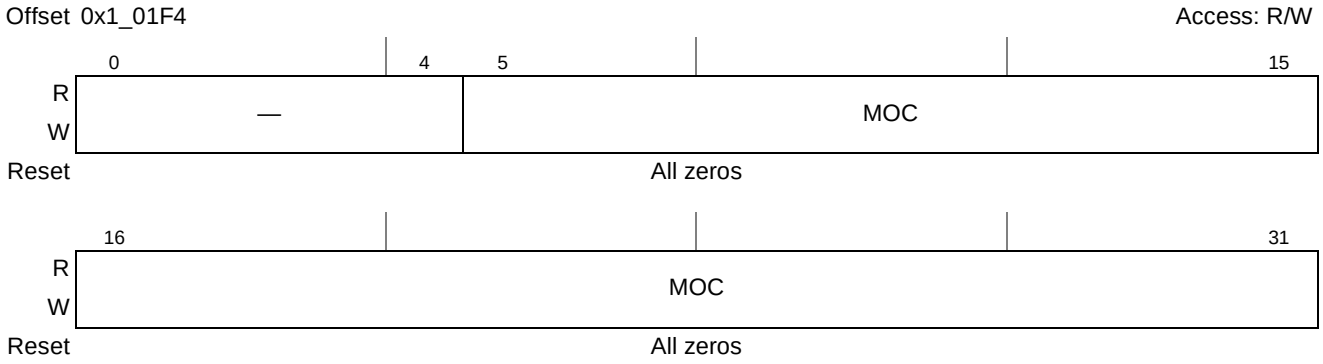


Figure 17-44. SRE Error Configuration 3 Register Format

Table 17-47. SRE Error Configuration 3 Register Field Descriptions

Bits	Name	Description
0	—	Reserved
5–31	MOC	Maximum allocated 32-byte offset into SRE Context table.

17.3.5 Deflate Engine Registers

17.3.5.1 Statistic—Deflate Bytes Consumed (STDBC)

The STDBC register counts the number of input bytes consumed by the Deflate Engine. It does not count bytes that are pass-through, only Deflate, GZIP or ZLIB. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

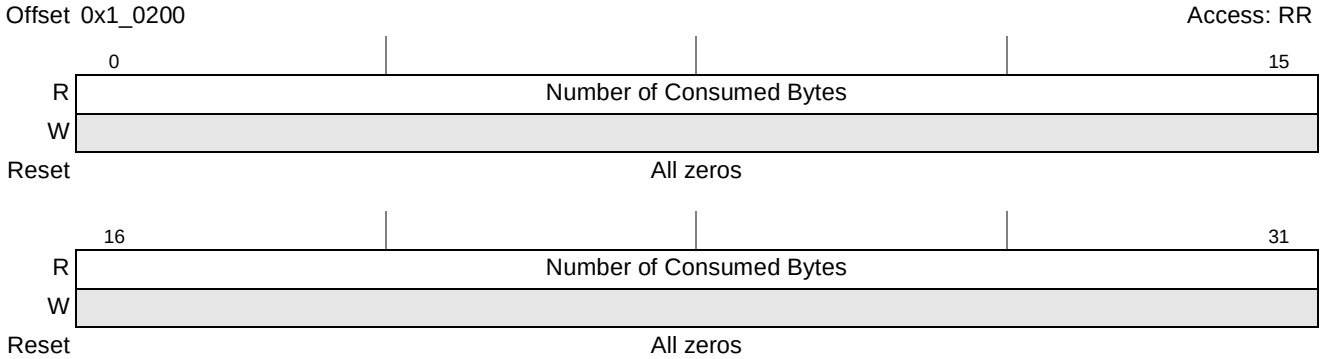


Figure 17-45. STDBC Register Format

Table 17-48. STDBC Register Field Descriptions

Bits	Name	Description
0–31	—	Number of consumed bytes.

17.3.5.2 Statistic—Deflate Bytes Produced (STDBP)

The STDBP register counts the number of output bytes produced by the Deflate Engine. It does not count bytes that are pass-through, only Deflate, GZIP or ZLIB. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

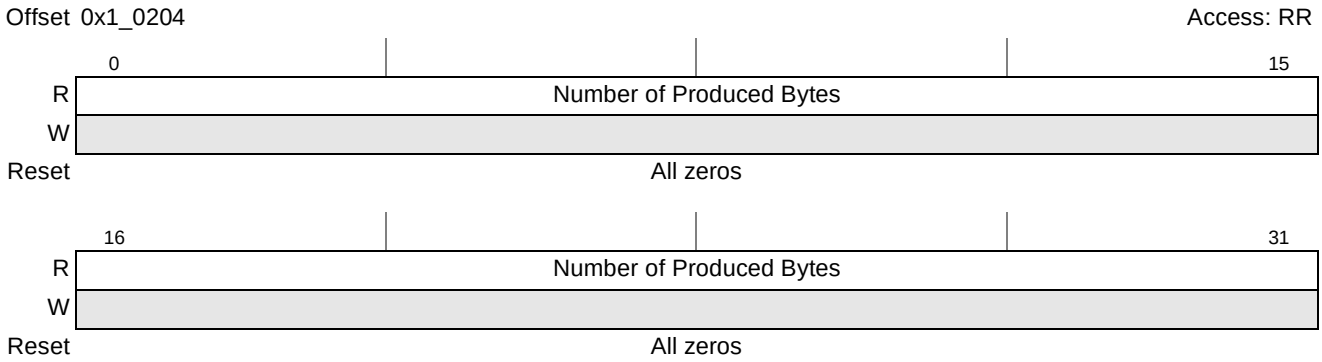


Figure 17-46. STDBP Register Format

Table 17-49. STDBP Register Field Descriptions

Bits	Name	Description
0–31	—	Number of produced bytes.

17.3.5.3 Statistic—Deflate Work Units Consumed (STDWC)

The STDWC register counts the number of input work units consumed by the Deflate Engine. It does not count work units that are passed-through, only Deflate, GZIP or ZLIB. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

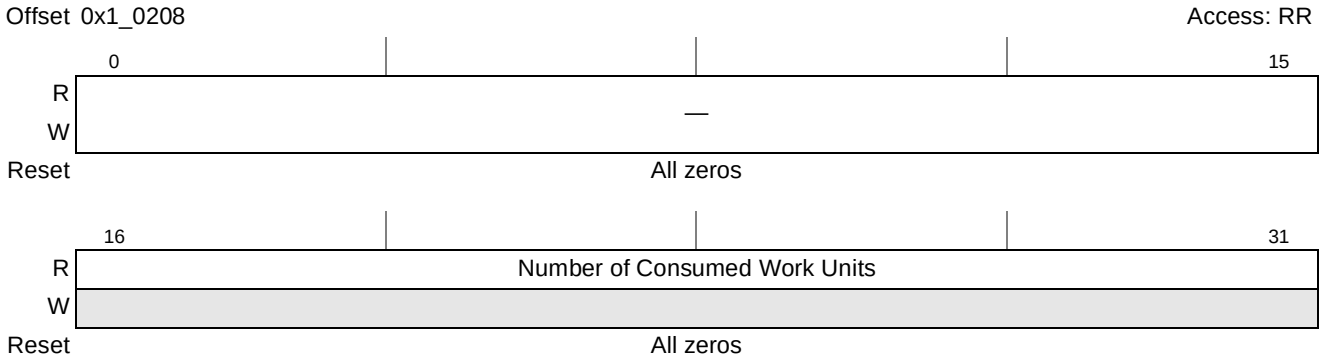


Figure 17-47. STDWC Register Format

Table 17-50. STDWC Register Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	—	Number of consumed work units.

17.3.5.4 Deflate Bytes Produced Per Work Unit Limit (DBPPWL)

The DBPPWL register limits the quantity of output generated by the decompression process to limit the adverse effects of a “decompression bomb”. A decompression bomb is a specially crafted compressed payload that generates a maximal data expansion ratio. In the example of Deflate this maximum ratio is approximately 1000:1 since in the worst case each byte could generate over 1 Kbyte of data on the output. This would result in a very effective denial-of-service attack.

If decompression of a work unit exceeds the configured limit the decompression **may** be aborted and the error is reported with the “Deflate Output Limited Failure” Notification FIFO exception code set in the Notification FIFO entry produced for that work unit (see Table 17-156). If the DBPPWL is exceeded by more than 285*2=570 bytes decompression **is** aborted and the error reported as in the former case. This error doess not cause a channel to halt operation.

The DBPPWL should be set to a value that is equal or smaller than the value set in the NDLL[0..3] register (see Section 17.3.8.24, “Notification Deflate Length Limit Register Channel 0 (NDLL0)) in the DMA Engine such that if the DMA Engine exceeds its configured data output limit and starts throwing away data the Deflate Engine is not performing (and generating) unnecessary work.

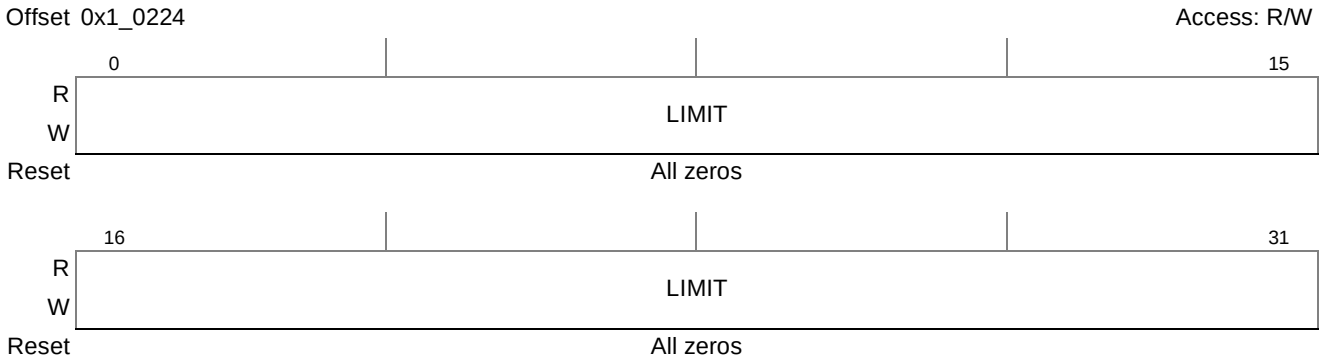


Figure 17-48. DBPPWL Register Format

Table 17-51. DBPPWL Register Field Descriptions

Bits	Name	Description
0–31	LIMIT	Set the number of bytes that a work unit is allowed to produce before the processing is aborted. Does not affect PASSTHRU type work units. if set to 0, this mechanism is disabled.

17.3.6 Free Buffer Manager Common Control Registers

This section describes the Free Buffer Manager (FBM) common control registers. FBM common control registers are written to initialize the eight free buffer lists and assign them as virtual Free Buffer Lists A and B to each of the four DMA Engine channels.

17.3.6.1 Free Buffer List Buffer Size Registers (FBL0SIZE to FBL7SIZE)

The FBL0SIZE to FBL7SIZE registers are used to specify the size of buffers in use on Free Buffer Lists 0 through 7 (physical free buffer lists). Free buffers are restricted to being multiples of 32 bytes in size, with a minimum buffer size of 128 bytes and a maximum buffer size of 64 Kbytes.

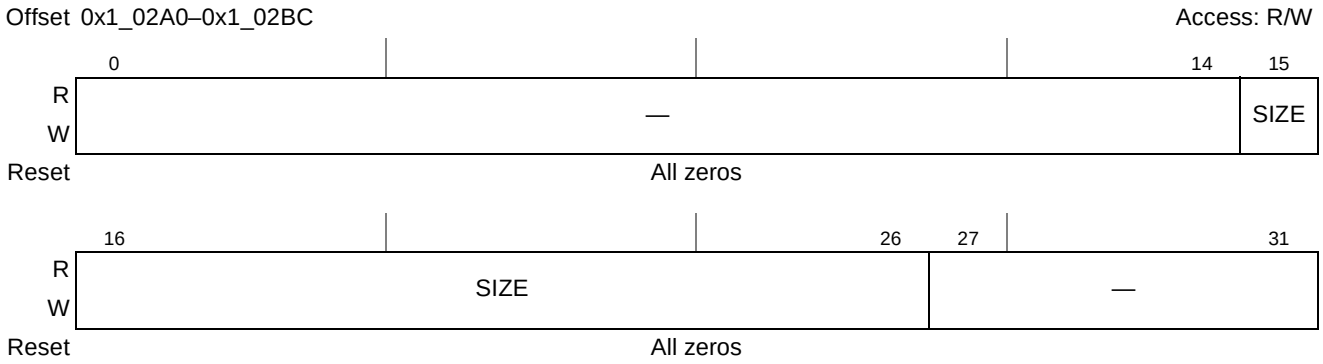


Figure 17-49. Free Buffer List Buffer Size Register Format

Table 17-52. Free Buffer List Buffer Size Register Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	RIF	Read In Flight. When read as 1 indicates that the FBM is waiting for outstanding read requests to return before completing an update of this free buffer list. FBM assignment and configuration registers should not be written to until all client channels of this list have been put in reset and this bit is read as 0.
8–14	—	Reserved, should be cleared.
15–26	SIZE	Size. This field specifies the size of free buffers for this free buffer list in 32 byte quantities. The minimum buffer size is 128 bytes (SIZE=4), the maximum buffer size is 64K bytes (SIZE=2048).
27–31	—	Reserved, should be cleared.

17.3.6.2 Free Buffer List Assignment A Register (FBLAAR)

The FBLAAR register is used to specify and control the assignment of hardware managed free buffer lists to DMA Engine channels for the purposes of buffering data. Each DMA Engine channel offers two virtual free buffer lists: A and B. This register assigns one of eight physical free buffer lists to each DMA Engine channel as its Virtual Free Buffer List A. It is legal to assign the same physical free buffer list as Virtual Free Buffer List A for more than one channel. It is not safe to modify list assignments once the Pattern Matcher is operating; this register is intended to be configured once at initialization and remain static during operation.

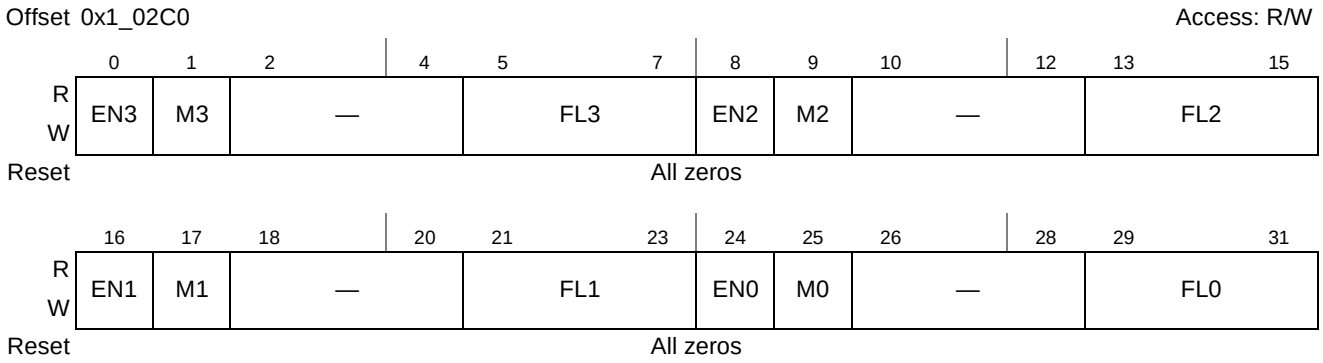


Figure 17-50. Free Buffer List Assignment A Register Format

Table 17-53. Free Buffer List Assignment A Register Field Descriptions

Bits	Name	Description
0	EN3	Free Buffer List A Enable Channel 3. When set, enables DMA Engine use of the specified free buffer list.
1	M3	Master Indication Channel 3. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_A3 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
2–4	—	Reserved, should be cleared.
5–7	FL3	Free Buffer List A Channel 3. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 3. All codings are valid.
8	EN2	Free Buffer List A Enable Channel 2. When set, enables DMA Engine use of the specified free buffer list.
9	M2	Master Indication Channel 2. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_A2 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
10–12	—	Reserved, should be cleared.
13–15	FL2	Free Buffer List A Channel 2. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 2. All codings are valid.
16	EN1	Free Buffer List A Enable Channel 1. When set, enables DMA Engine use of the specified free buffer list.
17	M1	Master Indication Channel 1. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_A1 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
18–20	—	Reserved, should be cleared.
21–23	FL1	Free Buffer List A Channel 1. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 1. All codings are valid.
24	EN0	Free Buffer List A Enable Channel 0. When set, enables DMA Engine use of the specified free buffer list.
25	M0	Master Indication Channel 0. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_A0 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
26–28	—	Reserved, should be cleared.
29–31	FL0	Free Buffer List A Channel 0. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 0. All codings are valid.

17.3.6.3 Free Buffer List Assignment B Register (FBLABR)

The FBLABR register is used to specify and control the assignment of hardware managed free buffer lists to DMA Engine channels for the purposes of buffering data. This register assigns one of eight physical free buffer lists to each DMA Engine channel as its Virtual Free Buffer List B. It is legal to assign the same physical free buffer list as Virtual Free Buffer List B for more than one channel. It is not safe to modify list assignments once the Pattern Matcher is operating; this register is intended to be configured once at initialization and remain static during operation.

Offset 0x1_02C4 Access: R/W

	0	1	2	4	5	7	8	9	10	12	13	15
R W	EN3	M3	—		FL3		EN2	M2	—		FL2	
Reset	All zeros											

	16	17	18	20	21	23	24	25	26	28	29	31
R W	EN1	M1	—		FL1		EN0	M0	—		FL0	
Reset	All zeros											

Figure 17-51. Free Buffer List Assignment B Register Format

Table 17-54. Free Buffer List Assignment B Register Field Descriptions

Bits	Name	Description
0	EN3	Free Buffer List B Enable Channel 3. When set, enables DMA Engine use of the specified free buffer list.
1	M3	Master Indication Channel 3. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_B3 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
2–4	—	Reserved, should be cleared.
5–7	FL3	Free Buffer List B Channel 3. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 3. All codings are valid.
8	EN2	Free Buffer List B Enable Channel 2. When set, enables DMA Engine use of the specified free buffer list.
9	M2	Master Indication Channel 2. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_B2 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
10–12	—	Reserved, should be cleared.
13–15	FL2	Free Buffer List B Channel 2. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 2. All codings are valid.
16	EN1	Free Buffer List B Enable Channel 1. When set, enables DMA Engine use of the specified free buffer list.
17	M1	Master Indication Channel 1. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_B1 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.
18–20	—	Reserved, should be cleared.
21–23	FL1	Free Buffer List B Channel 1. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 1. All codings are valid.
24	EN0	Free Buffer List B Enable Channel 0. When set, enables DMA Engine use of the specified free buffer list.
25	M0	Master Indication Channel 0. This bit is provided for the convenience of driver software. Its state (0 or 1) is reflected in the per-channel FBL_BS_B0 register. If a single physical free buffer list is being assigned to more than one channel's virtual free buffer list it may be useful for initialization software to designate one channel as the "master" and indicate so using this bit.

Table 17-54. Free Buffer List Assignment B Register Field Descriptions (continued)

Bits	Name	Description
26–28	—	Reserved, should be cleared.
29–31	FL0	Free Buffer List B Channel 0. This field specifies the physical free buffer list that the DMA Engine uses to buffer output on channel 0. All codings are valid.

17.3.6.4 Free Buffer Manager Control Register (FBM_CR)

The FBM_CR register is used is used to provide software with control over cache aware attributes and system memory transaction priority for the Free Buffer Manager within the Pattern Matcher Block.

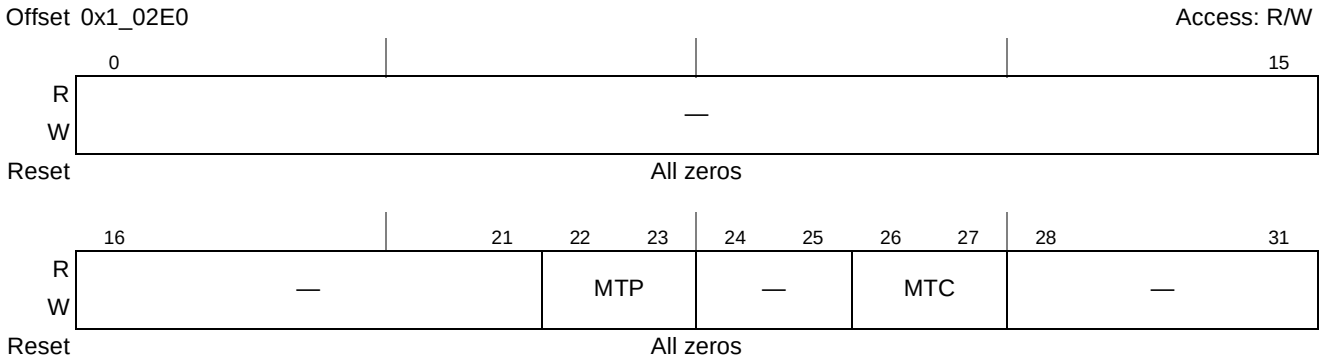


Figure 17-52. Free Buffer Manager Control Register Format

Table 17-55. Free Buffer Manager Control Register Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
22–23	MTP	Memory Transaction Priority. Specifies the transaction priority applied to FBM memory transactions.
24–25	—	Reserved, should be cleared.
26–27	MTC	Memory Transaction Cache Aware. Controls cache aware attributes during FBM transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved
28–30	—	Reserved, should be cleared.
31	—	Reserved, should be cleared.

17.3.7 Memory Interface Arbiter Registers

17.3.7.1 MIA Byte Count Register (MIA_BYC)

The MIA_BYC register counts the number of bytes transferred for all memory transactions that reach the system bus. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

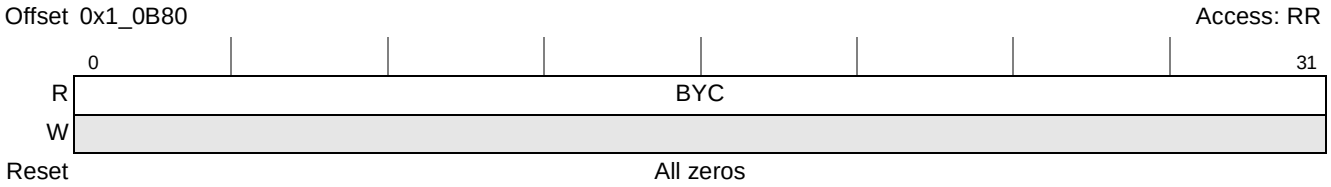


Figure 17-53. MIA Byte Count Register (MIA_BYC)

Table 17-56. MIA_BYC Field Descriptions

Bits	Name	Description
0–31	BYC	Total system bus transferred Byte Count. Byte count for all transactions that reach the system bus. Transactions from each client read and write port attached to MIA can be included or omitted from this count by settings in the MIA_CE register. Client read transactions that are satisfied by the local snoop interface, and client write transactions that are cancelled by a local snoop hit while pending, are not included in this count. Reading this register resets the count to 0 after the read completes. When the counter reaches its max value, it rolls over to 0. If this counter rolls over, an overflow bit is set in the SCOS register.

17.3.7.2 MIA Block Count Register (MIA_BLC)

The MIA_BLC register counts the number of bus aligned memory block accesses for all memory transactions that reach the system bus. This register is clear on read. If this counter rolls over, an overflow bit is set in the SCOS register.

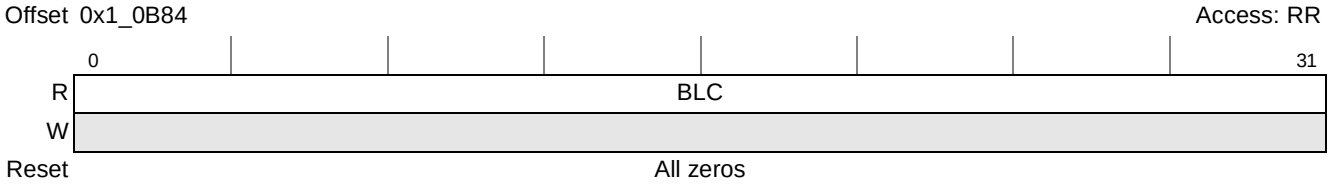


Figure 17-54. MIA Block Count Register (MIA_BLC)

Table 17-57. MIA_BLC Field Descriptions

Bits	Name	Description
0–31	BLC	Total system bus aligned Block access Count. Counts the number of aligned memory blocks accessed for all transactions that reach the system bus. For example, when counting 32 byte aligned blocks, a 1 byte transaction at any address increments BLC by one, a 32 byte transaction with address[4:0] = 0 increments BLC by one, but a 2 byte transaction with address[4:0] = 0x1F increments BLC by two, since it spans two 32 byte aligned blocks. Transactions from each client read and write port attached to MIA can be included or omitted from this count by settings in the MIA_CE register. MIA_CE[BLC_SIZE] controls the size of the aligned blocks counted by MIA_BLC. Client read transactions that are satisfied by the local snoop interface, and client write transactions that are cancelled by a local snoop hit while pending, are not included in this count. Reading this register resets the count to 0 after the read completes. When the counter reaches its max value, it rolls over to 0. If this counter rolls over, an overflow bit is set in the SCOS register.

17.3.7.3 MIA Count Enable Register (MIA_CE)

The MIA_CE register is used specify the client read and write ports that are included in the MIA_BLC and MIA_BYC counts. It also used to specify the size of the aligned blocks counted by MIA_BLC.

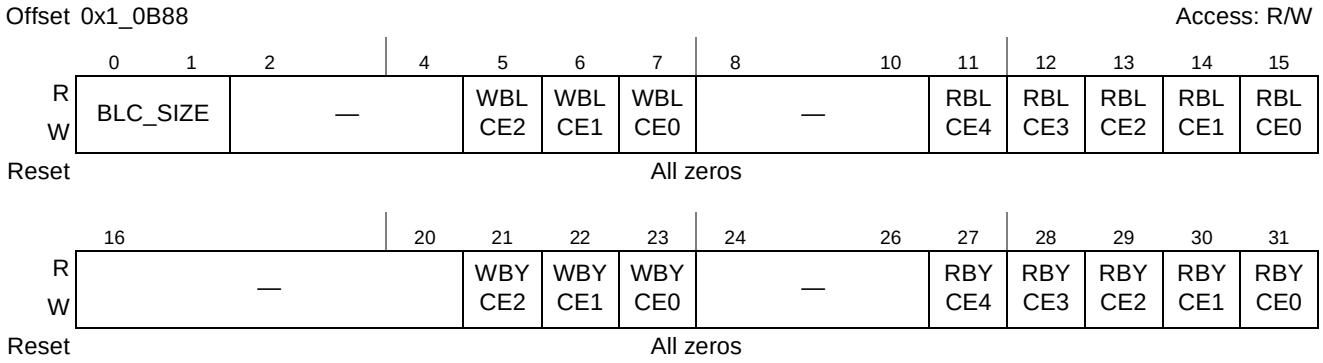


Figure 17-55. MIA Count Enable Register (MIA_CE)

Table 17-58. MIA_CE Field Descriptions

Bits	Name	Description
0–1	BLC_SIZE	Block Count Size. Determines the size of aligned memory block accesses that are counted by MIA_BLC. 00 MIA_BLC counts 16 byte aligned blocks. 01 MIA_BLC counts 32 byte aligned blocks. 10 MIA_BLC counts 64 byte aligned blocks. 11 MIA_BLC counts 128 byte aligned blocks.
2–4	—	Reserved, read as 0.
5	WBLCE2	Write Port Block Count Enable 2. This is the write port attached to the FBM. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.

Table 17-58. MIA_CE Field Descriptions (continued)

Bits	Name	Description
6	WBLCE1	Write Port Block Count Enable 1. This is the write port attached to the SRE for writing into the SRE Context table. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
7	WBLCE0	Write Port Block Count Enable 0. This is the write port attached to the DMA Engine. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
8–10	—	Reserved, read as 0.
11	RBLCE4	Read Port Block Count Enable 4. This is the read port attached to the SRE for reading into the Pattern Description and Stateful Rule table. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
12	RBLCE3	Read Port Block Count Enable 3. This is the read port attached to the DXE. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
13	RBLCE2	Read Port Block Count Enable 2. This is the read port attached to the FBM. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
14	RBLCE1	Read Port Block Count Enable 1. This is the read port attached to the SRE for reading into the SRE Context table. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
15	RBLCE0	Read Port Block Count Enable 0. This is the read port attached to the DMA Engine. 0 Transactions on this port which reach the system bus are not included in the MIA_BLC total block count. 1 Transactions on this port which reach the system bus are included in the MIA_BLC total block count.
16–20	—	Reserved, read as 0.
21	WBYCE2	Write Port Byte Count Enable 2. This is the write port attached to the FBM. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.

Table 17-58. MIA_CE Field Descriptions (continued)

Bits	Name	Description
22	WBYCE1	Write Port Byte Count Enable 1. This is the write port attached to the SRE for writing into the SRE Context table. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
23	WBYCE0	Write Port Byte Count Enable 0. This is the write port attached to the DMA Engine. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
24–26	—	Reserved, read as 0.
27	RBYCE4	Read Port Byte Count Enable 4. This is the read port attached to the SRE for reading into the Pattern Description and Stateful Rule table. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
28	RBYCE3	Read Port Byte Count Enable 3. This is the read port attached to the DXE. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
29	RBYCE2	Read Port Byte Count Enable 2. This is the read port attached to the FBM. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
30	RBYCE1	Read Port Byte Count Enable 1. This is the read port attached to the SRE for reading into the SRE Context table. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.
31	RBYCE0	Read Port Byte Count Enable 0. This is the read port attached to the DMA Engine. 0 Transactions on this port which reach the system bus are not included in the MIA_BYC total byte count. 1 Transactions on this port which reach the system bus are included in the MIA_BYC total byte count.

17.3.7.4 MIA Control Register (MIA_CR)

The MIA_CR register is used to configure various client read and write ports attributes.

Offset 0x1_0BA0 Access: R/W

	0–4	5	6	7	8–12	13	14	15
R	—	WCE2	WCE1	WCE0	—	LSE2	LSE1	LSE0
W								
Reset	All zeros							

	16–20	21	22	23	24–26	27	28	29	30	31
R	—	WD2	WD1	WD0	—	RD4	RD3	RD2	RD1	RD0
W										
Reset	All zeros									

Figure 17-56. MIA Control Register (MIA_CR)

Table 17-59. MIA_CR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, read as 0.
5	WCE2	Write Cancellation Enable 2. Controls write cancellation for the read/write port pair attached to the FBM, for example, write port 2 and read port 2. 0 Write cancellation is disabled. 1 Write cancellation is enabled, if LSE2 is also set to 1, and MIA_ACR[AM2] = 1. This setting is invalid, FBM does not support write cancellation since it does not support local read_priority_weighted mode.
6	WCE1	Write Cancellation Enable 1. Controls write cancellation for the read/write port pair attached to the SRE for accessing the SRE Context table, for example, write port 1 and read port 1. 0 Write cancellation is disabled. 1 Write cancellation is enabled, if LSE1 is also set to 1, and MIA_ACR[AM1] = 1.
7	WCE0	Write Cancellation Enable 0. Controls write cancellation for the read/write port pair attached to the DMA Engine, for example, write port 0 and read port 0. 0 Write cancellation is disabled. 1 Write cancellation is enabled, if LSE0 is also set to 1, and MIA_ACR[AM0] = 1. This setting is invalid, DMA Engine does not support write cancellation since it does not support read_priority_weighted mode.
8–12	—	Reserved, read as 0.
13	LSE2	Local Snoop Enable 2. Controls local snoop operation for the read/write port pair attached to the FBM, for example, write port 2 and read port 2. 0 Local snoop is disabled. 1 Local snoop is enabled. This setting is invalid, FBM does not support local snoop since it does not support read_priority_weighted mode.
14	LSE1	Local Snoop Enable 1. Controls local snoop operation for the read/write port pair attached to the SRE for accessing the SRE Context table, for example, write port 1 and read port 1. 0 Local snoop is disabled. 1 Local snoop is enabled.

Table 17-59. MIA_CR Field Descriptions (continued)

Bits	Name	Description
15	LSE0	Local Snoop Enable 0. Controls local snoop for the read/write port pair attached to the DMA Engine, for example, write port 0 and read port 0. 0 Local snoop is disabled. 1 Local snoop is enabled. This setting is invalid, DMA Engine does not support local snoop since it does not support read_priority_weighted mode.
16–20	—	Reserved, read as 0.
21	WD2	Write Port Disable 2. This is the write port attached to the FBM. 0 This memory access port is enabled. 1 This memory access port is disabled.
22	WD1	Write Port Disable 1. This is the write port attached to the SRE for writing into the SRE Context table. 0 This memory access port is enabled. 1 This memory access port is disabled.
23	WD0	Write Port Disable 0. This is the write port attached to the DMA Engine. 0 This memory access port is enabled. 1 This memory access port is disabled.
24–26	—	Reserved, read as 0.
27	RD4	Read Port Disable 4. This is the read port attached to the SRE for reading into the Pattern Description and Stateful Rule table 0 This memory access port is enabled. 1 This memory access port is disabled.
28	RD3	Read Port Disable 3. This is the read port attached to the DXE. 0 This memory access port is enabled. 1 This memory access port is disabled.
29	RD2	Read Port Disable 2. This is the read port attached to the FBM. 0 This memory access port is enabled. 1 This memory access port is disabled.
30	RD1	Read Port Disable 1. This is the read port attached to the SRE for reading into the SRE Context table. 0 This memory access port is enabled. 1 This memory access port is disabled.
31	RD0	Read Port Disable 0. This is the read port attached to the DMA Engine. 0 This memory access port is enabled. 1 This memory access port is disabled.

17.3.7.5 MIA Arbitration Control Register (MIA_ACR)

The MIA_ACR register is used to configure arbitration related attributes.

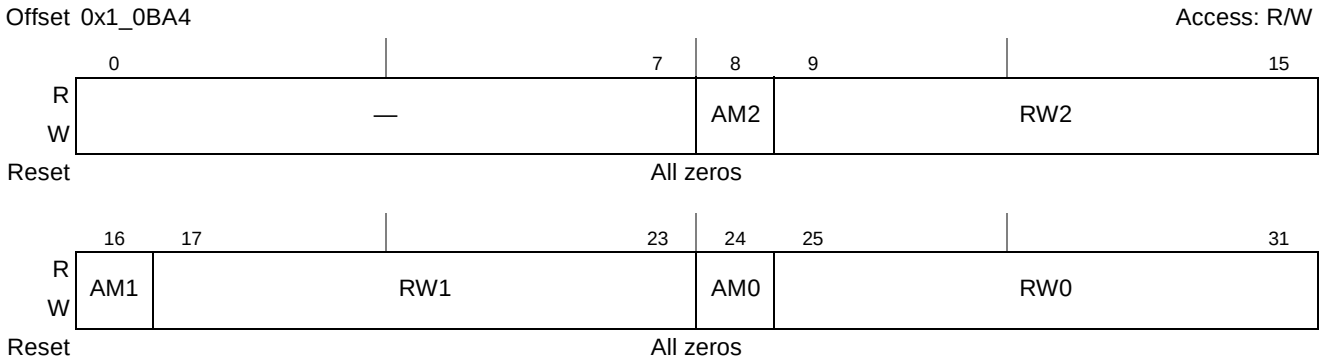


Figure 17-57. MIA Arbitration Control Register (MIA_ACR)

Table 17-60. MIA_ACR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, read as 0.
8	AM2	Arbitration Mode on the read/write port pair attached to the FBM. 0 serialized. See Section 17.4.7.2.1, “Serialized Mode”. 1 read_priority_weighted. See Section 17.4.7.2.2, “Read Priority Weighted Mode. This setting is invalid, FBM does not support read_priority_weighted mode.
9–15	RW2	Read Weight on the read/write port pair attached to the FBM. Used only if AM2 is set to read_priority_weighted_mode. See Section 17.4.7.2.2, “Read Priority Weighted Mode. This setting is invalid, FBM does not support read_priority_weighted mode.
16	AM1	Arbitration Mode on the read/write port pair attached to the SRE for accessing the SRE Context table. 0 serialized. See Section 17.4.7.2.1, “Serialized Mode. 1 read_priority_weighted. See Section 17.4.7.2.2, “Read Priority Weighted Mode.
17–23	RW1	Read Weight on the read/write port pair attached to the SRE for the SRE Context table. Used only if AM1 is set to read_priority_weighted_mode. See Section 17.4.7.2.2, “Read Priority Weighted Mode.
24	AM0	Arbitration Mode on the read/write port pair attached to the DMA Engine. 0 serialized. See Section 17.4.7.2.1, “Serialized Mode. 1 read_priority_weighted. See Section 17.4.7.2.2, “Read Priority Weighted Mode. This setting is invalid, DMA Engine does not support read_priority_weighted mode.
25–31	RW0	Read Weight on the read/write port pair attached to the DMA Engine. Used only if AM0 is set to read_priority_weighted_mode. See Section 17.4.7.2.2, “Read Priority Weighted Mode. This setting is invalid, DMA Engine does not support read_priority_weighted mode.

17.3.7.6 MIA Local Snoop Hit Count Registers (MIA_LSHC0/1/2)

The MIA_LSHC0/1/2 registers count the number of local snoop hits. These registers are clear on read. If any of these counters roll over, a corresponding overflow bit is set in the SCOS register.

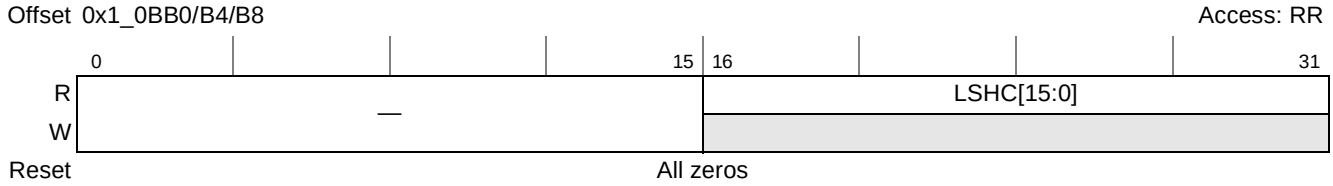


Figure 17-58. MIA Local Snoop Hit Count Registers (MIA_LSHC0/1/2)

Table 17-61. MIA_LSHC0/1/2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, read as 0.
16–31	LSHC	Local Snoop Hit Count. A 16 bit count of local snoop hits is maintained for each associated read/write port pair. Reading the LSHCn register resets the count to 0 after the read completes. When the counter reaches its max value, it rolls over to 0. If this counter rolls over, an overflow bit is set in the SCOS register. LSHC0 = Count of local snoop hits on the read/write port pair attached to the DMA Engine. Always set to 0 since DMA Engine does not support local snoop. LSHC1 = Count of local snoop hits on the read/write port pair attached to the SRE for writing into the SRE Context table. LSHC2 = Count of local snoop hits on the read/write port pair attached to the FBM. Always set to zero since FBM does not support local snoop.

17.3.7.7 MIA Cancelled Write Count Registers (MIA_CWC0/1/2)

The MIA_CWC0/1/2 registers count the number of cancelled writes. These registers are clear on read. If any of these counters roll over, a corresponding overflow bit is set in the SCOS register.

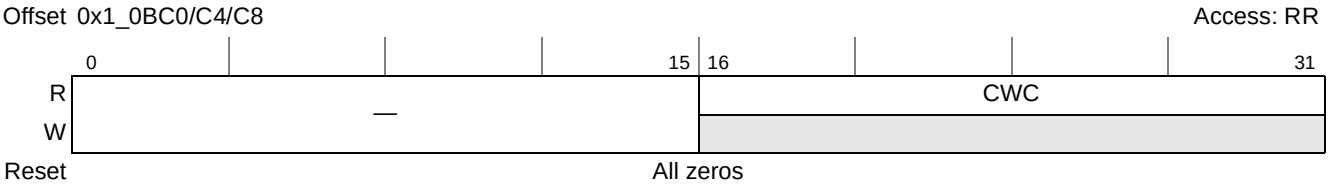


Figure 17-59. MIA Cancelled Write Count Registers (MIA_CWC0/1/2)

Table 17-62. MIA_CWC0/1/2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, read as 0.
16–31	CWC	Cancelled Write Count. A 16 bit count of cancelled writes is maintained for each associated read/write port pair. Reading the CWCn register resets the count to 0 after the read completes. When the counter reaches its max value, it rolls over to 0. If this counter rolls over, an overflow bit is set in the SCOS register. CWC0 = Count of cancelled writes on the read/write port pair attached to the DMA Engine. Always set to 0 since DMA Engine does not support write cancellation. CWC1 = Count of cancelled writes on the read/write port pair attached to the SRE for writing into the SRE Context table. CWC2 = Count of cancelled writes on the read/write port pair attached to the FBM. Always set to 0 since FBM does not support write cancellation.

17.3.7.8 PM IP Block Revision 1 Register (PM_IP_REV_1)

The PM_IP_REV_1 register is a read-only register which provides the unique IP block ID for the Pattern Matcher (PM) block, as well as major and minor revision numbers.

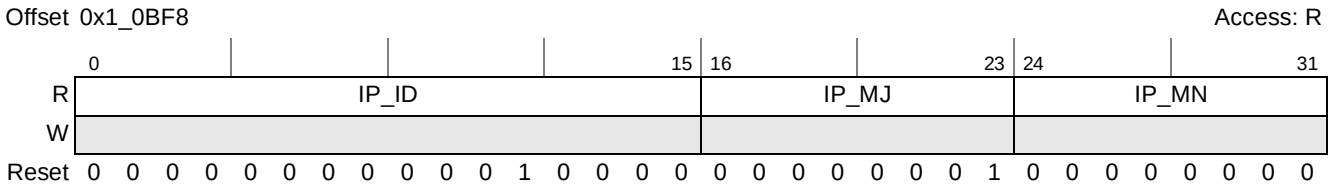


Figure 17-60. PM IP Block Revision 1Register (PM_IP_REV_1)

Table 17-63. PM_IP_REV_1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	IP Block ID. For the Patter Matcher, this value is 0x0010
16–23	IP_MJ	Major revision. This is currently set to 0x01.
24–31	IP_MN	Minor revision. This is currently set to 0x01.

17.3.7.9 PM IP Block Revision 2 Register (PM_IP_REV_2)

The PM_IP_REV_2 register is a read-only register which provides Pattern Matcher (PM) block integration and configuration options.

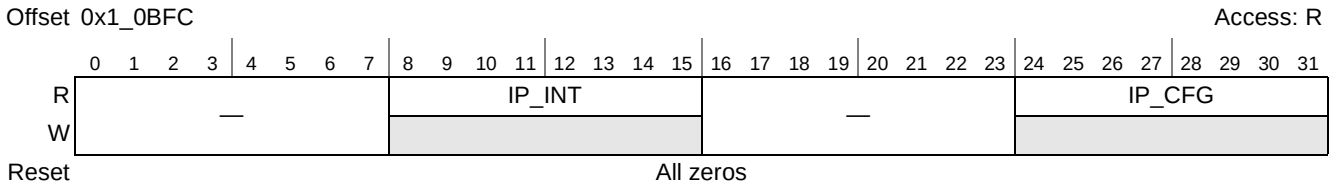


Figure 17-61. PM IP Block Revision 2 Register (PM_IP_REV_2)

Table 17-64. PM_IP_REV_2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved, read as 0.
8–15	IP_INT	Integration options. This is currently set to 0x00.
16–23	—	Reserved, read as 0.
24–31	IP_CFG	Configuration options. This is currently set to 0x00.

17.3.8 DMA Engine Channel 0 Registers

The DMA Engine control registers are divided into four per-channel register groups and one common DMA Engine control register group. This section describes the per-channel registers for channel 0. Registers for channels 1–3 are identical except for their offsets (see Table 17-4).

17.3.8.1 Interrupt Status Register 0 (ISR0)

If an event or condition occurs that sets a bit in the ISR0 register, shown in Figure 17-62, the Pattern Matcher DMA Channel 0 interrupt is generated if the corresponding bit in the IER0 register is also set and the corresponding bit in the ISDR0 register is cleared. Clearing the ISR0 bit clears the interrupt signal (as long as no other ISR0/IER0 bit pairs are set). The bit in the ISR0 register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

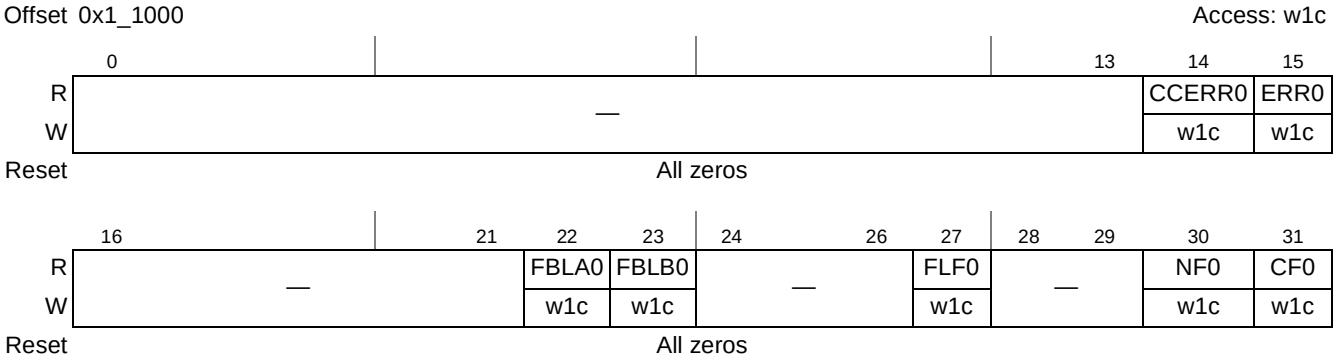


Figure 17-62. Interrupt Status Register 0 Format

Table 17-65. Interrupt Status Register 0 Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	CCERR0	Common Control Error Channel 0. This bit is asserted when a common-control error has been detected that caused processing to gracefully halt on all channels. This bit exists to inform per-channel driver software that processing on channel 0 has stopped due to a global error condition. Any work units that were in flight at the time the error was detected, and were affected by the error, have the appropriate error code (see Table 17-156) set in their Notification FIFO entry. This error code is also reflected in the CHERR0 register. 0 No error detected. 1 A Common Control error was detected.

Table 17-65. Interrupt Status Register 0 Field Descriptions (continued)

Bits	Name	Description
15	ERR0	Error Channel 0. This bit is asserted when a per-channel error has been detected that caused processing to gracefully halt on this channel. Any work units that were in flight at the time the error was detected, and were affected by the error, have the appropriate error code (Table 17-156) set in their Notification FIFO entry. This error code is also reflected in the CHERR0 register. 0 No error detected. 1 A per-channel error was detected.
16–21	—	Reserved, should be cleared.
22	FBLA0	Free Buffer List A Channel 0. This bit is asserted when the Virtual Free Buffer List A length for channel 0 has become less than the programmed threshold in the FBL_THRES_A0 register. 0 Threshold not reached. 1 Threshold reached.
23	FBLB0	Free Buffer List B Channel 0. This bit is asserted when the Virtual Free Buffer List B length for channel 0 has become less than the programmed threshold in the FBL_THRES_B0 register. 0 Threshold not reached. 1 Threshold reached.
24–26	—	Reserved, should be cleared.
27	FLF0	Free Buffer Deallocate FIFO Channel 0. This bit is asserted when the Free Buffer Deallocate FIFO fill level for channel 0 is less than the programmed threshold in the FBFIFO_THRES0 register. If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed. 0 Threshold not reached. 1 Threshold reached.
28–29	—	Reserved, should be cleared.

Table 17-65. Interrupt Status Register 0 Field Descriptions (continued)

Bits	Name	Description
30	NF0	Notification FIFO Channel 0. This bit is asserted when the Notification FIFO for channel 0 is not empty (NPI not equal to NCI). If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed unless the following two conditions are true: • The Command FIFO entry that caused the creation of this Notification FIFO entry had the “Interrupt on Notification” bit set. • The Notification FIFO fill level for channel 0 has become greater than the programmed threshold in the NFIFO_THRES0 register. Under these conditions, the Pattern Matcher DMA Channel interrupt signal is asserted immediately even if the prescribed interval set in the IICR0 register has not elapsed. 0 FIFO is empty. 1 FIFO is not empty
31	CF0	Command FIFO Channel 0. This bit is asserted when the Command FIFO for channel 0 has completed/processed commands (CCI not equal to CEI). If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed unless the following two conditions are true: • The Command FIFO entry has the “Interrupt On Completion” bit set. • The Command FIFO fill level for channel 0 has become greater than the programmed threshold in the CFIFO_THRES0 register. Under these conditions, the Pattern Matcher DMA Channel interrupt signal is asserted immediately even if the prescribed interval set in the IICR0 register has not elapsed. 0 FIFO is empty. 1 FIFO is not empty

17.3.8.2 Interrupt Enable Register 0 (IER0)

The IER0 register provides control over which possible interrupt events or conditions are allowed to generate an actual Pattern Matcher DMA Channel 0 interrupt. All implemented bits in this register are R/W. This register is cleared upon hardware reset. If the corresponding bits in both ISR0 and IER0 registers are set, the Pattern Matcher DMA Channel 0 interrupt is generated. The interrupt signal can be cleared by clearing the corresponding ISR0 bit. The state of an IER0 bit does not affect setting of the corresponding ISR0 bit, it controls whether or not setting of the ISR0 bit causes an interrupt.

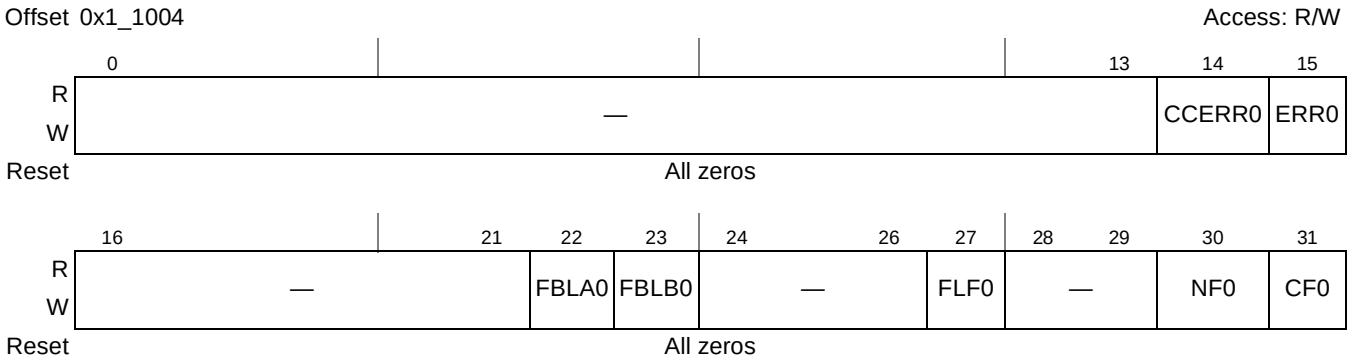


Figure 17-63. Interrupt Enable Register 0 Format

Table 17-66. Interrupt Enable Register 0 Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	CCERR0	Common Control Error Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
15	ERR0	Error Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
16–21	—	Reserved, should be cleared.
22	FBLA0	Free Buffer List A Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
23	FBLB0	Free Buffer List B Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
24–26	—	Reserved, should be cleared.
27	FLF0	Free Buffer Deallocate FIFO Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
28–29	—	Reserved, should be cleared.
30	NF0	Notification FIFO Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
31	CF0	Command FIFO Channel 0 Interrupt Enable. 0 Disable interrupt 1 Enable interrupt

17.3.8.3 Interrupt Status Disable Register 0 (ISDR0)

The ISDR0 register controls ISR0 reporting. An ISR0 bit is not set if the corresponding bit in ISDR0 is set (interrupt status is disabled).

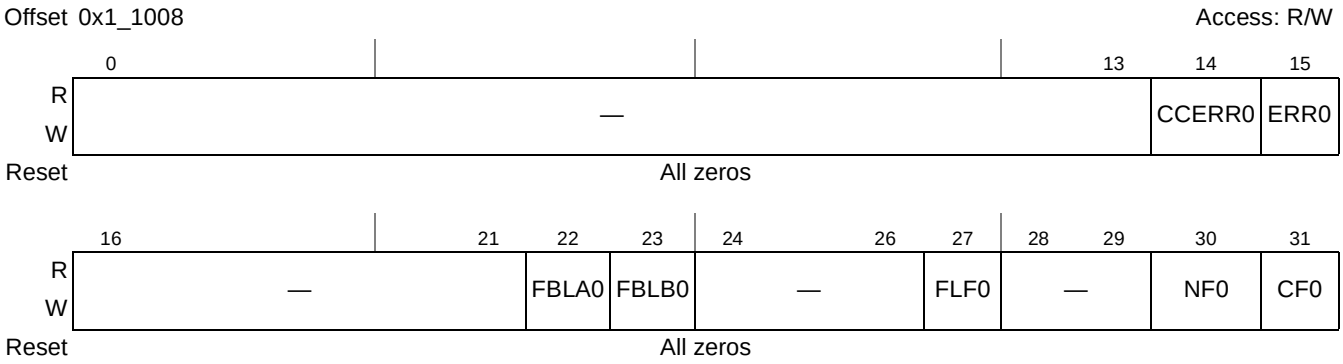


Figure 17-64. Interrupt Status Disable Register 0 Format

Table 17-67 describes the fields of the ISDR0 register.

Table 17-67. Interrupt Status Disable Register 0 Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	CCERR0	Common Control Error Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
15	ERR0	Error Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
16–21	—	Reserved, should be cleared.
22	FBLA0	Free Buffer List A Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
23	FBLB0	Free Buffer List B Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
24–26	—	Reserved, should be cleared.
27	FLF0	Free Buffer Deallocate FIFO Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
28–29	—	Reserved, should be cleared.
30	NF0	Notification FIFO Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status
31	CF0	Command FIFO Channel 0 Interrupt Status Disable. 0 Enable status 1 Disable status

17.3.8.4 Interrupt Inhibit Register 0 (IIR0)

The IIR0 register allows the Pattern Matcher DMA Channel 0 interrupt to be inhibited without modifying the enable or status disable registers.

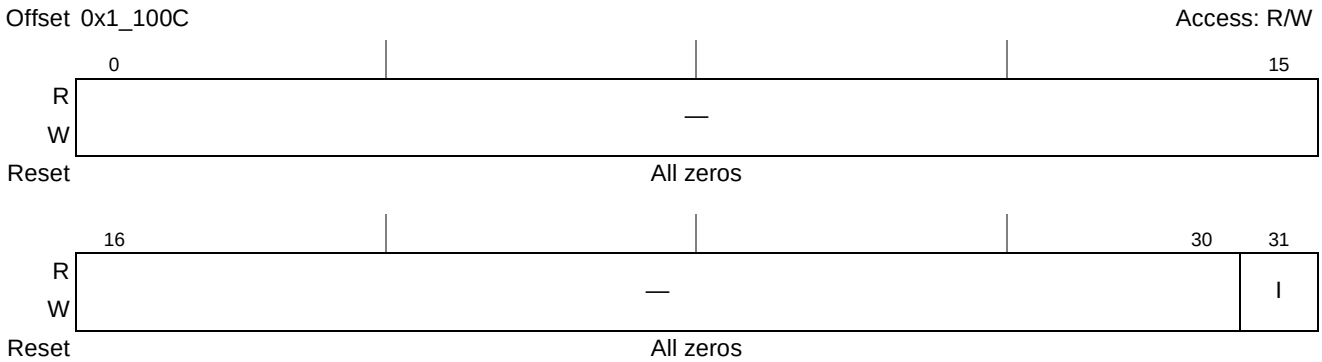


Figure 17-65. Interrupt Inhibit Register 0 Format

Table 17-68. Interrupt Inhibit Register 0 Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	I	Inhibit. 0 Interrupt not inhibited. Pattern Matcher DMA Channel 0 interrupt may assert. 1 Interrupt inhibited. Pattern Matcher DMA Channel 0 interrupt does not assert.

17.3.8.5 Interrupt Interval Control Register 0 (IICR0)

The IICR0 register allows for throttling of the Pattern Matcher DMA Channel 0 interrupt signal to a programmable maximum rate for Command FIFO, Notification FIFO or Free Buffer Deallocate FIFO interrupt sources. The Pattern Matcher block ensures that upon assertion of the interrupt signal (active going edge), a subsequent assertion does not occur until at least a number of clock ticks have occurred. The IICR0 register allows for setting the upper 12 bits of a 20 bit count value. At 333MHz this allows for a minimum interrupt interval in the range 0 to 3.1 msec in 0.768 usec increments, or unlimited down to ~320 interrupts per second.

The IICR0 setting does not affect the behavior of status bits in ISR0, rather it simply guarantees a minimum interval between interrupt signal assertions if the source is a Command FIFO, a Notification FIFO and/or Free Buffer Deallocate FIFO interrupt.

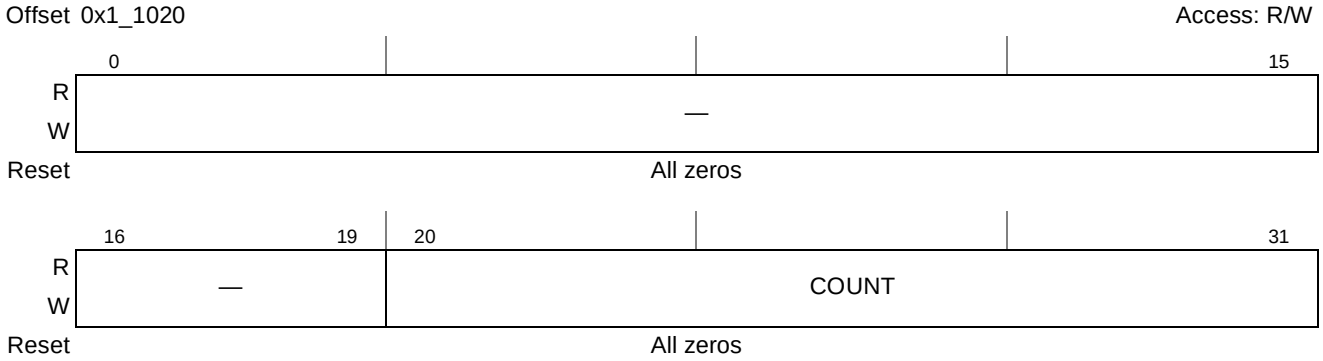


Figure 17-66. Interrupt Interval Control Register 0 Format

Table 17-69. Interrupt Interval Control Register 0 Field Descriptions

Bits	Name	Description
0–19	—	Reserved, should be cleared.
20–31	COUNT	COUNT. Upper 12 bits of a 20 bit counter comparison value. The counter is reset upon interrupt assertion or hardware reset and sticks when it reaches COUNT*256. Interrupts are inhibited when (counter < COUNT*256).

17.3.8.6 Notifications Produced Index Register Channel 0 (NPIR0)

The NPIR0 register provides the value of the Notifications Produced Index (NPI) on channel 0. NPIR0 is written by the DMA Engine to indicate the FIFO entry in which the next notification is written by the DMA Engine.

The actual size of the Notification FIFO is configured through the NFIFO_DEPTH0 register. Allowable Notification FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. NPI is actually represented with n+1 bits where the top bit (n+1 bit) of NPI must be masked off to map it to a FIFO entry. The extra bit is used on all Notification FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Notification FIFO is empty when NPI and NCI are equal. The Notification FIFO is full when NPI is exactly x positions in front of NCI where x is the size of the Notification FIFO. This register is read only. This register can only be reset by a device or channel reset.

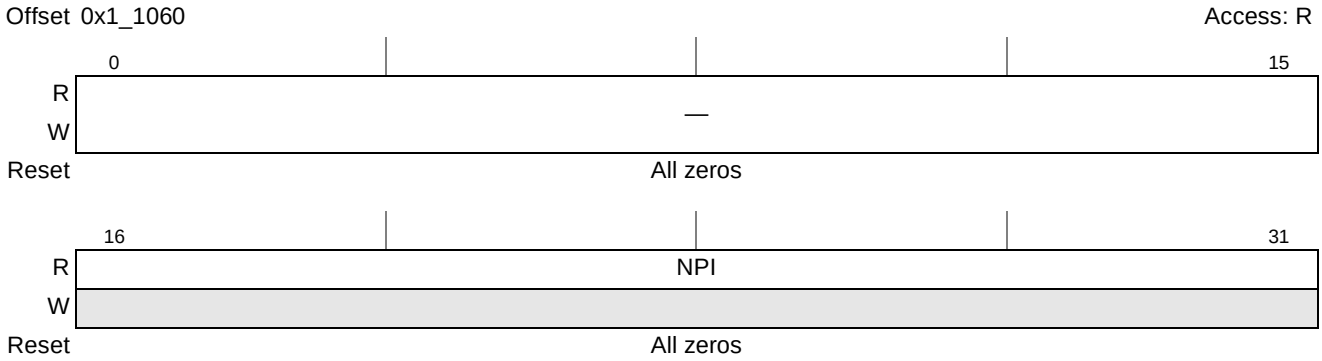


Figure 17-67. Notifications Produced Index Register Channel 0 Format

Table 17-70. Notifications Produced Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	NPI	Notifications Produced Index (NPI).

17.3.8.7 Commands Consumed Index Register Channel 0 (CCIR0)

The CCIR0 register provides the value of the Commands Consumed Index (CCI) on channel 0. CCIR0 is written by the DMA Engine to publish the FIFO entry representing the next command that the DMA Engine processes.

The actual size of the Command FIFO is configured through the CFIFO_DEPTH0 register. Allowable Command FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. CCI is actually represented with n+1 bits where the top bit (n+1 bit) of CCI must be masked off to map it to a FIFO entry. The extra bit is used on all Command FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Command FIFO is empty when CPI and CCI are equal. The Command FIFO is full when CPI is exactly x positions in front of CCI where x is the size of the Command FIFO. This register is read only. This register can only be reset by a device or channel reset.

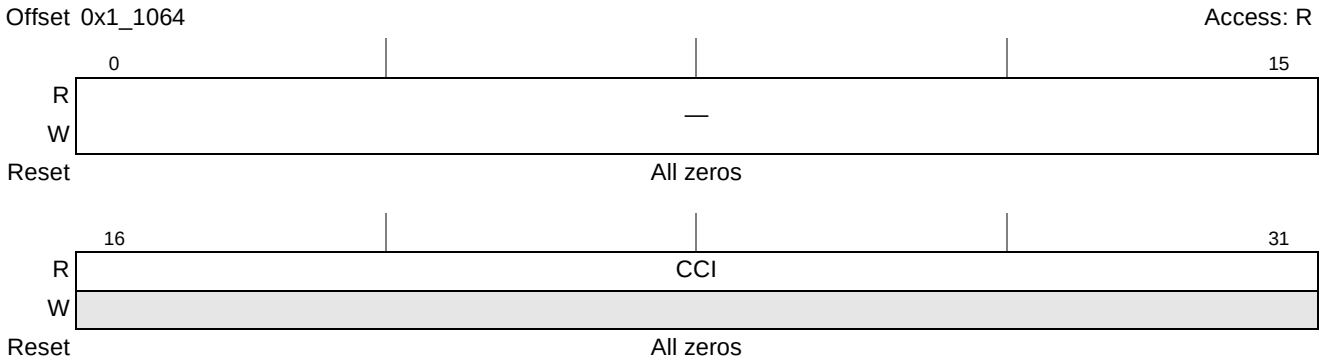


Figure 17-68. Commands Consumed Index Register Channel 0 Format

Table 17-71. Commands Consumed Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	CCI	Commands Consumed Index (CCI).

17.3.8.8 Free Buffer Deallocate FIFO Consumer Index Register Channel 0 (FBCIR0)

The FBCIR0 register provides the value of the Consumed Index (CI) of the Free Buffer Deallocate FIFO on channel 0. FBCIR0 is written by the DMA Engine to publish the FIFO entry representing the next FIFO entry (deallocate buffers) that the DMA Engine processes.

The actual size of the Free Buffer Deallocate FIFO is configured through the FBFIFO_DEPTH0 register. Allowable Free Buffer Deallocate FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. CI is actually represented with n+1 bits where the top bit (n+1 bit) of CI must be masked off to map it to a FIFO entry. The extra bit is used on all Free Buffer Deallocate FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Free Buffer Deallocate FIFO is empty when PI and CI are equal. The Free Buffer Deallocate FIFO is full when PI is exactly x positions in front of CI where x is the size of the Free Buffer Deallocate FIFO. This register is read only. This register can only be reset by a device or channel reset.

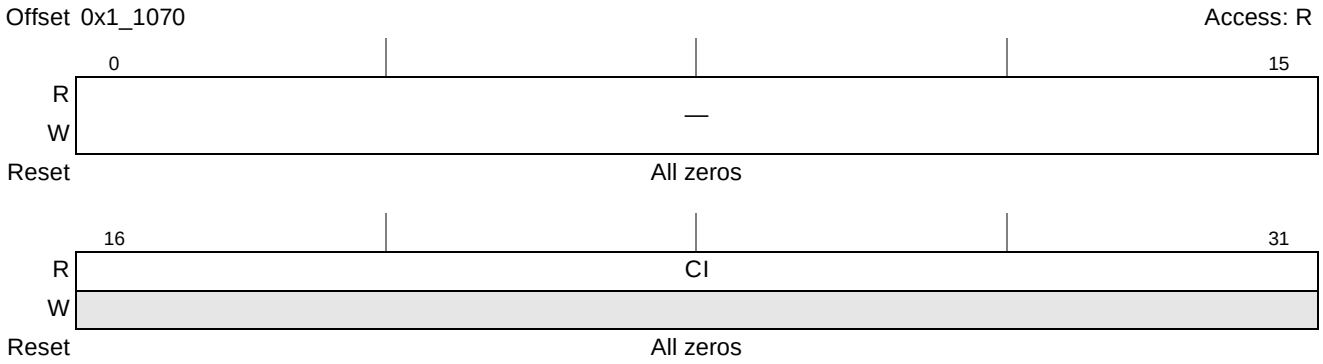


Figure 17-69. Free Buffer Deallocate FIFO Consumer Index Register Channel 0 Format

Table 17-72. Free Buffer Deallocate FIFO Consumer Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	CI	Consumer Index (CI).

17.3.8.9 Channel Error Status Channel 0 (CHERR0)

The CHERR0 register contains the error/exception code (see Table 17-156) of the highest precedence “Channel Error” or “Common-Control Error” (Section 17.4.5.6, “Error Handling) that the DMA Engine has seen but not necessarily communicated to software via the Notification FIFO. This register is intended to trap exceptions for which there may be no Notification FIFO entry created to carry the exception code, such as the ones resulted from processing the Free Buffer Deallocate FIFO. This register is read only. The error code provided by this register can only be reset by a device or channel reset.

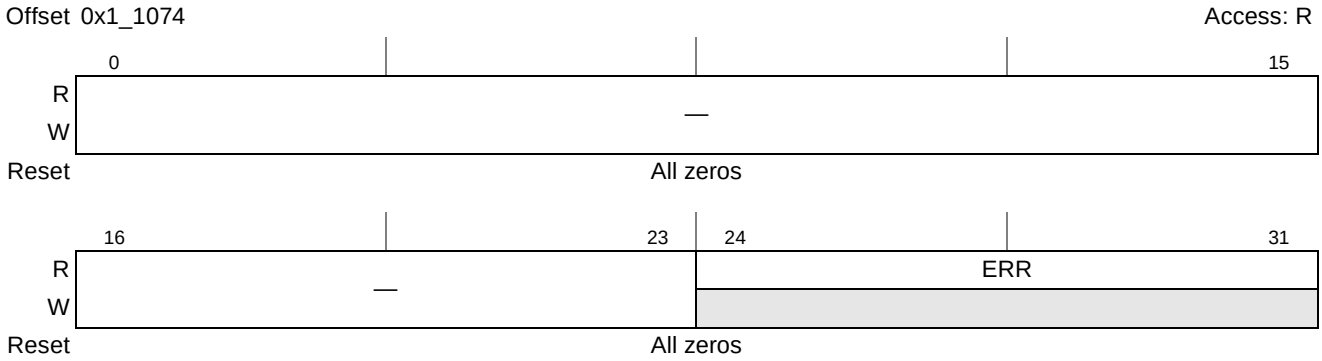


Figure 17-70. Channel Error Status Register Channel 0 Format

Table 17-73. Channel Error Status Register Channel 0 Field Descriptions

Bits	Name	Description
0–23	—	Reserved, should be cleared.
24–31	ERR	“Channel Error” or “Common-Control Error” error/exception code (see Table 17-156).

17.3.8.10 Notifications Consumed Index Register Channel 0 (NCIR0)

The NCIR0 register is used to provide the value of the Notifications Consumed Index (NCI) on channel 0. NCIR0 is written by software to publish the FIFO entry representing the next notification that software processes.

The actual size of the Notification FIFO is configured through the NFIFO_DEPTH0 register. Allowable Notification FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. NCI is actually represented with n+1 bits where the top bit (n+1 bit) of NCI must be masked off to map it to a FIFO entry. The extra bit is used on all Notification FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Notification FIFO is empty when NPI and NCI are equal. The Notification FIFO is full when NPI is exactly x positions

in front of NCI where x is the size of the Notification FIFO. This register is read only. This register can only be reset by a device or channel reset.

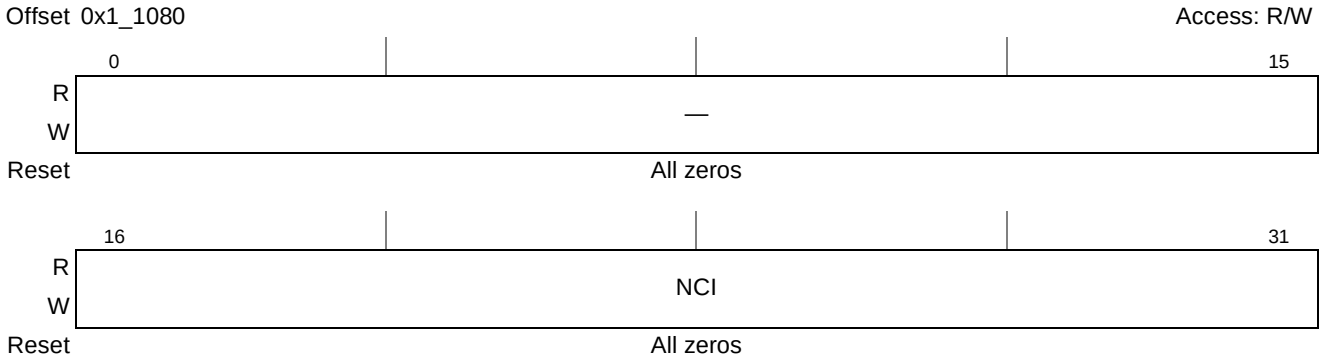


Figure 17-71. Notifications Consumed Index Register Channel 0 Format

Table 17-74. Notifications Consumed Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	NCI	Notifications Consumed Index.

17.3.8.11 Commands Produced Index Register Channel 0 (CPIR0)

The CPIR0 register is used to provide the value of the Commands Produced Index (CPI) on channel 0. CPIR0 is written by software to indicate the FIFO entry in which the next command is written by software.

The actual size of the Command FIFO is configured through the CFIFO_DEPTH0 register. Allowable Command FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. CPI is actually represented with n+1 bits where the top bit (n+1 bit) of CPI must be masked off to map it to a FIFO entry. The extra bit is used on all Command FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Command FIFO is empty when CPI and CCI are equal. The Command FIFO is full when CPI is exactly x positions in front of CCI where x is the size of the Command FIFO. This register is read only. This register can only be reset by a device or channel reset.

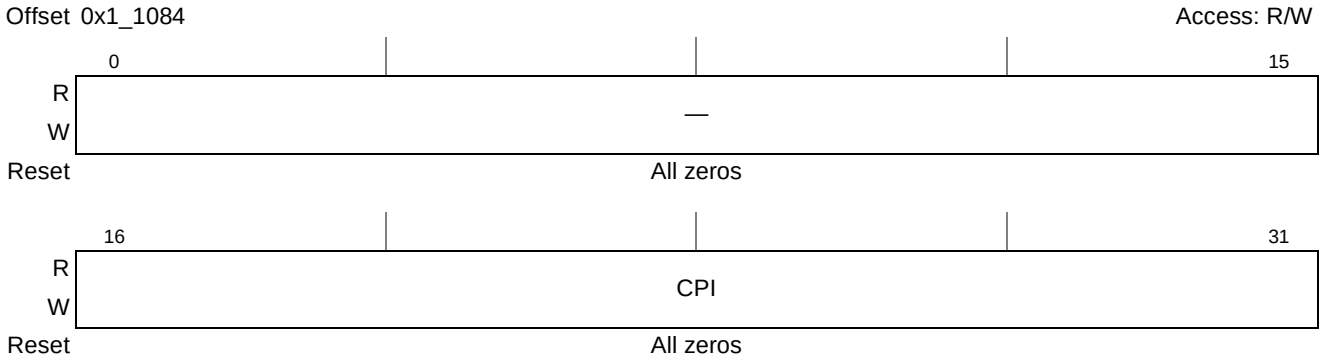


Figure 17-72. Command Produced Index Register Channel 0 Format

Table 17-75. Command Produced Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	CPI	Commands Produced Index.

17.3.8.12 Commands Expired Index Register Channel 0 (CEIR0)

The CEIR0 register is used to provide the value of the Commands Expired Index (CEI) on channel 0. CEIR0 is written by software to indicate the next FIFO entry which software needs to expire (cleaned up).

The actual size of the Command FIFO is configured through the CFIFO_DEPTH0 register. Allowable Command FIFO sizes are 2^n where n is an integral number ranging from 1 to 15 inclusively. CEI is actually represented with n+1 bits where the top bit (n+1 bit) of CEI must be masked off to map it to a FIFO entry. The extra bit is used on all Command FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Command FIFO is empty when CPI and CCI are equal. The Command FIFO is full when CPI is exactly x positions in front of CCI where x is the size of the Command FIFO. This register is read only. This register can only be reset by a device or channel reset.

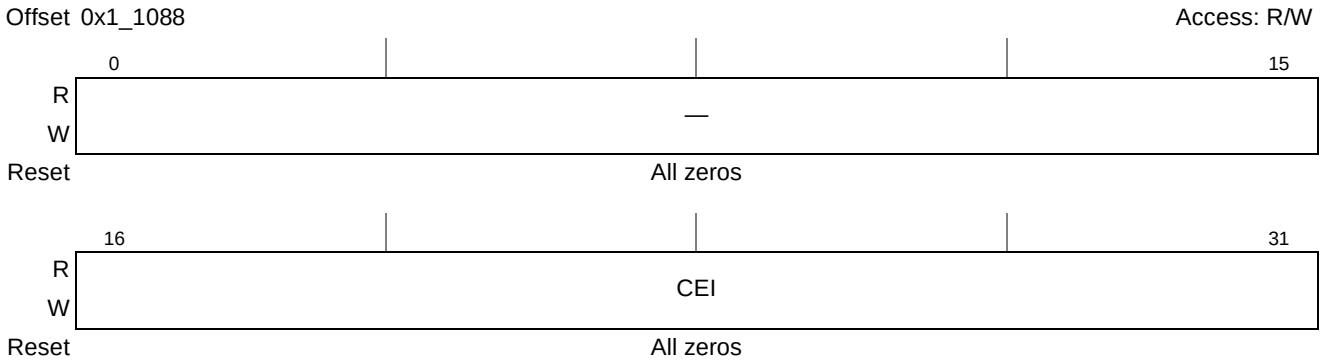


Figure 17-73. Commands Expired Index Register Channel 0 Format

Table 17-76. Commands Expired Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	CEI	Commands Expired Index.

17.3.8.13 Free Buffer Deallocate FIFO Producer Index Register Channel 0 (FBPIR0)

The FBPIR0 register is used to provide the value of the Produced Index (PI) on channel 0. FBPIR0 is written by software to indicate the FIFO entry in which the next free buffers to be deallocated is added by software.

The actual size of the Free Buffer Deallocate FIFO is configured through the FBFIFO_DEPTH0 register. Allowable Free Buffer Deallocate FIFO sizes are 2^n where n is an integral number ranging from 1 to 15

inclusively. PI is actually represented with n+1 bits where the top bit (n+1 bit) of PI must be masked off to map it to a FIFO entry. The extra bit is used on all Free Buffer Deallocate FIFO indexes to differentiate between empty and full FIFO conditions without having to use an empty entry or an extra bit in every entries. The Free Buffer Deallocate FIFO is empty when PI and CI are equal. The Free Buffer Deallocate FIFO is full when PI is exactly x positions in front of CI where x is the size of the Free Buffer Deallocate FIFO. This register is read only. This register can only be reset by a device or channel reset.

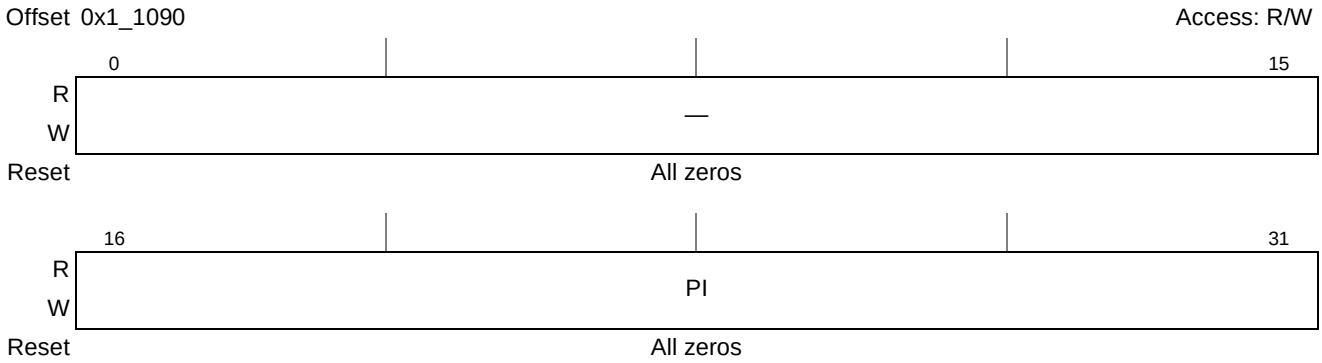


Figure 17-74. Free Buffer Deallocate FIFO Producer Index Register Channel 0 Format

Table 17-77. Free Buffer Deallocate FIFO Producer Index Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	PI	Producer Index (PI).

17.3.8.14 Output Truncated Incidence Counter Channel 0 (TRUNCIC0)

The TRUNCIC0 register provides a 16-bit COUNT field. COUNT increments by 1 for every notification entry that is produced where the output data has been truncated due to insufficient provided buffer space or free buffer list exhaustion. If this counter rolls over, an overflow bit is set in the SCOS0 register.

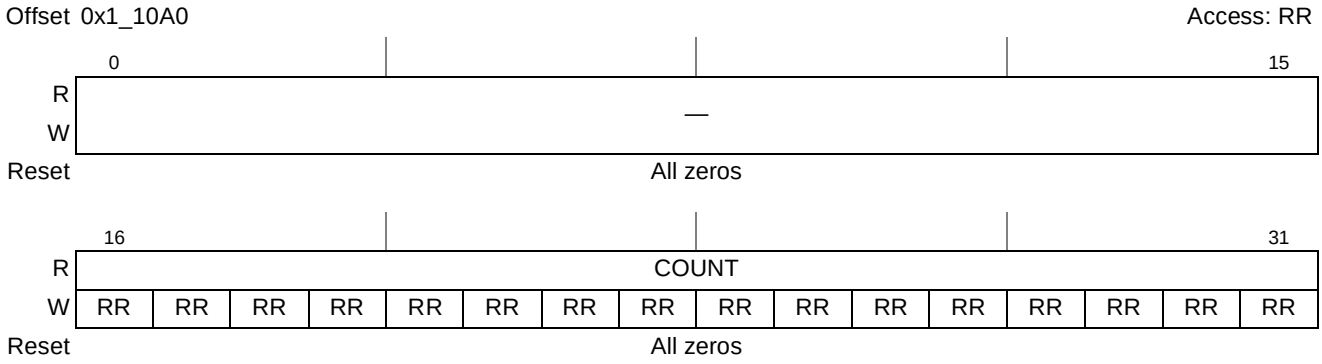


Figure 17-75. Output Truncated Incidence Counter Channel 0 Format

Table 17-78. Output Truncated Incidence Counter Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	COUNT	Count. Indicates the number of output truncations that have occurred since the last read of this location.

17.3.8.15 Read Byte Counter Channel 0 (RBC0)

The RBC0 register provides a 32-bit COUNT field. COUNT increments by 1 for every byte that is read by the DMA Engine on behalf of channel 0. All FIFO and buffer reads contribute to COUNT. If this counter rolls over, an overflow bit is set in the SCOS0 register.

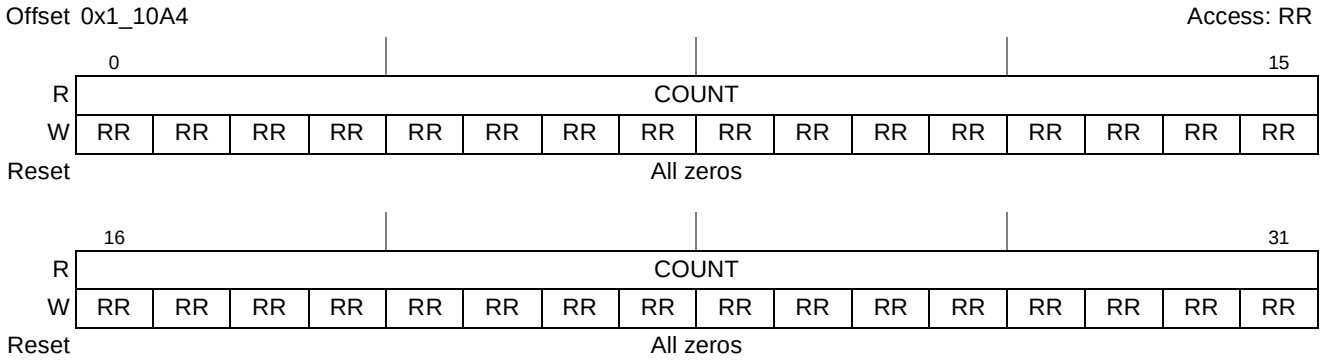


Figure 17-76. Read Byte Counter Channel 0 Format

Table 17-79. Read Byte Counter Channel 0 Field Descriptions

Bits	Name	Description
0–31	COUNT	Count. Indicates the number of read bytes since the last read of this location.

17.3.8.16 Statistics Counters Overflow Status Register Channel 0 (SCOS0)

The SCOS0 register contains overflow bits for the various channel 0 statistics counters in the Pattern Matcher. If a statistics counter rolls over, the corresponding bit in SCOS0 is set. Each overflow status bit is individually cleared by writing a 1 in that bit location.

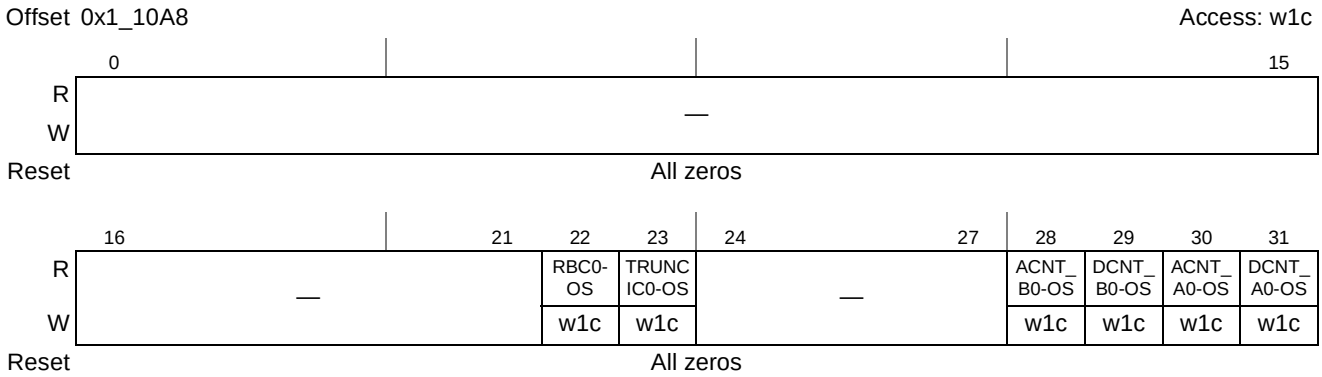


Figure 17-77. Statistics Counters Overflow Status 0 Register (SCOS0)

Table 17-80. SCOS0 Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	RBC0-OS	Overflow Status for the RBC0 counter
23	TRUNCIC0-OS	Overflow Status for the TRUNCIC0 counter
24–27	—	Reserved, should be cleared.
28	ACNT_B0-OS	Overflow Status for the ACNT counter located in the FBL_AD_CB0 register.
29	DCNT_B0-OS	Overflow Status for the DCNT counter located in the FBL_AD_CB0 register.
30	ACNT_A0-OS	Overflow Status for the ACNT counter located in the FBL_AD_CA0 register.
31	DCNT_A0-OS	Overflow Status for the DCNT counter located in the FBL_AD_CA0 register.

17.3.8.17 Channel Reset and Control Register Channel 0 (CRCR0)

The CRCR0 register provides reset and control capability for channel 0 logic in the Pattern Matcher block. The ability to reset and control a single channel in isolation is useful because it allows for recovery from per-channel errors without impacting the other Pattern Matcher channels.

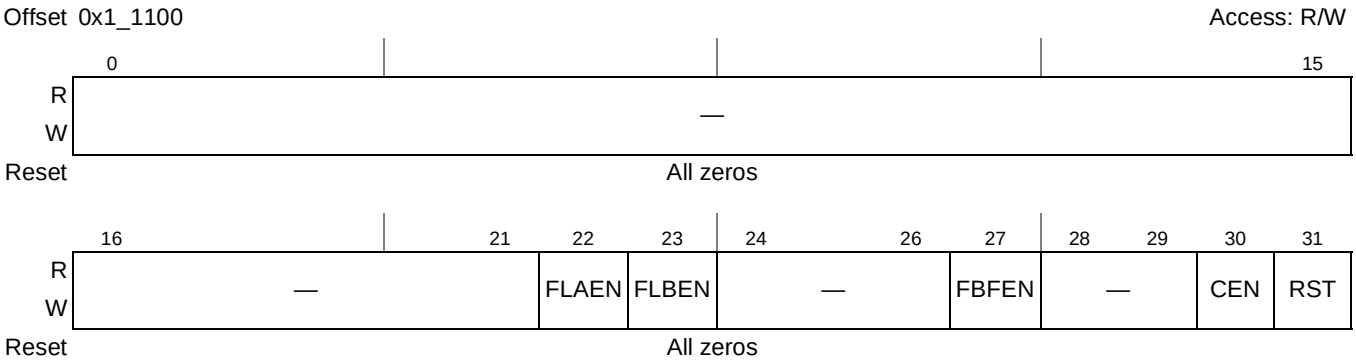


Figure 17-78. Channel Reset and Control Register Channel 0 Format

Table 17-81. Channel Reset and Control Register Channel 0 Field Descriptions

Bits	Name	Description
0–21	—	Reserved, should be cleared.
22	FLAEN	Free Buffer List A Enable. This bit enables use of the Free Buffer List A for this channel. If Free Buffer List A is not enabled, the DMA Engine does not attempt to use buffers from Free Buffer List A. 0 Free Buffer List A Disabled. 1 Free Buffer List A Enabled.
23	FLBEN	Free Buffer List B Enable. This bit enables use of the Free Buffer List B for this channel. If Free Buffer List B is not enabled, the DMA Engine does not attempt to use buffers from Free Buffer List B. 0 Free Buffer List B Disabled. 1 Free Buffer List B Enabled.
24–26	—	Reserved, should be cleared.

Table 17-81. Channel Reset and Control Register Channel 0 Field Descriptions (continued)

Bits	Name	Description
27	FBFEN	Free Buffer Deallocate FIFO Enable. This bit enables/disables Free Buffer Deallocate FIFO operation for this channel. If the Free Buffer Deallocate FIFO is not enabled then the DMA Engine does not attempt to service the Free Buffer Deallocate FIFO. 0 Free Buffer Deallocate FIFO disabled. Free Buffer Deallocate FIFO is not serviced. 1 Free Buffer Deallocate FIFO enabled. Free Buffer Deallocate FIFO is serviced.
28–29	—	Reserved, should be cleared.
30	CEN	Command FIFO Enable. This bit enables/disables Command FIFO operation for this channel. If used during operation to disable the Command FIFO operation, the Pattern Matcher gracefully completes any commands that are in flight. 0 Command FIFO disabled. Command FIFO is not serviced. 1 Command FIFO enabled. Command FIFO is serviced.
31	RST	Reset. This bit allows for resetting of this channel. If used to reset all Pattern Matcher internal state related to this channel. This is not intended to be a graceful reset, it is not safe to reset a channel while traffic is being processed. 0 Takes channel out of reset. Take note that if the channel is being held in reset, then you can't clear this bit and write “1s” (enabling) to the other fields of this register in the same access (in this case a separate access writing the other fields would be required). 1 Puts channel into reset and channel is held in reset till a 0 is written into this field. Take note that a channel is not able to enter reset while a common-control error remains asserted in the ISRCC register.

17.3.8.18 Command FIFO Base Address High and Low Registers Channel 0 (CFIFO_BH0 and CFIFO_BL0)

The CFIFO_BH0 and CFIFO_BL0 registers are used to provision the Pattern Matcher DMA Engine with the address in system memory where the Command FIFO is located. These registers allow the location to be programmed only on natural 32 byte address boundaries. During operation, the DMA Engine accesses Command FIFO entries at the location indicated by ADDR + CCI * (Command FIFO Entry Size). The Command FIFO Entry Size is configured through the CFIFO_DEPTH0 register.

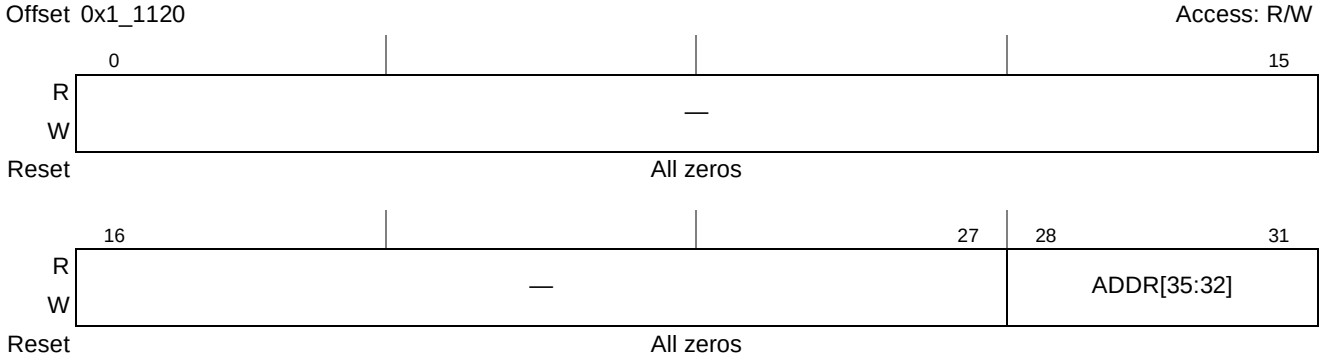


Figure 17-79. Command FIFO Base Address High Register Channel 0 Format

Table 17-82. Command FIFO Base Address High Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Command FIFO in system memory.

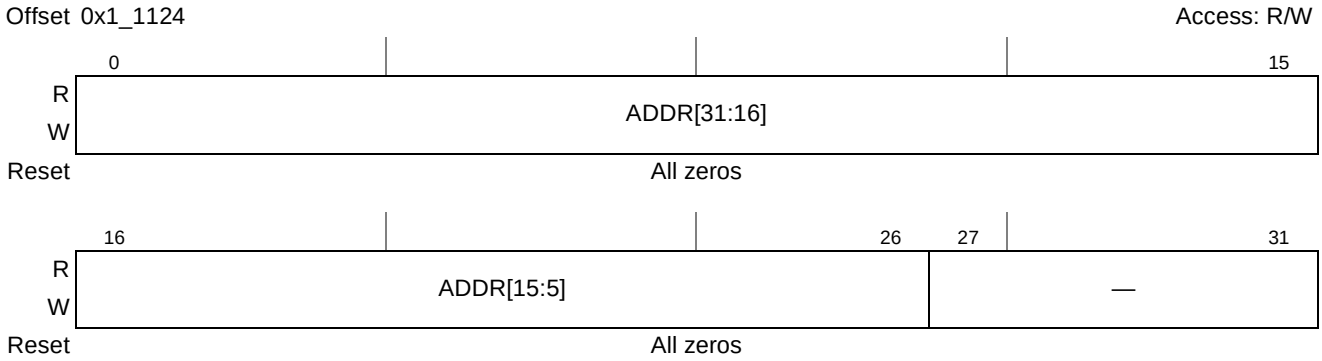


Figure 17-80. Command FIFO Base Address Low Register Channel 0 Format

Table 17-83. Command FIFO Base Address Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	ADDR	The base address of the Command FIFO in system memory.
27–31	—	Reserved, should be cleared.

17.3.8.19 Command FIFO Depth Register Channel 0 (CFIFO_DEPTH0)

The CFIFO_DEPTH0 register is used to provision the number of entries in the Command FIFO as well as the size of Command FIFO entries. FIFO depth is computed as 2^{DEPTH}. The FIFO entries are either 64 or 128 bytes in size.

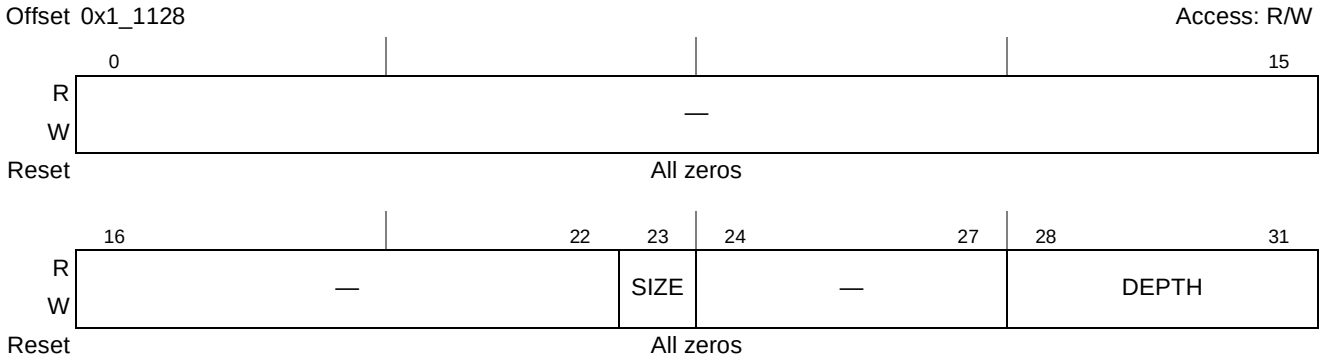


Figure 17-81. Command FIFO Depth Register Channel 0 Format

Table 17-84. Command FIFO Depth Register Channel 0 Field Descriptions

Bits	Name	Description
0–22	—	Reserved, should be cleared.
23	SIZE	Size. This bit controls the size of Command FIFO Entries. Coding: 0 64 byte command FIFO Entries. 1 128 byte command FIFO Entries.
24–27	—	Reserved, should be cleared.
28–31	DEPTH	Depth. The exponent of the desired number of FIFO entries. Valid settings are 1–15.

17.3.8.20 Command FIFO Threshold Register Channel 0 (CFIFO_THRES0)

The CFIFO_THRES0 register is used to provision a FIFO fill level threshold such that the Pattern Matcher DMA Channel interrupt signal interrupt is generated (if the Command FIFO interrupt source is enabled) when the FIFO fill level is greater than the threshold. Note that when this condition occurs, the Pattern Matcher DMA Channel interrupt signal is asserted immediately (if the corresponding bit in the IER0 register is also set) even if the prescribed interval set in the IICR0 register has not elapsed.

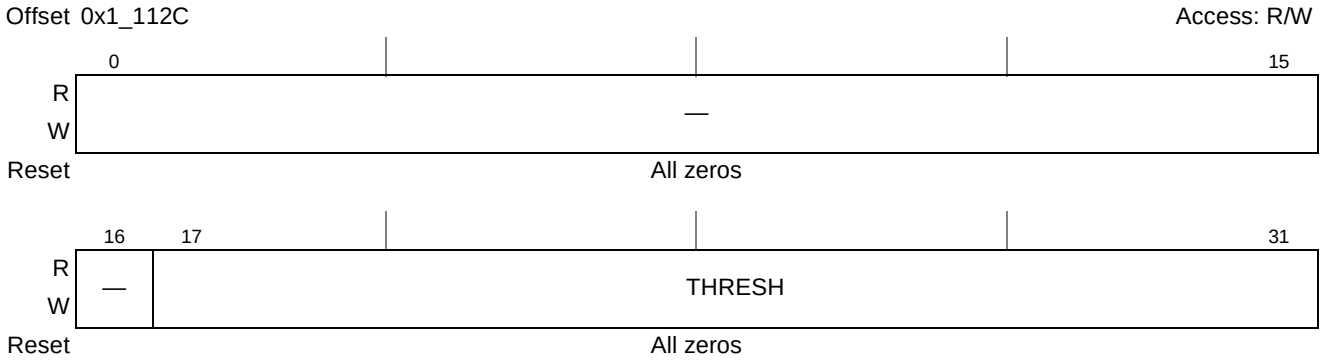


Figure 17-82. Command FIFO Threshold Register Channel 0 Format

Table 17-85. Command FIFO Threshold Register Channel 0 Field Descriptions

Bits	Name	Description
0–16	—	Reserved, should be cleared.
17–31	THRESH	Threshold. This field defines the fill level which, when exceeded, results in a Pattern Matcher DMA Channel interrupt signal if the Command FIFO source interrupt is enabled. Fill level is computed as the difference between CCI and CEI.

17.3.8.21 Notification FIFO Base Address High and Low Registers Channel 0 (NFIFO_BH0 and NFIFO_BL0)

The NFIFO_BH0 and NFIFO_BL0 registers are used to provision the Pattern Matcher DMA Engine with the address in system memory where the Notification FIFO is located. These registers allow the location to be programmed only on natural 32 byte address boundaries. During operation, the DMA Engine writes Notification FIFO entries at the location indicated by ADDR + NPI*64.

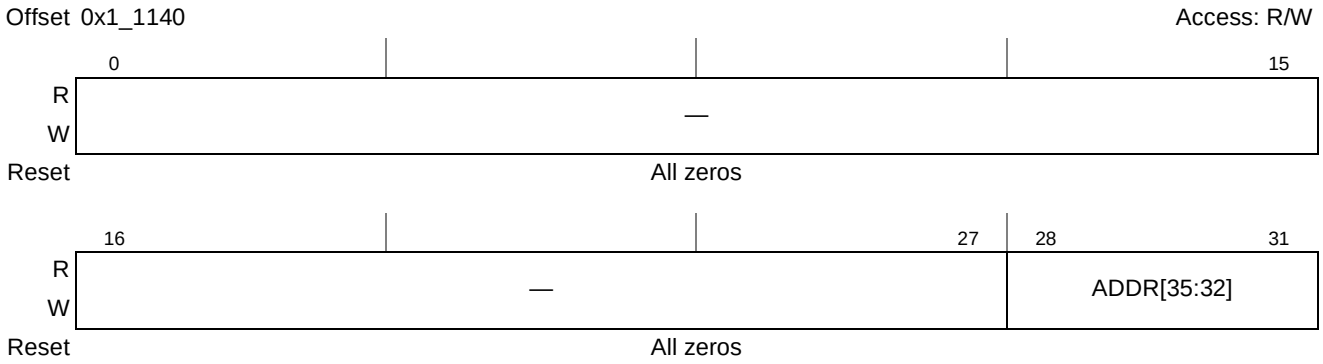


Figure 17-83. Notification FIFO Base Address High Register Channel 0 Format

Table 17-86. Notification FIFO Base Address High Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Notification FIFO in system memory.

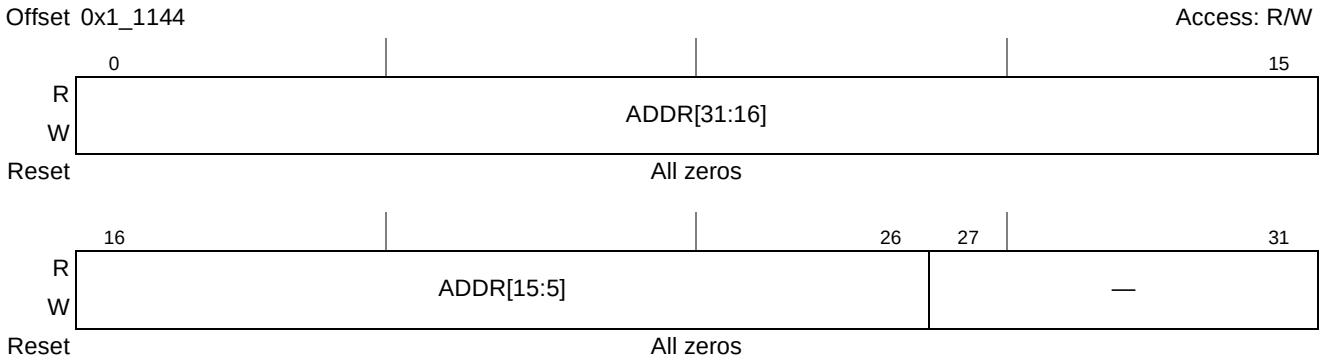


Figure 17-84. Notification FIFO Base Address Low Register Channel 0 Format

Table 17-87. Notification FIFO Base Address Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	ADDR	The base address of the Notification FIFO in system memory.
27–31	—	Reserved, should be cleared.

17.3.8.22 Notification FIFO Depth Register Channel 0 (NFIFO_DEPTH0)

The NFIFO_DEPTH0 register is used to provision the number of entries in the Notification FIFO. The FIFO depth is computed as 2^{DEPTH}.

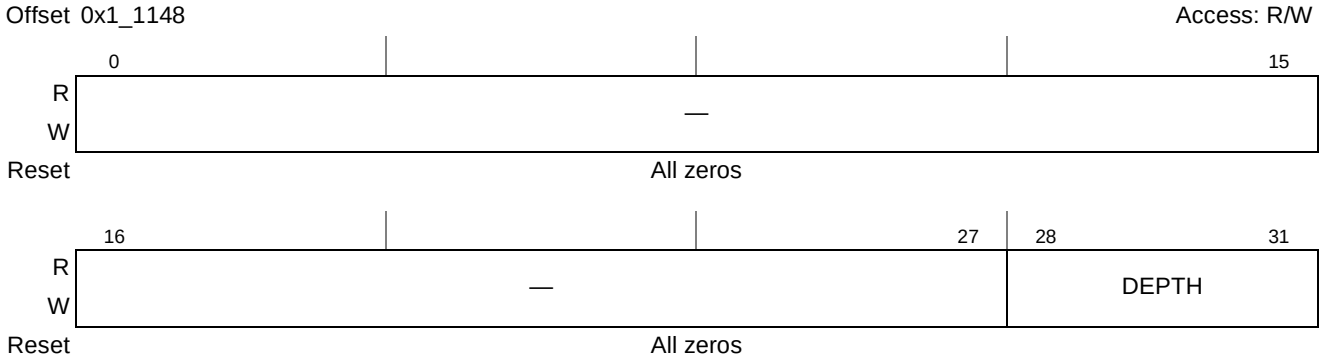


Figure 17-85. Notification FIFO Depth Register Channel 0 Format

Table 17-88. Notification FIFO Depth Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	DEPTH	Depth. The exponent of the desired number of FIFO entries. Valid settings are 1–15.

17.3.8.23 Notification FIFO Threshold Register Channel 0 (NFIFO_THRES0)

The NFIFO_THRES0 register is used to provision a FIFO fill level threshold such that the Pattern Matcher DMA Channel interrupt signal interrupt is generated (if the Notification FIFO interrupt source is enabled) when the FIFO fill level is greater than the threshold. Note that when this condition occurs, the Pattern Matcher DMA Channel interrupt signal is asserted immediately (if the corresponding bit in the IER0 register is also set) even if the prescribed interval set in the IICR0 register has not elapsed.

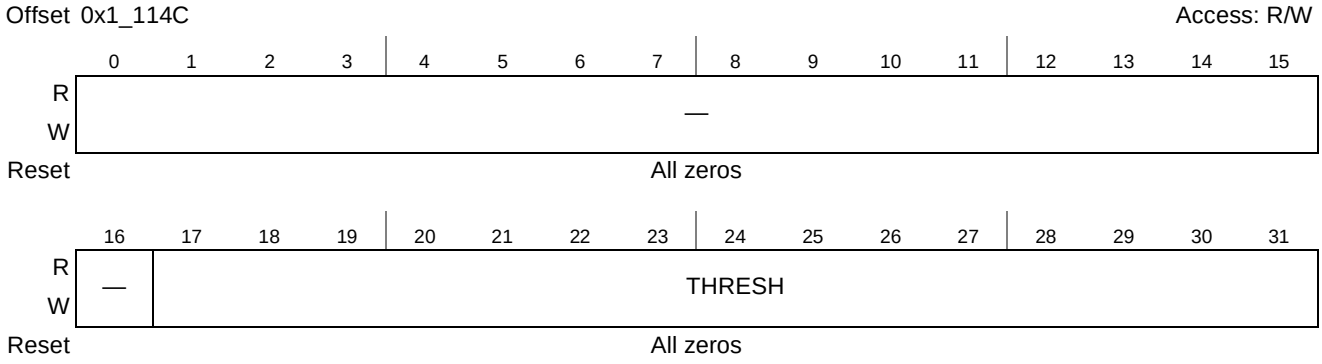


Figure 17-86. Notification FIFO Threshold Register Channel 0 Format

Table 17-89. Notification FIFO Threshold Register Channel 0 Field Descriptions

Bits	Name	Description
0–16	—	Reserved, should be cleared.
17–31	THRESH	Threshold. This field defines the fill level which, when exceeded, results in a Pattern Matcher DMA Channel interrupt if the Notification FIFO interrupt source is enabled. Fill level is computed as NPI - NCI.

17.3.8.24 Notification Deflate Length Limit Register Channel 0 (NDLL0)

The NDLL0 register allows software to impose a length limit on a Notification's decompressed data when the buffers being used to store the data are taken from the free buffer list.

In the case where a buffer was provided with the command, the NDLL0 register setting is ignored.

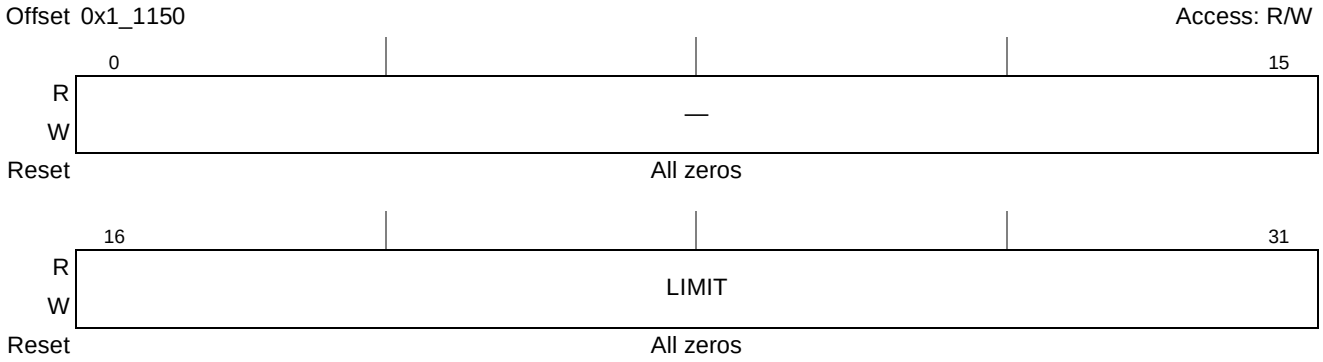


Figure 17-87. Notification Deflate Length Limit Register Channel 0 Format

Table 17-90. Notification Deflate Length Limit Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	LIMIT	Limit. This field specifies the maximum number of free buffers that can be allocated for a single Notification FIFO entry. Setting this field to 0x0 disables length limiting.

17.3.8.25 Notification Result Length Limit Register Channel 0 (NRLL0)

The NRLL0 register allows software to impose a length limit on a Notification's result data when the buffers being used to store the result are taken from the free buffer list.

In the case where a buffer was provided with the command, the NRLL0 register setting is ignored.

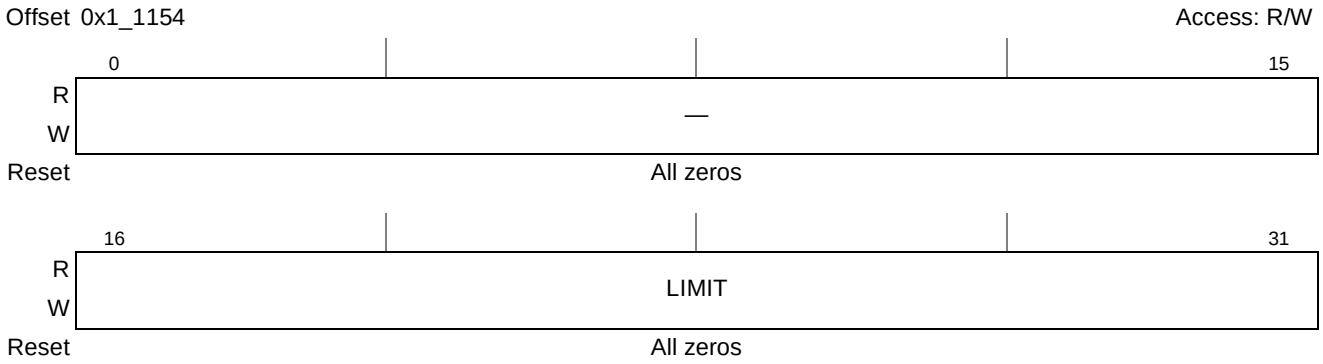


Figure 17-88. Notification Result Length Limit Register Channel 0 Format

Table 17-91. Notification Result Length Limit Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	LIMIT	Limit. This field specifies the maximum number of free buffers that can be allocated for a single Notification FIFO Entry. Setting this field to 0x0 disables length limiting.

17.3.8.26 Free Buffer Deallocate FIFO Base Address High and Low Registers Channel 0 (FBFIFO_BH0 and FBFIFO_BL0)

The FBFIFO_BH0 and FBFIFO_BL0 registers are used to provision the Pattern Matcher DMA Engine with the address in system memory where the Free Buffer Deallocate FIFO is located. These registers allow the location to be programmed only on natural 32 byte address boundaries. During operation, the DMA Engine reads Free Buffer Deallocate FIFO entries at the location indicated by ADDR + CI*32.

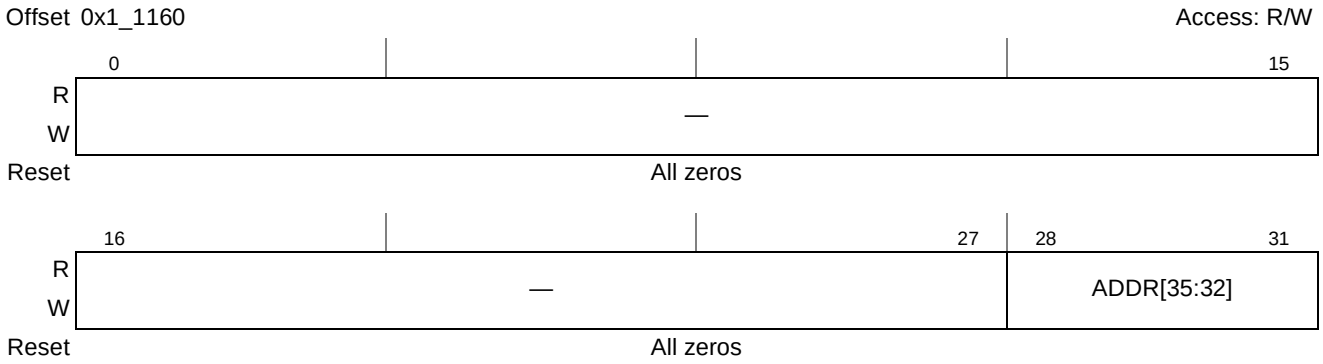


Figure 17-89. Free Buffer Deallocate FIFO Base Address High Register Channel 0 Format

Table 17-92. Free Buffer Deallocate FIFO Base Address High Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Free Buffer Deallocate FIFO in system memory.

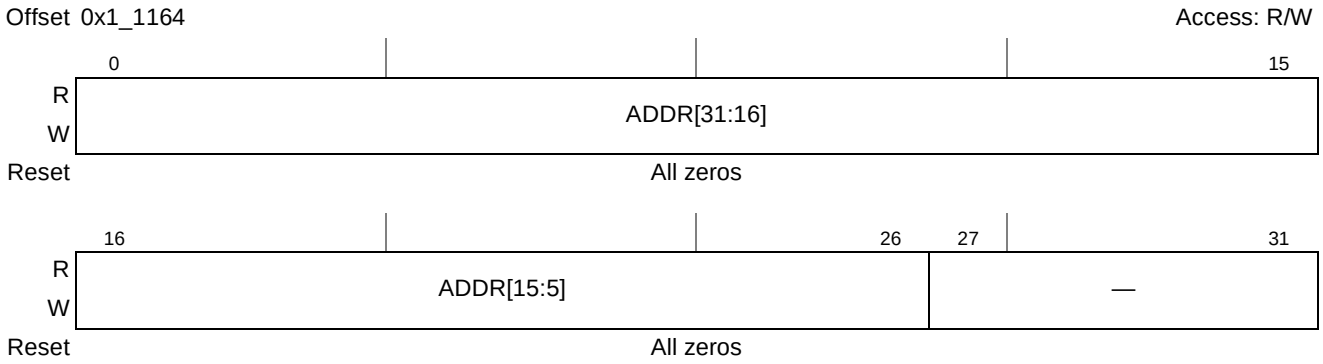


Figure 17-90. Free Buffer Deallocate FIFO Base Address Low Register Channel 0 Format

Table 17-93. Free Buffer Deallocate FIFO Base Address Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	ADDR	The base address of the Free Buffer Deallocate FIFO in system memory.
27–31	—	Reserved, should be cleared.

17.3.8.27 Free Buffer Deallocate FIFO Depth Register Channel 0 (FBFIFO_DEPTH0)

The FBFIFO_DEPTH0 register is used to provision the number of entries in the Free Buffer Deallocate FIFO. The FIFO depth is computed as 2^{DEPTH}.

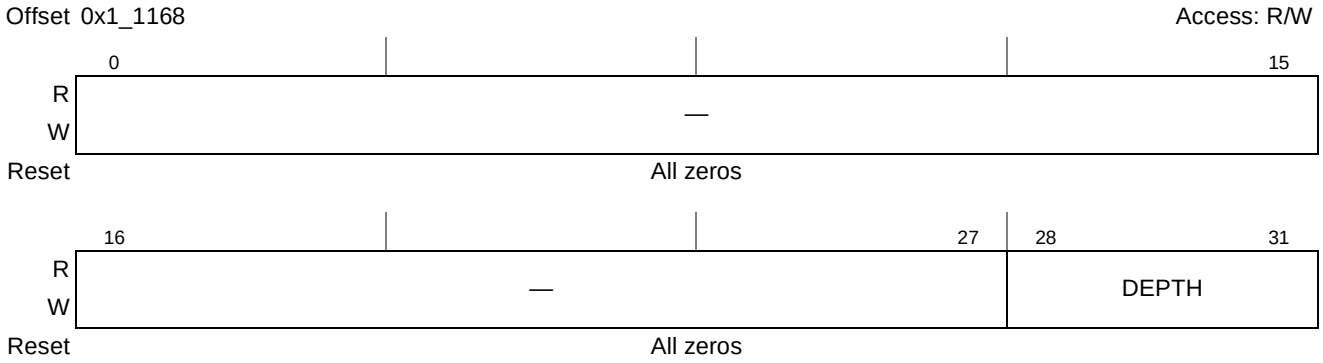


Figure 17-91. Free Buffer Deallocate FIFO Depth Register Channel 0 Format

Table 17-94. Free Buffer Deallocate FIFO Depth Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	DEPTH	Depth. The exponent of the desired number of FIFO entries. Valid settings are 1–15.

17.3.8.28 Free Buffer Deallocate FIFO Threshold Register Channel 0 (FBFIFO_THRES0)

The FBFIFO_THRES0 register is used to provision a FIFO fill level threshold such that the Free Buffer Deallocate FIFO status interrupt bit in the ISR0 register is set (if the Free Buffer Deallocate FIFO interrupt source is enabled) when the FIFO fill level is less than the threshold.

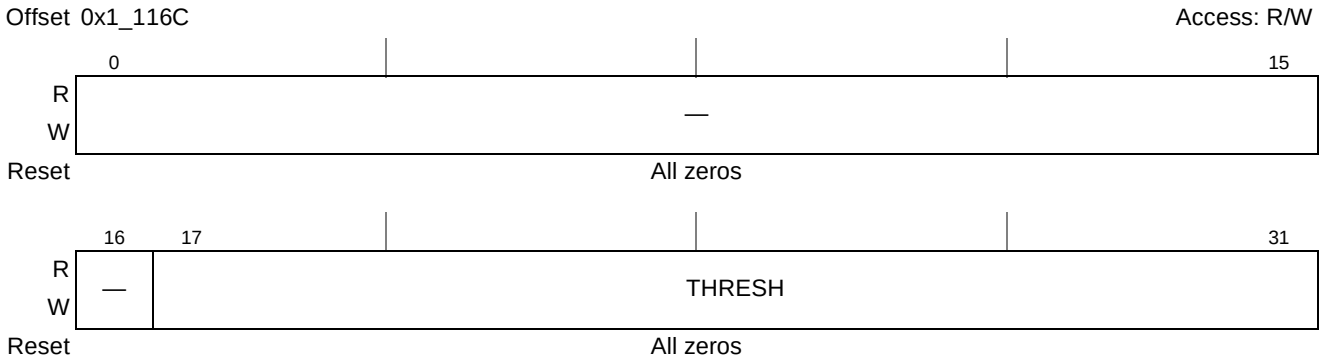


Figure 17-92. Free Buffer Deallocate FIFO Threshold Register Channel 0 Format

Table 17-95. Free Buffer Deallocate FIFO Threshold Register Channel 0 Field Descriptions

Bits	Name	Description
0–16	—	Reserved, should be cleared.
17–31	THRESH	Threshold. This field defines the fill level, which results in the Free Buffer Deallocate FIFO Interrupt Status bit being set in the ISR0 register; for example, when fill is less than threshold. Fill level is computed as PI - CI. If this field is set to zero, then the Free Buffer Deallocate FIFO interrupt source is implicitly disabled.

17.3.8.29 Stream Context Configuration Register Channel 0 (SC_CONFIG0)

The SC_CONFIG0 register provides control over the Stream Context addressing scheme. Stream Context entries can be configured as tabular (stored in a large contiguous table in system memory) or non-tabular (stored at arbitrary locations in system memory). In the tabular case, stream ID represents an index into the table. In the non-tabular case, stream ID represents the memory address that the stream context is found at.

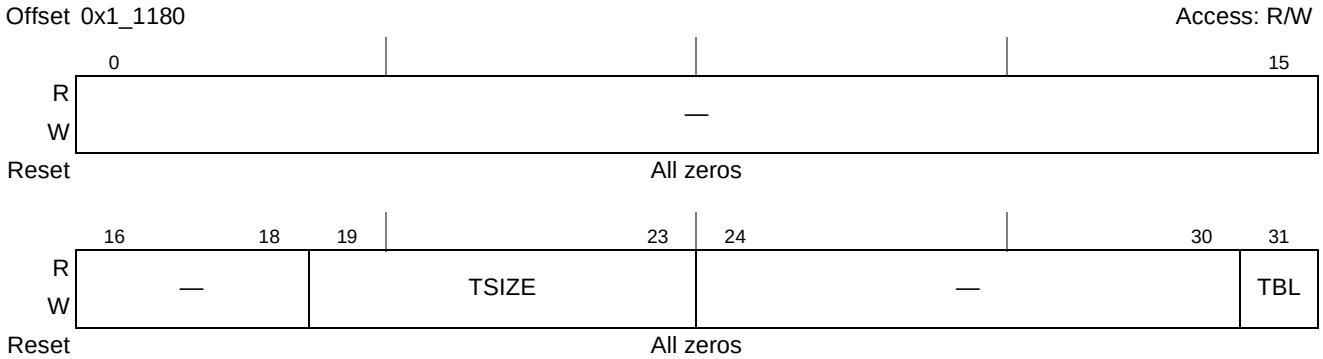


Figure 17-93. Stream Context Configuration Register Channel 0 Format

Table 17-96. Stream Context Configuration Register Channel 0 Field Descriptions

Bits	Name	Description
0–18	—	Reserved, should be cleared.
19–23	TSIZE	Table Size. This field encodes the size of the stream context table, and thus the number of supported streams, for the case where stream contexts are stored in a fixed size table. Table size is computed as 2^{TSIZE}, for example: • TSIZE=4 evaluates to a stream table of 2^4=16 streams. • TSIZE=20 evaluates to a stream table of 2^{20}=1,048,576 streams. This register also implicitly defines the bit-wise mask that is applied to stream ID's when they are processed by the DMA Engine. This allows software to use the full stream ID value in the Command FIFO, and have the full value echoed in the Notification FIFO entry, with the understanding that the DMA Engine only uses the bottom TSIZE bits of the stream ID. This capability may be useful for software as it allows the upper bits to be used for software mux/demux purposes.
24–30	—	Reserved, should be cleared.
31	TBL	Tabular. This bit indicates whether stream context entries are stored in a fixed table or at arbitrary locations in memory. If stream contexts are stored in a tabular form, then the SC_BH0 and SC_BL0 registers define the table's base address and the TSIZE field of this register defines the table's size. 0 Non-tabular. Stream context entries are stored at arbitrary locations in system memory. Stream ID values are considered to be memory pointers. 1 Tabular. Stream context entries are stored in a fixed size table at the address indicated by the SC_BH0 and SC_BL0 registers.

17.3.8.30 Stream Context Base Address High and Low Registers Channel 0 (SC_BH0 and SC_BL0)

The SC_BH0 and SC_BL0 registers are used to provision the location of the channel's stream context table in system memory when stream contexts are stored in tabular form (SC_CONFIG0[TBL] set to 1). Stream Context entries occupy a 64 byte memory footprint and the table base address must be aligned to a natural 64 byte boundary.

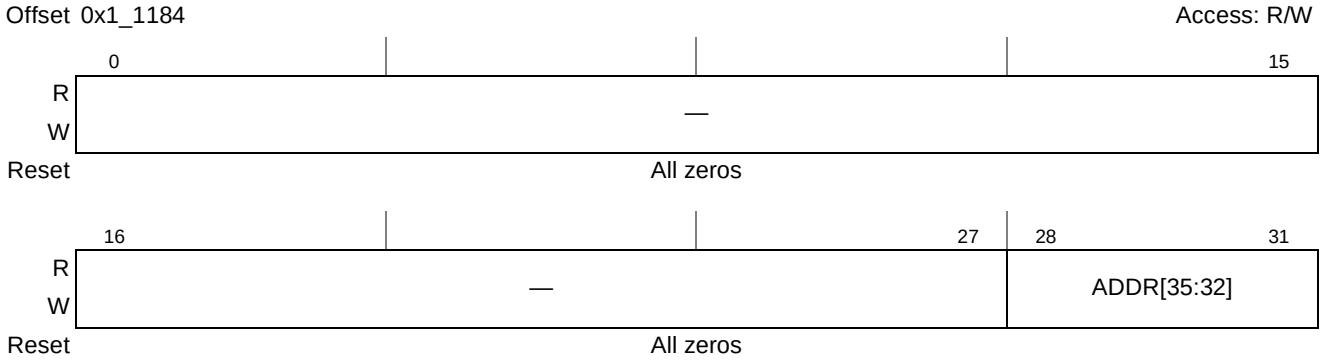


Figure 17-94. Stream Context Base Address High Register Channel 0 Format

Table 17-97. Stream Context Base Address High Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Stream Context table in system memory.

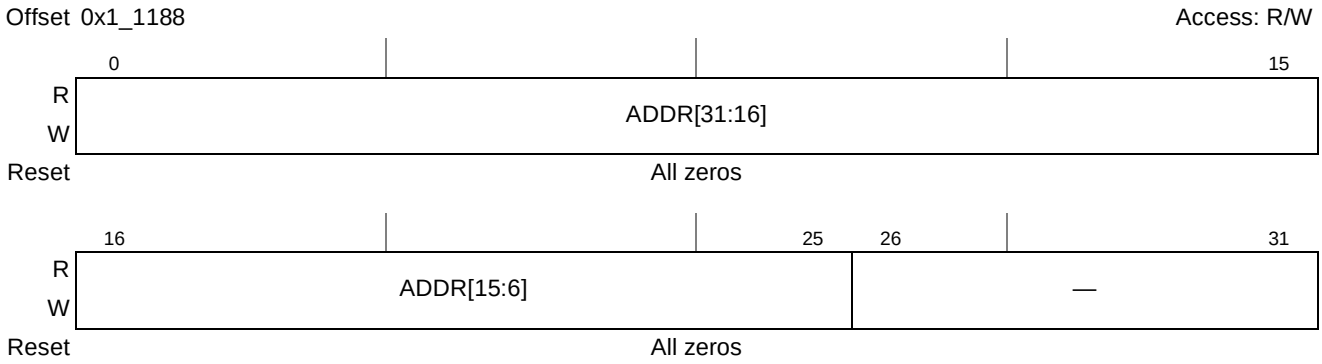


Figure 17-95. Stream Context Base Address Low Register Channel 0 Format

Table 17-98. Stream Context Base Address Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–25	ADDR	The base address of the Stream Context table in system memory.
26–31	—	Reserved, should be cleared.

17.3.8.31 Residue Configuration Register Channel 0 (RES_CONFIG0)

The RES_CONFIG0 register provides control over the Residue Context addressing scheme. Residue Context entries can be configured as tabular (stored in a large contiguous table in system memory) or non-tabular (stored at arbitrary locations in system memory). In the tabular case, residue pointers represent an index into the table. In the non-tabular case, residue pointers represent the memory address that the stream context is found at. The RES_CONFIG0 register also allows for programming of the maximum residue size that the hardware maintains.

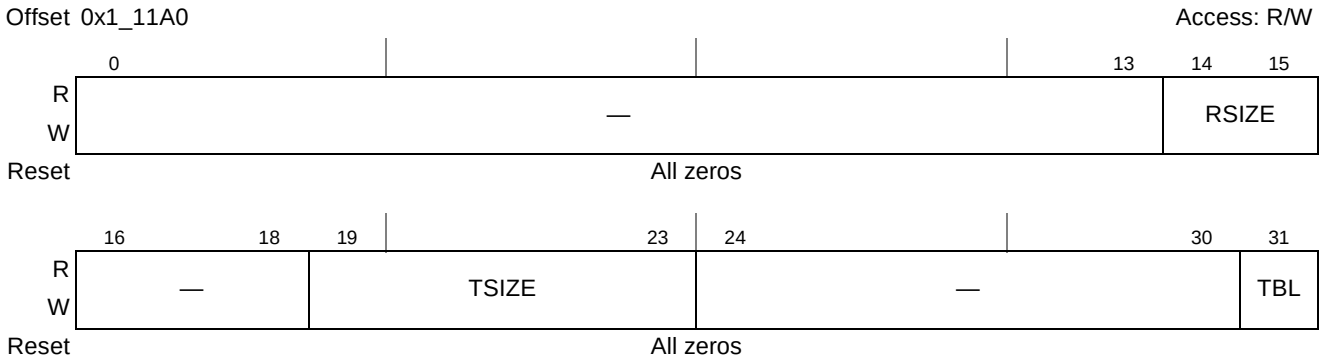


Figure 17-96. Residue Configuration Register Channel 0 Format

Table 17-99. Residue Configuration Register Channel 0 Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14–15	RSIZE	Residue Size. This field allows the maximum size of the residue context to be configured. The DMA Engine maintains residue up to this maximum. 00 Encoding 0. Maximum residue set to 32 bytes. 01 Encoding 1. Maximum residue set to 64 bytes. 10 Encoding 2. Maximum residue set to 96 bytes. 11 Encoding 3. Maximum residue set to 127 bytes.
16–18	—	Reserved, should be cleared.
19–23	TSIZE	Table Size. This field encodes the size of the residue table, and thus the number of streams that can use residue, for the case where residues are stored in a fixed size table. Table size is computes as 2^{TSIZE}, for example: • TSIZE=4 evaluates to a residue table of 2^4,=16 entries. • TSIZE=20 evaluates to a residue table of 2^{20},=1,048,576 entries.
24–30	—	Reserved, should be cleared.
31	TBL	Tabular. This bit indicates whether residues are stored in a fixed table or at arbitrary locations in memory. If residues are stored in a tabular form, then the RES_BH0 and RES_BL0 registers define the table's base address and the TSIZE field of this register defines the table's size. 0 Non-tabular. Residues are stored at arbitrary locations in system memory. Residue index values from stream context are considered to be memory pointers. 1 Tabular. Residues are stored in a fixed size table at the address indicated by the RES_BH0 and RES_BL0 registers.

17.3.8.32 Residue Base Address High and Low Registers Channel 0 (RES_BH0 and RES_BL0)

The RES_BH0 and RES_BL0 registers are used to provision the location of the channel's residue context table in system memory when residue contexts are stored in tabular form (RES_CONFIG0[TBL] set to 1). The table base address must be aligned to a natural 128 byte address boundary.

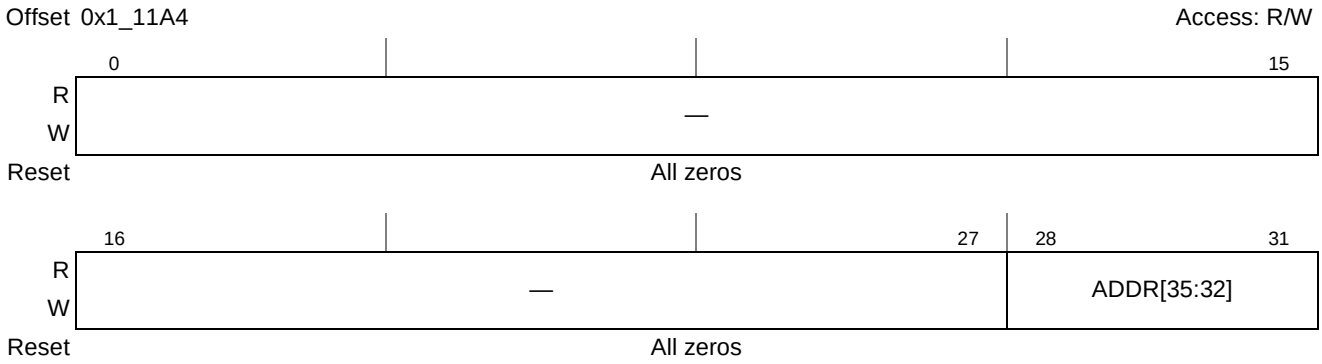


Figure 17-97. Residue Base Address High Register Channel 0 Format

Table 17-100. Residue Base Address High Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	The base address of the Residue table in system memory.

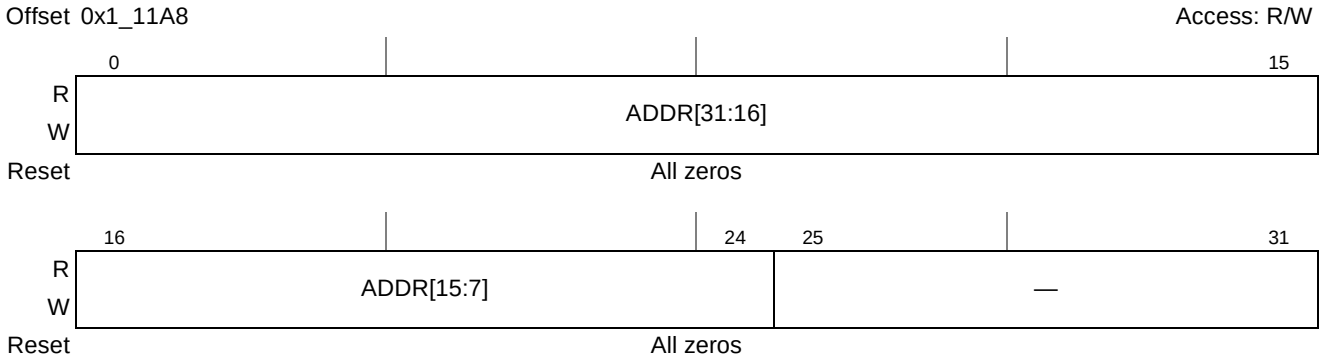


Figure 17-98. Residue Base Address Low Register Channel 0 Format

Table 17-101. Residue Base Address Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–24	ADDR	The base address of the Residue table in system memory.
25–31	—	Reserved, should be cleared.

17.3.8.33 Cache Awareness Control Register Channel 0 (CACR0)

The CACR0 register is used to provide control over cache aware attributes for DMA Engine system memory transactions. See Section 17.4.5.8, “Cache Awareness for more information.

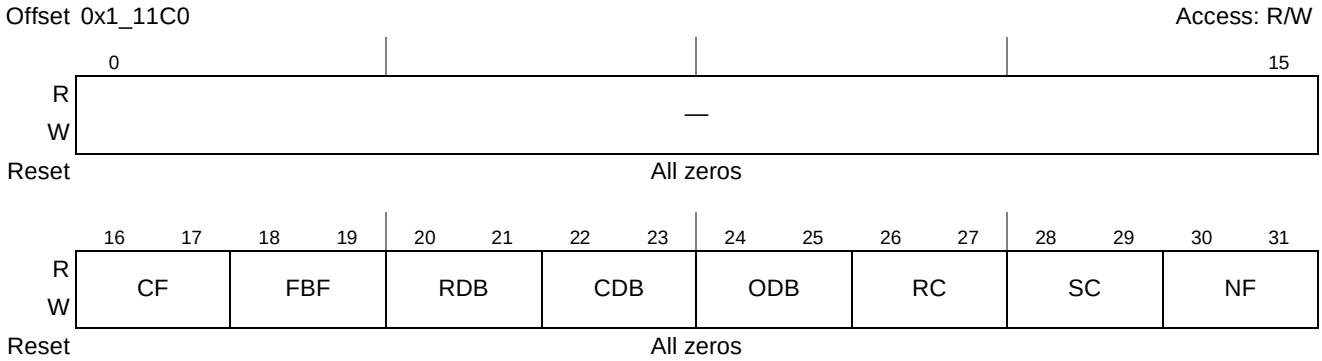


Figure 17-99. Cache Awareness Control Register Channel 0 Format

Table 17-102. Cache Awareness Control Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–17	CF	Command FIFO reads. Controls cache aware attributes during Command FIFO reads transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved
18–19	FBF	Free Buffer Deallocate FIFO reads. Controls cache aware attributes during Free Buffer Deallocate FIFO reads transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved
20–21	RDB	Result Buffer writes. Controls cache aware attributes during Result Buffer writes transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved
22–23	CDB	Command Data Buffer reads. Controls cache aware attributes during Command Data Buffer reads transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Reserved 11 Reserved
24–25	ODB	Output Deflate Data Buffer writes. Controls cache aware attributes during Deflate Data Buffer writes transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved
26–27	RC	Residue Context reads and writes. Controls cache aware attributes during Residue Context reads and writes transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved

Table 17-102. Cache Awareness Control Register Channel 0 Field Descriptions (continued)

Bits	Name	Description
28 SC	29	Stream Context reads and writes. Controls cache aware attributes during Stream Context reads and writes transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved
30–31	NF	Notification FIFO writes. Controls cache aware attributes during Notification FIFO writes transactions. 00 Encoding 0. Non-coherent. Neither Snoop nor Allocate are set during transactions. 01 Encoding 1. Coherent. Snoop is set during transactions, but Allocate is not. 10 Encoding 2. Coherent with Stashing. Snoop is set for all transactions and Allocate for write transactions. 11 Reserved

17.3.8.34 Memory Transaction Priority Control Register Channel 0 (MTPCR0)

The MTPCR0 register allows for control of the priority for the system memory transactions performed by the DMA Engine. Priority values from 0 to 3 are supported, with 0 (default) being the lowest priority setting.

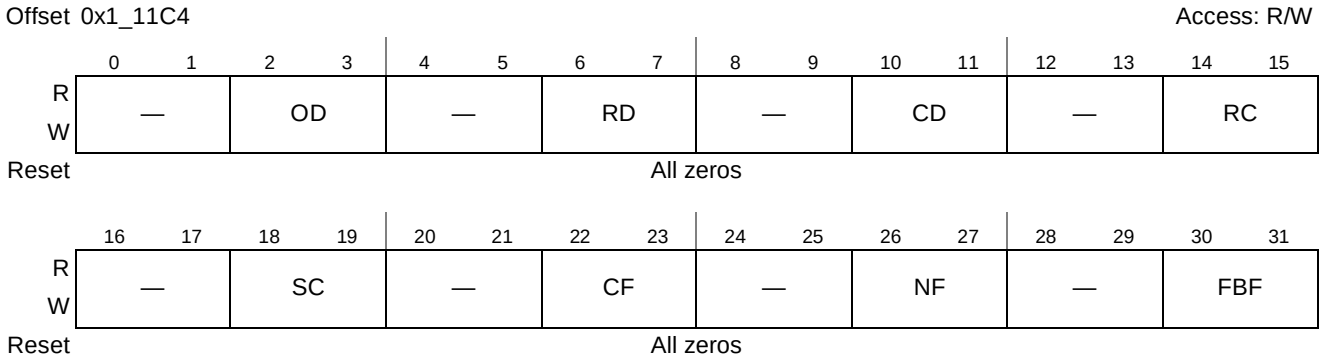


Figure 17-100. Memory Transaction Priority Control Register Channel 0 Format

Table 17-103. Memory Transaction Priority Control Register Channel 0 Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–3	OD	Output Deflate Data Buffer Priority. Specifies the transaction priority applied to Deflate Data Buffer writes. Valid codings are 0 (lowest) to 3 (highest).
4–5	—	Reserved, should be cleared.
6–7	RD	Result Data Priority. Specifies the transaction priority applied to Result Data writes. Valid codings are 0 (lowest) to 3 (highest).
8–9	—	Reserved, should be cleared.

Table 17-103. Memory Transaction Priority Control Register Channel 0 Field Descriptions (continued)

Bits	Name	Description
10–11	CD	Command Data Priority. Specifies the transaction priority applied to Command Data reads. Valid codings are 0 (lowest) to 3 (highest).
12–13	—	Reserved, should be cleared.
14–15	RC	Residue Context Priority. Specifies the transaction priority applied to Residue Context reads and writes. Valid codings are 0 (lowest) to 3 (highest).
16–17	—	Reserved, should be cleared.
18–19	SC	Stream Context Priority. Specifies the transaction priority applied to Stream Context reads and writes. Valid codings are 0 (lowest) to 3 (highest).
20–21		Reserved, should be cleared.
22–23	CF	Command FIFO Priority. Specifies the transaction priority applied to Command FIFO reads. Valid codings are 0 (lowest) to 3 (highest).
24–25	—	Reserved, should be cleared.
26–27	NF	Notification FIFO Priority. Specifies the transaction priority applied to Notification FIFO writes. Valid codings are 0 (lowest) to 3 (highest).
28–29	—	Reserved, should be cleared.
30–31	FBF	Free Buffer Deallocate FIFO Priority. Specifies the transaction priority applied to Free Buffer Deallocate FIFO reads. Valid codings are 0 (lowest) to 3 (highest).

17.3.9 Free Buffer Manager Channel 0 Registers

17.3.9.1 Free Buffer List Threshold A Register Channel 0 (FBL_THRES_A0)

The FBL_THRES_A0 register is used to provision two low-water thresholds for Channel 0's Free Buffer List A:

- Interrupt Threshold. Once the Free Buffer List A length is less than this threshold the Free Buffer List A Channel 0 interrupt status bit in the ISR0 register is set if this source of interrupt is enabled.
- Stop Service Threshold. Once the Free Buffer List A length is less than this threshold the DMA Engine stops servicing this channel, if Free Buffer List A is enabled for this channel.

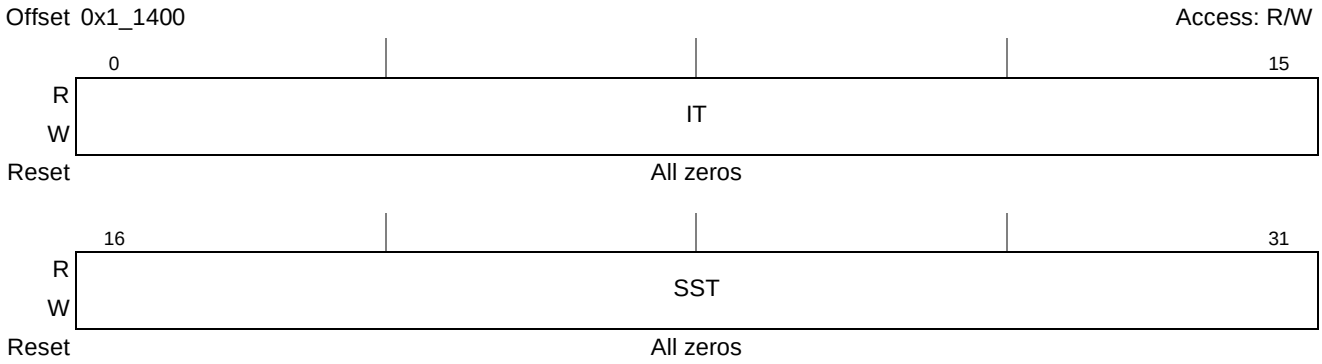


Figure 17-101. Free Buffer List Threshold Register Channel 0 Format

Table 17-104. Free Buffer List Threshold Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	IT	Interrupt Threshold. This field defines minimum length, which results in a Free Buffer List A Interrupt. Status bit to be set in the ISR0 register. If this field is set to zero, then the Free Buffer List A interrupt source is implicitly disabled.
16–31	SST	Stop Service Threshold. This field defines minimum length, which results in the DMA Engine stopping service to this channel.

17.3.9.2 Free Buffer List High and Low Watermark A Register Channel 0 (FBL_HLWM_A0)

The FBL_HLWM_A0 register provides the Free Buffer List A high and low watermark statistical information. The HWM field rolls over silently if the Free Buffer List A number of buffers exceeds 64K buffers.

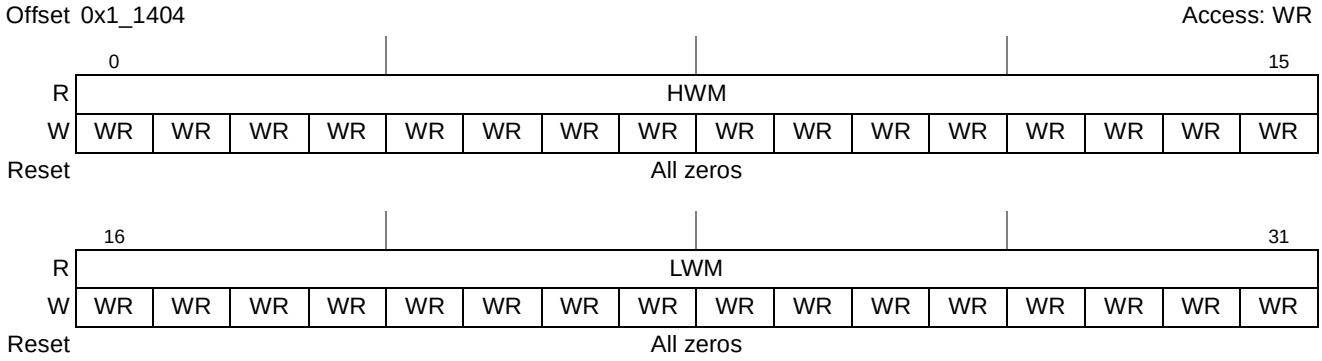


Figure 17-102. Free Buffer List High and Low Watermark A Register Channel 0 Format

Table 17-105. Free Buffer List High and Low Watermark A Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	HWM	High Watermark. Highest value of the Free Buffer List A length since the last reset of this register.
16–31	LWM	Low Watermark. Lowest value of the Free Buffer List A length since the last reset of this register.

17.3.9.3 Free Buffer List Allocation and Deallocation Counter Register A Channel 0 (FBL_AD_CA0)

The FBL_AD_CA0 register provides Free Buffer List A allocation and deallocation statistical information. If these counters roll over, an overflow bit is set in the SCOS0 register.

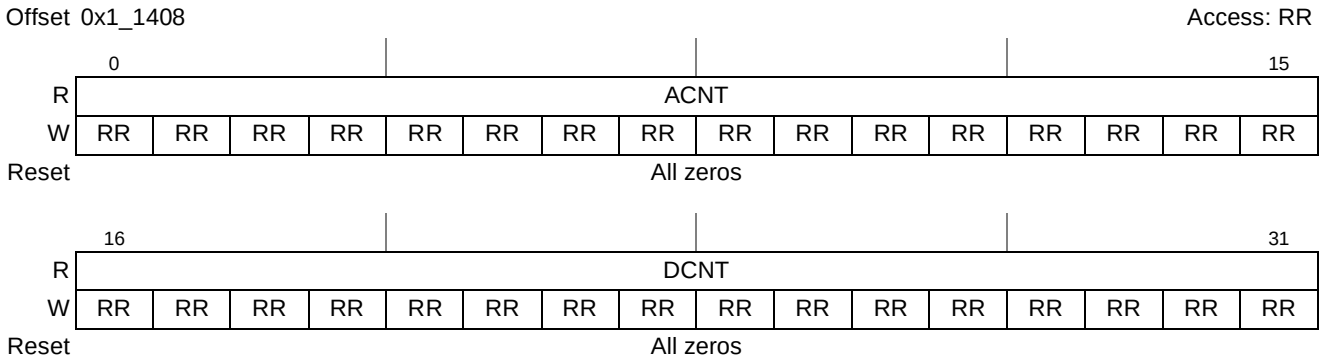


Figure 17-103. Free Buffer List Allocation and Deallocation Counter A Register Channel 0 Format

Table 17-106. Free Buffer List Allocation and Deallocation Counter A Register Channel 0 Field Descriptions

Bits	Name	Description
0–15	ACNT	Allocation Count. The number of free buffers that have been allocated from this free buffer list since the last reset of this register.
16–31	DCNT	Deallocation Count. The number of free buffers that have been deallocated to this free buffer list since the last reset of this register.

17.3.9.4 Free Buffer List On Deck Disable A Register Channel 0 (FBL_ODD_A0)

The FBL_ODD_A0 register is used to prevent the FBM from putting a free buffer into Channel 0's Free Buffer List A on deck internal registers. The on deck registers contain the physical pointer and virtual pointer of a free buffer that has been removed from the free buffer list and is ready for use by the DMA Engine, but does not appear in any data structures.

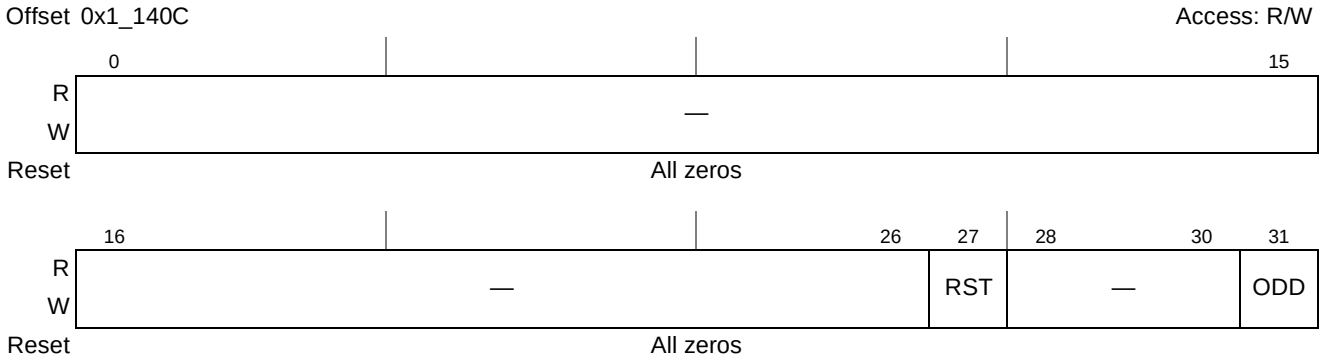


Figure 17-104. Free Buffer List On Deck Disable A Register Channel 0 Format

Table 17-107. Free Buffer List On Deck Disable A Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27	RST	Reset. When set, causes the physical head, virtual head, tail, and length of the physical free buffer list assigned to channel 0's list A to be reset to 0x0. The free buffer list must be enabled in the FBLAAR register in order for this to have an effect. It is only safe to set this bit when it is known that the physical free buffer list is in a static state (all client channels using this free buffer list are static and all memory read activity has had sufficient time to complete).
28–30	—	Reserved, should be cleared.
31	ODD	On Deck Disable. When set, prevents the FBM from placing a free buffer in channel 0's on deck A internal registers (physical pointer and virtual pointer of a free buffer).

17.3.9.5 Free Buffer List Physical Head Address High and Low A Registers Channel 0 (FBL_PH_AH0 and FBL_PH_AL0)

The FBL_PH_AH0 and FBL_PH_AL0 registers offer visibility of the physical address of the head buffer on Channel 0's Free Buffer List A.

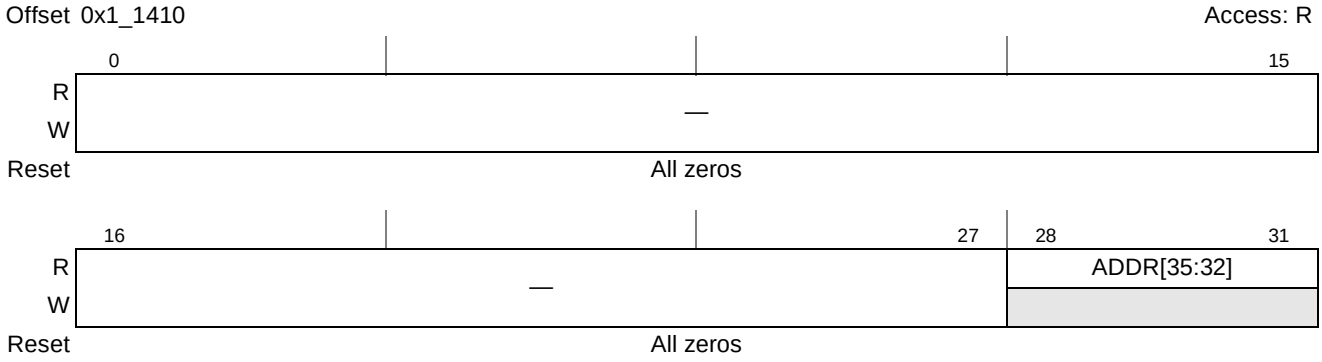


Figure 17-105. Free Buffer List Physical Head Address High A Register Channel 0 Format

Table 17-108. Free Buffer List Physical Head Address High A Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	Address. The upper bits of the physical address of the head buffer on channel 0's Free Buffer List A.

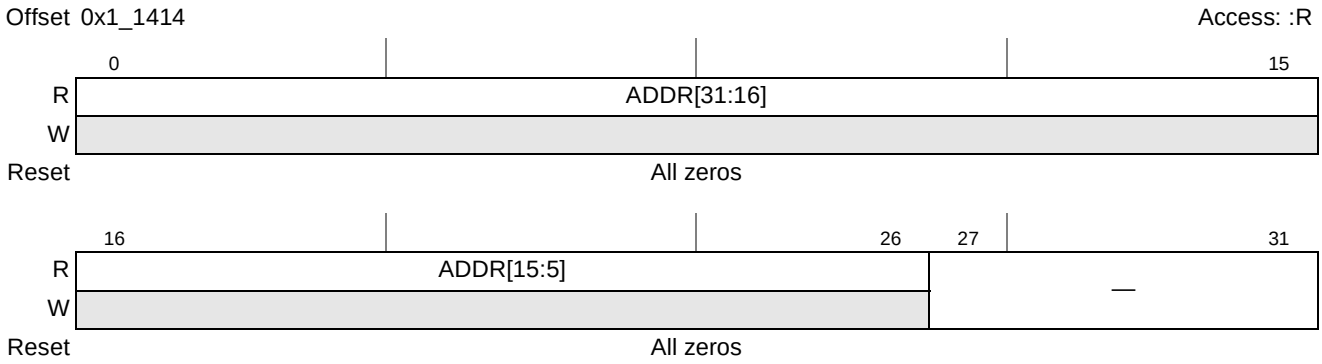


Figure 17-106. Free Buffer List Physical Head Address Low A Register Channel 0 Format

Table 17-109. Free Buffer List Physical Head Address Low A Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	ADDR	Address. The lower bits of the physical address of the head buffer on channel 0's Free Buffer List A.
27–31	—	Reserved, should be cleared.

17.3.9.6 Free Buffer List Virtual Head Address High and Low A Registers Channel 0 (FBL_VH_AH0 and FBL_VH_AL0)

The FBL_VH_AH0 and FBL_VH_AL0 registers offer visibility of the virtual address of the head buffer on Channel 0's Free Buffer List A.

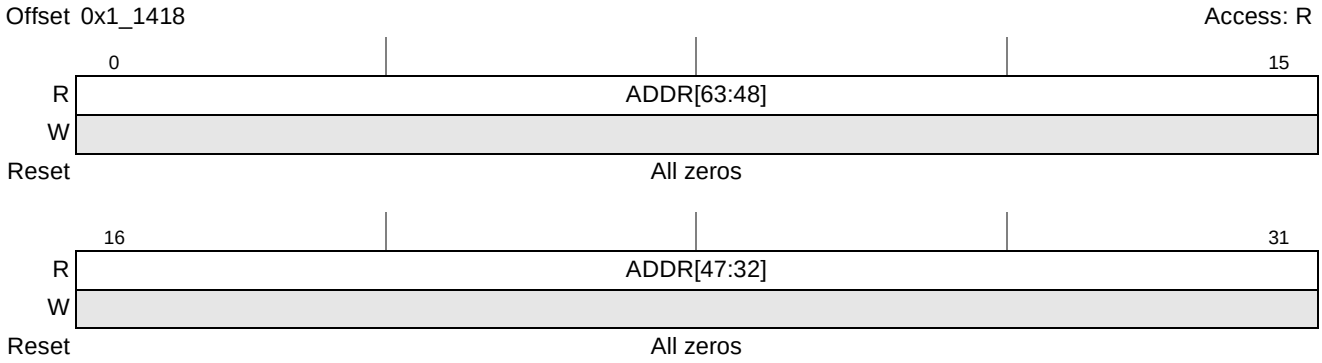


Figure 17-107. Free Buffer List Virtual Head Address High A Register Channel 0 Format

Table 17-110. Free Buffer List Virtual Head Address High A Register Channel 0 Field Descriptions

Bits	Name	Description
0–31	ADDR	Address. The upper 32 bits of the virtual address of the head buffer on channel 0's Free Buffer List A.

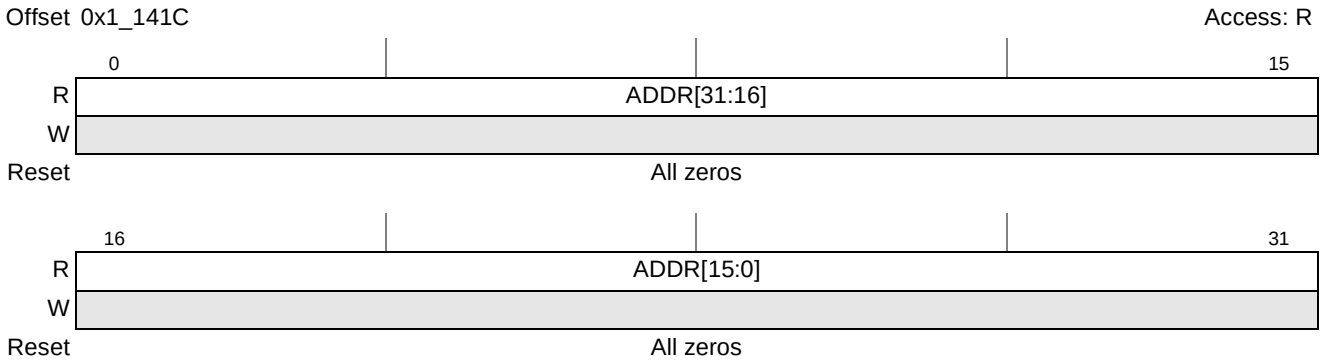


Figure 17-108. Free Buffer List Virtual Head Address Low A Register Channel 0 Format

Table 17-111. Free Buffer List Virtual Head Address Low A Register Channel 0 Field Descriptions

Bits	Name	Description
0–31	ADDR	Address. The lower bits of the virtual address of the head buffer on channel 0's Free Buffer List A.

17.3.9.7 Free Buffer List Physical Tail Address High and Low A Registers Channel 0 (FBL_PT_AH0 and FBL_PT_AL0)

The FBL_PT_AH0 and FBL_PT_AL0 registers offer visibility of the physical address of the tail buffer on Channel 0's Free Buffer List A.

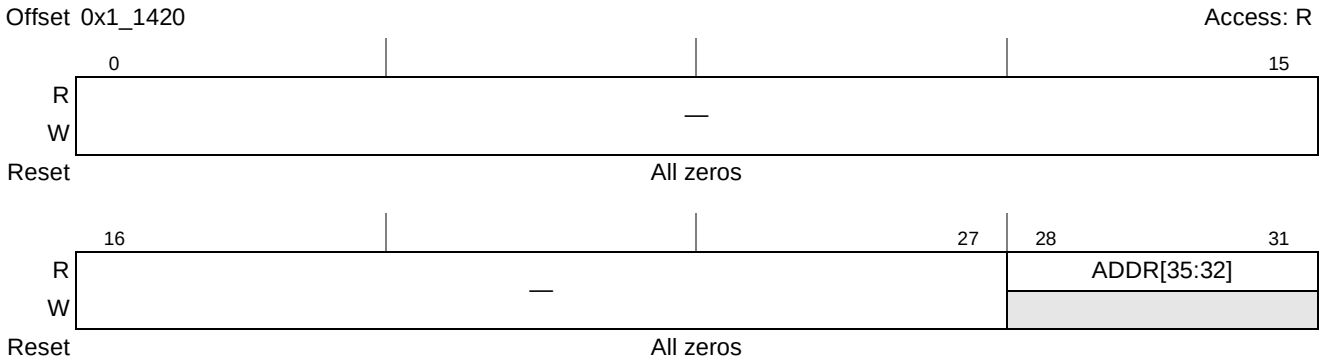


Figure 17-109. Free Buffer List Physical Tail Address High A Register Channel 0 Format

Table 17-112. Free Buffer List Physical Tail Address High A Register Channel 0 Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	ADDR	Address. The upper bits of the physical address of the tail buffer on channel 0's Free Buffer List A.

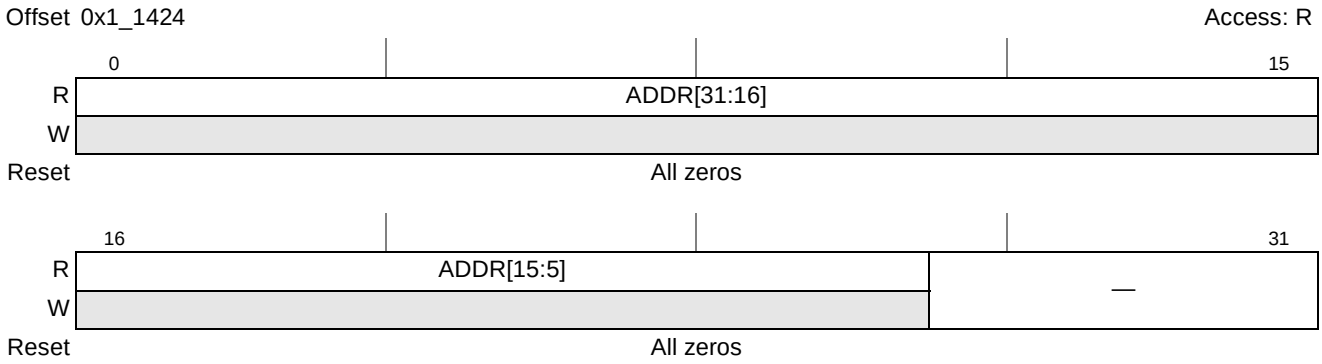


Figure 17-110. Free Buffer List Physical Tail Address Low A Register Channel 0 Format

Table 17-113. Free Buffer List Physical Tail Address Low A Register Channel 0 Field Descriptions

Bits	Name	Description
0–26	ADDR	Address. The lower bits of the physical address of the tail buffer on channel 0's Free Buffer List A.
27–31	—	Reserved, should be cleared.

17.3.9.8 Free Buffer List Number of Buffers A Register Channel 0 (FBL_NB_A0)

The FBL_NB_A0 register provides the number of free buffers on Channel 0's Free Buffer List A.

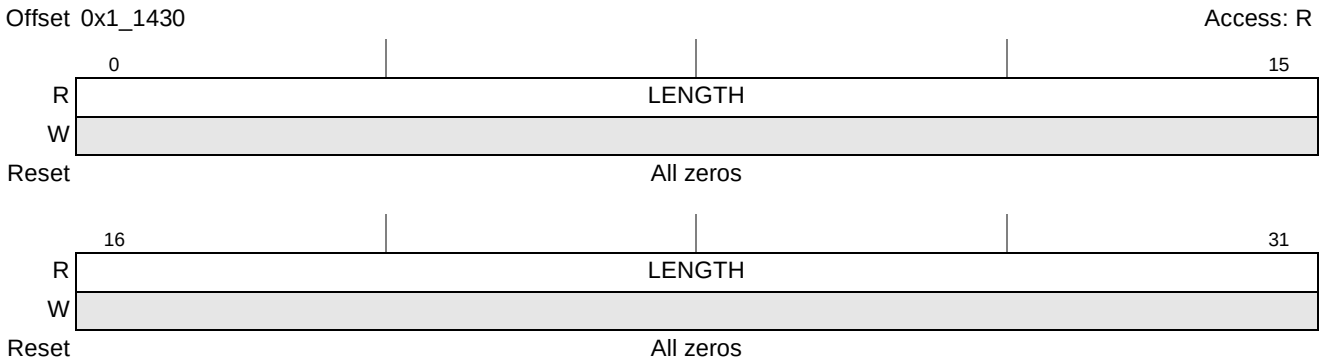


Figure 17-111. Free Buffer List Number of Buffers A Register Channel 0 Format

Table 17-114. Free Buffer List Number of Buffers A Register Channel 0 Field Descriptions

Bits	Name	Description
0–31	LENGTH	Length. The length of channel 0's Free Buffer List A.

17.3.9.9 Free Buffer List Buffer Size A Register Channel 0 (FBL_BS_A0)

The FBL_BS_A0 register provides information about Channel 0's Free Buffer List A.

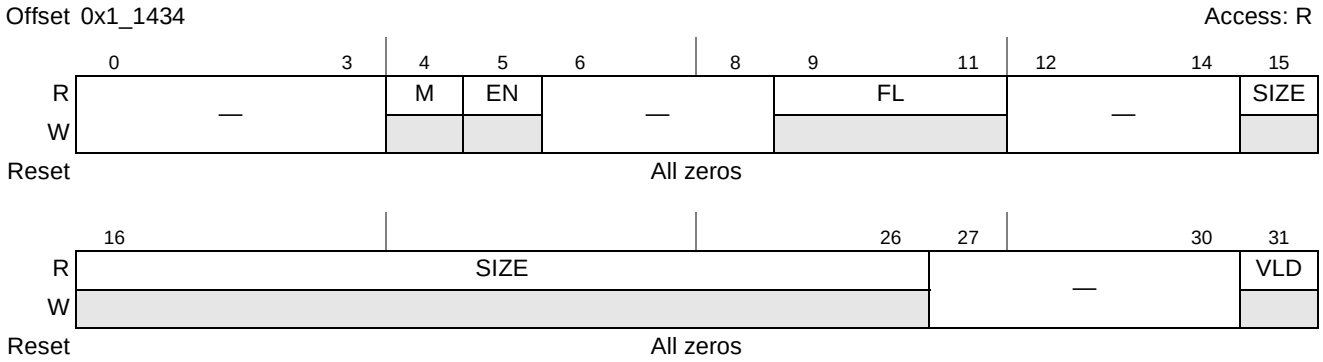


Figure 17-112. Free Buffer List Buffer Size A Register Channel 0 Format

Table 17-115. Free Buffer List Buffer Size A Register Channel 0 Field Descriptions

Bits	Name	Description
0–3	—	Reserved, should be cleared.
4	M	Master. This bit reflects the state of channel 0's Free Buffer List A Master bit in the FBLAAR register.
5	EN	Enable. This bit reflects the state of channel 0's Free Buffer List A Enable bit in the FBLAAR register.
6–8	—	Reserved, should be cleared.
9–11	FL	Free buffer list. This field reflects the physical free buffer list that is mapped to channel 0's Free Buffer List A.
12–14	—	Reserved, should be cleared.
15–26	SIZE	Size. This field gives the size of free buffers for this free buffer list in 32 byte quantities. The minimum buffer size is 128 bytes (SIZE=4), the maximum buffer size is 64K bytes (SIZE=2048). A setting of 0x0 specifies 64K bytes sized buffers.
27–30	—	Reserved, should be cleared.
31	VLD	Valid. Indicates that the on deck pointer is valid and not reflecting a stale value.

17.3.9.10 Free Buffer List Virtual On Deck Pointer A High and Low Registers Channel 0 (FBL_PD_AH0 and FBL_PD_AL0)

The FBL_PD_AH0 and FBL_PD_AL0 registers offer visibility of the virtual pointer of the free buffer stored in channel 0's Free Buffer List A on deck virtual pointer register. The on deck virtual pointer is a pointer of a free buffer that has been removed from the free buffer list and is ready for use by the DMA Engine, but does not appear in any data structures.

The FBL_BS_A0[VLD] bit indicates whether the on deck pointer published in the FBL_PD_AH0 and FBL_PD_AL0 registers is valid and not reflecting a stale value.

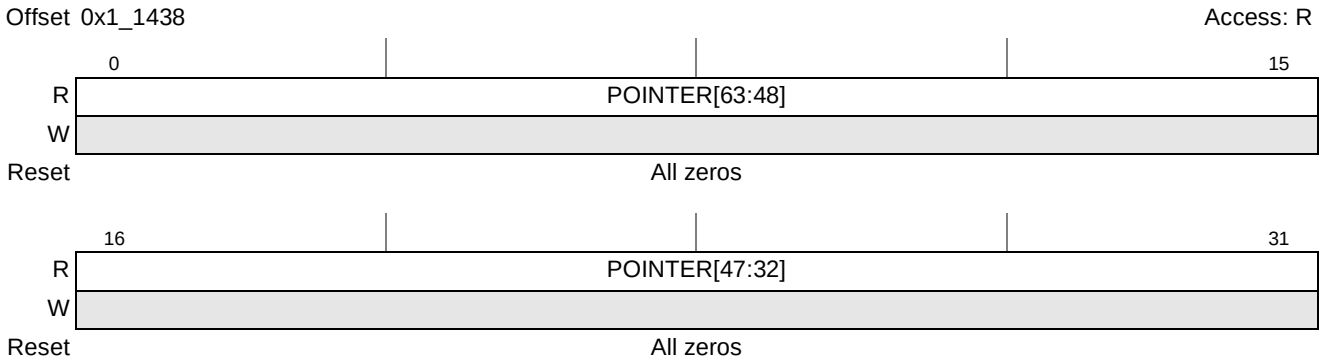


Figure 17-113. Free Buffer List Virtual On Deck Pointer A High Register Channel 0 Format

Table 17-116. Free Buffer List Virtual On Deck Pointer A High Register Channel 0 Field Descriptions

Bits	Name	Description
0–31	—	Pointer. The upper 32 bits of the on deck pointer.

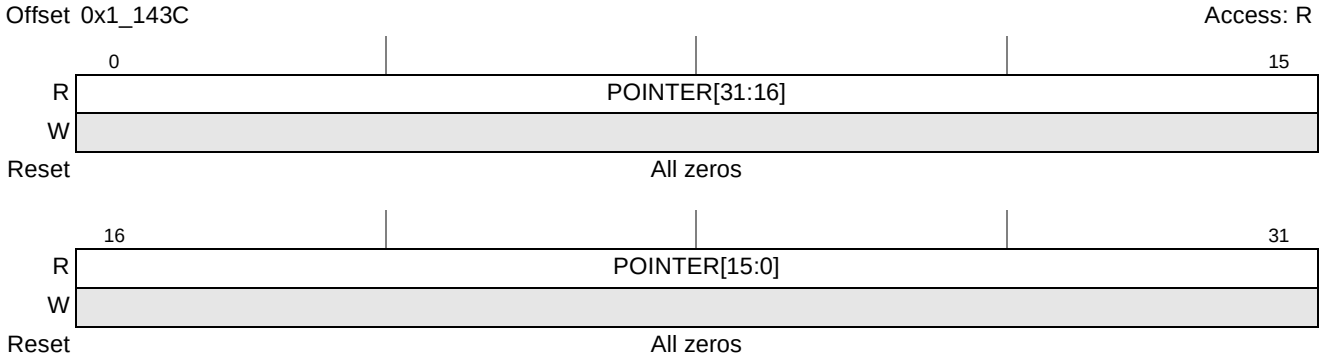


Figure 17-114. Free Buffer List Virtual On Deck Pointer A Low Register Channel 0 Format

Table 17-117. Free Buffer List Virtual On Deck Pointer A Low Register Channel 0 Field Descriptions

Bits	Name	Description
0–31	—	Pointer. The lower bits of the virtual address of the on deck buffer on channel 0's Free Buffer List A.

17.3.9.11 Free Buffer List Threshold B Register Channel 0 (FBL_THRES_B0)

The FBL_THRES_B0 register is identical to the FBL_THRES_A0 register (see Section 17.3.9.1, "Free Buffer List Threshold A Register Channel 0 (FBL_THRES_A0)), except the thresholds are applied to channel 0's Free Buffer List B instead of Free Buffer List A.

17.3.9.12 Free Buffer List High and Low Watermark B Register Channel 0 (FBL_HLWM_B0)

The FBL_HLWM_B0 register is identical to the FBL_HLWM_A0 register (see Section 17.3.9.2, "Free Buffer List High and Low Watermark A Register Channel 0 (FBL_HLWM_A0)), except it applies to Channel 0's Free Buffer List B instead of Free Buffer List A.

17.3.9.13 Free Buffer List Allocation and Deallocation Counter Register B Channel 0 (FBL_AD_CB0)

The FBL_AD_CB0 register is identical to the FBL_AD_CA0 register (see Section 17.3.9.3, “Free Buffer List Allocation and Deallocation Counter Register A Channel 0 (FBL_AD_CA0)), except it applies to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.14 Free Buffer List On Deck Disable B Register Channel 0 (FBL_ODD_B0)

The FBL_ODD_B0 register is identical to the FBL_ODD_A0 register (see Section 17.3.9.4, “Free Buffer List On Deck Disable A Register Channel 0 (FBL_ODD_A0)), except it applies to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.15 Free Buffer List Physical Head Address High and Low B Registers Channel 0 (FBL_PH_BH0 and FBL_PH_BL0)

The FBL_PH_BH0 and FBL_PH_BL0 registers are identical to the FBL_PH_AH0 and FBL_PH_AL0 registers (see Section 17.3.9.5, “Free Buffer List Physical Head Address High and Low A Registers Channel 0 (FBL_PH_AH0 and FBL_PH_AL0)), except they apply to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.16 Free Buffer List Virtual Head Address High and Low B Registers Channel 0 (FBL_VH_BH0 and FBL_VH_BL0)

The FBL_VH_BH0 and FBL_VH_BL0 registers are identical to the FBL_VH_AH0 and FBL_VH_AL0 registers (see Section 17.3.9.6, “Free Buffer List Virtual Head Address High and Low A Registers Channel 0 (FBL_VH_AH0 and FBL_VH_AL0)), except they apply to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.17 Free Buffer List Physical Tail Address High and Low B Registers Channel 0 (FBL_PT_BH0 and FBL_PT_BL0)

The FBL_PT_BH0 and FBL_PT_BL0 registers are identical to the FBL_PT_AH0 and FBL_PT_AL0 registers (see Section 17.3.9.7, “Free Buffer List Physical Tail Address High and Low A Registers Channel 0 (FBL_PT_AH0 and FBL_PT_AL0)), except they apply to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.18 Free Buffer List Number of Buffers B Register Channel 0 (FBL_NB_B0)

The FBL_NB_B0 register is identical to the FBL_NB_A0 register (see Section 17.3.9.8, “Free Buffer List Number of Buffers A Register Channel 0 (FBL_NB_A0)), except it applies to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.19 Free Buffer List Buffer Size B Register Channel 0 (FBL_BS_B0)

The FBL_BS_B0 register is identical to the FBL_BS_A0 register (see Section 17.3.9.9, “Free Buffer List Buffer Size A Register Channel 0 (FBL_BS_A0)), except it applies to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.3.9.20 Free Buffer List Virtual On Deck Pointer B High and Low Registers Channel 0 (FBL_PD_BH0 and FBL_PD_BL0)

The FBL_PD_BH0 and FBL_PD_BL0 registers are identical to the FBL_PD_AH0 and FBL_PD_AL0 (see Section 17.3.9.10, “Free Buffer List Virtual On Deck Pointer A High and Low Registers Channel 0 (FBL_PD_AH0 and FBL_PD_AL0)), except they apply to Channel 0’s Free Buffer List B instead of Free Buffer List A.

17.4 Functional Description

The Pattern Matcher IP block includes both pattern matching and decompression functionality in a single integrated block.

The Pattern Matcher IP block consists of the functional units listed in Table 17-118.

Table 17-118. Pattern Matcher Functional Units

Functional Unit	Description
Key Element Scanner	Implements a multi-stage, hash based, proprietary algorithm to determine the likelihood of the data being a match with a pattern. Matches found at this stage are delivered to the Data Examination Engine which performs a more stringent comparison to determine if the match actually exists or not.
Data Examination Engine	Implements a Non-deterministic Finite Automaton (NFA) (state machine) capable of implementing a significant subset of the regular expression (regex) pattern definition language. It also provides the ability to express patterns with capabilities beyond that provided by regex languages.
Stateful Rule Engine	Provides a means to link multiple pattern matches together and maintains state between matches including information extracted from the data under inspection.
Deflate Engine	Implements data decompression using the Deflate algorithm. Handles raw Deflate, as well as gzip and zlib format headers containing Deflate payloads.
DMA Engine	Offers four independent DMA channels which driver software uses to command pattern matching/decompression work and to receive notifications of completed pattern matching/decompression work.
Free Buffer Manager	Manages pools of free buffers that are used by the DMA Engine for generating pattern matching reports and decompressed data outputs.
Memory Interface Arbiter	Provides access to system memory for all of the functional units within the Pattern Matcher allowing them to initiate read and write transactions over the system bus.
Internal Register Interface	Provides access to all registers within the Pattern Matcher.

The Pattern Matcher is implemented as a series of processing functional units pipelined together to achieve desired functionality and performance. Up to 8 work units can be in flight in the pipeline at any given time. A work unit represents an atomic work request operation to the Pattern Matcher. Work request operations are conveyed to the Pattern Matcher through the Pattern Matcher DMA Command FIFOs (one work unit per Command FIFO entry). The Pattern Matcher implements a single pipeline meaning that processing of work units are completed in the same order as they were initially selected.

The first stage of the pipeline is the DMA Engine. The DMA Engine selects a work unit request operation from one the four DMA channels according to the configured scheduling policy, processes the request operation and in the case of a pattern matching/decompression work request sends the request and initial data read from the input data buffer to the next stage which is the Deflate Engine. The Deflate Engine and subsequent stages in the pipeline process the work units in a “streaming” fashion. That is, they don't wait for all of the input data before they start processing the work unit (it is not necessary to have the entire content of the work unit stored inside the Pattern Matcher block).

The Deflate Engine performs data decompression using the Deflate algorithm if data is required to be decompressed. If data is not required to be decompressed, the data is sent directly to the next stage for pattern matching. If both decompression and pattern matching are required, the decompression data is sent directly to the next stage for pattern matching without writing the decompressed data into system memory. Optionally, decompressed data can also be written into system memory by having the same decompressed data sent to the DMA Engine as well. If only decompression is required, the decompressed data is only sent to the DMA Engine. The DMA Engine writes the decompressed output in system memory and populates the Notification FIFO accordingly.

The core pattern matching functionality is implemented as a three stage pipeline. The first stage is the Key Element Scanner (KES) which serves as preliminary filter by detecting the possibility of matches of up to 16,000 patterns simultaneously through a single pass through the data at a scan rate of up to 1 byte per cycle. The Key Element Scanner implements a multi-stage, hash based, proprietary algorithm using on-chip memory to determine the likelihood of the data being a match with a pattern. Matches found at this stage are delivered to the Data Examination Engine which performs a more stringent comparison to determine if the match actually exists or not. The Data Examination Engine implements a Non-Deterministic Finite Automaton (NFA) capable of implementing a significant subset of the Regular Expression (regex) pattern definition language as well as many constructs which cannot be expressed in regex. Successfully matched patterns are passed to the next stage called the Stateful Rule Engine. The Stateful Rule Engine’s main function is to execute Stateful rules against pattern match events. Stateful rules provide the means to track state and context information between matches. The Stateful Rule Engine also formats the reports of matches including pattern match events that don't trigger the execution of Stateful rules. The reports are passed to the DMA Engine which in turn produces an entry in the Notification FIFO that contains a pointer to an output data descriptor containing the pattern reports associated for a given work unit.

The remaining functional units provide common functions to the Pattern Matcher functional units. The Memory Interface Arbiter provides access to system memory for all other Pattern Matcher functional units allowing them to initiate read and write transactions over the system bus. The Internal Register Interface provides access to all Pattern Matcher registers. Finally the Free Buffer Manager manages pools of free buffers that are used by the DMA Engine for generating pattern matching reports and decompressed data outputs for work unit requests that don't explicitly specify output buffers.

The core pattern matching functional units (KES, DXE and SRE) are described in more details in the next sub-sections. The decompression functionality description (such as Deflate Engine) follows after the core pattern matching functionality description. Next the DMA Engine and Free Buffer Manager are described followed by the Memory Interface Arbiter and Internal Register Interface functional units. Finally, the last sub-section describes Pattern Matcher events (from all functional units) that can be counted by the central Performance Monitor.

17.4.1 Key Element Scanner

The purpose of the Key Element Scanner (KES) is to pre-scan the data (here after referred as a String Under Inspection, or SUI) using internal hash tables and to identify at each alignment within the SUI the patterns that could possibly produce a positive match (such as the DXE NFA that need to be run) and thereby significantly reducing the work of the DXE as full pattern evaluations are substantially reduced. A SUI is composed of the contents of a work unit input data, prepended with the residue bytes from the previous SUI within a given stream. The residue refers to the tail end of previously scanned data. The size of residue may be up to 127 bytes. The function of prepending residue bytes is selectable on a per stream basis. The residue feature allows the Pattern Matcher to match patterns that cross work units.

If the Data Examination Engine (DXE) was to operate alone, it would have to execute an NFA description for each pattern and at each byte alignment within the SUI to catch all matches of all patterns. Take note that the actual hash functions implemented by KES aren't described in this document. A description of these functions can be made available separately if a customer wishes to construct the KES hash tables using its own software libraries and not the software libraries that accompany the Pattern Matcher.

SUI is fed to both the KES and the DXE simultaneously and each accesses and maintains its own flow-through buffered version of the SUI such that they can work on the same SUI (or different SUI's) in a skewed manner (the DXE always lags the KES). The SRE does not directly scan any data, instead it reacts to pattern matches found by the previous two stages to perform its work.

The SUI is stored in its original format and 3 different normalized forms which are:

- Equivalent Byte—Each byte value maps to a byte value. The mapping is user defined and is configured by writing entries into the Equivalent Byte Mapping internal table (see Table 17-152 for details on the Write command).
- Pre-Defined Group—Each byte value maps to a 3 bit group value (0..7). The mapping is pre-defined. See Table 17-119 for the mapping description.
- User-Defined Group—Each byte value maps to a 3 bit group value (0..7). The mapping is user defined and is configured by writing entries in the User-Defined Group Mapping internal table (see Table 17-152 for details on the Write command).

Table 17-119. Pre-Defined Group

Group	Description	Byte Value (hex)
—	Non-printing control characters(excludes Tab, LineFeed, FormFeed, CarriageReturn)	00-08,0b,0e-1f,7f
—	Whitespace (such as Tab, LineFeed, FormFeed, CarriageReturn, Space)	09,0a,0c,0d,20
—	Symbol Characters (such as !"#$%&'()*+,-./:;< = >?@ [\ ]^`{|}~)	21-2f,3a-40,5b-5e,60,7b-7e
—	UnderScore (such as _)	5f

Table 17-119. Pre-Defined Group (continued)

Group	Description	Byte Value (hex)
—	Digit 0-9 (such as 0123456789)	30-39
—	Upper case alpha A-Z(such as ABCDEFGHIJKLMNOPQRSTUVW)	41-5a
—	Lower case alpha a-z(such as abcdefghijklmnopqrstuvwxyz)	61-7a
—	Extended ASCII characters	80-ff

To identify possible pattern matches, the KES scans for a “Key Element” from each of the patterns at each possible alignment. The Key Element of a pattern is a set of symbol compares that:

- Must match their SUI in order for that pattern to possibly match.
- Are of type “Original Byte” or “Equivalent Byte”, or a mix of the two. However take note that the Confidence entry generated must be either all of type “Original Byte” or all of type “Equivalent Byte”. For instance, the Key Element “[Aa]bc” which is a mix of type “Original Byte” and “Equivalent Byte” can have its Confidence entry defined in the “Equivalent Byte” form (for example, “[Aa][Bb][Cc]”) or in the “Original Byte” form (for example, “bc”).
- Have a fixed positional relationship to each other when be applied to the SUI.
- Cover as much of the pattern as possible (up to 62 symbols).

An alternative way of viewing a Key Element is as a set contiguous symbol compares that includes “wildcard” compares (for example, don’t cares) of finite widths within it.

Below are examples of Key Elements using regex syntax (_ implies a fixed gap in a Key Element)

Regex: ABCDXYZ Key Element: ABCDXYZ

Regex: ABC[0-9]XYZ Key Element: ABC_XYZ

Regex: ABC(EFG|JKL)XYZ Key Element: ABC___XYZ

Regex: ABC(EFG|JKLM)XYZ Key Element: ABC or XYZ

Regex: ABC(XYZ|JKL) Key Element: ABC

Regex: ABCD{2,5}XYZ Key Element: either ABCDD or DDXYZ

Regex: (ABC|DEF)(QRS|XYZ) Key Element: regular expression split required (for example, ABC(QRS|XYZ) with a Key Element of ABC and DEF(QRS|XYZ) with a Key Element of DEF)

Regex: ABC(QRS|XYZ) Key Element: ABC

Regex: DEF(QRS|XYZ) Key Element: DEF

Regex: [Aa][Bb][Cc] Key Element: [Aa][Bb][Cc]

Regex: [Aa]bc Key Element: [Aa]bc

Information about the Key Elements is pre-loaded into tables stored in internal memories dedicated to KES and searched using a proprietary multi-level hash algorithm implemented as two stages. The first stage is referred as the Trigger stage and the second stage is referred as the Confidence stage. The Trigger stage and Confidence stage are depicted in Figure 17-115. They are shown in the figure as directly coupled in

but in the actual implementation they are pipelined functions. They are decoupled by a FIFO that operates on a skewed region of the SUI array. This allows the Trigger stage to continue to scan a new alignment at every clock cycle despite the fact that the Confidence stage may need to spend multiple cycles at a particular alignment.

The Trigger stage detects the possibility of a pattern match by detecting the possibility of a fingerprint of a pattern at a particular alignment. A fingerprint for a pattern is a set of contiguous symbols of type "Equivalent Byte" of length 1, 2, or K from within the Key Element. K is configurable from 2..16 but must be static (see Section 17.3.2.9, "KES Variable Length Trigger Size (KVLTS)." Operating on the "Equivalent Byte" form allows to define Key Elements and fingerprints with symbols that represent multiple symbols in the "Original Byte" form. For instance, a Key Element/fingerprint symbol can represent "a" and "A" in the original SUI if both values were mapped to "A" in the "Equivalent Byte" representation. This capability provides the means to support case-insensitive matches in a very efficient manner.

The Trigger stage detects the presence of these fingerprints through "look-up" methods referred respectively as 1-byte Trigger, 2-byte Trigger and Variable Length Trigger[1]. The Trigger stage employs hashing as a "look-up" method for the 2-byte Trigger and Variable Length Trigger and direct indexing for the 1-byte Trigger. Each look-up of the 3 fingerprint sizes yields an indication of a possible fingerprint match and an index to a confidence entry (or the first of a chain of confidence entries) to be processed by the Confidence stage. The three "look-up" methods access independent internal trigger memories and are performed in parallel for each SUI alignment. Note that no attempt is made to fully confirm the occurrence of the fingerprint because the Confidence stage builds confidence more effectively and efficiently by confirming the possible occurrence of a Key Elements of patterns which by definition includes the fingerprint.

In addition to identifying the possibility of a fingerprint, the Trigger stage can detect special positions in the SUI (for example, start/end of stream, start/end of work unit, and start/end of new line). These position matches are referred to as Special Triggers. Any occurrence of these special positions in the SUI causes a look-up into the Special Trigger table to determine whether or not access to the Confidence table (Confidence stage) is to be performed. The Special Triggers and fingerprint look-ups are performed in parallel accessing independent internal memories for each SUI alignment.

Based on the indication of a possible match and index from the Trigger stage, the Confidence stage accesses a "confidence entry" or a chain of confidence entries. Each confidence entry contains a description of a "Key Element" (indicates positions relative to fingerprint/Special Trigger that are part of the Key Element and whether the symbols are all of type "Original Byte" or "Equivalent Byte") and a "confidence hash value" of the Key Element. The Confidence stage performs a hash over the SUI bytes that are necessary to be the Key Element for a matching pattern and the result is a compared to the "confidence hash value".

1. The Variable Length Trigger (VLT) configured with a value of 2 (such as 2 bytes) provides the same functionality as the 2-byte Trigger but the hash performance for the VLT is better as the VLT table contains more entries than the 2-byte Trigger table. Thus it may be advantageous to use a 2-byte VLT instead of a 2-byte Trigger in situations where fingerprints longer than 2 bytes are not required.

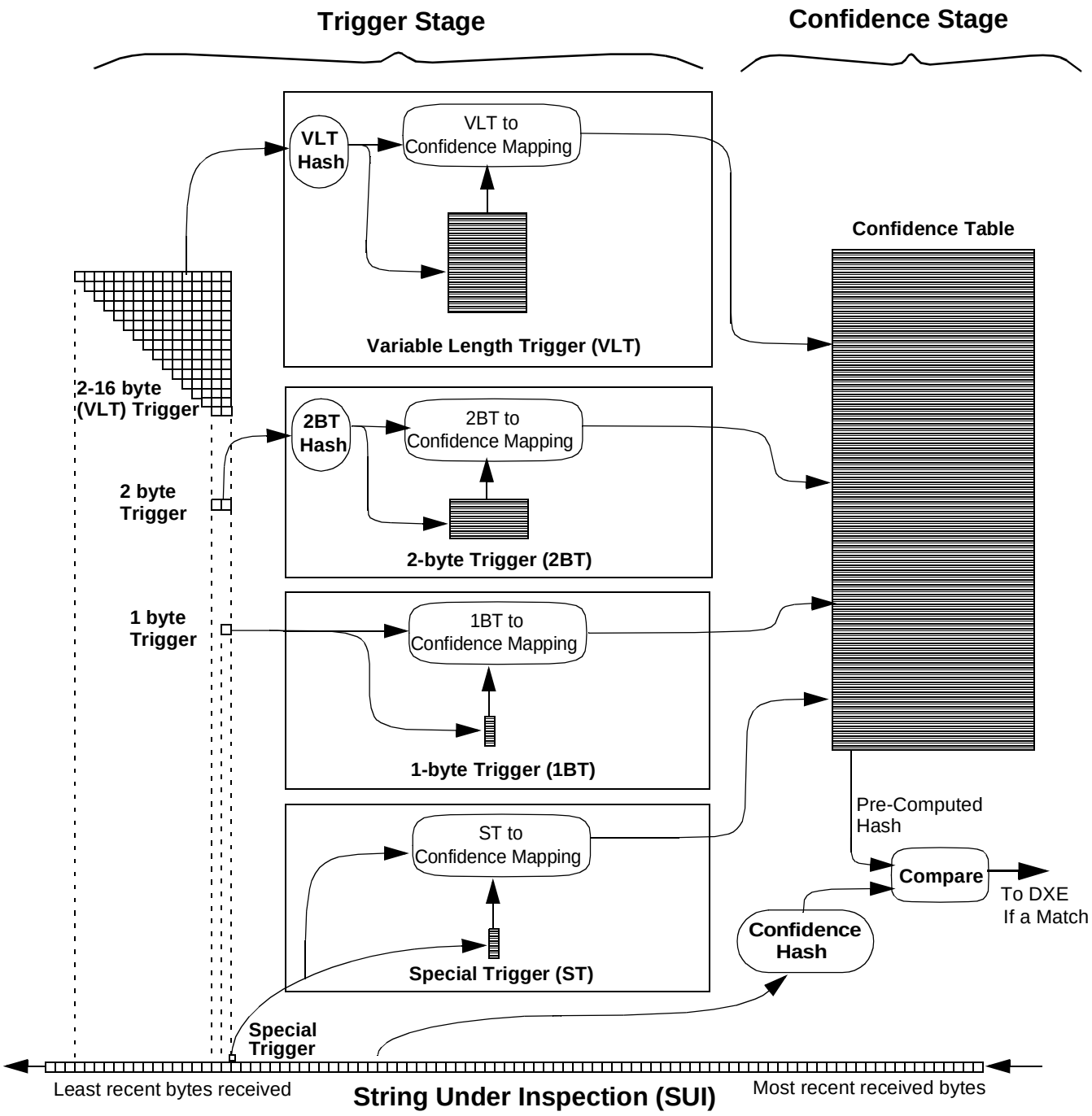


Figure 17-115. Trigger and Confidence Stage

17.4.1.1 Trigger Stage

The Trigger stage detects the possibility of a pattern match by detecting the possibility of a fingerprint of a pattern or a position at which a pattern may be anchored. It performs this function by executing multiple "look-up" methods in parallel for each SUI alignment. These "look-up" methods are: 1-byte Trigger,

2-byte Trigger, Variable Length Trigger and Special Triggers. These methods are described in more details in the following sub-sections.

17.4.1.1.1 Variable Length Trigger

The Variable Length Trigger (VLT) consists of a fingerprint hash (for example, VLT hash) performed over K contiguous bytes (2..16) of the SUI stored in its “Equivalent Byte” form. The VLT hash is performed at each SUI alignment. A register setting (see Section 17.3.2.9, “KES Variable Length Trigger Size (KVLTS)) determines the number of bytes (such as “K” value) to be covered by the VLT hash function. The output of the VLT hash function is used as the index into the VLT table. The VLT table is a compressed hash table of a “uncompressed VLT table” which is structured as a 2-dimension array of 4096 rows and 128 columns where each element is a single bit which indicates whether or not there is a pattern for this hash value. This is illustrated in Figure 17-116.

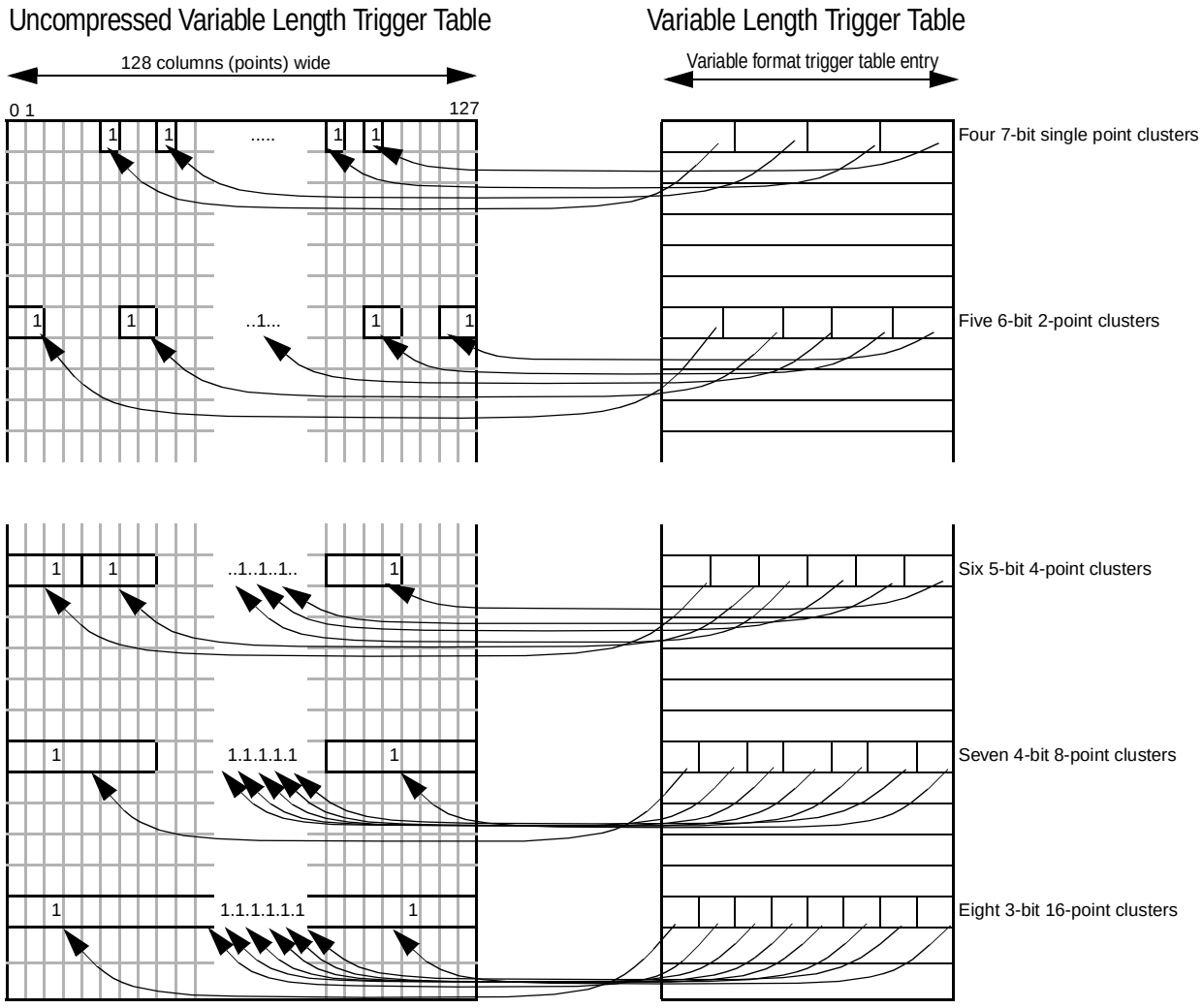


Figure 17-116. Variable Length Trigger Table

Each row of the VLT table is constructed by collapsing columns of the uncompressed VLT table into one of the following row formats:

- 4x7 format - Each valid entry in a row of the uncompressed VLT table is represented by a 7-bit identifier. The 7-bit identifier represents the column index of the valid entry in the uncompressed VLT table. This format supports a maximum of four 7-bit identifiers for a total of 4 valid entries within a row since each identifier represents a single valid entry.
- 5x6 format - The 128 columns of the uncompressed VLT table are grouped into 64 clusters (column 0 and 1 map to cluster 0, column 2 and 3 map to cluster 1 and so on) where each cluster is identified by a 6-bit identifier that identifies the cluster in which a valid entry belongs to. There is the possibility of having *compression collisions* since valid entries belonging to the same cluster is represented by the same 6-bit identifier. This format supports a maximum of five 6-bit identifiers for a total of 10 valid entries within a row since each identifier may represent 2 valid entries.
- 6x5 format - The 128 columns of the uncompressed VLT table are grouped into 32 clusters (column 0..3 map to cluster 0, column 4..7 map to cluster 1 and so on) where each cluster is identified by a 5-bit identifier that identifies the cluster in which a valid entry belongs to. There is the possibility of *compression collisions* since valid entries belonging to the same cluster is represented by the same 5-bit identifier. This format supports a maximum of six 5-bit identifiers for a total of 24 valid entries within a row since each identifier may represent 4 valid entries.
- 7x4 format - The 128 columns of the uncompressed VLT table are grouped into 16 clusters (column 0..7 map to cluster 0, column 8..15 map to cluster 1 and so on) where each cluster is identified by a 4-bit identifier that identifies the cluster in which a valid entry belongs to. There is the possibility of *compression collisions* since valid entries belonging to the same cluster is represented by the same 4-bit identifier. This format supports a maximum of seven 4-bit identifiers for a total of 56 valid entries within a row since each identifier may represent 8 valid entries.
- 8x3 format - The 128 columns of the uncompressed VLT table are grouped into 8 clusters (column 0..15 map to cluster 0, column 16..31 map to cluster 1 and so on) where each cluster is identified by a 3-bit identifier that identifies the cluster in which a valid entry belongs to. There is the possibility of *compression collisions* since valid entries belonging to the same cluster is represented by the same 3-bit identifier. This format supports a maximum of eight 3-bit identifiers for a total of 128 valid entries within a row since each identifier may represent 16 valid entries.

The VLT hash function always produces 19 bits of output hash irrespective of the input data size (2..16 bytes). As illustrated in Figure 17-117, the upper 12 output bits of the VLT hash selects one of 4196 rows (or entries) from the VLT table whereby the lower 7 output bits of the VLT hash represents the identifier of the cluster to which K bytes from the SUI are mapping to. If the identifier is present in the selected row, then this yields an indication of a possible pattern match and results in KES performing a look-up into the Confidence table in order to gain better confidence that there may be a pattern match.

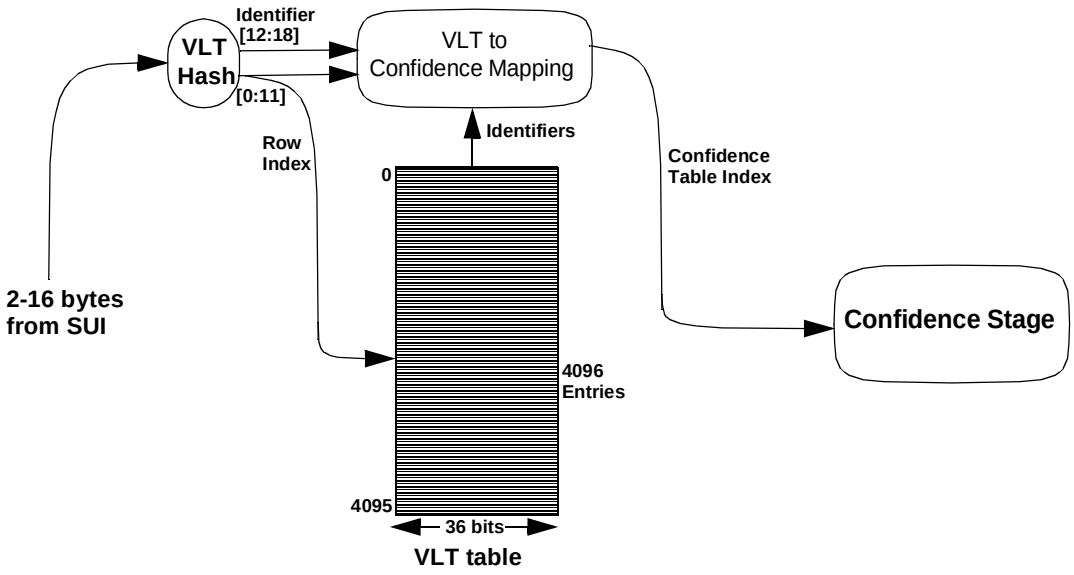


Figure 17-117. Variable Length Trigger (VLT) Table Hash

Each row of the VLT table has one of six formatting modes as described by Table 17-118. All VLT table rows initially start out empty, with all bits 0.

Variable Length Trigger Table Row Structure																																		
mode	0	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	32	33	34	35	36
empty	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4x7	0	0	0	0	0	0	0	1	identifier 1[0:6]							identifier 2[0:6]							identifier 3[0:6]							identifier 4[0:6]					
5x6	gxc				0	1	identifier 1[0:5]						identifier 2[0:5]						identifier 3[0:5]						identifier 4[0:5]						identifier 5[0:5]					
6x5	gxc				1	0	identifier 1[0:4]						identifier 2[0:4]						identifier 3[0:4]						identifier 4[0:4]						identifier 5[0:4]					
7x4	gxc				0	0	1	0	ident. 1[0:3]				ident. 2[0:3]						ident. 3[0:3]						ident. 4[0:3]						ident. 5[0:3]					
8x3	gxc				0	0	1	1	x	x	x	x	id.1[0:2]			id.2[0:2]			id.3[0:2]			id.4[0:2]			id.5[0:2]			id.6[0:2]			id.7[0:2]			id.8[0:2]	

Figure 17-118. Variable Length Trigger Table Row Structure

As illustrated in Figure 17-117, the lower 7 bits of the VLT hash provides an identifier (id) from 0 to 127. This id must be stored in the VLT table at the row selected. The following list describes the semantics and rules to follow for filling up the rows:

- If the entry is all 0's, then the row is emptied.
- If the mode is 4x7, the entire 7 bits of the id are stored in the row.
- If the mode is 5x6, the upper 6 bits of the id are stored in the row.
- Similarly to the above, mode 6x5 = upper 5 bits, mode 7x4 = upper 4 bits, mode 8x3 = upper 3 bits
- ids are stored in numerical order in each row.
- If *i* ids are stored in a row of type *nxm*,and *i*<*n*, then the remaining *n-i* empty ids in that row must be set to the same id value as the id at position *i*.

- If a new id needs to be stored, but an identical id is already stored in the row, then no change is needed to this row. However, this does result in a *trigger collision*. If the mode is non-4x7, and the new id is only identical after losing its lower bits per the table mode, then again no change is required to the row. However, this results in a *compression collision*.
- If the mode is 4x7, a new id is to be added, and 4 non-identical ids already exist in this row, then the mode of this row must be switched to 5x6. The 5 ids are stored in the same row, but now in the 5x6 mode. Note that the switch may result in less that five ids being stored in the 5x6 mode row. For example, if the 5 ids expressed in their entire 7 bits are 4,22,23,50,81 the resulting ids in the 5x6 are 2,11,25,40. The id 22 and 23 both result in 11 when the lowest bit is removed as per the 5x6 row format.
- Attempting to add a 6th id into a 5x6 row already holding 5 non-identical ids causes the mode to be switched to 6x5 through the same process.
- Similarly, 6x5 switches to 7x4, 7x4 to 8x3.

The guest transform code (gxc) is used as an input to the "Variable Length Trigger to Confidence Mapping" function to calculate an index into the Confidence table. The usage of this field is described in more details in Section 17.4.1.2.1, "Variable Length Trigger to Confidence Mapping."

The Variable Length Trigger table rows are updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables such the Variable Length Trigger table. Table 17-152 details the format of the Write Table Entry Command.

17.4.1.1.2 2-Byte Trigger

The 2-byte Trigger (2BT) consists of a fingerprint hash (such as 2BT hash) performed over 2 contiguous bytes of the SUI stored in its "Equivalent Byte" form. The output of the 2BT hash function is used as the index into the 2BT table. Similarly to the Variable Length Trigger (VLT) table, the 2-byte Trigger table is a compressed hash table of a "uncompressed trigger table" structured as a 2-dimension array of 512 rows and 128 columns where each element is a single bit which indicates whether or not there is a pattern for this hash value (see Figure 17-116). Each row of the 2BT table is constructed by collapsing columns of the uncompressed 2BT table into one of the same six formatting modes as defined for the VLT table. See Section 17.4.1.1.1, "Variable Length Trigger," for a description of the row formatting modes.

The 2-byte Trigger hash outputs 16 bits. As illustrated in Figure 17-119, the upper 9 output bits of the 2BT hash selects one of 512 rows (or entries) from the 2BT table whereby the lower 7 output bits of the 2BT hash represents the identifier of the cluster to which the 2 bytes from the SUI are mapping to. If the identifier is present in the selected row, then this yields an indication of a possible pattern match and thus results in KES performing a look-up into the Confidence table in order to gain better confidence that there may be a pattern match.

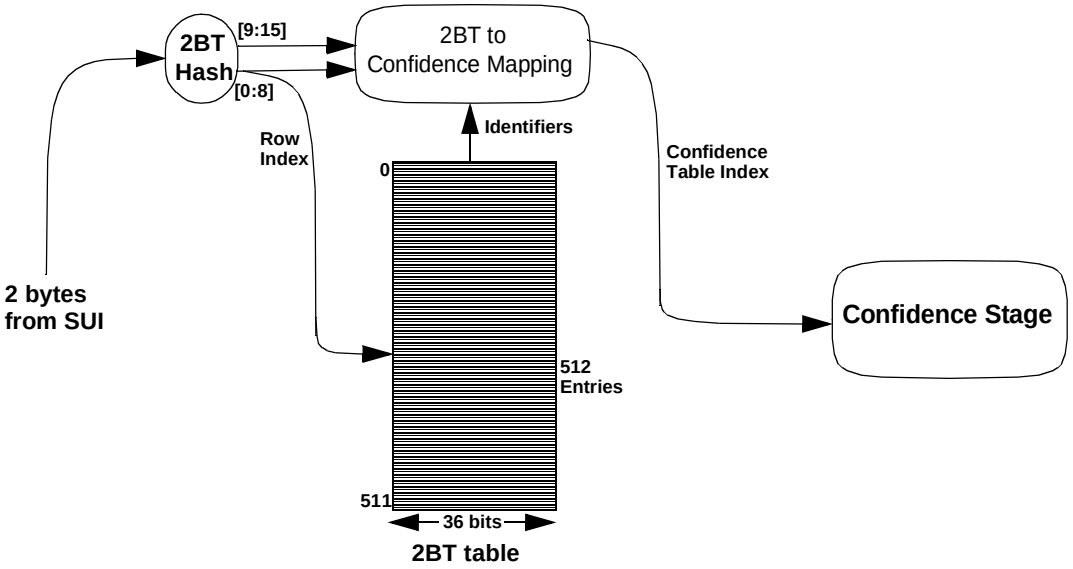


Figure 17-119. 2-Byte Trigger (2BT) Table Hash

Each row of the 2BT table has one of six formatting modes as described by Table 17-120. All 2BT table rows initially start out empty, with all bits 0.

Variable Length Trigger Table Row Structure																																		
mode	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	32	33	34	35
empty	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4x7	0	0	0	0	0	0	0	1	identifier 1[0:6]							identifier 2[0:6]							identifier 3[0:6]							identifier 4[0:6]					
5x6	gxc				0	1	identifier 1[0:5]					identifier 2[0:5]					identifier 3[0:5]						identifier 4[0:5]						identifier 5[0:5]					
6x5	gxc				1	0	identifier 1[0:4]					identifier 2[0:4]					identifier 3[0:4]						identifier 4[0:4]						identifier 5[0:4]					
7x4	gxc				0	0	1	0	ident. 1[0:3]				ident. 2[0:3]				ident. 3[0:3]				ident. 4[0:3]				ident. 5[0:3]				ident. 6[0:3]				ident. 7[0:3]	
8x3	gxc				0	0	1	1	x	x	x	x	id.1[0:2]			id.2[0:2]			id.3[0:2]			id.4[0:2]			id.5[0:2]			id.6[0:2]			id.7[0:2]			id.8[0:2]

Figure 17-120. 2-Byte Trigger Table Row Structure

The semantics and rules to follow for filling up the rows are the same as for the Variable Length Trigger method (see Section 17.4.1.1.1, “Variable Length Trigger).

The 2-byte Trigger table rows are updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables such the 2-byte Trigger table. Table 17-152 details the format of the Write Table Entry Command.

17.4.1.1.3 1-Byte Trigger

Each 1 byte of the SUI stored in its “Equivalent Byte” form provides an index into the internal 1-byte Trigger table (see Figure 17-121). If the bit entry is set to one, then this yields an indication of a possible

pattern match and thus results in KES performing a look-up into the Confidence table in order to gain better confidence that there may be a pattern match.

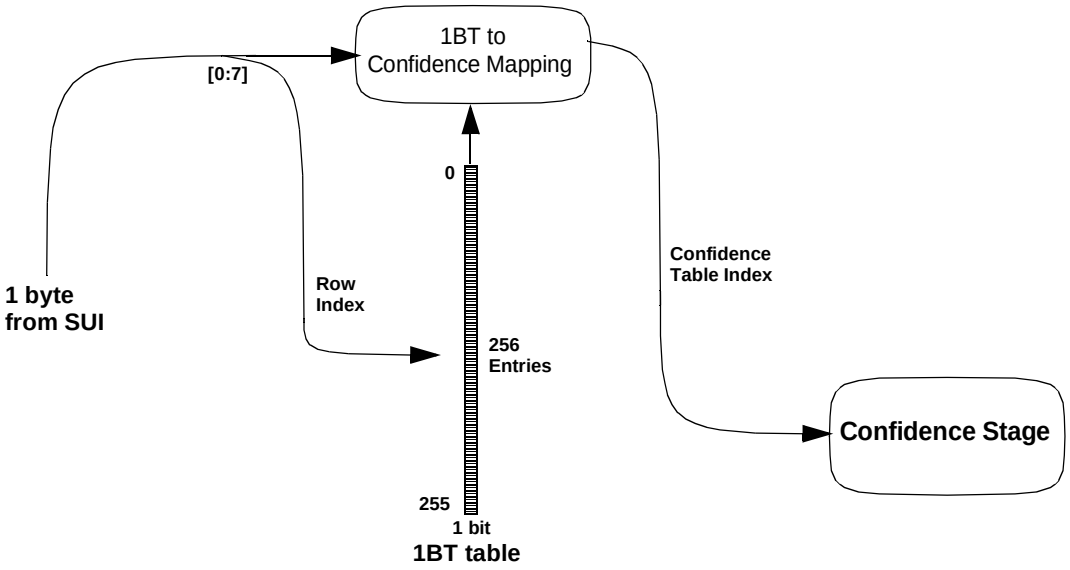


Figure 17-121. 1-Byte Trigger (1BT)

The 1-byte Trigger table is updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables. Table 17-152 details the format of the Write Table Entry Command.

17.4.1.1.4 Special Triggers

The Trigger stage can detect special positions in the SUI (for example, position immediately after a CR or LF character). These position matches are referred to as Special Triggers. Any occurrence of these special positions in the SUI causes a look-up into the Special Trigger table to determine whether or not access to the Confidence table (Confidence stage) is to be performed. As illustrated in Figure 17-122, the Special Trigger table consists of 256 entries of 1 bit wide. A Special Trigger entry set to one causes KES to perform a look-up into the Confidence table if the Special Trigger condition is detected.

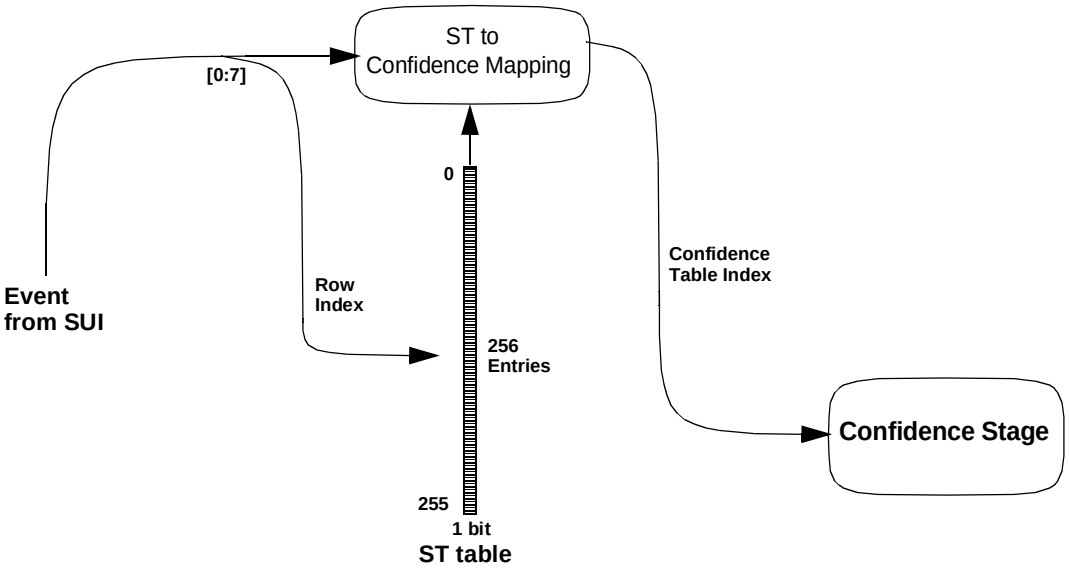


Figure 17-122. Special Trigger (ST)

Although there is space in the Special Trigger table for 256 Special Triggers, 8 Special Triggers have been defined. These 8 Special Triggers are described in Table 17-120.

Table 17-120. Special Triggers Description

Special Trigger Table Index	Special Trigger Description
0	Trigger on End of Stream
1	Trigger on End of Work Unit
2	Trigger on End of Stream or Carriage Return/Linefeed
3	Trigger on End of Work Unit or Carriage Return/Linefeed
4	Trigger on Start of Work Unit or Carriage Return/Linefeed
5	Trigger on Start of Stream or Carriage Return/Linefeed
6	Trigger on Start of Work Unit
7	Trigger on Start of Stream

The Special Trigger table is updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables. Table 17-152 details the format of the Write Table Entry Command.

17.4.1.2 Confidence Stage

Based on the indication of a possible match from the trigger stage, the Confidence stage performs a more rigorous examination of the SUI to determine whether or not if there is indeed a possible pattern match at

the position where the trigger event was detected. The Confidence stage performs a hash over the SUI bytes that are necessary to be the Key Element for a matching pattern and the result is compared to the pre-computed hash value of the Key Element stored in the Confidence table entry that is tied to the trigger event. If the compare operation produces a positive match, then the Key Element Scanner notifies the Data Examination Engine for further examination of the SUI.

The Confidence table is structured as a 2-dimension array of 4096 rows and 4 columns where each entry is 32 bits wide (see Figure 17-123).

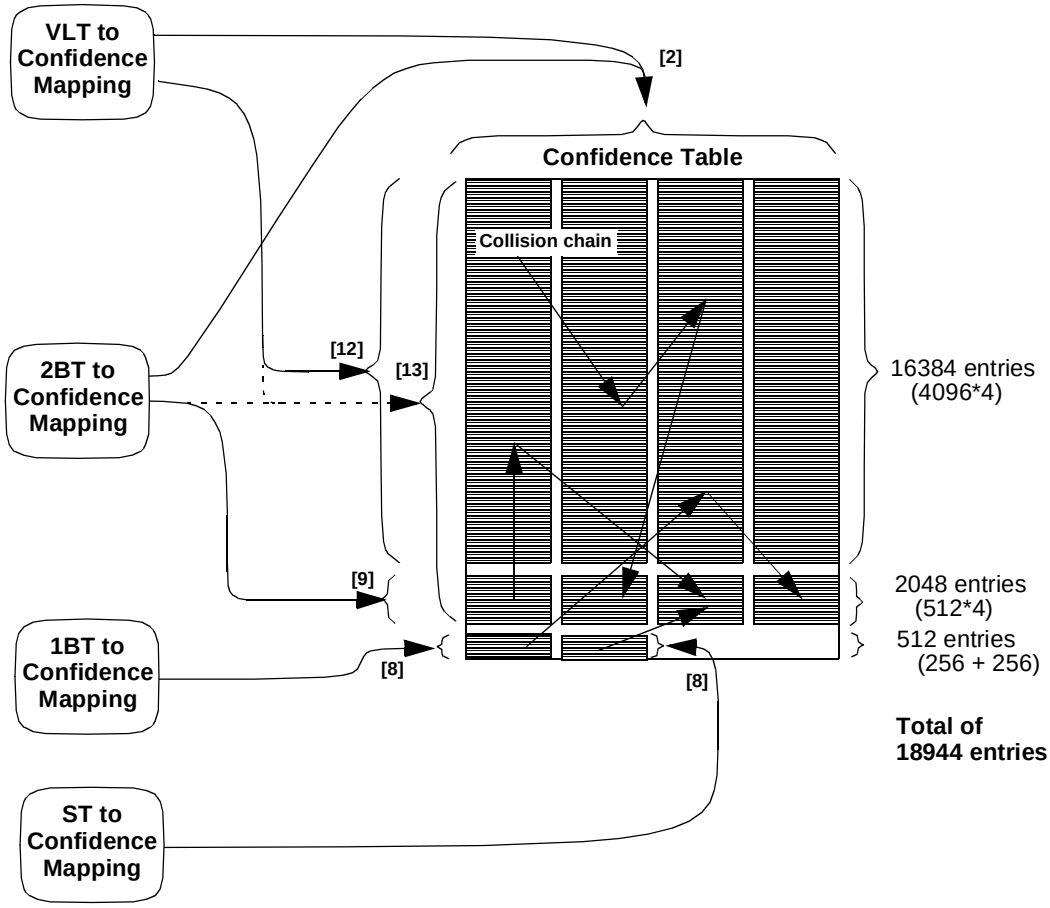


Figure 17-123. Confidence Table

Each Confidence table entry contains a description of a “Key Element” (called “confidence mask code”) which indicates positions relative to the fingerprint (or special trigger) that are part of the Key Element and whether the symbols are all of type “Original Byte” (indicated by having the T bit field set to 1) or “Equivalent Byte” (indicated by having the T bit field set to 0) and a “confidence hash value” of the Key Element. A detailed description of the confidence mask code is provided in Section 17.4.1.2.6, “Confidence Mask Codes.” A Confidence table entry may also contain a pointer (or address) to another Confidence table entry if the entry is part of a collision chain (singly-linked list). The structure of the Confidence table entries is depicted in Table 17-24. Note that if an entry contains an address to another Confidence table entry, then this next address is encoded in the “expected confidence hash result” field. This encoding is described in Section 17.4.1.2.5, “Confidence Collision Resolution.” The C bit field

determines whether or not there is a next address stored in the entry. When the C field is set to 0, it indicates that there is a next address stored in the entry.

Confidence Table Row 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
C	T	confidence mask code														expected confidence hash result [16:31]															
C - last entry in collision chain, T - include ‘transform bits’ in hash calculation																															

Figure 17-124. Confidence Table Row Structure

There is a one to one relationship between confidence entries and complete pattern descriptions. Complete pattern descriptions are stored in system memory (see Section 17.4.2, “Data Examination Engine). The Confidence table is sized to operate with a high load factor since the table is stored in internal memory. Internal memory is a relatively expensive resource and as result we must make effective use of it. The Confidence table size is fixed at 18,944 entries to support a maximum of 16,000 patterns.

When the Trigger stage detects a possible pattern match, it provides the Confidence stage with 64 bytes of the SUI in the Equivalent Byte format. As illustrated in Figure 17-125, the right most byte of the fingerprint (or trigger) is positioned in the middle of the 64 byte window (such as, position “-1”). In the case where the possible pattern match event was detected by a Special Trigger representing a location just to the left of the pattern (for example, Start of Work Unit), the first byte immediately after the Special Trigger location is at position “-1” within the 64 byte window (-32 to 31). Similarly, a Special Trigger representing a location just to the right of the pattern (for example, End of Work Unit), the last byte immediately before the Special Trigger location is at position “-1” within the 64 byte window (-32 to 31).

Associated with each 1-byte symbol (which is in the Equivalent Byte format) is one bit field called the “transform” bit. When set, this bit indicates that there is a difference between the Original and Equivalent Byte representation for that particular byte or symbol of the SUI. The T bit field in the Confidence table entry when set indicates that these “transform” bits are included in the confidence hash calculation. The T bit field must be set when a Confidence entry contains a description of a Key Element in its “Original Byte” format. If the Confidence entry contains a description of a Key Element in its “Equivalent Byte” format, then the T bit field must be set to 0 indicating that the “transform” bits are not to be included in the confidence hash computation since symbols that are inputted into confidence hash function are in the Equivalent Byte format. Also note that the Pattern Set is included in the hash calculation.

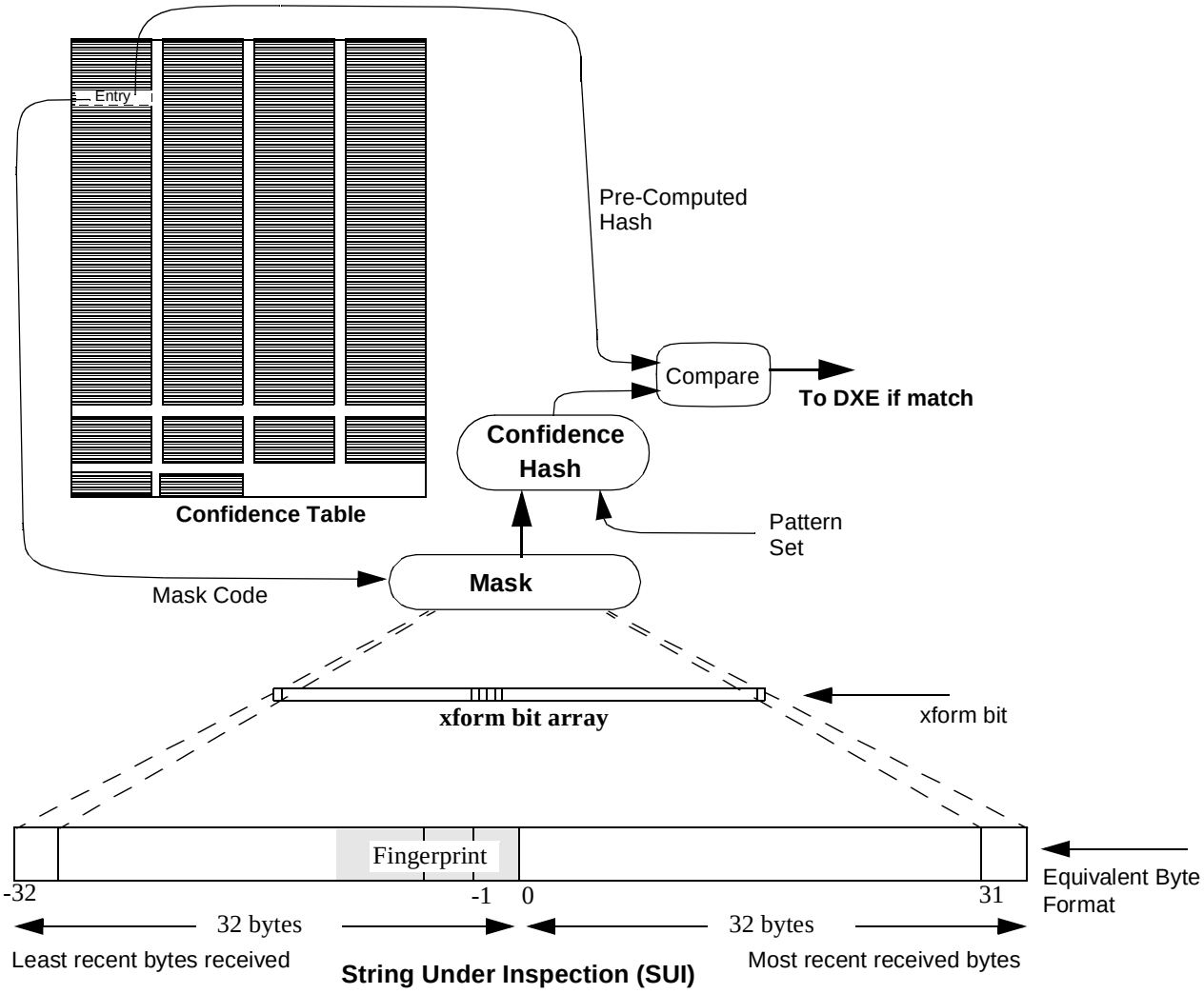


Figure 17-125. Confidence Hash

The Confidence table entries are updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables. Table 17-152 details the format of the Write Table Entry Command.

17.4.1.2.1 Variable Length Trigger to Confidence Mapping

The Variable Length Trigger (VLT) hash lower output bits provide an identifier that must be stored in the VLT table at the row selected by the VLT hash upper output bits. If an identifier (id) is found to match, then this yields an indication of a possible fingerprint match and an index to a confidence entry called Confidence Index Head (CIH) is generated through a mapping described below. The CIH index (row and column) is used to access a “confidence entry” or a chain of confidence entries (collision).

There are two mapping cases. The first case occurs when the id field number is <= 4. For a VLT row r, each id field, x, where x<=4, the CIH is set as follows:

- CIH row set to r (for example, same row as the VLT table row that caused the trigger), range is 0 to 4095.
- CIH column set to x-1, range is 0 to 3.

The second mapping case only applies to non-4x7 row format modes, and to id field numbers 5,6,7,8. Since the Confidence table row indexed by the VLT hash upper 12 output bits is full (occupied by id field 1,2,3,4), the id field numbers 5,6,7,8 cannot be stored directly in this row. The “guest transform code” (gxc) stored in this row (row indexed by the VLT hash upper 12 output bits) indicates which of the methods was used to store the id field numbers 5,6,7,8. If *gxc is non-zero*, then the ids were stored in another row using a hash distribution function. This hash distribution function uses as input the gxc code and the current row index to produce a uniformly distributed output row index over the Confidence table row range of 0 to 4607. The row produced by the hash distribution function must have sufficient empty entries to store directly the id fields. The id field numbers 5, 6,7, and 8 are stored respectively in column 3,2,1 and 0. If *gxc=0,* then this indicates that the id field numbers 5,6,7,8 were stored through collision chaining. That is, an empty entry (must be in the Confidence table row range of 0 to 4607) in the Confidence table was found to store the id and this entry was linked to a singly linked list of chained Confidence entries. The head entry of this linked list is located in column 3 of the current row for the id field number 5, column 2 of the current row for the id field number 6, and so on. This is referred as a *foldback collision*. Take note that these linked lists may contain other entries as a result of *compression and trigger collisions*.

17.4.1.2.2 2-Byte Trigger to Confidence Mapping

The 2-byte Trigger (2BT) hash lower output bits provide an identifier that must be stored in the 2BT table at the row selected by the 2BT hash upper output bits. If an identifier (id) is found to match, then this yields an indication of a possible fingerprint match and an index to a confidence entry called Confidence Index Head (CIH) is generated through a mapping described below. The CIH index (row and column) is used to access a “confidence entry” or a chain of confidence entries (collision).

There are two mapping cases. The first case occurs when the id field number is <= 4. For a 2BT row r, each id field, x, where x<=4, the CIH is set as follows:

- CIH row set to r+4096, range is 4096 to 4607.
- CIH column set to x-1, range is 0 to 3.

The second mapping case only applies to non-4x7 row format modes, and to id field numbers 5,6,7,8. Since the Confidence table row indexed by the VLT hash upper 12 output bits is full (occupied by id field 1,2,3,4), the id field numbers 5,6,7,8 cannot be stored directly in this row. The “guest transform code” (gxc) stored in this row (row indexed by the VLT hash upper 12 output bits) indicates which of the methods was used to store the id field numbers 5,6,7,8. If *gxc is non-zero*, then the ids were stored in another row using a hash distribution function. This hash distribution function uses as input the gxc code and the current row index to produce a uniformly distributed output row index over the Confidence table row range of 0 to 4607. The row produced by the hash distribution function must have sufficient empty entries to store directly the id fields. The id field numbers 5, 6,7, and 8 are stored respectively in column 3,2,1 and 0. If *gxc=0,* then this indicates that the id field numbers 5,6,7,8 were stored through collision chaining.

That is, an empty entry (must be in the Confidence table row range of 0 to 4607) in the Confidence table was found to store the id and this entry was linked to a singly linked list of chained Confidence entries. The head entry of this linked list is located in column 3 of the current row for the id field number 5, column 2 of the current row for the id field number 6, and so on. This is referred as a *foldback collision*. Take note that these linked lists may contain other entries as a result of *compression and trigger collisions*.

17.4.1.2.3 1-Byte Trigger to Confidence Mapping

Each 1 byte of the SUI stored in its “Equivalent Byte” form provides an index into the internal 1-byte Trigger table. Each entry bit provides an indication of a possible fingerprint match. If set, then an index to a confidence entry called Confidence Index Heads (CIH) is generated using the 1-byte Trigger table index bits (8), *i* as shown below:

- CIH row set to i+4608, range is 4608 to 4863.
- CIH column set 0.

Take note that these entries may represent the head entry of a collision chain if they were compression and/or trigger collisions on those entries. Confidence entries that fall in the row range of 4608 to 4863 cannot be used to store items that resulted in collision (empty entry must be located in the row range of 0 to 4607).

17.4.1.2.4 Special Trigger to Confidence Mapping

Each Special Trigger provides an index into the internal Special Trigger table where each entry bit provides an indication of a possible match. If set, then an index to a confidence entry called Confidence Index Heads (CIH) is generated using the Special Trigger table index bits (8), *i* as shown below:

- CIH row set to i+4608, range is 4608 to 4863.
- CIH column set 1.

17.4.1.2.5 Confidence Collision Resolution

If a *trigger collision*, *compression collision*, and/or *foldback collision* has occurred for an entry, then the Confidence stage runs a confidence check (to determine the possibility of a match) on each entity of the confidence collision chain. The collision chain linking algorithms is described below.

The address to another Confidence table entry is encoded in the “expected confidence hash result”. Three types of chain address information may be used which provide successively more addressing capability while reducing hash precision

The collision linking algorithm for 2-byte and Variable Length Triggers is described in Figure 17-126.

Confidence Entry "expected confidence hash result" Field Format When Used For Collision Chaining

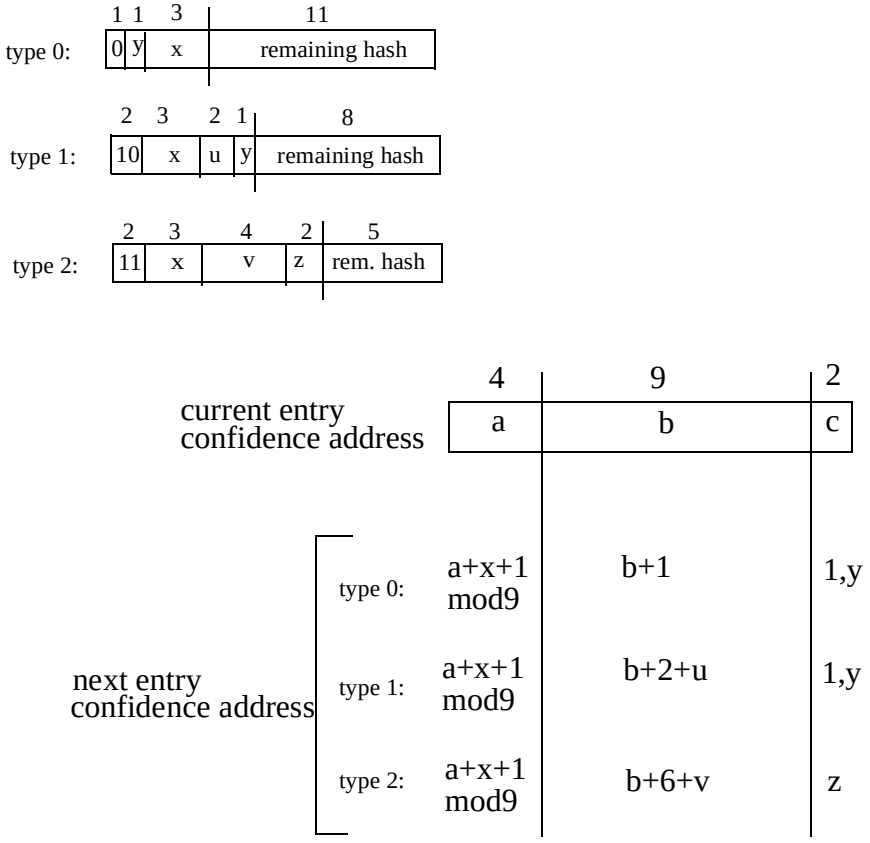


Figure 17-126. 2-Byte and Variable Length Trigger Collision Linking Algorithm

Collisions on 1-byte triggered patterns are also possible. Since the space for the 1-byte Trigger portion of the confidence table is very limited (see Figure 17-115), the 1-byte Trigger collision chain uses emptied entries from the space of the 2-byte and Variable-Length Trigger portions of the Confidence table. The collision linking algorithm for 1-byte Trigger is described in Figure 17-127. This algorithm only applies for the first link in the chain. All subsequent links in the chain follow the 2-byte/Variable Length Trigger collision resolution algorithm.

X[7:0] is the index in the 1-byte confidence space (0-255) or the index of the special trigger
X[8] is a 1 if the confidence entry is a special trigger, 0 if it is a 1B trigger

Y[7:0] is the upper 8 bits stolen from the 'expected confidence hash result' in the confidence entry
Z[14:0] is the new index into the 2-byte/Variable Length Trigger confidence space

```
Z[1:0] = Y[1:0]
Z[3:2] = X[1:0]
Z[5:4] = 0
temp[7:0] = Y[7:2] + X[8:2]
if (temp >= 144)
   Z[14:8] = temp - 144
   Z[7:6] = 3
else if (temp >= 72)
   Z[14:8] = temp - 72
   Z[7:6] = 2
else
   Z[14:8] = temp
   Z[7:6] = 0
```

Note: Bit 0 represents here the least significant bit

Figure 17-127. 1-Byte Trigger Collision Linking Algorithm

17.4.1.2.6 Confidence Mask Codes

Confidence mask codes are described in Figure 17-128. Confidence mask codes are 14-bit wide where the most significant 4 bits represents a mask type and the least significant 10 bits represent the actual mask value. For mask type 0, the mask code [4:8] value represents how many bytes to the left from the middle of the 64 byte window to mask (-1 to -31) whereas the mask code [9:13] value represents how many bytes to the right from the middle of the 64 byte window to mask (0 to 31). For mask type 1, the "-8 to -1" range is always included, mask code [4] bit represents the mask for byte -18, mask code [5] bit represents the masks for byte -17 and so on. Other mask types follows similar pattern.

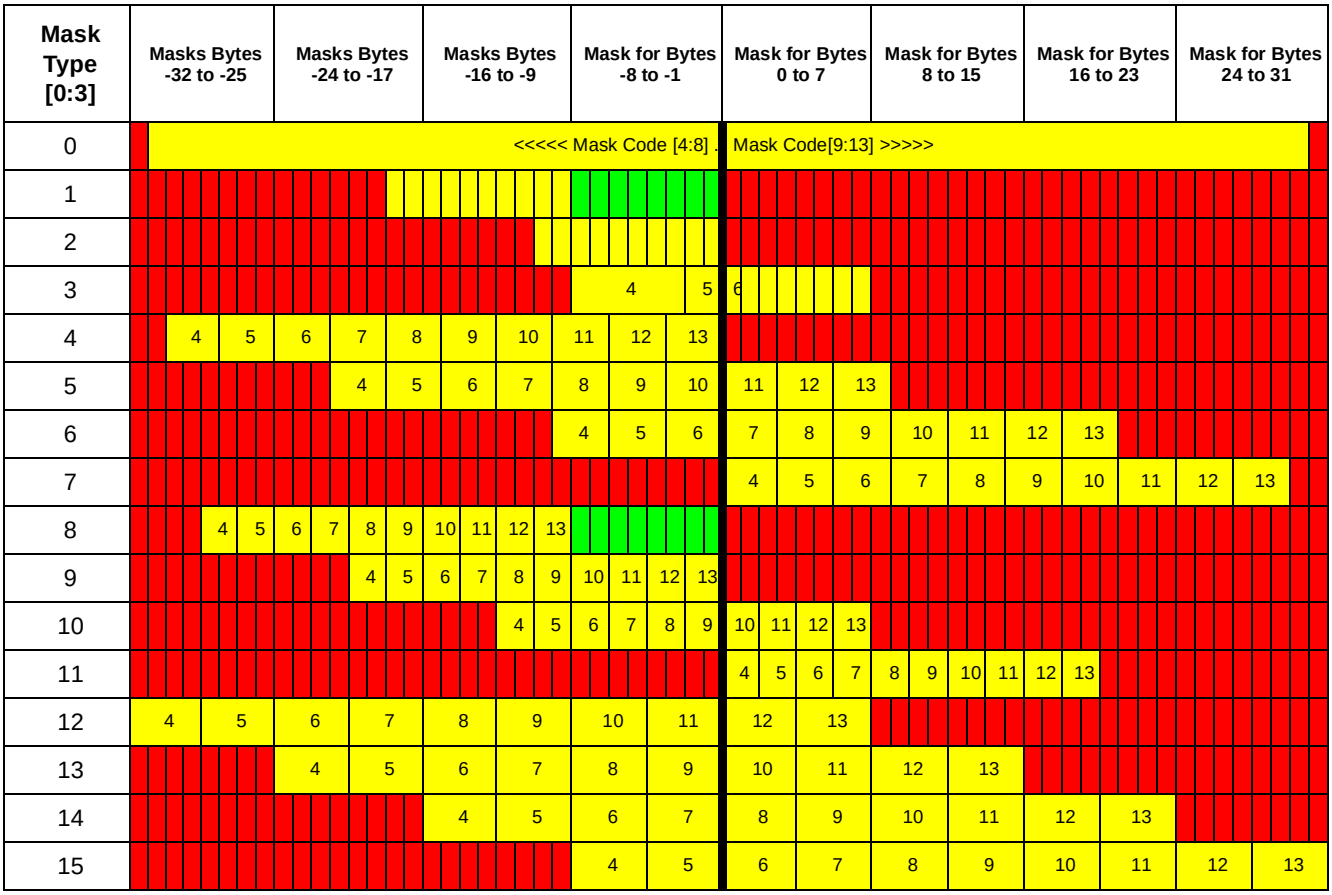


Figure 17-128. Confidence Mask Codes

The confidence mask code “mask type 15, mask value 0” is a special code that can be used to implement the concept of a block anchor. This special mask code dictates that 32 bytes to the left from the middle of the 64 byte window are masked and that a positive confidence match can only be reported if the 'bytes scanned' is a multiple of 32. For patterns with long consecutive repeating characters, the use of this special mask code can significantly reduce the number of repeating sequences reported by KES (reported at every 32 byte positions instead of every position).

17.4.2 Data Examination Engine

From the perspective of the Data Examination Engine (DXE), each pattern is an independent description (set of specialized instructions) of related symbol comparisons that are applied relative to a specific anchor position of a SUI and evaluated (using a Non-deterministic Finite Automaton) as defined by the description and results in match or non match indication. Symbols comparisons are 10-bit units (2-bit Compare Type and 8-bit Compare Value) that describe a compare action to a single original or normalized byte of the SUI.

The Compare Type can be one of the following:

- “Original Byte” compare
- “Equivalent Byte” compare

- “Pre-Defined Group” compare
- “User-Defined Group” compare

The Compare Value is defined as follows:

- Compare Type set to Original Byte or Equivalent Byte—Compare Value is the actual value to match.
- Compare Type set to Pre-Defined Group or User-Defined Group—Compare Value contains an 8-bit map indicating the set of groups to match. Can be used to support wildcarding efficiently (no pattern explosion).

The pattern descriptions are stored in system memory, and are capable of implementing a significant subset of the regex pattern definition syntax, as well as many constructs which cannot be expressed in regex. Each pattern description entry is up to 512 bytes long and can contain up to 240 unique symbol comparisons.

That is to say the DXE natively performs anchored pattern matches[1] of regex complexity[2]. This requires the DXE to have a certain level of algorithmic sophistication because although at least one symbol compare is in a fixed position (the anchor position) subsequent symbol compares may involve alternatives and/or repetitions which means that symbol comparisons positions may vary based on the outcome of previous symbol compares.

In theory the DXE could perform the requirement of unanchored pattern matching for 16,000 patterns by executing the instructions for each pattern in the database at each possible anchor position. However, even with the specialized DXE instructions that can perform multiple (8) symbol compares in a single cycle, using the DXE in a brute force manner does not meet the PM performance requirements. Thus, the DXE is preceded by the Key Element Scanner (KES) which pre-scans the SUI using internal hash tables to identify the possible positive matches.

17.4.2.1 Pattern Description

Patterns are described by set of specialized instructions referred to as “test lines”. Each pattern is described by a maximum of 32 test lines.

The first two test lines are special and are referred to as the “head test lines”. The head test lines contain important information relating to the overall pattern match such as how many test lines are valid and if a report is requested.

The rest of the test lines (referred as instruction test lines) each describe a particular instruction be performed. Each test line (including the head test lines) is 16 bytes worth of data. The following section provides a detailed description of each field within the test lines.

The operation of the DXE NFA can be logically represented by the diagram on Figure 17-129. In that diagram a box represents the execution of an instruction (or instruction test line) and the arrow represents a pointer to another instruction. The execution begins at the circle labelled “start”. From the start, there can be one or two choices to follow. If there are two choices, the top arrow represents first choice to follow and the second choice is saved on a stack in case the match does not succeed on the first choice path.

1. An Anchored Pattern Match should not be confused with a fixed position match which is a degenerate form of an Anchored Pattern Match.
2. The DXE can evaluate a pattern of regex complexity and structure and would permit a 1 to 1 relationship between regex Expressions and Patterns. However, the structure of some regex Expressions are incompatible with the requirements of the KES and therefore in some cases it is necessary to split regex Expressions into multiple patterns

Similarly, if a particular instruction passes (matches), there can also be two choices to follow. If there are two choices, the top arrow represents the first choice to follow and the second choice (bottom arrow) is saved on a stack in case the match does not succeed on the first choice path. The first choice (including the case where there is only one choice) is referred as the “subsequent” instruction. The second choice is referred as the “alternate” instruction. If a leaf box is reached (next instruction corresponds to the circle labelled “end”), a pattern match (entire pattern) has been found and the execution stops. If a particular instructions does not yield a match (match fails), it backtracks by executing the last saved instruction which is retrieved (popped) from the top of the stack (a last in, first out (LIFO)). If the stack becomes empty, then it declares that the entire pattern is not match and stops execution.

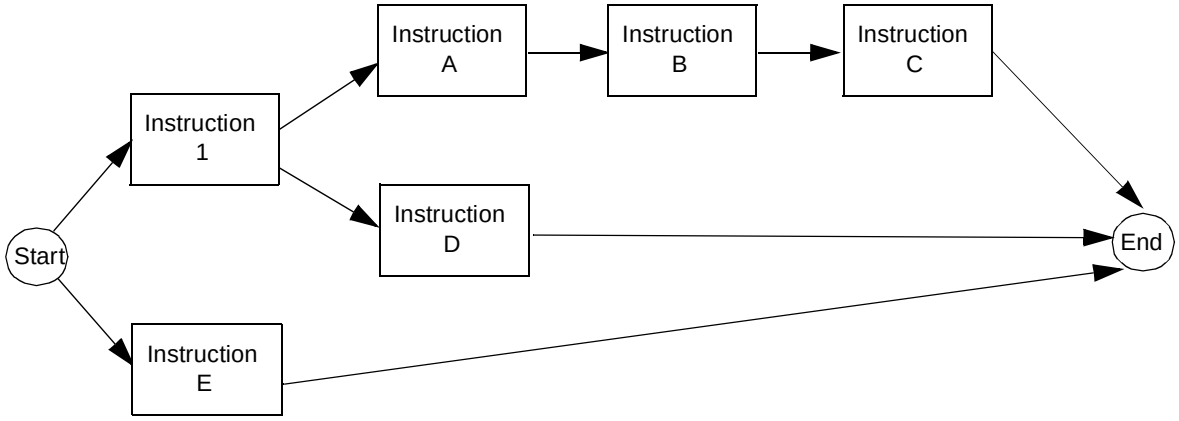


Figure 17-129. DXE NFA Operation

In the example shown in Figure 17-129, the DXE NFA would traverse the instructions as follows:

1. Instruction E is first saved on the stack followed by the execution of instruction 1.
2. If instruction 1 matches, then instruction D is saved on the stack followed by the execution of instruction A.
3. If instruction A matches, then instruction B is executed.
4. If instruction B matches, then instruction C is executed
5. If instruction C matches then an entire pattern match has been found and execution stops since the next instruction points to the “end”.
6. If instruction A, B or C match fails, then it backtracks by executing the last saved instruction which corresponds in this case to the execution of instruction D.
7. If instruction D matches then an entire pattern match has been found and execution stops since the next instruction points to the “end”.
8. If instruction D match fails then it backtracks by executing the last saved instruction which corresponds in this case to the execution of instruction E.
9. If instruction E matches then an entire pattern match has been found and execution stops since the next instruction points to the “end”.

10. If instruction E match fails then it backtracks by executing the last saved instruction which in this case is “none” since the stack is empty. Execution stops and the DXE NFA declares that the pattern is not a match.

Each instruction test line contains two instruction pointers: a subsequent instruction pointer and an alternate instruction pointer. Before an instruction test line begins execution the alternate instruction pointer stored in that instruction test line is always saved on a internal stack for the purpose of backtracking. The instruction test line execution then proceeds. If the instruction test line matches then the subsequent instruction pointer provides the next instruction to execute. Going back to Figure 17-129, the arrow from the start to instruction 1 corresponds to KES reporting a possible pattern match which in turn causes the DXE to execute the first instruction test line (such as instruction 1) for that pattern. The arrow from “start” to instruction E is stored in instruction 1 alternate instruction pointer. Similarly the arrow from instruction 1 to instruction D is stored in instruction A alternate instruction pointer.The arrow from instruction 1 to instruction A is stored in instruction 1 subsequent instruction pointer. A value of 0 for the alternate instruction pointer means that there is no alternate instruction present. For the above example, instruction B, C, D and E have their alternate instruction pointer set to 0. A value of 0 for the subsequent instruction pointer represents the end (such as leaf node/box) unless they are extended instructions. Extended instructions are discussed later in this sub-section. For the above example, instruction C, D and E have their subsequent instruction pointer set to 0.

When KES detects a possible pattern match, it provides to the DXE, the pointer to the first test line block of that pattern (see Section 17.4.2.1.6, “Test Line Block Format for a detailed description of the test line block format). It also provides 128 bytes of the SUI in four different formats (also referred as the SUI history memory) as shown in Figure 17-130 in which the right most byte of the fingerprint (or trigger) is positioned in the middle of the 128 byte window (such as position “-1”). In the case where the event was detected by a Special Trigger representing a location just to the left of the pattern (for example, Start of Work Unit), the first byte immediately after the Special Trigger location is at position “-1” within the 128 byte window (-64 to 63). Similarly, a Special Trigger representing a location just to the right of the pattern (for example, End of Work Unit), the last byte immediately before the Special Trigger location is at position “-1” within the 128 byte window (-64 to 63).

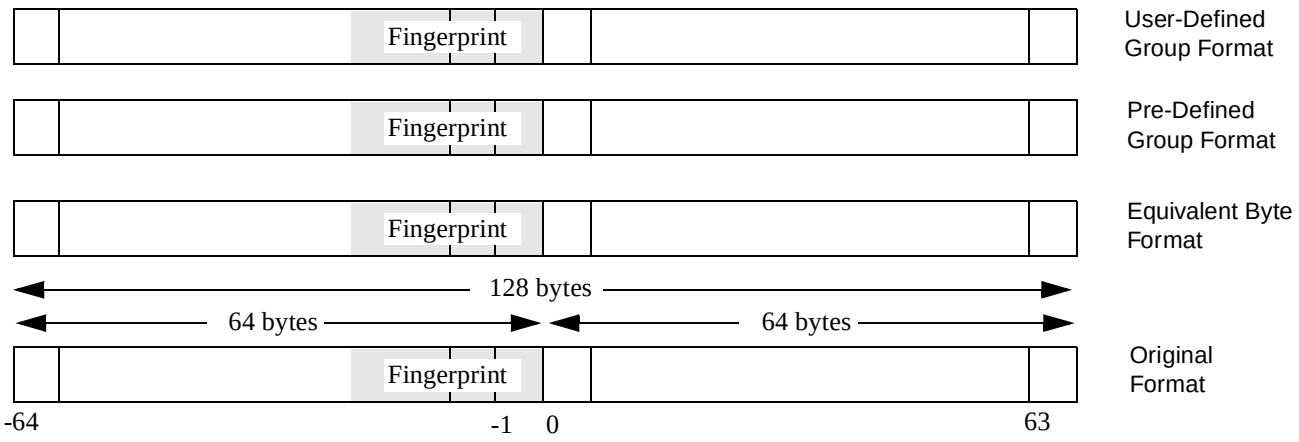


Figure 17-130. DXE 128 byte SUI Window Formats

The starting position relative to the above 128 window must be specified in the head test line. The allowable starting position is from “-64” to -”1”. The DXE instructions provides the ability to scan the 128 byte (SUI) window in both directions. This capability is provided to handle the case where the exact starting position of the entire pattern cannot be determined because there are a variable number of repeats in front of the fingerprint. However, the instruction execution flow must start by moving from “left to right” and when it reaches the end it can switch to moving from “right to left” if required. It can only switch once, it is not allowed to switch back to the “left to right” direction once moving from the “right to left” direction. DXE starts the instruction execution just before the starting position specified in the head test line. When a direction switch is made, the position is set back to the original starting position (for example, specified in the head test line) and DXE resumes execution just before that position but moving from “right to left”. This means that the symbol stored at the starting position is not examined when moving from “right to left”.

There are two types of test line instructions: Anchor instructions and Element Compare instructions.

17.4.2.1.1 Anchor Instruction

An Anchor instruction represents a position in the SUI to be matched. The following position tests are supported:

- Start of a stream
- End of a stream
- Start of a work unit
- End of a work unit
- Start of a stream or position immediately after a CR or LF character within a stream
- Start of a work unit or position immediately after a CR or LF character within a work unit
- End of a stream or position immediately before a CR or LF character within a stream
- End of a work unit or position immediately before a CR or LF character within a work unit

17.4.2.1.2 Element Compare Instruction

An Element Compare (EC) instruction compares the SUI with 1 to 8 bytes of information. A single EC instruction can process alternatives (elements) in parallel. This is referred as a parallel compare. When executing a parallel compare, the DXE picks the first match from left-to-right. Note that if the rest of the pattern fails after this first match, it cannot backtrack to that parallel compare instruction and attempt to find the next match using the same compare instruction. As a result, the alternatives in a parallel compare instruction should be mutual-exclusive (no overlapping).

The DXE provides the option of grouping multiple instructions so that they are processed as if they were a single instruction. When turned on, this option improves the overall efficiency of the DXE NFA. Note that the parallel compare behavior described above applies as whole to all of the alternatives contained in the group of instructions. As a result, all of these alternatives should be mutual-exclusive. This option is turned on by setting the PEC flag in the first instruction and in any adjacent instructions (referred early on as extended instructions) that belongs to the group. Take note that the subsequent instruction pointer, alternate instruction pointer and repetition (described below) fields are only relevant in the first instruction of the group. Furthermore, when other instruction test lines have a “group” instruction as the next

instruction to execute, these instruction test lines must have their pointers (for example, subsequent instruction pointer and/or alternate instruction pointer) set to the first instruction of that group. A “group” instruction execution starting at an instruction other than the first instruction of the group has unpredictable results.

17.4.2.1.3 Repetition

EC instructions provide support for repetitions. The repetition is specified as a minimum and a maximum number of allowable compare matches (“repeat minimum” and “repeat maximum” fields). Repetitions with “no upper bound” are supported (“no upper bound” meaning up to the boundary of the 128 byte window). The repetition can be specified as either “greedy” or “lazy”. A “greedy” repetition indicates that it should match as much as possible (up to the maximum number of allowed times) but without having the rest of the pattern failing. A “lazy” repetition indicates that it should match the smallest number of times (from the minimum number of times allowed.). When an EC instruction matches (including individual repeats), the position is advanced by the number of bytes (1 to 8) being compared by the instruction.

For the “greedy” operation, there is a “cautious” flag in the instruction indicating if there is overlap between the elements in the repeating instruction, and the subsequent instruction. When executing cautiously (because there is an overlap) DXE saves the shorter possibilities on the stack so that it can backtrack if a longest choice fails to produce a match for the rest of the pattern. If the “cautious” flag is not set, then DXE does not save on the stack the shorter possibilities.

For the “lazy” operation, once the minimum number of matches is met, DXE saves the current position on the stack so that it can backtrack and try another repetition if the rest of the pattern fails to produce a match. It then follows by executing the next instruction.

It is important to note that the desired “lazy” behavior may not be produced if an EC repetition instruction is scanning the 128 byte window from “right to left” since languages such as regex always expect that the regex engine scans the data from “left to right”. A shortest pattern match found from scanning from “right-to-left” may not yield the shortest match when scanning the same pattern occurrence from “left-to-right”. Take for example the regex expression /a(.+?)c/ applied to the string “aXXXcXXXc”. A “right-to-left” scan starting from the end of the string would yield the pattern occurrence “aXXXcXXXc” as the shortest match but a scan from “left-to-right” starting at the beginning of the string would yield the pattern “aXXXc” as the shortest match. From a regex language perspective, the “aXXXc” match would be the expected behavior and not the “aXXXcXXXc” match. Desired “lazy” behavior can be achieved by ensuring that the fingerprint of patterns that contain variable number of repetitions be located at the front of the pattern.

Note that multiple occurrences of the same pattern, including all overlapping occurrences are reported since the DXE finds matches from every position where the fingerprint is found. However from a given starting position, if there are variable repetitions (meaning that there could be multiple matches starting at the same point) either the longest or the shortest match is reported depending on the “Greedy” or “Lazy” nature of the repetitions. When looking for matches that cross work unit boundaries, for example, the use of residue bytes, it is possible that DXE matches a shorter greedy match in the first work unit and then another longer match in the second work unit.

When multiple instructions are grouped (PEC flag set), the repeat minimum field, repeat maximum field and greedy/lazy related fields (“Cautious “and “lazy” flags) are only relevant in the first instruction of the

group and the repetition behavior described above applies at the group level since instructions are processed as if they were a single instruction (for example, in every repetition, all instructions in the group get executed).

17.4.2.1.4 Capture

EC instructions provide the ability to capture data in internal registers for later use. These can be used in Stateful rules and/or included in pattern matching reports. In some text-based protocols, the captured data may be an ASCII-encoded numeric value, which, if stored as-is, would be difficult to interpret. It would therefore be useful to have the capability to convert this ASCII-encoded value into a numeric value before storing. The option to perform this conversion prior to storage is supported for ASCII-encoded binary, octal, decimal, and hexadecimal.

17.4.2.1.5 Inconclusive Match

If an Element Compare instruction reaches either the left edge of the 128-byte window or the right edge of the 128 window before it completes its compare operations (including repeat loops) and the symbols compared up to this boundary location are matching, then the result of the instruction execution is considered “inconclusive” because it can not continue to match the pattern further.[1] Take note that an instruction executing any types of repeat loops (for example, exact, no upper bound) that exceeds the boundary of the 128 byte window is considered as an “inconclusive” match (if the symbols compared up to the boundary are matching).

When an instruction results in an “inconclusive” match there are two possible modes of operation.

One mode treats the instruction as it found a “match” and records that an “inconclusive” condition was encountered. Since it considers the instruction as a “match”, the subsequent instruction pointer provides the next instruction to execute. If the next instruction operates in the same direction as the previous one, this instruction is not executed but is considered immediately a “match”. This instruction flow continues until a subsequent instruction pointing to the “end” is encountered or an instruction that switches direction (from right to left) is encountered. Left side proceeds in the same fashion where an “inconclusive” match is treated as a “match”. This mode is referred as to the “default” inconclusive mode.

The other mode treats the instruction as it found a “no match” and records that an “inconclusive” condition was encountered. Since it considers the instruction as a “no match”, the alternative instruction pointer provides the next instruction to execute. If a “conclusive” match is discovered in executing these alternatives, this condition (conclusive matching alternative paths) is also recorded and execution continues to the left (only if the instruction switches direction). Left side proceeds in the same fashion where an “inconclusive” match is treated as a “no match”. This mode is referred as to the “alternate” inconclusive mode.

The inconclusive mode is configured through the SRE Configuration register (see Section 17.3.4.7, “SRE Configuration Register (SREC)”).

The result of a pattern match operation is reported by DXE in the “$I” Pattern Match Event Meta Data register. See Table 17-121 for a description of the valid codings of this register when operating in the

1. If there is not enough bytes in the SUI beyond the 128 byte window to complete the compare operations, then the match performed by this instruction is considered “failed/no match”.

"default" inconclusive mode. See Table 17-122 for a description of the valid codings of this register when operating in the "alternate" inconclusive mode. Note that in the "alternate" inconclusive mode, the capture lengths ($Xn and $Yn) and capture data ($X and $Y) is zero for all inconclusive matches (for example, $I[5] set to 0 and ($I[6] and/or $I[7] set to 1)).

Table 17-121. DXE Pattern Match Reporting—"Default" Inconclusive Mode

Right Side Condition	Left Side Condition	$I[0:7]
Match within window	Match within window or no leftside checks	0: Pattern match
Match within window	Inconclusive	2: Left side inconclusive
Match within window	No match within window	No report
Inconclusive	Match within window or no leftside checks	1: Right side inconclusive
Inconclusive	Inconclusive	3: Left side and right side inconclusive
Inconclusive	No match within window	No report

Table 17-122. DXE Pattern Match Reporting—"Alternate" Inconclusive Mode

Right Side Condition	Left Side Condition	$I[0:7]			
		I[0:4]	$I[5] Pattern Match	$I[6] Left Inconclusive	$I[7] Right Inconclusive
Match within window	Match within window or no leftside checks	0	1	0	0
Match within window	Possible match beyond window and match found within window in alternate paths	0	1	1	0
Match within window	Possible match beyond window and no match found within window in alternate paths	0	0	1	0
Match within window	No match within window	no report	no report	no report	no report
Possible match beyond window and match found within window in alternate paths	Match within window or no leftside checks	0	1	0	1
Possible match beyond window and match found within window in alternate paths	Possible match beyond window and match found within window in alternate paths	0	1	1	1
Possible match beyond window and match found within window in alternate paths	Possible match beyond window and no match found within window in alternate paths	0	0	1	1
Possible match beyond window and match found within window in alternate paths	No match within window	no report	no report	no report	no report
Possible match beyond window and no match found within window in alternate paths	Match within window or no leftside checks	0	0	0	1

Table 17-122. DXE Pattern Match Reporting—“Alternate” Inconclusive Mode (continued)

Right Side Condition	Left Side Condition	$I[0:7]			
		I[0:4]	$I[5] Pattern Match	$I[6] Left Inconclusive	$I[7] Right Inconclusive
Possible match beyond window and no match found within window in alternate paths	Possible match beyond window and match found within window in alternate paths	0	0	1	1
Possible match beyond window and no match found within window in alternate paths	Possible match beyond window and no match found within window in alternate paths	0	0	1	1
Possible match beyond window and no match found within window in alternate paths	No match within window	no report	no report	no report	no report

17.4.2.1.6 Test Line Block Format

Test lines are stored in fixed sized 128-byte blocks. First block, which is always present, contains the first two “head test” lines and 1 to 6 “instruction” test lines. If additional instructions are required, then up to 3 additional 128-byte blocks (called extension blocks) each containing additional instruction test lines can be linked from the first block. This allows for patterns of up to 30 instructions (6+8+8+8) to be defined. Each block must be packed with instruction test lines except if it is the last block.

Structure of the first 128-byte block is shown in Table 17-131.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[1:7] First Test Pos.	[0] SimpleMatch Report [1:2] ------ [3:7] Num. Instr. Lines	------	[0:7] Pat. Set	---	---	[0:15] - Subset Mask		[14:31] Reaction Pointer				[0:31] Pattern Tag			
-------				[14:31] DXE Extension Pointer 1 (Inst 7-14)				[14:31] DXE Extension Pointer 2 (Inst 15-22)				[14:31] DXE Extension Pointer 3 (Inst 23-30)			
DXE Instruction 1															
DXE Instruction 2															
DXE Instruction 3															
DXE Instruction 4															
DXE Instruction 5															
DXE Instruction 6															

Figure 17-131. Test Line First Block (128-Byte) Format

Table 17-123. Test Line First Block Field Description

Offset	Bits	Field Name	Description
Offset + 0	[1:7]	First Test Position	Two's-complement indicating where within the window the first test occurs. For a right-side element test it specifies left-most position to be tested, For a left-side test it specifies a position one byte to the right of the right most position to be tested. The 128 positions of the examination window are reference as -64 (left extreme) to +63 (right extreme). A 2-byte trigger (or fingerprint) appears in positions -2,-1. An 8-byte trigger appears in positions -8 to -1. The allowable range for the first test position is -64 to -1.
Offset + 1	[0]	Simple Match Report	The pattern results in the generation of a Simple Match report
Offset + 1	[3:7]	Number Instruction Lines	Valid values 1-30. Number of instruction lines in the DXE routine (excludes the head test lines) in this and all attached DXE extension blocks
Offset + 3	[0:7]	Pattern Set	This pattern belongs to this set
Offset + 6	[0:15]	Pattern Subset Mask	This pattern belongs to a number of subsets (1's in mask)
Offset + 8	[14:31]	Reaction Pointer	Pointer to the start block of the Stateful Rule reactions to be executed if this pattern matches. The pointer is an index of a 128-byte-block relative to the start of the"DXE pattern descriptor memory space." Valid pointer values for this field are 0 (no reaction pointer) and 0x0_4908 to 0x3_fffc.
Offset + 12	[0:31]	Pattern Tag	User context. 32 bit value passed back to user in a Simple Match report, or usable in Stateful Rule reactions

Table 17-123. Test Line First Block Field Description (continued)

Offset	Bits	Field Name	Description
Offset + 20	[14:31]	DXE Extension Pointer 1	Pointer to an additional blocks that contains additional instructions (instruction 7-14) for this same pattern. That block must be packed with DXE instructions (8) except if it is the last block. The pointer is an index of a 128-byte-block relative to the start of the “DXE pattern descriptor memory space.” Valid pointer values for this field are 0x0_4908 to 0x3_fffc.
Offset + 24	[14:31]	DXE Extension Pointer 2	Pointer to an additional blocks that contains additional instructions (instruction 15-22) for this same pattern. That block must be packed with DXE instructions (8) except if it is the last block. The pointer is an index of a 128-byte-block relative to the start of the “DXE pattern descriptor memory space.” Valid pointer values for this field are 0x0_4908 to 0x3_fffc.
Offset + 28	[14:31]	DXE Extension Pointer 3	Pointer to an additional blocks that contains additional instructions (instruction 23-30) for this same pattern. That block must be packed with DXE instructions (8) except if it is the last block. The pointer is an index of a 128-byte-block relative to the start of the “DXE pattern descriptor memory space.” Valid pointer values for this field are 0x0_4908 to 0x3_fffc.

Structure of the 128-byte extension blocks are shown in Table 17-124.

Table 17-124. Test Line Extension Block (128-byte) Format

Byte 0-3	Byte 4-7	Byte 8-11	Byte 12-15
DXE Instruction 7or 15 or 23			
DXE Instruction 8 or 16 or 24			
DXE Instruction 9 or 17 or 25			
DXE Instruction 10 or 18 or 26			
DXE Instruction 11 or 19 or 27			
DXE Instruction 12 or 20 or 28			
DXE Instruction 13 or 21 or 29			
DXE Instruction 14 or 22 or 30			

Format of instruction test line is shown in Table .

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[0] Anchor [1] PEC or Anchor Start/End [2] Cautious or Anchor WU/Stream [3] Lazy or Anchor CRLF [4:7] Capture Code	[0] Side [1:2] ---- [3:7] Subsequent Link	[0:2] ---- [3:7] Alt. Link	[0] ---- [1:7] Repeat Min.	[0] ---- [1:7] Repeat Max.	[0:7] EC	[0:15] Type Code		[0:63] Compare Values							

Figure 17-132. Instruction Test Line Format

Table 17-125. Instruction Test Line Field Description

Offset	Bits	Field Name	Description
Offset + 0	[0]	Anchor	Specifies that an Anchor test is to be executed at this position. The type of test does not perform any character tests, it merely does a positional test on the current match boundaries. This test should only be performed as the LAST step of a right test or left test. When this bit is set the Anchor Start/End, Anchor WU/Stream, and Anchor CRLF bits are valid. The Subsequent and Alternate Links are the only other valid fields when this bit is set.
Offset + 0	[1]	Anchor Start/End	1 start anchor test (^) 0 end anchor test ($) This bit is valid when the Anchor bit is set
Offset + 0	[1]	PEC	Parallel Element Compare flag. A parallel element compare is used for alternations and extended tests (EC_8_E). For example (a|bc|def). DXE can look at the alternatives (elements) in parallel. If this flag is set, the test line following the current one is considered part of the current parallel compare. This optimizes hardware cycles when testing multiple alternatives. Example: (a|bc|def|ghij|klmnop). This alternation uses 5 elements and all 5 do not fit into one test line. The compiler would do something like: EC_6_2 {klmnop,bc} followed by EC_4_3_1 {ghij,def,a} and set the PEC flag on the first test line. This tells hardware that the first and second test lines are one larger parallel compare.
Offset + 0	[2]	Anchor WU/Stream	1 anchor test is executed relative to a work unit 0 anchor test is executed relative to a stream This bit is valid when the Anchor bit is set

Table 17-125. Instruction Test Line Field Description (continued)

Offset	Bits	Field Name	Description
Offset + 0	[2]	Cautious	Cautious Repeat Flag. This flag should only be set on repeating test lines (with min!=1 or max!=1) and only if there is some direct ambiguity or overlap between the elements in this repeating test line, and the following test line. Example: if the regex expression were /a{1,3}(a|b)/, there is overlap between the a{1,3} element and the (a|b) element. The Cautious repeat flag must be set in this case if a greedy match is required. DXE scans for the repeating element in a less-than optimal way as far as performance is concerned, but by scanning in this way guarantees the greedy behavior of the match.
Offset + 0	[3]	Lazy	Lazy Repeat Flag. This flag should only be set on test lines requiring a regex lazy repeating match (with min!=1 or max!=1) and only if there is some direct ambiguity or overlap between the elements in this repeating test line, and the following test line. If there is no ambiguity, then there is no difference between a lazy and a greedy match, therefore this flag should not be set. Example: if the regex expression were /a{1,3}(a|b)/, there is overlap between the a{1,3} element and the (a|b) element. The Lazy flag should be set in this case if a lazy match is desired. The Cautious and the Lazy flag cannot be set together. DXE scans for the repeating element in a less-than optimal way as far as performance is concerned, but by scanning in this way guarantees the lazy behavior of the match.
Offset + 0	[3]	Anchor CRLF	1 The anchor test also looks for CR or LF within a stream or a work unit in addition to start/end stream or start/end work unit. This bit is valid when the Anchor bit is set
Offset + 0	[4:7]	Capture Code	Capture code. If this test line is involved in a capture, then one of the following codes is used: 0 - None 1 - Left align X 2 - Right Align X 3 - Hash X 4 - Binary X 5 - Octal X 6 - Hex X 7 - Decimal X 8 - Continue Capture 9 - Left align Y 10 - Right align Y 11 - Hash Y 12 - Binary Y 13 - Octal Y 14 - Hex Y 15 - Decimal Y If a capture of $X or $Y is required, the appropriate code must be set on the “FIRST” test line to be captured. Subsequent Test lines of the capture series must contain the Continue Capture code. $X and $Y cannot be captured simultaneously. See Table 17-126 for a detailed description of the capture codes
Offset + 1	[0]	Side	Side (Left = 0, Right = 1). Since the trigger is likely to fall somewhere in the middle of the pattern, we may have to test both left and right of the trigger.
Offset + 1	[3:7]	Subsequent Link	Describes the next test line to go to if this test line passes.

Table 17-125. Instruction Test Line Field Description (continued)

Offset	Bits	Field Name	Description
Offset + 2	[3:7]	Alternate Link	Alternate Link. Describes an alternative choice to follow.
Offset + 3	[1:7]	Repeat Minimum	Minimum repeat value. 0-127. Note that the DXE can perform a maximum of 64 repeats from the right and a maximum of 63 repeats from the left. A value in the range of 65 to 127 is interpreted as a repeat with no upper bound.
Offset + 4	[1:7]	Repeat Maximum	Maximum repeat value. 1-127. Note that the DXE can perform a maximum of 64 repeats from the right and a maximum of 63 repeats from the left. A value in the range of 65 to 127 is interpreted as a repeat with no upper bound.
Offset + 5	[0:7]	Element Code	This code describes how to interpret the 8 possible compare values and which ones (starting from the left) are valid. The values may be grouped in various ways if we are doing a parallel compare (an alternation). For example, a simple compare of 6 characters would have an EC code of EC_6. An alternation such as (a|bc|def) would have a code of EC_1_2_3. A string of characters that continues into the next test line (such as more than 8) would be EC_8_EXTEND. See Table 17-127 for a complete list of codes.
Offset + 6	[0:15]	Type Code	Type codes (2 bits for each value). The compare code tells the DXE how to treat each value. There are 4 choices for each value: 0 Pre-Defined Group 1 User-Defined Group 2 Original Byte 3 Equivalent Byte Pre-Defined group values: 0 ASCII 0-31 (excluding group 1) and 127 (DEL) 1 \t \n \f \r SPACE 2 Punctuation symbols (such as !"#$%, etc.) 3 UNDERSCORE 4 Digits 5 Upper case alpha 6 Lower case alpha 7 Extended ASCII (128-255)
Offset + 8	[0:63]	Compare Values	Compare values (8 bits each). The values the DXE looks for based on the Element Code and the Compare codes.

Table 17-126. Capture Codes Description

Capture Code	Description
0	None
1	Left align X - The capture of $X is to be treated as a literal left-justified, meaning that the first 64 bits of the data are to be stored in the capture register starting at the most significant byte location (with the first 8 bits of data stored in the most significant byte, the next 8 bits of data stored in the next most significant byte, and so on). If the data to be captured is less than 64 bits, the register least significant bytes are padded with 0.
2	Right Align X - The capture of $X is to be treated as a literal right-justified, meaning that the last 64 bits of the data are to be stored in the capture register starting at the least significant byte location (with the last 8 bits of data stored in the least significant byte, the next 8 bits of data to be stored in the next least significant byte, and so on). If the data to be captured is less than 64 bits, the register most significant bytes are padded with 0.

Table 17-126. Capture Codes Description (continued)

Capture Code	Description
3	Hash X - The data is hashed to 64 bits prior to storing into the capture $X register. The length of the data over which the hash is computed can be up to 127 bytes.
4	Binary X - The data to be captured is expected to be an ASCII-encoded number and the data should be interpreted as ASCII-encoded binary and be converted to its numeric equivalent prior to storing into the capture $X register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
5	Octal X - The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded octal, and be converted to its numeric equivalent prior to storing into the capture $X register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
6	Hex X - The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded hexadecimal, and be converted to its numeric equivalent prior to storing into the capture $X register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
7	Decimal X - The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded decimal, and be converted to its numeric equivalent prior to storing into the capture $X register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
8	Continue Capture - The appropriate code must be set on the “FIRST” test line to be captured. Subsequent Test lines of the capture series must contain the Continue Capture code. Also take note that test lines of the capture series must have their “Side” field set to the same value. That is, you cannot switch direction within a continue capture series.
9	Left align Y - The capture of $Y is to be treated as a literal left-justified, meaning that the first 64 bits of the data are to be stored in the capture register starting at the most significant byte location (with the first 8 bits of data stored in the most significant byte, the next 8 bits of data stored in the next most significant byte, and so on). If the data to be captured is less than 64 bits, the register least significant bytes are padded with 0.
10	Right Align Y- The capture of $Y is to be treated as a literal right-justified, meaning that the last 64 bits of the data are to be stored in the capture register starting at the least significant byte location (with the last 8 bits of data stored in the least significant byte, the next 8 bits of data to be stored in the next least significant byte, and so on). If the data to be captured is less than 64 bits, the register most significant bytes are padded with 0.
11	Hash Y- The data is hashed to 64 bits prior to storing into the capture $X register. The length of the data over which the hash is computed can be up to 127 bytes.
12	Binary Y- The data to be captured is expected to be an ASCII-encoded number and the data should be interpreted as ASCII-encoded binary and be converted to its numeric equivalent prior to storing into the capture $Y register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.

Table 17-126. Capture Codes Description (continued)

Capture Code	Description
13	Octal Y- The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded octal, and be converted to its numeric equivalent prior to storing into the capture $Y register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
14	Hex Y- The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded hexadecimal, and be converted to its numeric equivalent prior to storing into the capture $Y register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.
15	Decimal Y- The data to be captured is expected to be an ASCII-encoded number and that the data should be interpreted as ASCII-encoded decimal, and be converted to its numeric equivalent prior to storing into the capture $Y register. If the resulting number requires fewer than 64 bits to store, the register most significant bytes are padded with 0. If the resulting number requires more than 64 bits to store, the number is truncated down to 64 bits, with the most significant bytes dropped, prior to storage. If the data is encoded incorrectly, the value captured is unpredictable.

Table 17-127. Element Codes

Element	Valu	Element	Valu	Element	Valu	Element	Valu
EC_8_EXTE	0	EC_2_6	64	EC_1_7	128	EC_1_1_5_1	192
EC_8	1	EC_2_5_1	65	EC_1_6_1	129	EC_1_1_5	193
EC_7_1	2	EC_2_5	66	EC_1_6	130	EC_1_1_4_2	194
EC_7	3	EC_2_4_2	67	EC_1_5_2	131	EC_1_1_4_1	195
EC_6_2	4	EC_2_4_1_1	68	EC_1_5_1_1	132	EC_1_1_4_1	196
EC_6_1_1	5	EC_2_4_1	69	EC_1_5_1	133	EC_1_1_4	197
EC_6_1	6	EC_2_4	70	EC_1_5	134	EC_1_1_3_3	198
EC_6	7	EC_2_3_3	71	EC_1_4_3	135	EC_1_1_3_2	199
EC_5_3	8	EC_2_3_2_1	72	EC_1_4_2_1	136	EC_1_1_3_2	200
EC_5_2_1	9	EC_2_3_2	73	EC_1_4_2	137	EC_1_1_3_1	201
EC_5_2	10	EC_2_3_1_2	74	EC_1_4_1_2	138	EC_1_1_3_1	202
EC_5_1_2	11	EC_2_3_1_1	75	EC_1_4_1_1	139	EC_1_1_3_1	203
EC_5_1_1_1	12	EC_2_3_1_1	76	EC_1_4_1_1	140	EC_1_1_3_1	204
EC_5_1_1	13	EC_2_3_1	77	EC_1_4_1	141	EC_1_1_3	205
EC_5_1	14	EC_2_3	78	EC_1_4	142	EC_1_1_2_4	206
EC_5	15	EC_2_2_4	79	EC_1_3_4	143	EC_1_1_2_3	207
EC_4_4	16	EC_2_2_3_1	80	EC_1_3_3_1	144	EC_1_1_2_3	208
EC_4_3_1	17	EC_2_2_3	81	EC_1_3_3	145	EC_1_1_2_2	209
EC_4_3	18	EC_2_2_2_2	82	EC_1_3_2_2	146	EC_1_1_2_2	210
EC_4_2_2	19	EC_2_2_2_1	83	EC_1_3_2_1	147	EC_1_1_2_2	211
EC_4_2_1_1	20	EC_2_2_2_1	84	EC_1_3_2_1	148	EC_1_1_2_2	212
EC_4_2_1	21	EC_2_2_2	85	EC_1_3_2	149	EC_1_1_2_1	213
EC_4_2	22	EC_2_2_1_3	86	EC_1_3_1_3	150	EC_1_1_2_1	214
EC_4_1_3	23	EC_2_2_1_2	87	EC_1_3_1_2	151	EC_1_1_2_1	215
EC_4_1_2_1	24	EC_2_2_1_2	88	EC_1_3_1_2	152	EC_1_1_2_1	216
EC_4_1_2	25	EC_2_2_1_1	89	EC_1_3_1_1	153	EC_1_1_2_1	217
EC_4_1_1_2	26	EC_2_2_1_1	90	EC_1_3_1_1	154	EC_1_1_2_1	218
EC_4_1_1_1	27	EC_2_2_1_1	91	EC_1_3_1_1	155	EC_1_1_2_1	219
EC_4_1_1_1	28	EC_2_2_1_1	92	EC_1_3_1_1	156	EC_1_1_2_1	220
EC_4_1_1	29	EC_2_2_1	93	EC_1_3_1	157	EC_1_1_2	221
EC_4_1	30	EC_2_2	94	EC_1_3	158	EC_1_1_1_5	222
EC_4	31	EC_2_1_5	95	EC_1_2_5	159	EC_1_1_1_4	223
EC_3_5	32	EC_2_1_4_1	96	EC_1_2_4_1	160	EC_1_1_1_4	224
EC_3_4_1	33	EC_2_1_4	97	EC_1_2_4	161	EC_1_1_1_3	225
EC_3_4	34	EC_2_1_3_2	98	EC_1_2_3_2	162	EC_1_1_1_3	226

Table 17-127. Element Codes (continued)

Element	Valu	Element	Valu	Element	Valu	Element	Valu
EC_3_3_2	35	EC_2_1_3_1	99	EC_1_2_3_1	163	EC_1_1_1_3	227
EC_3_3_1_1	36	EC_2_1_3_1	100	EC_1_2_3_1	164	EC_1_1_1_3	228
EC_3_3_1	37	EC_2_1_3	101	EC_1_2_3	165	EC_1_1_1_2	229
EC_3_3	38	EC_2_1_2_3	102	EC_1_2_2_3	166	EC_1_1_1_2	230
EC_3_2_3	39	EC_2_1_2_2	103	EC_1_2_2_2	167	EC_1_1_1_2	231
EC_3_2_2_1	40	EC_2_1_2_2	104	EC_1_2_2_2	168	EC_1_1_1_2	232
EC_3_2_2	41	EC_2_1_2_1	105	EC_1_2_2_1	169	EC_1_1_1_2	233
EC_3_2_1_2	42	EC_2_1_2_1	106	EC_1_2_2_1	170	EC_1_1_1_2	234
EC_3_2_1_1	43	EC_2_1_2_1	107	EC_1_2_2_1	171	EC_1_1_1_2	235
EC_3_2_1_1	44	EC_2_1_2_1	108	EC_1_2_2_1	172	EC_1_1_1_2	236
EC_3_2_1	45	EC_2_1_2	109	EC_1_2_2	173	EC_1_1_1_2	237
EC_3_2	46	EC_2_1_1_4	110	EC_1_2_1_4	174	EC_1_1_1_1	238
EC_3_1_4	47	EC_2_1_1_3	111	EC_1_2_1_3	175	EC_1_1_1_1	239
EC_3_1_3_1	48	EC_2_1_1_3	112	EC_1_2_1_3	176	EC_1_1_1_1	240
EC_3_1_3	49	EC_2_1_1_2	113	EC_1_2_1_2	177	EC_1_1_1_1	241
EC_3_1_2_2	50	EC_2_1_1_2	114	EC_1_2_1_2	178	EC_1_1_1_1	242
EC_3_1_2_1	51	EC_2_1_1_2	115	EC_1_2_1_2	179	EC_1_1_1_1	243
EC_3_1_2_1	52	EC_2_1_1_2	116	EC_1_2_1_2	180	EC_1_1_1_1	244
EC_3_1_2	53	EC_2_1_1_1	117	EC_1_2_1_1	181	EC_1_1_1_1	245
EC_3_1_1_3	54	EC_2_1_1_1	118	EC_1_2_1_1	182	EC_1_1_1_1	246
EC_3_1_1_2	55	EC_2_1_1_1	119	EC_1_2_1_1	183	EC_1_1_1_1	247
EC_3_1_1_2	56	EC_2_1_1_1	120	EC_1_2_1_1	184	EC_1_1_1_1	248
EC_3_1_1_1	57	EC_2_1_1_1	121	EC_1_2_1_1	185	EC_1_1_1_1	249
EC_3_1_1_1	58	EC_2_1_1_1	122	EC_1_2_1_1	186	EC_1_1_1_1	250
EC_3_1_1_1	59	EC_2_1_1_1	123	EC_1_2_1_1	187	EC_1_1_1_1	251
EC_3_1_1_1	60	EC_2_1_1_1	124	EC_1_2_1_1	188	EC_1_1_1_1	252
EC_3_1_1	61	EC_2_1_1	125	EC_1_2_1	189	EC_1_1_1	253
EC_3_1	62	EC_2_1	126	EC_1_2	190	EC_1_1	254
EC_3	63	EC_2	127	EC_1_1_6	191	EC_1	255

17.4.2.1.7 Pattern Description Memory Space

The pattern descriptions are stored in the Pattern Description and Stateful Rule table which is located in system memory. The registers defined in Section 17.3.3.5, “Pattern Description and Stateful Rule Base Address High and Low Registers (PDSRBAH and PDSRBAL),” are used to configure the location of the Pattern Description and Stateful Rule table in system memory. Each pattern occupies at a minimum one fixed sized 128-byte block. If additional instructions are required to describe the pattern, then up to 3 additional fixed sized 128-byte blocks (called extension blocks) can be linked from the first block. The system memory space used to store the pattern description is shown on Figure 17-133. The DXE can be configured to verify that pointers to blocks don’t exceed the maximum allowed block index. The maximum allowed block index is configured through the DXE Error Configuration register (see Section 17.3.3.7, “DXE Error Configuration (DEC)”).

These blocks are updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables such the Pattern Description and Stateful Rule table. Table 17-152 details the format of the Write Table Entry Command.

Although the Pattern Description and Stateful Rule table has space for 18668 patterns, the maximum number of patterns supported is 16,000. This is a result of the nature of the hashing algorithm used for the Confidence table which requires additional entries beyond 16,000 to support 16,000 patterns effectively and the fact that the Pattern Description and Stateful Rule table must have the same number of Pattern head entries as the Confidence table.

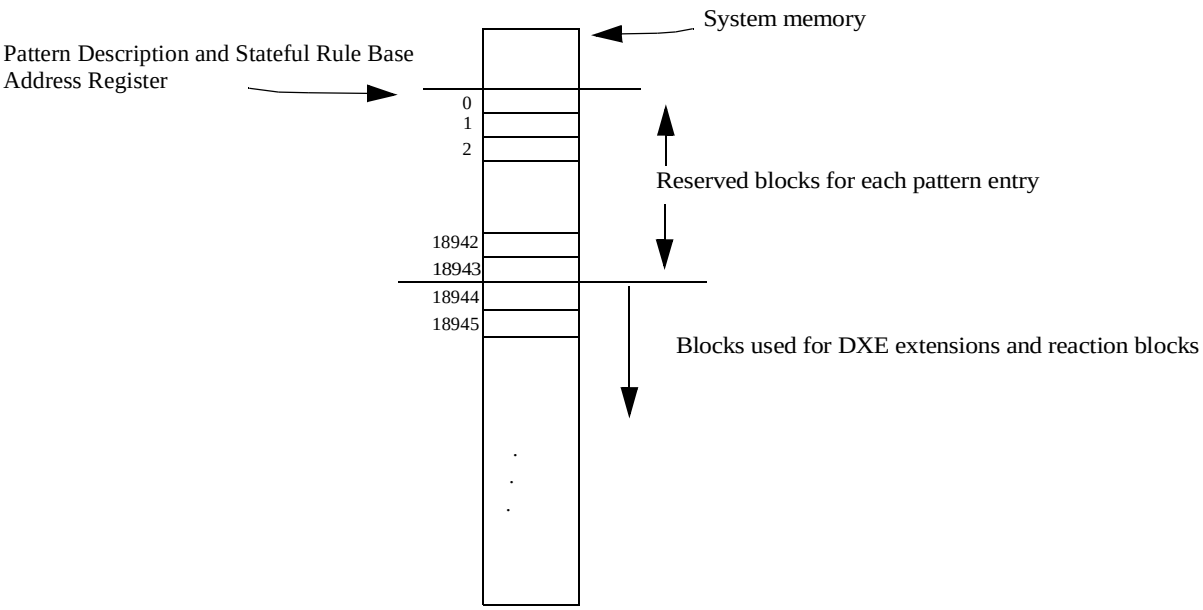


Figure 17-133. Pattern Description and Stateful Rule memory space

17.4.2.1.8 Test Line Examples

The following subsections show examples on how a regular expression can be converted to test lines.

Test Line Example 1

Regex expression: /abc(de)+(f|gh)/

Fingerprint: abcde

Table 17-128. Example 1 Head Test Lines

First Test Position	Number Instruction Lines	Set	Pattern Subset Mask
–5	3	0x00	0xffff

Table 17-129. Example 1 Test Lines

Test Line #	PEC	Cautious	Lazy	Side	Sub. Link	Alter. Link	Repeat Min	Repeat Max	EC	Type Code	Compare Values
1	0	0	0	Right	2	0	1	1	EC_5	22222000	0x6162636465000000 {abcde___}

Table 17-129. Example 1 Test Lines (continued)

Test Line #	PEC	Cautious	Lazy	Side	Sub. Link	Alter. Link	Repeat Min	Repeat Max	EC	Type Code	Compare Values
2	0	1	0	Right	3	0	1	127	EC_2	22000000	0x6465000000000000 {de_____}
3	0	0	0	Right	0	0	1	1	EC_2_1	22200000	0x6768660000000000 {ghf_____}

Test Line Example 2

Regex expression: /html:\/\/(foo|boo|zoo)\.bar/

Fingerprint: html://

Table 17-130. Example 2 Head Test Lines

First Test Position	Number Instruction Lines	Set	Pattern Subset Mask
–7	4	0x00	0xffff

Table 17-131. Example 2 Test Lines

Test Line #	PEC	Cautious	Lazy	Side	Sub. Link	Alter. Link	Repeat Min	Repeat Max	EC	Type Code	Compare Values
1	0	0	0	Right	2	0	1	1	EC_7	22222220	0x68746d6c3a2f2f00 {html://_}
2	1	0	0	Right	4	0	1	1	EC_3_3	22222200	0x666f6f626f6f0000 {fooboo__}
3	0	0	0	Right	0	0	1	1	EC_3	22200000	0x7a6f6f0000000000 {zoo_____}
4	0	0	0	Right	0	0	1	1	EC_4	22220000	0x2e62617200000000 {.bar____}

17.4.3 Stateful Rule Engine

The Stateful rule engine (SRE) is capable of detecting and reporting the occurrence of specified complex scenarios within the data content of individual sessions. A session is defined here as a logical grouping of one or more flows, for instance a pair of flows representing the two directions of communication in a connection. The specification of these scenarios are referred to as Stateful rules.

Stateful rules are Finite State Machines (FSMs) that operate on a per session basis. Stateful rules are expressed as a collection of states. Each state accepts a finite number of events generated by the Data Examination Engine. These events are called pattern match events. Pattern match events are either actual matched patterns (the confirmed detection of a pattern by the Data Examination Engine) or the End of SUI event. The End of SUI event is generated when the Data Examination Engine finishes scanning a work

unit. The generation of the End of SUI event is optionally selectable on a per-stream basis (see Table 17-148).

For every valid input pattern match event, there is a reaction which describes a sequence of actions to be performed by the Stateful Rule Engine. Actions supported by the SRE are:

- Change State
- Report
- Arithmetic operations
- Logical operations
- Conditional branching operations (within a reaction)
- Data/control information movement operations

The above actions are implemented through the SRE instructions. The SRE instructions are described in details in Section 17.4.3.2, “SRE Instruction Set.”

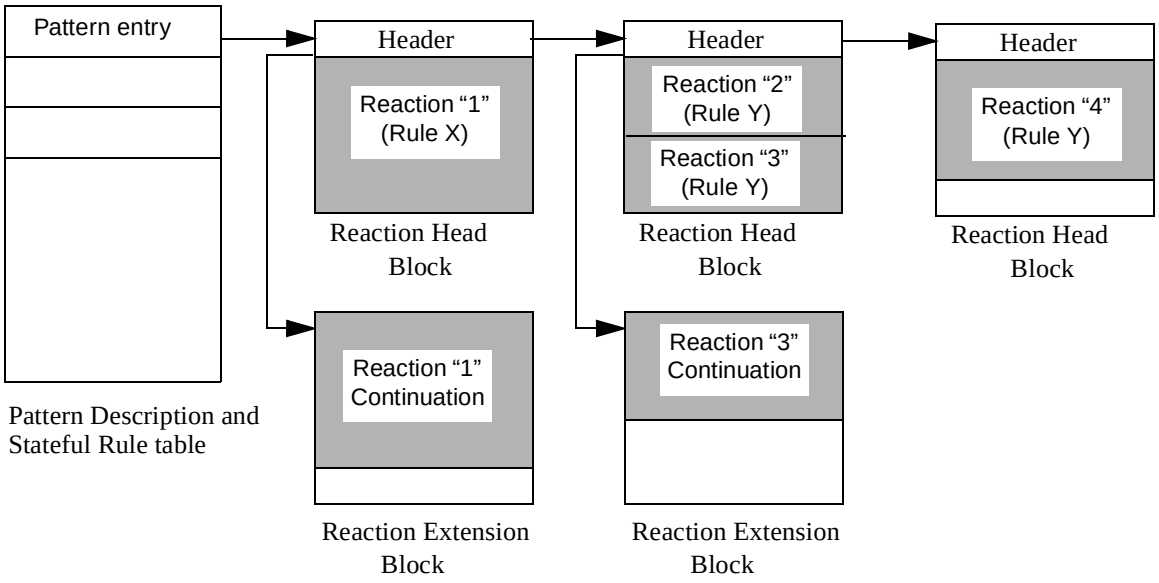
A Stateful rule must have a minimum of 2 states, one of which is to be the “Reset” state (initial state). A Stateful rule may have a maximum of 256 states. Each state must have a minimum of one “reaction”.

As stated earlier, Stateful rules are applied on a per session basis. For every session, the SRE maintains and tracks state and related context information for up to 8192 Stateful rules. This information is stored in the SRE Context table which is located in system memory. This table is indexed using the Session ID stored in the Stream Context.Valid range for the Session ID is 0 to 131.071 million. Session ID “0” has a special meaning where the session is considered active only during the processing of the SUI. Once the processing of the SUI is complete, the context (of session ID 0) is automatically cleared.

17.4.3.1 Stateful Rule Physical Structure

The Stateful Rule Engine (SRE) does not directly examine the content but instead reacts to notification of successfully matched patterns from the Data Examination Engine (DXE). The DXE also supplies additional information specific to that match (possibly derived/extracted from the content) which is made available to SRE instructions through internal registers.

A Stateful rule is defined as a set of reactions where each reaction is associated with a state and describes a sequence of actions (or instructions) to be performed for a pattern match event. Physically, the reactions are stored in system memory in a singly linked list of fixed size 128-byte blocks with the head of the linked list being the pattern they are associated with. There are multiple linked lists where each linked list contains all reactions defined in all Stateful rules that are associated with a pattern. The pointer to the linked list is stored in the Pattern Description and Stateful Rule table, in the first test line of the pattern description entry. For the End of SUI event, the pointer to the reaction linked list associated with the End of SUI event is configured through the SRE Configuration register (see Table 17-38). This pointer represents the head of the reaction linked list that is traversed by the SRE when the DXE detects a successful match on that pattern or an End of SUI. The reaction linked list structure is shown in Figure 17-134.



Note: Reaction “number” is not included in the structure, it is shown here to help visualize the Reaction delineation

Figure 17-134. Reaction Linked List Structure

The reaction linked list consists of singly linked 128-byte blocks (called “reaction head” block) where each block holds the instructions for one or more reactions. If additional space is required to store the instruction information, an additional 128-byte block (called “reaction extension” block) can be linked to the “reaction head” block. A reaction cannot continue onto the next “reaction head” reaction block. Reactions must begin with a “reaction start” instruction” and end with an “end reaction” instruction.See Section 17.4.3.2, “SRE Instruction Set,” for a detailed description of these instructions.

Structure of the “reaction head” 128-byte block is shown in Table 17-135 and the structure of the “reaction extension” 128-byte block is shown in Table 17-136.

The “Next Reaction Ptr” and “Reaction Extension Ptr” pointers are indices of a 128-byte-block relative to the start of the “DXE pattern descriptor memory space.” The registers in Section 17.3.3.5, “Pattern Description and Stateful Rule Base Address High and Low Registers (PDSRBAH and PDSRBAL),” are used to configure the location of the “DXE pattern descriptor memory space” in system memory. The SRE can be configured to verify that pointers to blocks don’t exceed the maximum allowed block index. The maximum allowed block index is configured through the SRE Error Configuration 2 register (see Section 17.3.4.15, “SRE Error Configuration 2 (SEC2)”).

The above reaction blocks are updated/populated using the Pattern Matcher Control command called Write Table Entry. This command allows software to write entries into the Pattern Matcher tables. Table 17-152 details the format of the Write Table Entry Command. These blocks can also be read using the Pattern Matcher Control command called Read Table Entry. Table 17-151 details the format of the Read Table Entry Command.

0	1	2	3	4	5	6	7	Byte 8-11		Byte 12-15	
----	[0]---[1:7] Num. instr. words	[14:31] Next Reaction Ptr (null for last reaction)			[14:31] Reaction Extension Ptr			Instr Word 0	Instr Word 1	Instr Word 2	Instr Word 3
Instr Word 4		Instr Word 5		.....							
									.....	Instr Word 58	Instr Word 59

Figure 17-135. 1 Reaction Head 128-Byte Block Format

Byte 0-3		Byte 4-7		Byte 8-11		Byte 12-15	
Instr Word 60	Instr Word 61	.....					
					.....	Instr Word 122	Instr Word 123

Figure 17-136. Reaction Extension 128-Byte Block Format

17.4.3.2 SRE Instruction Set

The Stateful Rule Engine (SRE) has special-purpose hardware for executing a unique set of instructions called the SRE instructions.

The SRE instructions are an integral number of 16-bit words (2 bytes) ranging from 1 to 5 words (2-10 bytes).

17.4.3.2.1 Instruction Set Operands

Most instructions don’t access system or internal memory directly, but instead use a set of internal SRE registers called SRE Instruction Operand registers to access operands. A subset of these registers are loaded automatically when the SRE begins processing of a reaction. In the case of persistent operands, they are saved automatically when the SRE encounters the end of a reaction. Other registers can be loaded and stored through a separate set of load and store instructions.

The SRE Instruction Operand registers (see Table 17-138) are:

- General Purpose Registered Values (GPRV) 0-15 registers—For rules using context (Dualstate with context and Multistate rules, see Section 17.4.3.2.2, “Instruction Set Description),” these sixteen 8-bits registers can be used to store additional information learned as the state of a rule progresses. GPRV 0-15 can be used individually as 8-bit values or up to 8 adjacent GPRVs may be used together for up to 64-bit operations. Note that for Multistate rules, the ‘State’ value occupies GPRV 0 and thus cannot be used. The values in these registers are persistent within each Stateful rule on a per session basis. These registers are loaded when the SRE is executing a “Reaction Start with Context” instruction unless the Stateful rule has been reset[1]. The “Reaction Start with Context” instruction specifies the location in the session context entry from where these registers are to be loaded. The value stored in these registers are saved in the session context entry (same location as specified in the “Reaction Start with Context” instruction) when the SRE executes an “End of Reaction” instruction. For rules using no context (Stateless and Dualstate without context rules, see Section 17.4.3.2.2, “Instruction Set Description),” or for Dualstate with context rules in the reset state (state is 0), these sixteen 8-bits registers can be used as temporary registers within a reaction. GPRV 0-15 can be used individually as 8-bit values or up to 8 adjacent GPRVs may be used together for up to 64-bit operations. These registers are initialized to zero at the start of the reaction. The value in these registers are not saved when the SRE executes an “end of reaction” instruction.
- General Purpose Registered Values (GPRV) 16-31 registers—These sixteen 8-bits registers can be used to store global information. GPRV 16-31 can be used individually as 8-bit values or up to 8 adjacent GPRVs may be used together for up to 64-bit operations. These registers are global across all sessions and rules. These registers are cleared when the Pattern Matcher block is reset.
- Pattern Match Event Meta Data registers—These registers contain additional information specific to a Pattern Match event (possibly derived/extracted from the content). These are read-only registers that are loaded from information passed from the DXE when the SRE starts processing a Pattern Match event.
- Accumulator registers—Two 64-bit registers (Accumulator A and B) which are used to hold the result of arithmetic and logical operations. These registers are initialized to zero at the start of the reaction. The value in these registers are not saved when the SRE encounters an “end of reaction” instruction.
- Flag register—This 32-bit register records conditions used by the branch instructions. The most significant 16 bits of this register are global for all Stateful rules within a given session and are referred as the Session Persistent flags. These flags must be loaded explicitly through the “fetch flag” instruction. The Session Persistent flags are saved in the session context entry at the location specified in the “fetch flag” instruction when the SRE executes an “End of Reaction” instruction. The least significant 16 bits of this register are local to a reaction and referred as the Volatile flags. The Volatile flags are initialized to zero at the start of the reaction. The Volatile flags are not saved when the SRE encounters an “end of reaction” instruction.
- Immediate Value registers—These registers hold the actual values specified within the instruction. Immediate values ranges from one to four 16-bit words. These registers are loaded automatically when the SRE is executing an instruction that contains immediate values.

1. Contrary to a “multistate” rule, a “dualstate” rule cannot maintain data in its context while its state is 0. While the state of a dualstate rule (with or without context) is 0, the GPRV 0-15 registers can only be used as temporary registers within a reaction.

17.4.3.2.2 Instruction Set Description

See Table 17-137 for a description of the SRE instructions format. A detailed description of the SRE instructions is given in Table 17-132. The SRE instruction fields are described in details in Table 17-133.

The instructions have been defined with the concept that rules fall under the following types:

- Stateless—A Stateless rule has no rule number, no state and does not use any context. These rules are used to perform stateless actions (for example, generating a report). All reactions within this rule must start with the “Reaction Start Stateless” instruction and end with the “End Stateless Reaction” instruction.
- Dualstate without context—This is a Stateful rule which has only two states and which does not use any context besides storing the state value. In this case the state value is stored in the Stateful Rules Summary area of the session context entry and thus it does not occupy any space in the Stateful Rules context area of the session context entry. All reactions within this rule must start with the “Reaction Start Dualstate” instruction and end with the “End Dualstate Reaction” instruction.
- Dualstate with context—This is a Stateful rule which has only two states and which does use additional context (in the Stateful Rules context area) besides storing the state value. All reactions within this rule must start with the “Reaction Start with Context” instruction and end with the “End Dualstate with Context Reaction” instruction.
- Multistate—This is a Stateful rule which can have more than two states. In this case, the state value is stored in the Stateful Rules context area of the session context entry. All reactions within this rule must start with the “Reaction Start with Context” instruction and end with the “End Multistate” instruction.

It is important to point out that the SRE does not automatically check the state value of a reaction. This must be done explicitly using the appropriate instructions.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Additional Words
Nop (0000)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—
Reaction Start Stateless (0001)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	—
End Stateless Reaction (0002)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	—
Reset Rule (0003)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	—
Unused (0004)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	—
End Dualstate Reaction (0005)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	—
End Dualstate with Context Reaction (0006)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	—
End Multistate (0007)	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	—
Set State Bit (0008)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	—
Clear State Bit (0009)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	—
Unused (000A .. 000F)																	—
Accumulator Modify (001x)	0	0	0	0	0	0	0	0	0	0	0	1	AS	Operations			—
Assign Flag Value (Set/Clr) (004x .. 007x)	0	0	0	0	0	0	0	0	0	1	V	Flag Num					—
Fetch Flag (0080 .. 00BF)	0	0	0	0	0	0	0	0	1	0	Flag Index						—
Unused (00C0 .. 057F)																	—
Report from ACCs (058x-05fx)	0	0	0	0	0	1	0	1	1	AS	Acc pos			Reprt Size			—
Unconditional Jump (060x .. 068x)	0	0	0	0	0	1	1	0	0	Absolute Jump Location							—
Test and Jump (068x .. 06Fx)	0	0	0	0	0	1	1	0	1	Absolute Jump Location							[0:4] [5] Inconclusive or State Test [6] Bit or Vector [7] Equal [8:15] State Value
Assign New State (07xx)	0	0	0	0	0	1	1	1	New State Value								—
Compare ACCs and Set Flag (08xx - 0Bxx)	0	0	0	0	1	0	Cmp Type			0	0	Flag Num					—
Compare ACCs and Jump (0Cxx - 0Fxx)	0	0	0	0	1	1	Cmp Type			Absolute Jump Location							—
Jump if Flag Not Set (1XXX)	0	0	0	1	Flag Num					Absolute Jump Location							—
Store Accumulator in GPRV (2xxx)	0	0	1	0	Target					AS	Acc pos			Store Size			—
Max Reaction Execution Limit (30xx-37xx)	0	0	1	1	0	Max Reaction Execution Limit											—
Store Accumulator in Flags (38xx-3fxx)	0	0	1	1	1	AS	Flag Pos					Num Flags					—
Reaction Start Dualstate (4xxx .. 5xxx)	0	1	0	Rule Number													—
Reaction Start with Context (6xxx .. 7xxx)	0	1	1	Rule Number													[0:1] Ctx Size [2:15] Ctx Offset
Load Accumulators (8xxx .. Fxxx)	1	Operation (64-bit)				AS	Operand Selection Code							Selected Operand Size			0,1,2,3,4 Words of Immediate Values

Figure 17-137. SRE Instructions Format

Table 17-132. SRE Instruction Descriptions[1]

Instruction	Description
Nop	No action performed
Reaction Start Stateless	Must be the first instruction issued for a stateless reaction. The stateless reaction does not have an associated rule number, and does not use any state context. The ‘set state bit’, ‘clear state bit’ and ‘assign new state’ instructions cannot be used. The reaction must be terminated by an ‘end stateless reaction’.
End Stateless Reaction	Must be the last instruction issued for a stateless reaction
Reset Rule	Clears the entire context for this rule. Can only be used if the ‘start reaction with context’ has been used and the reaction is of the type multistate. This instruction cannot be used for dualstate reactions. Note that for dualstate reactions with context, the “Clear State Bit” instruction automatically resets the context.
End Dualstate Reaction	Must be the last instruction issued for a dualstate reaction
End Dualstate with Context Reaction	Must be the last instruction issued for a dualstate reaction that uses context.
End Multistate	Must be that last instruction issued for a multistate reaction that uses context.
Set State Bit	Sets the state bit of a dualstate reaction. Can only be used for a dualstate reaction. For dualstate with context reactions, at least one of the GRRV registers must be written to or initialized to zero within state “0” before a transition to state”1” can be made.
Clear State Bit	Clears the state bit of a dualstate reaction. Can only be used for a dualstate reaction. For a dualstate reaction with context, the entire rule context is automatically reset. when the SRE executes the “End Dualstate with Context Reaction” instruction. This means that contrary to a mutistate reaction, context information cannot be maintained/saved in state “0” (reset state).
Accumulator Modify	Performs a modification on the contents held in an Accumulator. Results are kept in the same Accumulator.
Assign Flag Value	Assigns a 1 or 0 to a selected flag.
Fetch Flag	Must be issued by any reaction before using any Session Persistent Flags. The index offset (integral number of 32 byte relative to the start of the session context entry) where the flags are stored must be specified. The most significant 16 bits of the 32-byte entry must hold the stored value of the Session Persistent Flags. This instruction does not need to be called if the reaction only uses the Volatile Flags. The Session Persistent Flags are saved in the session context entry at the location specified in the “fetch flag” instruction when the SRE executes an “End of Reaction” instruction.
Report from Accumulator	This instruction sends a report from 1B-8B from the specified Accumulator and the specified position within that Accumulator.
Unconditional Jump	Perform an absolute jump relative to the start of the reaction to the specified 16-bit word. The first instruction word of a reaction (the ‘start’ reaction) is word number 0.
Test and Jump	Check equality or inequality of current state value (of the state vector or the state bit, depending on dualstate or multistate) compared to a specified state value or equality or inequality of any inconclusive bits. If the test is performed on the state bit, then only the lowest bit of the specified state value is valid. If the test is performed on the inconclusive bit, then either bit triggers the jump in the equality case, and both inconclusive bits must be zero for the inequality case to trigger a jump. An absolute jump relative to the start of the reaction is performed to the specified 16-bit word if the test is true.

Table 17-132. SRE Instruction Descriptions[1] (continued)

Instruction	Description
Assign New State	Sets the state vector to the specified value. Can only be used for a multistate reaction.
Compare Accumulator and Set Flag	The contents of both Accumulators are compared (in a method specified in the instruction) and the result is stored in a flag.
Compare Accumulator and Jump	The contents of both Accumulators are compared (in a method specified in the instruction). An absolute jump relative to the start of the reaction is performed to a specified 16-bit word if the compare results are true.
Jump if Flag Not Set	Tests specified flag. Jumps to an absolute 16-bit word (relative to the start of the reaction) if the flag is set to 0.
Store Accumulator in GPRV	Stores 1B-8B from selected Accumulator at the specified Accumulator position into the specified GPRV(s) (based on the GPRV target alignment specified in the instruction).
Store Accumulator in Flags	Stores the specified Accumulator (64b right aligned) into the 1-32 selected flags.
Max Reaction Execution Limit	If this instruction is called, the value in SRE Free Running Counter Configuration register (see Section 17.3.4.13, “SRE Free Running Counter Configuration (SFRCC)”) is superseded for the maximum instructions executed limit.
Reaction Start Dualstate	Must be the first instruction issued for a dualstate reaction where no context is needed. The dualstate reaction has an associated rule number, but does not use any state context. The ‘assign new state’ instructions cannot be used. Only ‘set state bit’ and ‘clear state bit’ can be used to alter the state. The reaction must be terminated by an ‘end dualstate reaction’.
Reaction Start with Context	Must be the first instruction issued for a dualstate or multistate reaction where context is required. The dualstate or multistate reaction with context has an associated rule number and uses 2B,4B,8B or 16B of state context. The context is located at the specified double-byte offset relative to the start of the session context entry. The ‘assign new state’ instruction must be used for multistate reactions. The ‘set state bit’ and ‘clear state bit’ must be used for dualstate reactions. For the GPRV 0-15 registers, the GPRVs beyond the reactions size cannot be accessed (for example, GPRV 0-7 are available when the context is 8B in size). If the reaction is a dualstate reaction, GPRV 0 is available for storage. If the reaction is a multistate reaction, GPRV 0 is reserved and cannot be assigned using the ‘store accumulator in GPRV’ instruction. The reaction must be terminated by an ‘end dualstate with context reaction’ or ‘end multistate’ depending on the reaction.
Load Accumulator	This instruction loads the selected Accumulator with 1B-8B of the selected operand while potentially performing a 64b math operation with the original value of the selected Accumulator. The load is always 64b right aligned. Note that for operation 7, 8, 9 and 10 the shift size corresponds to the lowest 6 bits of the operand.

1 Invalid operand for a given instruction may produce unknown results.

Table 17-133. SRE Instruction Field Descriptions

Field Name	Description
AS (Accumulator Select)	0: Accumulator A selected 1: Accumulator B selected
Operations	Accumulator modification functions 0: Byte endian Swap 1: Clear ACC 2: Clear ACC (if ACC is negative)
V (Flag Value)	0: clear 1: set
Flag Num	Selected Flag 0-31
Flag Index	Index number (integral number of 32-bytes relative to start of session context entry) where 16b of Session Persistent Flags is stored
Acc Pos	Accumulator Position Select. Selects right-most byte and ‘size’ bytes to the left.
Reprt Size	0-7 => 1B-8B reported from Accumulator
Absolute Jump Location	Jumps to a 16-bit word relative to the start of the reaction. Valid range is 1-123 (word 1- word 123). Word 0 is always a ‘start’ instruction and hence not a valid jump location.
Inconclusive or State Test	0: Test is performed on the state of the rule. ‘Bit or Vector’ field and ‘State Value’ field can be considered valid for use. 1: Test is performed on the inconclusive bits.
Bit or Vector	0: Test state is performed on state vector. Only for Multistate reactions. 1: Test state is performed on state bit. Only for dualstate reactions.
Equal	0: Tests state for inequality 1: Tests state for equality
State Value	Value to compare current state to. If testing state in a dualstate reaction, value can only be 0 or 1. In multistate reactions, test value can be 0-255.
New State Value	Value to set current state to. Can only be used in multistate reactions. Can be 0-255.
Cmp Type	Accumulator compare types 0: A < B 1: A <= B 2: A = B 3: A != B 4: A >= B 5: A > B 6,7: unused
Target	0-31. Selects right most GPRV for storage and ‘size’ GPRV to the left. (for example. if 2B needs to be stored in GPRV3-4, target=4,size=1).
Store Size	0-7 => 1B-8B to be stored in GPRVs selected by Target.
Flag Pos	0-31 Selects right most flag for storage, and ‘num flags’ flags to the left.
Num Flags	0-31 => 1-32 flags to store
Rule Number	0-8191

Table 17-133. SRE Instruction Field Descriptions (continued)

Field Name	Description
Ctx Size	0: 2B 1: 4B 2: 8B 3: 16B
Ctx Offset	Double-byte offset relative to the start of the session context entry. 2B context must be 2B aligned. 4B context must be 4B aligned. 8B context must be 8B aligned. 16B context must be 8B aligned. 0-16383. Accommodates maximum session context entry of 32KB.
Operation (64-bit)	0: AS <= Op 1: AS <= Op + AS 2: AS <= Op - AS 3: AS <= AS - Op 4: AS <= Op or AS 5: AS <= Op and AS 6: AS <= Op xor AS 7: AS <= Op >> AS 8: AS <= Op << AS 9: AS <= AS >> Op 10: AS <= AS << Op 11-15: unused
Operand Selection Code	See Table on page 17-148. Selects right-most byte of matrix and ‘size’ bytes to left. Regardless of alignment of selected bytes, the operation is always right aligned and always 64b. Examples: $X => code=0x67 size=7 $M => code=0x47 size=3 bytes 3-5 of $Y => code=0x6d size=2 $N => code=0x4f size=0 leftmost 2B of Acc B => code=0x29 size=1
Selected Operand Size	0-7 => 1B-8B for load accumulator operation.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
Byte: 0x	GPRV(0) State	General Purpose Registered Values(1:15) (Session and Rule Persistent)														
Byte: 1x	General Purpose Registered Values (16:31) (Global Persistent)															
Byte: 2x	Accumulator A								Accumulator B							
Byte: 3x	Session Persistent Flags(0..15)		Volatile Flags(16..31)		$T[0:31]				$Sc[0:47]						$Nr	$Nl
Byte: 4x	$P[0:31]				$M[0:31]				$Si[0:47]						$R	$N
Byte: 5x	$Ob[0:31]				$Ox[0:31]				--	--	--	--	--	$I	$Xn	$Yn
Byte: 6x	$X[0:63]								$Y[0:63]							
Byte: 7x	$C[0:31]				—	—	—	—-	Additional Instruction Word(0)		Additional Instruction Word(1)		Additional Instruction Word(2)		Additional Instruction Word(3)	

Figure 17-138. Operand Register Select Table (for Accumulator Load Instruction)

Table 17-134. Pattern Match Event Meta Data Registers

Registers	Matched Pattern Event	End of SUI Event
$C[0:31]	A 1 usec tick counter (rolls over silently). Configured using prescaler value in the SRE Free Running Counter Configuration register (see 17.3.4.13/17-42).	A 1 usec tick counter (rolls over silently). Configured using prescaler value in the SRE Free Running Counter Configuration register (see 17.3.4.13/17-42).
$T[0:31]	A generic 32 bit tag assigned to the pattern that invoked this reaction.	Not applicable; set to 0.
$I [0:7]	Indicates whether the match that invoked this reaction was inconclusive. See & Table 17-121 & Table 17-122 for possible values. $I[7]: When set, inconclusive on right because it reached the right edge of the 128-byte window (64 bytes after the trigger position) without eliminating the possibility of a match (although everything within the window matches). $I[6]: Inconclusive on left because it reached the left edge of the 128-byte window (63 bytes before the trigger position) without eliminating the possibility of a match (although everything within the window matches). $I[5]: Pattern match; only applicable when “alternate” inconclusive mode is used.	Not applicable; set to 0.
$P [0:31]	$P is the position within the String Under Inspection (SUI) at which the trigger byte was found. The trigger byte being the last (rightmost) symbol of the trigger fingerprint. The first byte of an SUI is considered to be 1.	$P is the position within the String Under Inspection (SUI) that corresponds to the last byte in the SUI. The first byte of an SUI is considered to be 1.

Table 17-134. Pattern Match Event Meta Data Registers (continued)

Registers	Matched Pattern Event	End of SUI Event
$Nl [0:7]	$Nl is the number of bytes to the left of the trigger byte (last symbol of the fingerprint) that are matched by the pattern. As an example usage, If one wanted to report the position within the SUI of the first matching byte of a pattern, the report would specify “$P-$Nl” as the argument.	Not applicable; set to 0.
$Nr [0:7]	$Nr is the number of bytes that match to the right of the trigger byte (last symbol of the fingerprint) that are matched by the pattern. As an example usage, if one wanted to report the position within the SUI of the last matching byte of a pattern, the report would specify “$P+$Nr” as the argument.	Not applicable; set to 0.
$N[0:7]	Number of bytes matched by the pattern (max value of 128). Equal to $Nl+1+$Nr.[1]	Not applicable; set to 1; equal to $Nl+1+$Nr.
$M[0:31]	$M is the position of rightmost byte of the match relative to the work unit passed in (the SUI without the prefixed Residue). The value is equal to $P+$Nr-$R. The first byte of a work unit is considered to be 1.	$M is the position of the “last byte in the SUI” relative to the work unit passed in (the SUI without the prefixed Residue). The value is equal to $P-$R. The first byte of a work unit is considered to be 1.
$Sc [0:47]	The number of bytes completely scanned (such as scanned and not held as residue) prior to the current SUI. As an example usage, if one wanted to report the position of the first matching byte of a pattern within the stream, the report would specify “$Sc+$P-$Nl” as the argument. The value is equal to $Si - $R	The number of bytes completely scanned (such as scanned and not held as residue) prior to the current SUI. The value is equal to $Si - $R
$Si [0:47]	The number of bytes scanned initially (such as scanned but possible still held as residue) prior to the current work unit. As an example usage, if one wanted to report the position of the last matching byte of a pattern within the stream, the report would specify “$Si+$M” as the argument.	The number of bytes scanned initially (such as scanned but possible still held as residue) prior to the current work unit.
$R[0:7]	The number of bytes of residue from a previous work unit which is prefixed to the current work unit to form the SUI. Alternatively can be thought of as the offset within the SUI where new previously unscanned data begins (such as not the residue from the previous work unit). As an example usage, if one wanted to report the trigger position of a match within the “work unit” passed to the PM, the report would specify “$P-$R” as the argument.	The number of bytes of residue from a previous work unit which is prefixed to the current work unit to form the SUI. Alternatively can be thought of as the offset within the SUI where new previously unscanned data begins (such as not the residue from the previous work unit).
$Ob[0:31]	The position within the SUI where a line break character (LF or CR) last occurred (closest to but not after the current value of $P). A value of $Ob=0 indicates that a break has not yet been seen in the current SUI. The first byte of an SUI is considered to be 1. As an example usage, $Ob allows a rule to be written such that it is effectively reset by a “CR” or “LF”.	The position within the SUI where a line break character (LF or CR) last occurred (closest to but not after the current value of $P). A value of $Ob=0 indicates that a break has not yet been seen in the current SUI. The first byte of an SUI is considered to be 1. As an example usage, $Ob allows a rule to be written such that it is effectively reset by a “CR” or “LF”.

Table 17-134. Pattern Match Event Meta Data Registers (continued)

Registers	Matched Pattern Event	End of SUI Event
$Ox[0:31]	The position within the SUI where an extended character (such as bit 7 set) last occurred (closest to but not after the current value of $P). A value of $Ox=0 indicates that an extended character has not yet been seen in the current SUI. The first byte of an SUI is considered to be 1.	The position within the SUI where an extended character (such as bit 7 set) last occurred (closest to but not after the current value of $P). A value of $Ox=0 indicates that an extended character has not yet been seen in the current SUI. The first byte of an SUI is considered to be 1.
$X[0:63]	A generic 64-bit value captured by the DXE captured during the pattern match that invoked this reaction.	Not applicable; set to 0.
$Xn[0:7]	The number of character positions that matched and contributed to the value $X. (A zero indicates no successful capture match occurred for the pattern match.) This value can be used to determine if an “overflow” of the capture value occurred as follows. $Xn > 8 : overflow of a “right justified” or “left justified” string capture. $Xn > 64 : overflow of a binary capture $Xn > 21 : overflow of a octal capture $Xn > 16 : overflow of a hexadecimal capture $Xn > 16 : overflow of a decimal capture (Although a 19 digit decimal value converts to less than 64-bits, the decimal capture has same limit as hexadecimal capture (16) because capture value is temporarily stored as binary-coded-decimal (4-bits per digit) before being converted to true binary.)	Not applicable; set to 0.
$Y[0:63]	A second generic 64-bit value captured by the DXE captured during the pattern match that invoked this reaction.	Not applicable; set to 0.
$Yn[0:7]	The number of character positions that matched and contributed to the value $Y Similar to $Xn.	Not applicable; set to 0.

[1] For 0 byte match, Nr is set to 1.

17.4.3.3 SRE Context Table

The SRE maintains the meta-state for up to 8192 Rules on a per-session basis in records called SRE session context entry. These entries are stored in the SRE Context table which is located in the system memory as shown in Figure 17-139.

Each session has access to a programmable total size of context. The memory used for each session can be 32 bytes, 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1 Kbyte, 2 Kbytes, 4 Kbytes, 8 Kbytes, 16 Kbytes, or 32 Kbytes based on a setting in Section 17.3.4.7, “SRE Configuration Register (SREC).” The registers in SRE Context Base Address High and Low Registers (SCBARH and SCBARL) (Section 17.3.4.5 on page 36) are used to configure the location of the SRE Context table in system memory. The maximum size for the SRE Context table is 4 Gbytes. The size of the table is configured through the register in SRE Error Configuration 3 (SEC3) (Section 17.3.4.16 on page 44) if the SRE has been configured (see Section 17.3.2.11, “PME Error Handling Disables (PEHD)”) to verify that the session ID passed in the Stream Context points to a valid entry (location within the space allocated to the SRE Context table). Take

note that the maximum number of sessions are constrained by the session context size. For example, the maximum number of sessions would be lower than 8.192 million if the session context size was configured to a size of 512 bytes or greater.

The session context entry is divided into two areas: Stateful Rules Summary area and the Stateful Rules Context area.

The Stateful Rules Summary area is located at the start of the session context entry. This area includes a 1-bit flag for each Stateful rule (first bit corresponds to rule number 0, second bit corresponds to rule number 1, and so on). This space can occupy up to 8192 bits since the maximum number of Stateful rules is 8192. For “multistate” Stateful rules (such as rules made of “multistate” reactions), this flag indicates whether or not the Stateful rule has valid data (context data that has not been reset) in its associated context. A value of “1” indicates that there is valid data in the context whereby a value of 0 indicates that the context data been reset. For “dualstate” Stateful rules (such as rules made up of “dualstate” reactions), this flag represents the state of the Stateful rule. Since a “dualstate” rule (contrary to “multistate” rule) cannot maintain data in its context while in the reset state (state = 0), this flag implicitly indicates whether or not the Stateful rule has valid data in its associated context.

The Stateful Rules Context area follows the Stateful Rules Summary area and is used to store context for all Stateful rules for a particular session. Individual rules can consume 0 (“dualstate” rule without context) 2,4,8 or 16 bytes of context each, depending on how the rule has been defined.

Note that a session context size of 32 bytes can only be used if there are only “Dualstate without context” Stateful rules since the Stateful Rules Summary area requires a minimum area of 32 bytes. In other words, a session context cannot have a Stateful Rules Context space if the context size has been configured to 32 bytes.

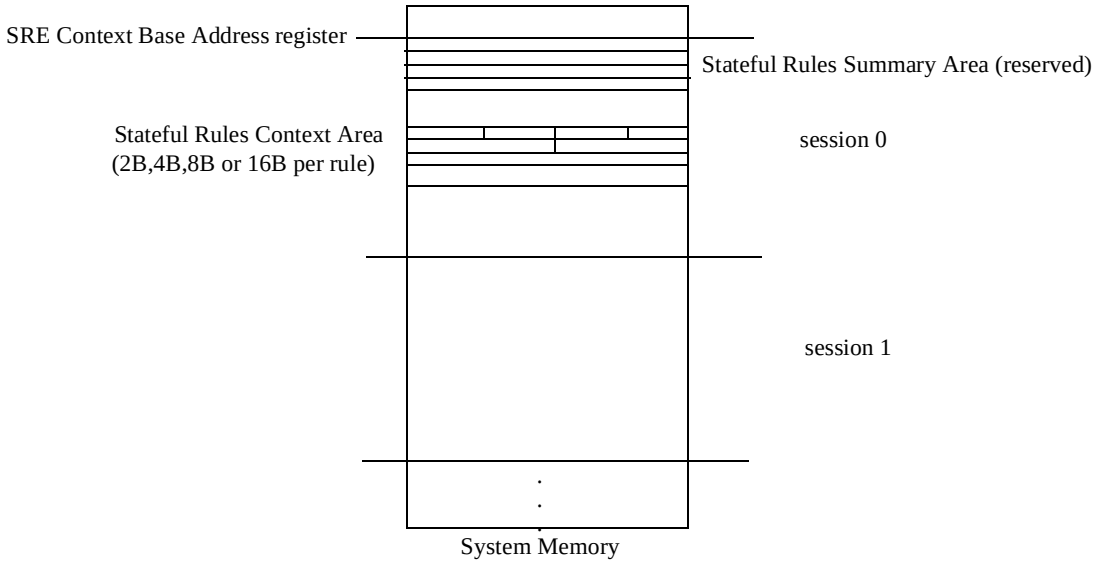


Figure 17-139. SRE Session Context Entries

The SRE Context table can be “write/read” at the 32-byte unit using the Pattern Matcher Control command called Write Table Entry (Table 17-152 details the format of the Write Table Entry Command) and the Pattern Matcher Control command called Read Table Entry (Table 17-151 details the format of the Read

Table Entry Command).There is also a command that allows individual session context entry to be cleared individually. This command is called Clear Session Context by Session ID (Table 17-153 details the format of this command). Furthermore, there is the capability of clearing context data associated with a set of Stateful rules through setting the appropriate SRE control registers. Specifically one could reset the context associated with a set of Stateful rules by following the sequence described below:

1. Set the rate and priority at which the reset operation is to be executed by the Pattern Matcher. This is achieved by configuring the SRRWC register (see Section 17.3.4.12, “SRE Rule Reset Work Configuration (SRRWC)”).
2. Set the size of the session context entry by configuring the SRRI register (see Section 17.3.4.10, “SRE Rule Reset 32-Byte Increment (SRRI)”).
3. Specify the Stateful Rules to be reset by first setting an index to point to a 32-byte entry within the Stateful Rules Summary area (first entry corresponds to Stateful 0–255, second entry corresponds to 255-511 and so on) of the first session context entry to be reset. This index is configured through the SRRFI register (see Section 17.3.4.9, “SRE Rule Reset First Index (SRRFI)”). Next, specify the reset mask for the above set of 256 Stateful rules (wether to reset or not). This mask is configured through the SRRV0-7 registers (see Section 17.3.4.8, “SRE Rule Reset Vector 0–7 (SRRV0–7)”).
4. Specify the number of sessions for which Stateful rules context need to cleared by configuring the SRRR register (see Section 17.3.4.11, “SRE Rule Reset Repetitions (SRRR)”). Writing into this register starts the reset operation. Subsequently reading a value of 0 from this register means that the reset operation is complete.
5. Repeat step 3–4 if additional rules need to be reset since one reset operation can reset up to 256 Stateful rules.

17.4.3.4 SRE Report Format

The SRE reports on a variety of events.

A pattern match produces a simple match report if the ‘Simple Match Report’ bit is set in the pattern Test Line Block. The Simple Match report is produced BEFORE any match Reaction reports generated by this pattern match.

An End of SUI produces a Simple End of SUI report if the ESR bit is set in the SREC register (Table 17-38). The Simple End of SUI report is produced AFTER any Reaction reports generated by the End of SUI reaction (if the EOSRP is non zero in the SREC register).

Reactions can create reports by using the ‘Report from ACCs’ instruction. This report is 1-8B in length and is always made up of bytes from either accumulator.

All reports for a given SUI are packed together within a single notification result.

17.4.3.4.1 Simple Match Report

The Simple Match report is described in Table 17-135. Refer to Table for a description of the Pattern Match Event Meta Data registers.

Table 17-135. Simple Match Report

Byte Offset	Bits	Description
0	0	Set to 0.
0	1–3	Lowest 3 significant bits of the $I Pattern Match Event Meta Data register.
0	4–7	Set to 0x1.
1	0–47	Value of the $N Pattern Match Event Meta Data register.
2	0–31	Value of the $Si Pattern Match Event Meta Data register.
8	0–31	Value of the $M Pattern Match Event Meta Data register.
12	0–31	Value of the $T Pattern Match Event Meta Data register.

17.4.3.4.2 Simple End of SUI Report

The Simple End of SUI report is described in Table 17-136.

Table 17-136. Simple End of SUI Report

Byte Offset	Bits	Description
0	0–7	Set to 0x80.
1	0–31	Total bytes in work unit.

17.4.3.4.3 Reaction Reports

The Reaction reports include 1 to 8 bytes of report as specified in the reaction instruction.

17.4.3.4.4 Verbose Reports

There are 4 levels of verbosity allowed for pattern match reports. The verbosity is defined in the stream context (see Table 17-148). In Verbose Mode 0, a pattern match produces a Simple Match report if the ‘Simple Match Report’ bit is set in the pattern Test Line Block. In Verbose Mode 1, 2 or 3, no Simple Match report is generated even if the ‘Simple Match Report’ bit is set. The Verbose Mode 1, 2 and 3 reports are described below.

Verbose Mode 1 Report

In Verbose Mode 1, verbose match information is reported for every pattern match. The Verbose Mode 1 report is described in Table 17-137. Refer to Table 17-134 for a description of the Pattern Match Event Meta Data registers.

Table 17-137. Verbose Mode 1 Report

Byte Offset	Bits	Description
0	0	Set to 0.
0	1–3	Lowest 3 significant bits of the $I Pattern Match Event Meta Data register.
0	4–7	Set to 0x2.
1	0–7	Value of the $N Pattern Match Event Meta Data register.
2	0–47	Value of the $Si Pattern Match Event Meta Data register.
8	0–31	Value of the $M Pattern Match Event Meta Data register.
12	0–31	Value of the $T Pattern Match Event Meta Data register.
16	0–31	Value of the $Ob Pattern Match Event Meta Data register.
20	0–31	Value of the $Ox Pattern Match Event Meta Data register.
24	0–7	Value of the $Xn Pattern Match Event Meta Data register.
25	0–7	Value of the $Yn Pattern Match Event Meta Data register.
26	0–7	Value of the $I Pattern Match Event Meta Data register.
27	0–63	Value of the $X Pattern Match Event Meta Data register (0 if no capture).
35	0–63	Value of the $Y Pattern Match Event Meta Data register (0 if no capture).

Verbose Mode 2 Report

In Verbose Mode 2, the Verbose Mode 1 report is generated on a pattern match, followed by Verbose Mode 2 report(s) providing a summary of the state of the each rule that executed due to that match. There is a Verbose Mode 2 report generated for each of the rule executed. The structure of the Verbose Mode 2 report depends on the type of rule:

- Stateless rule—See Table 17-138 for a description of the Verbose Mode 2 report for a Stateless rule.
- Dualstate without context rule—See Table 17-139 for a description of the Verbose Mode 2 report for a Dualstate without context rule.
- Dualstate with context rule—See Table 17-140 for a description of the Verbose Mode 2 report for a Dualstate with context rule.
- Multistate rule—See Table 17-141 for a description of the Verbose Mode 2 report for a Multistate rule.

Table 17-138. Verbose Mode 2 Report—Stateless Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x03. Represents a Stateless rule.

Table 17-139. Verbose Mode 2 Report—Dualstate Without Context Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x04. Represents Dualstate without context rule.
1	0–15	Rule number
3	0–6	Set to 0.
3	7	Rule state bit

Table 17-140. Verbose Mode 2 Report—Dualstate With Context Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x05. Represents Dualstate with context rule.
1	0–15	Rule number
3	0–6	Set to 0.
3	7	Rule state bit

Table 17-141. Verbose Mode 2 Report—Multistate Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x07. Represents Multistate rule.
1	0–15	Rule number
3	0–7	Rule state

Verbose Mode 3 Report

In Verbose Mode 3, the Verbose Mode 1 report is generated on a pattern match, followed by Verbose Mode 2 report(s) providing a summary of the state of the each rule that executed due to that match, followed by Verbose Mode 3 report(s) providing a summary of the state of the rule that executed due to that match and the context owned by that rule. There is a Verbose Mode 3 report generated for each of the rule "with context" executed. The structure of the Verbose Mode 3 report depends on the type of rule:

- Dualstate with context rule—See Table 17-142 for a description of the Verbose Mode 3 report for a Dualstate with context rule.
- Multistate—See Table 17-143 for a description of the Verbose Mode 3 report for a Multistate rule.

Table 17-142. Verbose Mode 3 Report—Dualstate with Context Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x08. Represents Dualstate with context rule.
1	0–15	Rule number
3	0–6	Set to 0.

Table 17-142. Verbose Mode 3 Report—Dualstate with Context Rule (continued)

Byte Offset	Bits	Description
3	7	Rule state bit
4	0–127	Rule Context (left aligned 16 bytes reported even if rule only uses 2 bytes, 4 bytes or 8 bytes)
20	0–127	Global context
36	0–15	Session Persistent Flags
38	0–15	Volatile Flags

Table 17-143. Verbose Mode 3 Report—Multistate Rule

Byte Offset	Bits	Description
0	0–7	Set to 0x06. Represents Dualstate with context rule.
1	0–15	Rule number
3	0–7	Rule state.
4	0–127	Rule Context (left aligned 16 bytes reported even if rule only uses 2 bytes, 4 bytes or 8 bytes)
20	0–127	Global context
36	0–15	Session Persistent Flags
38	0–15	Volatile Flags

17.4.4 Deflate Engine

The Deflate Engine performs decompression of Deflate, GZIP, and ZLIB encoded streams. Other GZIP and ZLIB algorithms aside from Deflate are not supported.

The Deflate Engine consists of the following major sub-blocks:

- Huffman Decoding Engine:
  - Handles header and trailer decoding (GZIP, ZLIB, and Deflate), Huffman tree configuration (Deflate), and Huffman code matching (Deflate)
- Data Expansion Engine:
  - Maintains a data history and expands distance/length pairs into data from the history store

17.4.4.1 Huffman Decoding Engine

The Huffman Decoding Engine (HDE) handles the complex process of expanding variable bit-width Huffman codes from their encoded values into their intermediate values. These intermediate values can consist of Literals, Lengths, and Distances that are encoded in a set of Huffman “alphabets”. These alphabets can be pre-set or, optionally, encoded in the Deflate header. A header Huffman “alphabet” must be decoded first, to in turn decode this payload alphabet. Multiple blocks can be contained in a single Deflate payload, each with a distinct Huffman alphabet. The resulting intermediate values are passed on to the Data Expansion Engine.

The HDE handles the headers and trailers associated with the GZIP, ZLIB and Deflate algorithms.

The HDE can also operate in a passthrough mode to allow non-compressed data to proceed to the pattern matcher engines.

17.4.4.2 Data Expansion Engine

The Data Expansion Engine (DEE) handles two types of input; literals and length/distance pairs. Literals are stored in the data history and sent to the output. Length/distance pairs refer to a number of bytes of data starting N bytes previous to the most recent byte.

The data history contains up to a maximum of 32KBytes of previous data.

The DEE also performs the additional tasks of maintaining a CRC and SIZE count for checking verses the GZIP and ZLIB trailer values.

17.4.5 DMA Engine

This section describes the functionality of the DMA Engine within the Pattern Matcher block.

The DMA Engine, like the other Pattern Matcher functional blocks, interfaces with the SoC bus system via the MIA and the IRI blocks. The DMA Engine offers four independent DMA channels which driver software uses to command pattern matching work and to receive notifications of completed pattern matching work. DMA configuration is flexible such that the four channels can be driven by the same or different CPUs in the system.

A DMA channel consists primarily of data structures that exist in system memory at programmable locations; plus some structure control values which exist in system memory (also at programmable locations) and are published to or from DMA Engine registers. The primary data structures that exist in system memory are:

- Command FIFO. A ring structure (array of entry locations) of programmable depth which the driver uses to dispatch commands to the DMA Engine; such as pattern scan commands.
- Notification FIFO. A ring structure (array of entry locations) of programmable depth which the DMA Engine uses to inform the driver of events; such as availability of a pattern scan result.
- Free Buffer Deallocate FIFO. A ring structure (array of entry locations) of programmable depth which the driver uses to add free buffers to the DMA Engine channels' free buffer lists.

Figure 17-140 illustrates the various structures within system memory that are accessed by the DMA Engine in order to both control and stream data to and from the Pattern Matcher. Each of the FIFO structures present in system memory is accessed and controlled via a producer/consumer index pair. In the case of the Command FIFO a third expiry index is used as well.

The driver publishes Command FIFO entries by writing new values of CPI (Commands Produced Index) to a register in the Pattern Matcher block.

Command FIFO entries contain a command code, a pointer to a stream context entry, an input data descriptor, instructions as to what the DMA Engine is to do with output from command, some command modifier bits, and a region of pass-through context provided for the benefit of driver software.

The command code informs the DMA Engine as to what type of command the entry contains. Typically the command code indicates that the DMA Engine is to consult the indicated stream context entry for pattern matcher specific instructions on what type of processing to apply to the indicated input data.

The indicated stream context entry contains pattern matcher specific instructions, context and state. The instructional portion consists primarily of an Activity Code that indicates the type of processing that is to be applied to any data that is processed on the stream.

Input data descriptors supplied in the Command FIFO entry can be of either linked list or scatter/gather format.

Instruction fields in the command instruct the DMA Engine as to what is to be done with output data resulting from processing of the command. Output data is typically either pattern matching result information, decompressed output data from the Deflate Engine, or both. Separate output buffer descriptors of scatter/gather format for each of these possible data sources can be supplied in the command. Alternatively, the command may direct the DMA Engine to store output from one or both of the output data sources in buffers taken from a hardware managed free buffer list. The command specifies which of the two per-channel free buffer lists to use for each source of output data and also specifies whether the DMA Engine is to create a scatter/gather chain or linked list structure out of free buffers.

Once processing of a command is complete, the DMA Engine informs the driver that it has consumed Command FIFO entries by publishing new values of CCI (Commands Consumed Index) to a register in the Pattern Matcher block and, optionally, generating an interrupt.

The driver informs the DMA Engine that it has expired (cleaned up) Command FIFO entries by publishing new values of CEI (Commands Expired Index) to a register in the Pattern Matcher block. See Section 17.4.5.1, “Command and Notification FIFO Operation for more information on the use of CEI.

The DMA Engine informs the driver that new Notification FIFO entries are available for processing by publishing new values of NPI (Notifications Produced Index) to a register in the Pattern Matcher block and, optionally, generating an interrupt.

Notification FIFO entries contain a stream ID token copied from the stream context indicated by the initiating command, a pointer to an output data descriptor containing either pattern matcher result data or decompressed output, a status code, and a region of pass-through context reflected from the command that initiates the pattern matcher activity.

The driver informs the DMA Engine that it has consumed Notification FIFO entries by publishing new values of NCI (Notifications Consumed Index) to a register in the Pattern Matcher Block.

The Free Buffer Deallocate FIFO mechanism allows driver software to add free memory buffers to the Free Buffer Manager’s free buffer lists for management and use by the DMA Engine. Free Buffer Deallocate FIFO entries contain data descriptors that may be of linked list or scatter/gather format.

The driver publishes Free Buffer Deallocate FIFO entries to the DMA Engine by writing new values of PI (Producer Index) to a register in the Pattern Matcher block.

The DMA Engine informs the driver that it has consumed Free Buffer Deallocate FIFO entries by publishing new values of CI (Consumer Index) to a register in the Pattern Matcher block and, optionally, generating an interrupt.

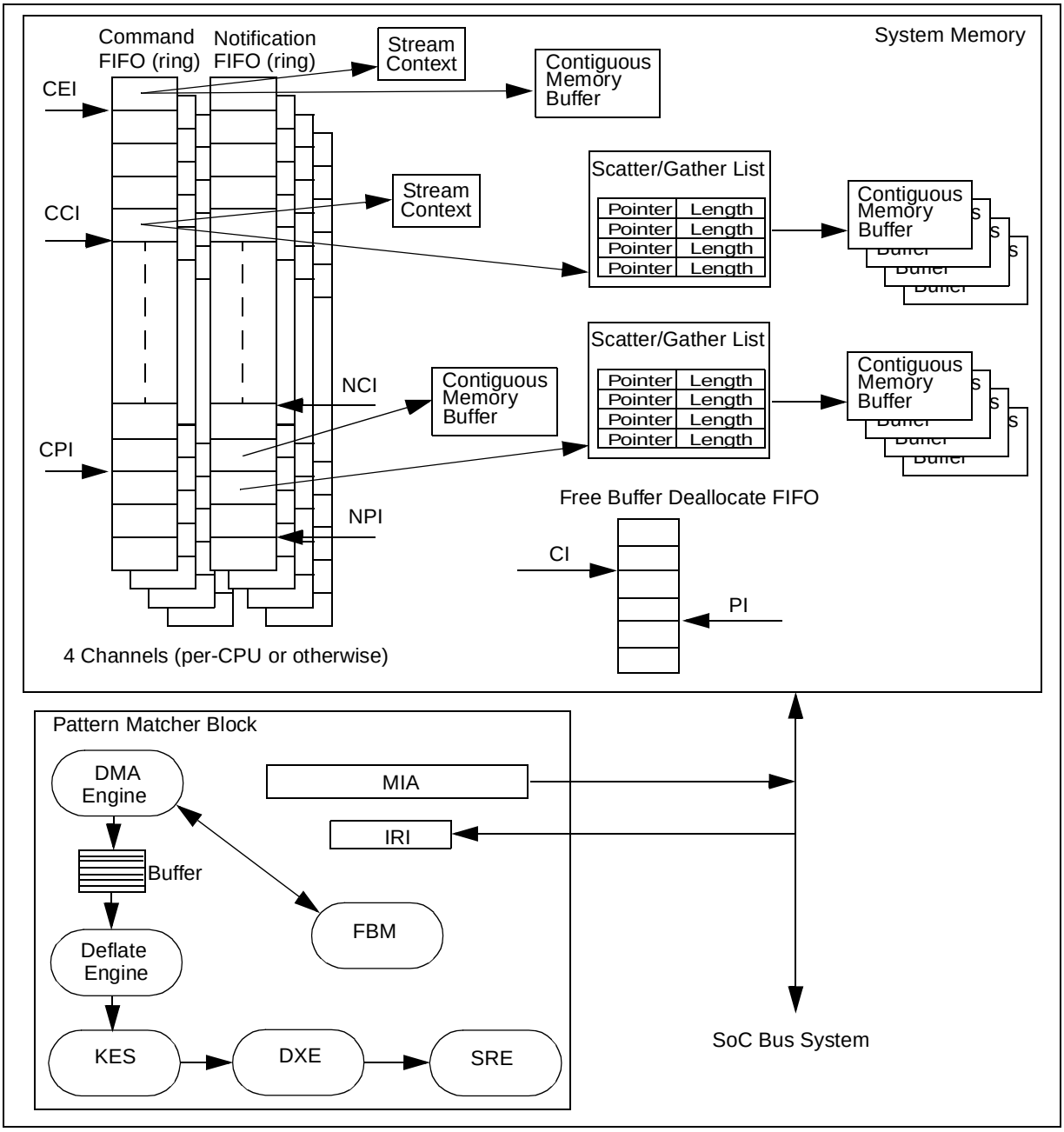


Figure 17-140. DMA Engine Memory Structures

17.4.5.1 Command and Notification FIFO Operation

The DMA Engine implements four separate Command and Notification channels. The channels may be divided amongst separate CPUs or may be used by different processes on the same CPU. The Pattern Matcher functional logic itself is single channel; the DMA Engine schedules work units from the different channels for servicing by the single channel pattern matching logic.

The Command FIFO is populated by the host, whereas the notification FIFO is populated by the DMA Engine. The FIFO indices are used to manage communications, with the DMA Engine implementing an

interrupt state-machine through the command and notification FIFO indices together with an interrupt status register.

Command FIFO entries may be either 64 or 128 bytes long, depending on the level of functionality required. Notification FIFO entries are fixed at 64 bytes long.

The 3-letter naming convention for the FIFO indices is trivial; the third letter is always ‘i’ for index, the first letter refers to the FIFO, and the second letter is either *p*roduced, *c*onsumed, or *e*xpired.

FIFO sizes are always 2^n for some integral n, but due to hardware design considerations, FIFO indices actually are represented with n+1 bits. This means that the top bit in a FIFO index must be masked off to map it to a FIFO entry. The benefit of this scheme is that wrap-around of a FIFO index to zero only occurs after wrapping around the actual FIFO twice—as such, there is never any ambiguity as to whether cei=cpi or nci=npi indicates an empty or full FIFO.

To initiate Pattern Matcher activity, software populates Command FIFO entries with field values as specified in Table 17-146.

Commanded work is dispatched as follows:

1. Software populates one or more formatted Command entries starting at the FIFO location pointed to by CPI.
2. Software writes CPI + N to the CPI register in the Pattern Matcher register space, where N is the number of Command entries being dispatched.
3. Upon completely consuming a command, the DMA Engine updates the CCI register in the Pattern Matcher register space with the new value CCI + 1.
4. Whether by polling or interrupt, software detects the completion of a command by comparing CCI and CEI. When CCI != CEI there are completed commands in the FIFO that can be expired by software.
5. As software expires Command FIFO entries it writes CEI + 1 to the Pattern Matcher register space. The Pattern Matcher need only be aware of the value of CEI for the purpose of interrupt generation.

The DMA Engine only consumes an available Command FIFO entry if there is sufficient space in the Notification FIFO to store the results of the operation. This is done to avoid blocking of service to all channels by the DMA Engine upon completion of a command and encountering of a full Notification FIFO. Once a Command is accepted its progress is not blocked by channel FIFO congestion.

As stated, the DMA Engine does not publish an updated CCI value to the driver until the corresponding commands (up to and including the new CCI) is fully consumed by the DMA Engine (all required data buffers have been read from system memory). Once the driver observes that CCI has been updated it is safe for him to deallocate any buffers that are pointed to by the consumed command entry.

The DMA Engine informs the CPU of completed work via the Notification FIFO. In the Pattern Matcher implementation, Notification FIFO entries are always the result of a previously issued Command FIFO entry. Table 17-154 details the Notification FIFO Entry fields and their meanings.

The process of creating a Notification FIFO Entry is as follows:

1. The DMA Engine populates the Notification FIFO entry pointed to by its NPI index with a formatted Notification entry.

2. The DMA Engine updates the NPI register in the Pattern Matcher register space with a value of NPI + 1.
3. Whether by polling or interrupt, software detects the presence of Notifications by comparing NPI and NCI. When NPI != NCI there are notifications in the FIFO that can be processed by software.
4. As software processes notifications he frees the Notification FIFO entries by writing NCI+N to the Pattern Matcher register space, where N is the number of entries that have been consumed.

17.4.5.2 Command FIFO Entries

An entry in the Command FIFO specifies the following information:

- Command code
- Stream ID
- Input data descriptor
  - DMA address and length of contiguous buffer, or
  - DMA address and length of array (scatter/gather or linked list) of input descriptors
- Result data descriptor
  - DMA address and length of contiguous buffer, or
  - DMA address and length of array (scatter/gather or linked list) of buffer descriptors, or
  - Nil (any result should use the free buffer list)
- Deflate data descriptor
  - DMA address and length of contiguous buffer, or
  - DMA address and length of array (scatter/gather or linked list) of buffer descriptors, or
  - Nil (any deflate should use the free buffer list)
- Command Modifier Bits
- Pass-through data

17.4.5.2.1 Command FIFO Command Codes

This section describes in detail the set of command codes supported by the Pattern Matcher via the Command FIFO. The supported commands are:

- NOP
- Stream Context Update
- Stream Context Driven

NOP

No Operation. This command is intended for use as a fence by software. The users may want to know when their commands (issued with notifications suppressed) have been consumed by the Pattern Matcher. This is done by having a NOP command (without notifications suppressed) follow a set of commands issued with notifications suppressed. The NOP notification indicates that the Pattern Matcher has consumed all preceding commands.

Stream Context Update

Stream Context Update commands are used for initializing (at startup) or updating/re-initializing 64 byte stream context entries. For this type of command, the input data descriptor in the Command FIFO entry describe an ordered {TYPE,VALUE} list in system memory that the DMA Engine interprets in order to modify stream context fields. TYPE is a 32 bit enumerated code and the value that follows it is provided in an integral number of 32 bit words, even if the value itself is not an integral multiple of 32 bits.

Bits [0-7] of Type indicate the type of stream context entry being initialized. Pattern Matcher stream context is identified as stream context type 0x01.

Bits [8-15] of Type indicate the length of the Value in bytes. The actual number of data bytes following Type is understood to be always mod 4. In cases where the value is not an integral number of 4 bytes the value is presumed to be right justified within the 4 byte word; for example, a 1 byte value would appear in the 4th byte of a 4 byte word as 0x000000NN where NN is the 1 byte value.

Bits [16-31] of Type indicate the type code itself.

Stream context update type value pairs must be aligned on 8 byte address boundaries in system memory.

Table 17-148 defines the type and value codes that are supported by the Pattern Matcher block.

Stream Context Driven

A Stream Context Driven command in the Command FIFO indicates that the stream context contains sufficient information for the Pattern Matcher Engine to process the specified input data buffer and to produce meaningful information in notification FIFO entries and possibly one or both of the indicated output data buffers.

The stream context's Activity Code indicates to the Pattern Matcher what activity is to be performed and the type of output that is expected. As described in Table 17-149, the Activity Code indicates that the input data buffer is either to be decompressed and/or scanned, or is to be interpreted as a string (batch) of Pattern Matcher Control commands.

Activity Code values are 32 bits. The upper 8 bits of the code identify the specific processing block to which they apply. The Pattern Matcher block is identified as block 0x01, and flags an exception if an Activity Code is read that identifies another processing block. As well, an unrecognized code in the Activity Code's lower 16 bits causes the same exception to be raised.

A decompression and/or scanning Activity Code indicates that the input data buffer is to be decompressed and/or scanned for pattern matches. The DMA Engine reads the specified data buffer from system memory, passes it through the Pattern Matcher block, and writes any resulting result data into the specified result buffer. If residue is involved, the DMA Engine prefixes the residue to the data prior to scanning and updates the residue once all of the scan data has been read. If the Activity Code instructs the Pattern Matcher to recover decompressed input data, then the DMA Engine writes the decompressed data into the deflate data buffer provided, or into a buffer from the channel's free buffer list if a deflate buffer was not supplied. Note that for decompression, the entire compressed data unit must be conveyed to the Pattern Matcher in one and only one work unit (there is one work unit per Command FIFO entry). In other words, the compressed data can not be interleaved between several work units.

A Pattern Matcher Control Activity Code indicates that the DMA Engine is to read and interpret the specified input data buffer as a list of Pattern Matcher Control commands. Pattern Matcher Control commands consist of reads and writes to Pattern Matcher data structures such as the trigger tables, confidence table, pattern memory, or stateful rule context.

The following Pattern Matcher Control commands are supported by the DMA Engine:

- Read Table Entry. This command allows software to read the entries in any of the Pattern Matcher tables. Table 17-150 details the format of the Read Table Entry Command.
- Write Table Entry. This command allows software to write entries in the internal trigger tables, internal confidence table, DXE table, and SRE table. Table 17-152 details the format of the Write Table Entry Command.
- Clear Session Context by Session ID. Table 17-153 details the format of the Clear Session Context by Session ID Command.

17.4.5.2.2 Command FIFO Stream ID

The Stream ID value in Command FIFO entries is either a full memory pointer to a stream context entry or an index into a contiguous table of stream context entries. In either case the stream context entries exist in system memory but need not be directly accessible by software.

The information present in the stream context includes:

- Result Stream ID—This is the “stream” the DMA Engine presents reports on for commands specifying this stream ID.
- Deflate Stream ID—This is the “stream” the DMA Engine presents decompressed data on for commands specifying this stream ID.
- Activity Code—This is the kind of activity that is to be performed on input data.
- Sequence counter—A stream byte counter, used by stateful rule matching. The DMA Engine increments the sequence counter as data bytes are fully processed on this stream context.
- Session ID—Used by stateful rule matching as an index into a contiguous table of session context entries called the SRE Context table. Valid range is 0 to 131.071 million. Session ID “0” has a special meaning where the session is considered active only during the processing of the SUI. Once the processing of the SUI is complete, the context (of session ID 0) is automatically cleared.
- Pattern set and subset—Used by pattern-matching.
- Residue length—How much pattern-matching residue is being held. The DMA Engine manages residue on behalf of the pattern matching logic.
- Residue ID—An index when in tabular mode, otherwise a memory pointer. ‘Residue length’ bytes of residue data are written and read to this location as the DMA Engine processed input data. Not all stream contexts used for pattern matching necessarily requires or have residue context; and residue storage is kept distinct from stream context to allow for flexibility in allocating resources.

When operating in tabular mode, memory areas for stream contexts and residues are provided, during initialization, in the form of tables of configurable (but fixed) sizes—run-time allocation of these entities amounts to assigning unused table indexes.

When not operating in tabular mode, the software may implement an allocation scheme of its own choosing - assigning residue to a stream context involves setting an address instead of an index. In this case, the DMA Engine only supports Stream ID values that are 32 byte block aligned (the bottom 5 address bits must be 0x0).

17.4.5.2.3 Command FIFO Input Data Descriptor

Input data may be described as either a linked list of memory buffers or as a scatter/gather chain of descriptor tables. In both cases the DMA Engine is capable of traversing the buffer structure autonomously in order to consume the data.

Driver software can direct the DMA Engine to deallocate all inputs buffers from a command FIFO entry to one of the channel's two free buffer lists upon completion of data consumption. This feature is intended as an offload capability that saves driver software from having to perform memory deallocation upon command completion. This feature is restricted in that all of the input buffers in the linked list or scatter gather chain must match the free buffer list's buffer size and must be 32 byte block aligned.

The supported input structure formats include both the physical and virtual addresses for individual buffers. The DMA uses only physical addresses. The virtual addresses are reserved for the software to track the buffer (for example, store the actual virtual/logical address of the buffer). The DMA does not make use of the virtual address but maintains it so that this information is returned to software when the buffers are returned to software. The software would then use the returned virtual address to access the buffer and thus avoiding the cost of physical to virtual address translation.

Input Data Linked List Structure

Figure 17-141 illustrates the supported input linked list data structure. The linked list structure is described by the following parameters:

- Length. The total number of data bytes in all the buffers of the linked list structure.
- Head Physical/Virtual Pointer Pair. The physical and virtual addresses of the first byte of the 64 byte next buffer field of the first buffer in the linked list chain. The physical address must be 32 byte block aligned.
- Buffer Data Offset. The position of the first byte of data within the first linked list buffer. A buffer data offset value of 0x0 points to the byte at physical address + 64; such as buffer data offset is relative to the first possible byte of data within the buffer, not the physical buffer address.
- Buffer Data Length. The number of valid data bytes within the first linked list buffer. The maximum Buffer Data Length is 65,536 bytes. Buffers within a chain having a Buffer Data Length of zero bytes are supported.

Every buffer in the linked list chain contains a 64 byte next buffer field located at the 32 byte block aligned physical buffer address. Table 17-144 defines the structural layout of a linked list buffer's Next Buffer Descriptor field.

An input linked list descriptor appearing in the Command FIFO does not include an explicit tail pointer. The DMA Engine relies upon the supplied Length value to determine when the end of the input data buffer has been reached. The DMA Engine does not recognize any sort of null pointer termination in the linked list tail buffer and continues processing until Length bytes have been consumed.

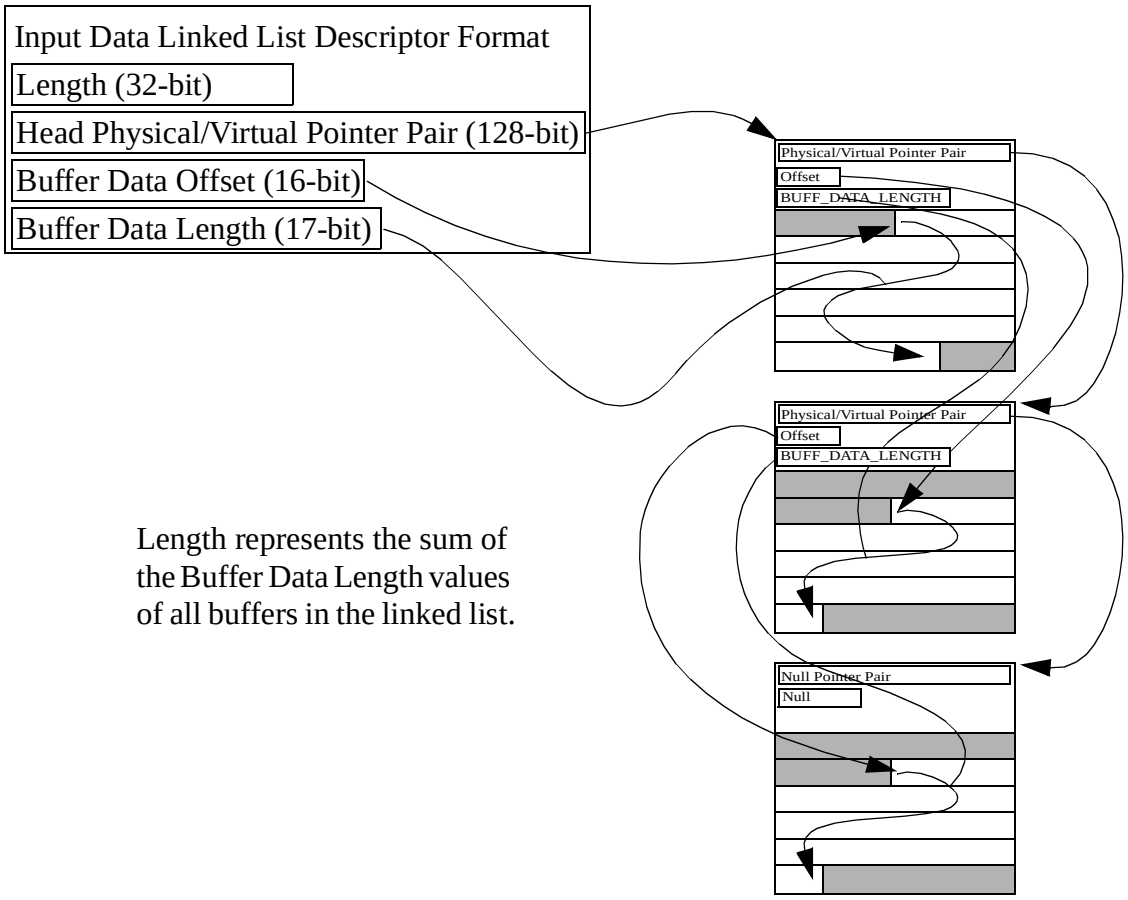


Figure 17-141. Input Data Linked List Structure

Input Data Scatter/Gather Data Structure

Figure 17-142 illustrates the supported input scatter/gather data structure. The scatter/gather structure is compatible with IEEE Std 1212.1-1993™, however, some variations have been introduced that are not compliant with IEEE 1212.1. These are discussed in this section.

The supported scatter/gather input data structure is described by the following parameters:

- Length. The total number of data bytes in all the buffers of the scatter/gather structure.
- Physical/Virtual Pointer Pair. The physical and virtual addresses of the first byte of the first buffer in the scatter/gather chain. The physical address need not be aligned to any particular alignment except if pointing to a scatter/gather table, when it must be 32-bit aligned as per IEEE 1212.1.
- Buffer Data Offset. The position of the first byte of data within the first data buffer. A buffer data offset value of 0x0 points to the byte at physical address + 0; for example, the first possible byte of data within the buffer is at the physical buffer address. This field is not part of the IEEE 1212.1 specification and buffer data offset values other than 0x0 creates an incompatible scatter/gather structure.
- Buffer Data Length. The number of valid data bytes within the data buffer. The maximum Buffer Data Length is 65,536 bytes. Buffers within a chain having a Buffer Data Length of zero bytes are

supported. This definition is not consistent with the IEEE 1212.1 definition of buffer size. Use of non-zero buffer data offset values within a scatter/gather chain creates an incompatible structure.

Table 17-145 defines the structural layout of block vector entries within a scatter/gather table. Note that the buffer data offset field is implemented in the region that is defined as Reserved by the IEEE 1212.1 specification. In order to avoid incompatibility issues when processing scatter/gather structures created by an unknown entity, an IEEE compatibility bit is offered per scatter/gather structure (such as, IEEE compatible bit in Command FIFO entry). When operating in IEEE compatible mode, input data block descriptors are assumed to have a buffer data offset of 0x0; ignoring any non-zero value seen in the reserved region. Likewise when the DMA produces output data into a provided scatter/gather structure, a buffer data offset of 0x0 is presumed when writing data into buffers regardless of any values seen in the block descriptor's reserved region.

In order to maintain the ability to track both physical and virtual address in scatter/gather data structures yet offer compatibility with the IEEE 1212.1 specification the DMA Engine is capable of performing scatter/gather table replication. Table replication involves treating a single scatter/gather table as though it consists of two adjacent tables in memory. One table is written with block descriptors pointing to physical addresses. The other table is written with block descriptors that are identical to the those in the first table, except containing virtual addresses rather than physical addresses. DMA Engine table replication behavior is as follows:

- When processing input scatter/gather chains, the DMA Engine follows physical pointers and consumes data buffers until Length Bytes have been read. The DMA Engine only redirects to a new scatter/gather table if a block descriptor with Extension=1 is encountered. This is consistent with IEEE 1212.1 specified behavior and is independent of whether or not the scatter/gather tables being processed are replicated.
- When writing to pre-constructed scatter/gather structures, the DMA Engine does not need to perform any extra steps when dealing with structures built using table replication because the structures are already fully defined. The one exception to this rule is the last block descriptor of the structure. If the last buffer is not completely filled then the Buffer Data Length of the physical block descriptor (and the virtual one if it exists) does not represent the number of valid bytes in the buffer. This is consistent with IEEE 1212.1 specified behavior, however, it is not consistent with this design's definition of Buffer Data Length.

  Because the DMA Engine is unaware of whether or not the table being populated is a replicated one, it is not safe to automatically adjust the Buffer Data Length field at the location of the presumed virtual scatter/gather table.
- When creating scatter/gather chains using free buffer list buffers the DMA Engine automatically performs table replication, dividing free buffers used as scatter/tables in half as it goes. If the eventual consumer of the structure is unaware of the replication they simply see scatter/gather tables that are half as big as they could be. If they are aware of the replication, then the replica table structure containing virtual addresses provides significant benefit by not requiring address translations as it is traversed.
- When reaping input data scatter/gather structures for return to free buffer list, the DMA Engine behaves as though the tables are replicated, reading a virtual address from the expected location for every physical address it reaps. In this situation, the input data scatter/gather structures must have their scatter/gather tables replicated (such as sufficient buffer space allocated to hold the block

descriptors containing the virtual addresses). If the reaped virtual addresses are meaningless, then it is up to software to ensure that possible consumers of the reaped buffers are not expecting valid virtual addresses in created structures.

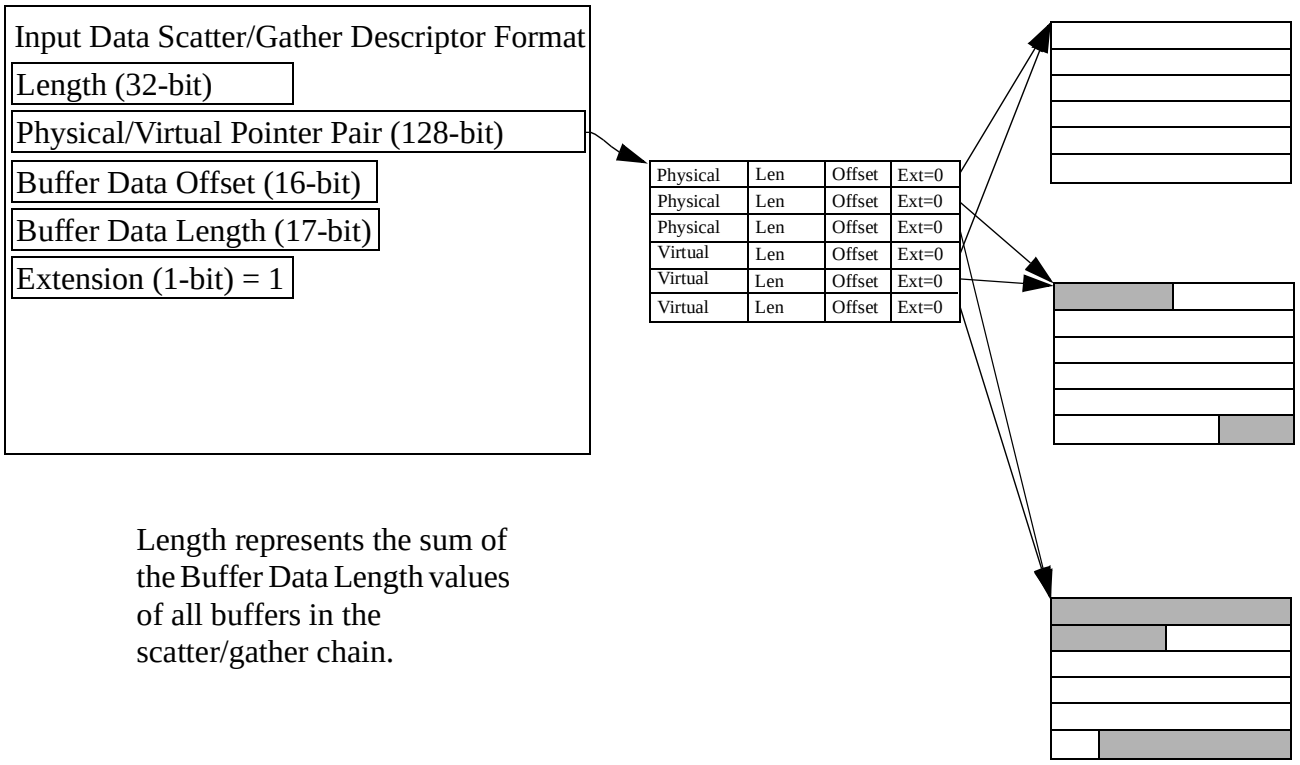


Figure 17-142. Input Data Scatter/Gather Structure

Start Of Stream and End Of Stream

In order to support anchored pattern searches the DMA Engine performs delineation of the start and end of a data stream.

If residue applies to the stream, then the first byte of data received after the residue length is reset to 0 is considered to the be the start of stream for anchored pattern searches. Note that the sequence number does not need to be reset, and a sequence number of 0 is not treated as the start of stream (sequence number may roll over during the lifetime of a stream).

If residue does not apply to the stream, then the first byte of data received after the sequence number is reset to 0 is considered to be the start of stream for anchored pattern searches. The sequence number may subsequently roll over during the lifetime of the stream but this is not considered a new start of stream.

A bit is provided in the Command FIFO Entry that instructs the DMA Engine to indicate an End of Stream to the Pattern Matcher engines. When a command with the End of Stream bit set is executed, the last byte of input data is considered as the end of stream. As well, the DMA Engine automatically resets the stream

context's sequence number and residue length context to zero when an End of Stream is encountered, such that the next data byte on the stream is considered a Start of Stream.

In order to accommodate protocols such as TCP, where it may not be known until later that the last processed byte was in fact the End of Stream byte, Commands with zero length input data but with the End of Stream bit set are allowed. In these cases, the stream's residue is recycled through the Pattern Matcher engines with the new End of Stream indication such that any anchored patterns present won't be missed.

17.4.5.2.4 Command FIFO Result and Deflate Data Descriptors

The Command FIFO entry contains fields that allow driver software full flexibility in specifying where any produced pattern matcher result data or decompressed deflate data should be written.

Two pre-constructed scatter/gather output buffer descriptors can be supplied in a Command FIFO entry when 128 byte sized command FIFO entries are being used. When 64 byte size entries are being used, only one output buffer descriptor can be supplied for either result or deflate data and the other source of data must use free buffer list buffers. Pre-constructed linked list output buffers are not supported.

The two output buffer specifiers are referred to generically as "Output X" and "Output Y". Controls are provided that allow per-command specification of which output stream (result or deflate) goes to each output. As mentioned, "Output X" and "Output Y" can be buffered in supplied, pre-constructed scatter/gather buffers. In addition, driver software can specify one of two DMA Engine free buffers lists for use by each output stream. The same free buffer list can be used for both output streams, if desired. Controls are provided that specify whether the DMA Engine is to create linked list or scatter/gather data structures using free buffer list buffers.

17.4.5.2.5 Command FIFO Modifier Bits

The DMA Engine offers several Command modifier bits that control certain aspects of command behavior. These are detailed in Table 17-146.

17.4.5.2.6 Command FIFO Pass Through Context

As an assist to driver software, the DMA Engine provides a 16 byte pass through context region in Command FIFO entries. The contents of this region are echoed in Notification FIFO entries created as a result of Command completion.

17.4.5.3 Notification FIFO Entries

The DMA Engine normally produces two Notification FIFO entries per Command FIFO entry. The first notification indicates the results of the decompression operation (if any) and points to recovered decompressed data (if any). The second notification points to the pattern matching result data (if any). Two notifications are also produced for commands where the indicated stream context Activity Code is a Pattern Matcher Control code.

The two notifications are produced in order with respect to each other but not necessarily with respect to notification pairs from other commands (interleaving may occur). Both notifications are produced regardless of which operations the SUI in question is undergoing.

If desired, driver software can independently suppress one or both notifications on a per-command basis. This may be useful in situations where it is known that one or both notifications contain no meaningful information; and also are not required for any software completion/callback reasons.

An entry in the Notification FIFO specifies the following information:

- Stream ID.
- Output Data Descriptor.
- Notification FIFO Exception Code and truncation indication.
- Pass through data.

17.4.5.3.1 Notification FIFO Stream ID

The Stream ID value appearing in Notification FIFO entries amounts to a 64-bit token value copied from the stream context referenced by the command that resulted in the Notification FIFO entry. The stream context contains independent stream ID's for both pattern matching result output and decompressed deflate data output. The stream ID appearing here can thus be used by driver software to determine which of the two output stream it has received, although the notifications are produced in a deterministic order.

17.4.5.3.2 Notification FIFO Output Data Descriptor

Notification FIFO entries contain either a linked list descriptor, a scatter/gather block vector type descriptor, or no data descriptor at all. NOP and Stream Context Update commands result in Notification FIFO entries that indicate an invalid output data buffer source because no data is referenced. Stream Context Driven commands indicate one of the data source types: supplied, Free Buffer List A, or Free Buffer List B. If a descriptor is present, the "Output Format" field indicates which of the two are present. In addition, the "Output Source" notification entry field indicates the source of the output data buffer (supplied, channel Free Buffer List A, channel Free Buffer List B, or no buffer present).

Notification FIFO Linked List Output Data Descriptor

The linked list descriptor present in Notification FIFO entries describes an identical structure to that detailed in Input Data Linked List Structure section on page 17-164. However, the descriptor fields are somewhat different. The following fields are present in the Notification FIFO linked list descriptor:

- Output Length. This field has the same meaning as in Input Data Linked List Structure section on page 17-164.
- Head Physical/Virtual Pointer Pair. This field has the same meaning as in Input Data Linked List Structure section on page 17-164.
- Tail Virtual Pointer. This is the virtual address of the tail block in the linked list. Driver software likely manages output data units in a linked list fashion and it is useful to know the virtual address of the tail block for linking operations.
- Output Buffer Count. This field indicates how many free buffer list buffers are present in the linked list. Like the tail pointer, it is useful for list linking and management. Also, when driver software deallocates these buffers back to the DMA Engine channel's free buffer list at a later time the buffer count is a required field in the Free Buffer Deallocate FIFO entry.

- Output Offset. This field has the same meaning as in Input Data Linked List Structure in Section , “Input Data Linked List Structure.” For buffers produced out of the free buffer list, the Output Offset field is set to 0.
- Output Buffer Length. This field has the same meaning as in Input Data Linked List Structure in Section , “Input Data Linked List Structure,” however, the DMA Engine never produces a linked list containing empty buffers.

Notification FIFO Scatter/Gather Output Data Descriptor

The scatter/gather descriptor present in Notification FIFO Entries describes an identical structure to that detailed in Input Data Scatter/Gather Data Structure in Section , “Input Data Scatter/Gather Data Structure,” except in the Notification FIFO entry the virtual address of the data buffer or scatter/gather list is present as well as the physical address. Also note that for buffers produced out of the free buffer list, the Output Offset field is set to 0.

17.4.5.3.3 Notification FIFO Exception Code and Truncation Indication

The Notification FIFO Exception Code indicates to driver software whether any exceptions were encountered during processing of the associated work unit. Refer to Section Table 17-156., “Notification FIFO Exception Code Enumeration,” for an explanation of the possible exception codes.

In addition to the exception code, the notification FIFO entry also indicates whether or not the output data associated with the notification was truncated due to insufficient buffer space. This can occur for two reasons:

- The software supplied output scatter/gather buffer was not large enough to contain all of the produced output.
- The DMA Engine channel’s free buffer list became exhausted while the DMA Engine was buffering output data.

17.4.5.3.4 Notification FIFO Pass Through Context

Pass through context from the originating command is echoed in the notification FIFO entry. Refer to Section 17.4.5.2.6, “Command FIFO Pass Through Context.”

17.4.5.4 Interrupts

The DMA Engine is responsible for creating and managing interrupts sent to the CPU.

The Pattern Matcher block supports five separate interrupt signals (wires) corresponding to the four separate command/notification channels (Pattern Matcher DMA Channel 0–3 interrupt signals), plus a Pattern Matcher Common Control interrupt signal.

Under normal mission mode conditions there are three main sources of interrupt that occur (per-channel):

- Notification FIFO work available. There are unprocessed entries (NPI not equal to NCI) in the Notification FIFO available for the driver to consume. If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA

Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed unless the following two conditions are true:

— The Command FIFO entry that caused the creation of this Notification FIFO entry had the Interrupt on Notification bit set.

— The Notification FIFO fill level has become greater than the programmed threshold in the NFIFO_THRES0 register.

- Command FIFO work available. There are processed command entries (CCI not equal to CEI) in the Command FIFO available for the driver to expire. If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed unless the following two conditions are true:

  — The Command FIFO entry has the Interrupt On Completion bit set.

  — The Command FIFO fill level has become greater than the programmed threshold in the CFIFO_THRES0 register.

- Free Buffer Deallocate FIFO. This interrupt source becomes valid when the Free Buffer Deallocate FIFO fill level has become less than the programmed threshold in the FBFIFO_THRES0 register.

In addition to these two main sources, the DMA Engine also offers the following per-channel interrupt sources:

- Free Buffer List Depletion. These interrupt sources become valid when the channel's free buffer List number of buffers becomes less than or equal to a programmed threshold. If this status bit becomes asserted and the prescribed interval in the IICR0 register since the previous assertion of the Pattern Matcher DMA Channel interrupt signal has not elapsed, the Pattern Matcher DMA Channel interrupt signal is not asserted until the prescribed interval has elapsed.
- Common Control Error. This interrupt source indicates a common-control error has been detected that caused processing to gracefully halt on all channels. This interrupt source is used to inform per-channel driver software that processing on this channel has stopped due to a global error condition. Any work units that were in flight at the time the error was detected, and were affected by the error, have the appropriate error code set in their Notification FIFO entry.
- Error. This interrupt source indicates a per-channel error has been detected that caused processing to gracefully halt on this channel. Any work units that were in flight at the time the error was detected, and were affected by the error, have the appropriate error code set in their Notification FIFO entry. These are considered channel errors. See Section 17.4.5.6.2, "Channel Errors," for a more detailed description on channel errors.

The per-channel interrupts are level triggered. The interrupt source appears valid as long as the condition is true. That is, clearing these interrupt status bits while their conditions are still true, cause the interrupt status bits to be reasserted. For the above two error conditions, the condition remains true until a channel reset is issued.

The Pattern Matcher Common Control interrupt signal. is used to pass errors that cannot be associated to a particular channel. These errors are:

- System memory bus write errors. Since write transactions are posted in MIA, there is no way to determine the channel associated with the failed write. When this error is detected the DMA engine

gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. These are considered common-control errors. See Section 17.4.5.6.3, “Common-Control Errors,” for a more detailed description on common-control errors.

- DMA Engine Pattern Matching Control Error. This interrupt source indicates an unrecognizable Pattern Matcher Control command or an ill-formed pattern matching data structure was encountered. When this error is detected the DMA engine gracefully halts processing on all channels, allowing all in-flight work to complete prior to interrupting. These are considered common-control errors. See Section 17.4.5.6.3, “Common-Control Errors,” for a more detailed description on common-control errors.
- DMA Engine Internal Memory Parity Error. This interrupt source indicates that a parity error was detected by the DMA Engine during a read from its internal context memory. A parity error on this specific internal memory is treated as a common-control error due to the fact that corruption of certain information stored in this memory may impact other channels. See Section 17.4.5.6.3, “Common-Control Errors,” for a more detailed description on common-control errors.

The Pattern Matcher Common Control interrupts are event triggered.

In addition to the standard status and interrupt enable registers the Pattern Matcher provides two additional interrupt control registers: Interrupt Interval Control register and Interrupt Inhibit register.

The IICR register provides a means of throttling the rate at which a Pattern Matcher DMA Channel interrupt signal can be asserted for Command FIFO, Notification FIFO or Free Buffer Deallocate FIFO interrupt sources. The IICR setting does not affect the setting of any bits in the status register, rather it simply guarantees a minimum interval between assertions of the interrupt signal. If new interrupt sources become present in the status register and the prescribed interval since the previous assertion of the interrupt signal has not elapsed, the interrupt signal does not assert until the prescribed interval has elapsed.

The Interrupt Inhibit Register (IIR) provides software with a means of globally inhibiting the Pattern Matcher block’s DMA Channel and Common Control interrupt signals without modifying the interrupt enable or source disable register. This is useful for increasing interrupt coalescing as the following example illustrates:

1. An interrupt asserts and the Interrupt Service Routine is invoked.
2. The Interrupt Service Routine collects all active interrupt bits from the status register.
3. The Interrupt Service Routine inhibits further interrupts on this signal with inhibit register.
4. The Interrupt Service Routine schedules the interrupt task to run at the next opportunity.
5. The Interrupt Service Routine exits.

At this point the task may execute immediately, or may be pushed off for some time if the system is under load. In the case where the task execution is delayed, the amount of work to be done for the collected interrupt sources may increase (such as the FIFO’s may get fuller). Also, additional sources of interrupt may arrive in the status register (such as the other FIFO may collect some work if it had none at the time of interrupt).

When the task executes, the following steps are performed by it:

1. The interrupt status register is read to determine all sources of work. This read returns the already known but uncleared bits and possibly some new bits.

2. The task services to exhaustion all sources of interrupt.
3. The task might decide to re-read the status register prior to exiting in case further work has arrived during execution.
4. Having exhausted all known work, the task uninhibits this interrupt signal and exits.

The advantages of having the interrupt inhibit capability are obvious in a system where interrupt processing is decoupled from the Interrupt Service Routine itself. As CPU load increases, the amount of work that is available when the task executes increases, resulting in better system efficiency.

17.4.5.5 Free Buffer Deallocate FIFO

Although the FBM is responsible for management of free buffer lists, the Free Buffer Deallocate FIFO mechanism used to deallocate free buffers to the FBM is implemented in the DMA Engine. The use of a buffer deallocation mechanism that is independent of the Command FIFO is necessary to avoid blocking situations within the DMA Engine. An example of this would see the DMA Engine cease servicing of a particular channel's Command FIFO due to free buffer list depletion. If the Command FIFO were the only means of re-populating the free list then this channel would be permanently blocked because software would have no other means of replenishing the free list.

The Free Buffer Deallocate FIFO is similar in concept to the Command FIFO. Driver software populates a formatted FIFO entry at the location indicated by PI and then publishes an incremented PI to the DMA Engine's internal register space. The DMA Engine reads the FIFO Entry, which contains a descriptor for either a linked list or a scatter/gather chain of buffers, and publishes an incremented value of CI via the DMA Engine block's internal register space.

Table 17-158 details the format of Free Buffer Deallocate FIFO linked list entries.

Table 17-159 details the format of Free Buffer Deallocate FIFO scatter/gather entries. When deallocating scatter/gather chains, the DMA Engine walks the scatter/gather table chain until it finds a table that does not have the EXT bit set in the last entry.

The "Offset" field and "Buffer Data Length" field in block descriptors are ignored. When a buffer is consumed from a free buffer list the "Offset" field is reset 0x0 and the "Buffer Data Length" field is set to the programmed size value for that specific free buffer list.

The Free Buffer Deallocate FIFO mechanism is completely independent of the Command FIFO mechanism such that Free Buffer Deallocate FIFO operations are never blocked by pending Command FIFO operations. This allows driver software to replenish a channel's free buffer list even when the DMA Engine has stopped servicing the command FIFO due to the free buffer depletion condition.

In addition to using the Free Buffer Deallocate FIFO, driver software can also specify that input data buffers provided in Command FIFO Entries be automatically deallocated to a specified channel free buffer list once they have been consumed by the DMA Engine.

If an exception, such as a system memory read error, is encountered during Free Buffer Deallocate FIFO (FBDF) operations, the DMA Engine does not cease servicing of the FBDF. This behavior differs from the DMA Engine Command FIFO response, where servicing of the Command FIFO gracefully ceases following an exception. The DMA Engine, however, does not allow free buffers to be deallocated to the FBM once an FBDF exception has been detected. This response has the potential to create memory leaks

in the system because the FBDF entry appears to have been consumed and processed by the DMA Engine. However, because the free list may be shared by other channels, the DMA Engine must protect the integrity of the free list by not polluting it with potentially corrupted buffers.

17.4.5.6 Error Handling

There are three types of errors that can be generated:

- No-halt errors
- Channel errors
- Common-control errors

No-halt errors are localized to the work unit itself and as a result dpes not cause the Pattern Matcher to halt operation. Channel and common-control errors are more severe, causing the Pattern Matcher to halt operations—on a specific channel in the case of channel errors, and on all channels for common-control errors.

17.4.5.6.1 No-Halt Errors

No-halt errors are constrained to an individual work unit. An example of such an error is a Deflate CRC failure, which indicates that the CRC, as recorded in the GZIP or ZLIB trailer, did not match the computed value at the conclusion of decompression operation. When the Pattern Matcher encounters such an error, it gracefullys cease operation on the associated work unit, report the error as a Notification FIFO exception in the Notification Entry produced for that work unit and proceed normally to the next work unit. There are no interrupts generated for these errors.

In the case where the command associated with the work unit produces two notification FIFO entries, the error is reported in both notification FIFO entries unless one or both of them encounter a higher precedence error as they are processed down the pipeline. Furthermore, both Notification FIFO entries may contain truncated results (recovered decompressed data and/or pattern matching results) produced prior to the detection of the error.

17.4.5.6.2 Channel Errors

A channel error is localized to the channel in which it occurs, causing processing on that channel to be halted in a graceful manner. In-flight work (work in the pipeline) is allowed to complete to ensure software is able to unwind, no new commands are consumed, and notifications for work units already in the pipeline affected by the error are tagged with an appropriate Notification FIFO exception code. Once the pipeline has been flushed, the Pattern Matcher sets the Error bit in the channel's Interrupt Status register. The driver is assured, once it sees this error bit, that processing of the channel has ceased and any consumed commands have had any corresponding notifications already produced by the Pattern Matcher—for example, it is safe to cleanup and/or reset the channel.

When the Pattern Matcher encounters a channel error, it ceases operation on the current work unit (no attempt is made to complete the processing on that work unit), report the error as a Notification FIFO exception in the Notification FIFO entry produced for that work unit and proceed normally to the next work unit in the pipeline.

In the case where the command associated with the work unit produces two Notification FIFO entries, the error is reported in one or both of the Notification FIFO entries (depends on where the error is detected in the pipeline) unless one or both of them encounter a higher precedence error or an earlier error of the same precedence as they are processed through the pipeline.

Furthermore, both Notification FIFO entries may contain truncated results (recovered decompressed data and/or pattern matching results) produced prior to the detection of the error. If the incomplete output (however intact) is produced using blocks from the free buffer list then it is the responsibility of the software to recycle these blocks.

If multiple channel and/or common-control errors occurred on a work unit then the Notification FIFO exception code with the highest precedence is reported, regardless of the order in which the errors were encountered. If within the same work unit there are multiple channel and/or common-control errors detected of the same precedence, then the first error encountered in the pipeline is reported. Although only one Notification FIFO exception is reported, interrupts from both error types are generated, and as a result, the condition that both types of errors occurred is not lost.

Only the work units who encounter a channel error are tagged unless the error condition is such that the results of subsequent commands in the pipeline can’t be trusted. The latter is referred to as a “sticky” error condition in which any work units in the pipeline following the sticky error detection is tagged with the same exception code as the initial errored work unit.

In addition to placing an error code into the affected Notification FIFO entry, the DMA Engine also updates the appropriate CHERR[0..3] register with error code. This is done to trap the error code in cases where a Notification FIFO may not be available to put the error code into, such as Free Buffer Deallocate FIFO exceptions.

Recovery from a channel error can be achieved by issuing a channel reset on the channel. A channel reset also clears the channel's interrupt status register. Take note that the status bits (for example, Error bit) in the channel Interrupt Status register are level triggered. The interrupt sources appear valid as long as the conditions are true. That is, clearing these interrupt status bits while their conditions are still true causes the interrupt status bits to be reasserted. When an Error condition is detected, it remains true until a channel reset is issued on the channel. See Section 17.3.8.17, “Channel Reset and Control Register Channel 0 (CRCR0),” for a description on how to reset a channel.

17.4.5.6.3 Common-Control Errors

Common-control errors are errors that either couldn't be contextualized to a particular channel (for example, memory write errors) or that require all channels be suspended due to the error having a (potentially) global impact (for example, corruption of the global pattern matching database).

A common-control error causes processing to be halted on all channels in a graceful manner. Channel operation is halted in the same fashion as for channel errors;

- in-flight work (work in the pipeline) is allowed to complete to ensure software is able to unwind,
- no new commands is consumed, and
- notifications for work units already in the pipeline affected by the error are tagged with an appropriate Notification FIFO exception code.

Once the entire pipeline for all channels has been flushed, the Pattern Matcher sets the corresponding common-control error bit in the ISRCC register and the Common Control Error bit in every channel Interrupt Status registers. Take note that there is a separate interrupt status bit for every source of memory write errors whereas the remaining common-control errors (for example, pattern matching database corruption, unrecognizable pattern matcher control command) are represented by a common interrupt status bit.

As in the case for channel errors;

- Pattern Matcher ceases operation on the current work unit (no attempt is made to complete the processing on that work unit) when a common-control error is detected,
- the error is reported as a Notification FIFO exception in the Notification FIFO entry produced for that work unit, and
- Pattern Matcher proceeds normally to the next work unit in the pipeline. Notification FIFO entries are produced in the same manner as for channel errors. See Section 17.4.5.6.2, “Channel Errors,” for a more detailed description on Notification FIFO entries generation when errors are encountered.

As in the channel error case, the DMA Engine also updates the CHERR[0..3] registers with the error code to haPattern Match Event Meta Datandle exceptions for which a Notification FIFO Entry may not be available to carry the exception code.

Recovery from a common-control error can be achieved by first clearing the common-control status interrupt followed by issuing a channel reset on each channel.

Take note that status bits in the interrupt status register Common Control register are event triggered whereas status bits (for example, Error and Common Control interrupts) in the channel Interrupt Status register are level triggered (interrupts appear valid as long as the condition is true). Clearing the channel interrupt status bits while their condition are still true, causes the interrupt status bits to be reasserted. When a Common Control error condition is detected, it remains true until a channel reset is issued on the channel. See Section 17.3.8.17, “Channel Reset and Control Register Channel 0 (CRCR0),” for a description on how to reset a channel.

A channel is not able to enter reset while a common-control error remains asserted in the ISRCC register. As long as the common-control error remains asserted, any channel that was already in reset when the common-control error was detected has the Common Control Error bit asserted in its channel interrupt status register when it leaves reset. In this way, handling of common-control errors must take place before channels are able to reset and any potential race conditions between Common Control exceptions and channel driver software are avoided. Channel driver software is responsible for reading the channel’s CRCR register after any attempt to put the channel into reset in order to verify that the write and its associated reset action were successful.

17.4.5.7 Channel Scheduling

Scheduling of service for the DMA work channels is programmable via the CSCR register in the Pattern Matcher Register Space. Table 17-157 details the available scheduling settings. This scheduling setting can be overwritten for a particular DMA channel by designated it as a high priority channel. The channel

designated as high priority is serviced before all others regardless of the configured channel scheduling algorithm.

17.4.5.8 Cache Awareness

The DMA Engine provides controls that allow for cache aware operation in order to improve system performance. Cache aware operation means applying one of the following attributes to read or write transactions on the SoC system bus:

- Snoop. Without cache awareness the DMA Engine would have to indicate that all transactions must be snooped. The DMA Engine can be configured such that some memory read/write transactions indicate that the transaction does not need to be snooped. This is done to save the overhead of snooping in cases where it is known that the data is not resident in any cache.
- Allocate. Without cache awareness the DMA Engine would only be aware that it is writing into system memory. The DMA Engine can be configured such that some memory write transactions are placed into cache. This is done to pre-position the data in cache when it is known that the data is consumed in the near future.

The memory spaces and the controls provided for each are as follows:

- Command FIFO: Snoop. The DMA Engine should only ever read Command FIFO locations that the CPU has recently visited, therefore it is likely that the Command FIFO location being read is resident in cache. In this scenario, the DMA Engine should always indicate that Command FIFO reads be snooped. The DMA Engine never writes to the Command FIFO.
- Notification FIFO: Allocate. The driver should only ever read Notification FIFO locations that the DMA Engine has recently visited, therefore the Notification FIFO location about to be read can be positioned in cache to improve driver code performance. The DMA Engine never reads from the Notification FIFO.
- Free Buffer Deallocate FIFO: Snoop. The DMA Engine should only ever read Free Buffer Deallocate FIFO locations that the CPU has recently visited, therefore it is likely that the Free Buffer Deallocate FIFO location being read is resident in cache. In this scenario, the DMA Engine should always indicate that Free Buffer Deallocate FIFO reads be snooped. The DMA Engine never writes to the Free Buffer Deallocate FIFO.
- Stream context: Snoop and Allocate. Stream context entries are read and written exclusively by the DMA Engine. Usually the stream context entry indicated by a Command FIFO entry is read-modified-written by the DMA Engine during servicing of that entry. It may be desirable therefore to have the DMA Engine both Snoop and Allocate during operation to optimize read latency when the number of stream contexts being visited is small.
- Residue context: Snoop and Allocate. Similar to stream context, this location is read-modified-written exclusively by the DMA Engine.
- Input data buffer: Snoop. This buffer is only ever read by the DMA Engine. In certain applications the data being interpreted by the DMA Engine (whether a control string or data to be scanned) may be known to not be resident in cache, or it may not be important that the modified portion in cache be scanned (L2/L3 header stashed and modified but payload untouched). In these applications snooping could be disabled to improve performance.

- Result data buffer: Allocate. This buffer is only written by the DMA Engine. It may be desirable to have result data positioned in cache.
- Deflate data buffer: Allocate. This buffer is only written by the DMA Engine. It may be desirable to have deflate data positioned in cache.
- SRE entries: Snoop and Allocate. A portion of the SRE entry is used as a state digest vector which is read and written by the SRE. It may be desirable to have this portion of session context snooped and allocated to realize a performance gain.

17.4.5.9 System Memory Transaction Priority

The DMA Engine allows configuration of the priority level used for system memory transactions. The priority can be programmed independently via a control register for each of the following locations on a per-channel basis:

- Command and Notification FIFO pair
- Stream Context
- Residue Context
- Command Data Buffer
- Result Data Buffer
- Deflate Data Buffer
- Free Buffer Deallocate FIFO

Each transaction type can be programmed to use one of four priority levels on the SoC system bus. Priority setting 0 (default) is the lowest priority setting.

This capability is intended for in-system performance tuning and its use varies by application.

17.4.5.10 DMA Engine Data Structures

Table 17-144. Linked List Next Buffer Descriptor Format

Offset	Bits	Name	Description
Offset + 0	0–63	Next Buffer Descriptor Physical Pointer	The physical address of the Next Buffer Descriptor of the next buffer in the linked list chain.
Offset + 8	0–63	Next Buffer Descriptor Virtual Pointer	The virtual address of the Next Buffer Descriptor of the next buffer in the linked list chain.
Offset + 16	0–15	—	Reserved
	16–31	Next Buffer Descriptor Offset	The offset of the first data byte in the next buffer in the linked list chain.
Offset + 20	0–31	Next Buffer Descriptor Buffer Data Length	The number of valid bytes in the next buffer in the linked list chain.

Table 17-145. Scatter/Gather Block Descriptor Format

Offset	Bits	Name	Description
Offset + 0	0–63	Pointer	The address of a buffer or scatter/gather table, depending on “Extension”.
Offset + 8	0–31	Buffer Data Length	The number of valid bytes in the buffer or scatter/gather table at “Pointer”.
Offset + 12	0	Extension	Extension. Meaning: • 0x0 “Pointer” field points to a contiguous data buffer. • 0x1 “Pointer” field points to a scatter/gather table.
—	1–14	—	Reserved
—	15–31	Offset	The offset of the first data byte in the buffer or scatter/gather table at “Pointer” field.

Table 17-146. Command FIFO Entry Fields

Offset	Bits	Name	Description
Offset + 0	0–63	Stream ID	Identifies the 64 byte stream context entry that this command pertains to. Depending on the channel’s Stream Context Configuration register setting, “Stream Id” represents either a pointer (memory address) or an index into a table of stream contexts. If this is a pointer, it must be 32 byte block aligned. If “Command Code” indicates NOP, then this value is echoed in the corresponding Notification FIFO entry.
Offset + 8	0–63	Input Physical Pointer	The physical memory address of the associated data buffer, linked-list head buffer, or head scatter/gather table.
Offset + 16	0–63	Input Virtual Pointer	The virtual memory address of the desired result buffer, linked-list head buffer, or head scatter/gather table.
Offset + 24	0–63	Output X Physical Pointer	The physical memory address of the desired output buffer or head scatter/gather table.
Offset + 32	0–31	Input Length	A number in the range 0 to 4GB-1 that represents the total number of data bytes contained in the specified input buffer, linked-list, or scatter/gather chain.
Offset + 36	0–15	Input Offset	A number in the range 0 to 64KB-1 that represents the location of the first byte of input data relative to “Input Physical Pointer”.
	16–31	Output X Offset	A number in the range 0 to 64KB-1 that represents the location of the first byte of output data to be written relative to “Output X Physical Pointer”. This field is only valid when “Output X Source” indicates that the output data bound for “Output X” is to be directed into the specified “Output X” buffer and not into buffers from the free buffer list.

Table 17-146. Command FIFO Entry Fields (continued)

Offset	Bits	Name	Description
Offset + 40	0–1	Input Treatment	A coded value specifying what the DMA Engine is to do with the specified input buffer, linked-list, or scatter/gather chain upon consuming the data from it. Codings: • 0x0 Do nothing. • 0x1 Deallocate buffer(s) to Free Buffer List A. • 0x2 Deallocate buffer(s) to Free Buffer List B. • 0x3 Reserved
—	2	Input Extension	Determines whether to jump to a chain of scatter/gather tables when referencing the “Input Physical Pointer” field. If “Input Format” indicates LL, then “Input Extension” is a don’t care and the input buffer is treated as a linked list of buffers. If “Input Format” indicates BV, then “Input Extension” has the following meaning: • 0x0 The “Input Physical Pointer” field points to a contiguous data buffer. • 0x1 The “Input Physical Pointer” field points to a chain of scatter/gather tables.
—	3	Input Format	This coded field indicates what descriptor format the input data buffer is described with. Codings: • 0x0 Block Vector (BV) • 0x1 Linked List (LL)
—	4	Input IEEE	IEEE Compliant. This bit, when set, forces the DMA Engine to ignore any set bits in the reserved region of IEEE 1212.1 format block descriptors when reading data from input scatter/gather buffers. This allows the DMA Engine to accept scatter/gather buffers of unknown origin and safely process them if the originator has set any bits in the reserved region.
—	5	Output X IEEE	IEEE Compliant. This bit, when set, forces the DMA Engine to ignore any set bits in the reserved region of IEEE 1212.1 format block descriptors when writing data to output scatter/gather buffers. This allows the DMA Engine to accept scatter/gather buffers of unknown origin and safely process them if the originator has set any bits in the reserved region.
—	6	Input End Of Stream	End of Stream. When set, this bit indicates to the DMA Engine that the last scanned data byte of this command is to be considered an “end of stream” for anchored search pattern matching purposes.
—	7	Exclusive	Multiple back-to-back FIFO entries that have this bit set (plus a final entry that has this bit cleared) are guaranteed to be processed atomically with respect to other command channels. In other words, once the hardware processes an entry on a channel with Exclusive set, it service sthat same channel exclusively until a subsequent entry is processed that has Exclusive clear Take note that exclusivity mode is released even if there is an error detected while processing the command with the Exclusive clear.

Table 17-146. Command FIFO Entry Fields (continued)

Offset	Bits	Name	Description
—	8	Suppress Result Notification	When set, causes the hardware to NOT create a notification FIFO entry upon completion of this command's scanning operation. This is useful in situations where a notification is not strictly required in order to improve performance; for example, doing a bulk stream context initialization. When set, the DMA Engine discards any output data produced as a result of processing the command.
—	9	Suppress Deflate Notification	When set, causes the hardware to NOT create a notification FIFO entry upon completion of this command's decompression operation. This is useful in situations where a notification is not strictly required in order to improve performance; for example, doing a bulk stream context initialization. When set, the DMA Engine discards any output data produced as a result of processing the command.
—	10	Interrupt On Completion	When set, causes the CPU to be interrupted without delay upon completion of this Command FIFO entry (CI incremented). This would be used to override the minimum interrupt interval timer in cases where time-sensitive commands are being issued and a non-zero minimum interrupt interval setting is used.
—	11	Interrupt On Notification	When set, causes the CPU to be interrupted without delay upon creation of the Notification entry corresponding to this command. In incarnations of this interaction model where notifications and commands are not linked this bit may not have meaning. This would be used to override the minimum interrupt interval timer in cases where time-sensitive commands are being issued and a non-zero minimum interrupt interval setting is used.
—	12	Output Assignment	This field determines which output data source is directed into the “Output X” buffer. Codings: • 0x0 Pattern Matcher result output, if any, is directed into the “Output X” buffer. Deflate output data, if any, is directed into the “Output Y” buffer. • 0x1 Pattern Matcher result output, if any, is directed into the “Output Y” buffer. Deflate output data, if any, is directed into the “Output X” buffer.
—	13–14	—	Reserved
—	15–31	Input Buffer Data Length	A number in the range 0 to 64KB that represents the number of bytes of input data to be read at “Input Physical Pointer” + “Input Offset”.
Offset + 44	0–7	Command Code	This is an enumerated type indicating to the DMA Engine what type of command this is. Table 17-147 defines the enumerations and their meanings.

Table 17-146. Command FIFO Entry Fields (continued)

Offset	Bits	Name	Description
—	8–9	Output Y Source	This coded field specifies where the DMA Engine is to get the data buffer used to store data for “Output Y”. Codings: • 0x0 This Command entry. • 0x1 Channel Free Buffer List A • 0x2 Channel Free Buffer List B • 0x3 Null, no buffer is being specified. The Null code point should only be used in commands where there is no possibility of output being produced, such as NOPs.
—	10	Output Y Format	This field indicates what descriptor format the buffer is to be described with. Codings: 0x0 Block Vector (BV) 0x1 Linked List (LL) Note that LL format is only supported when Output Y Source is set to Free Buffer List A or B. Pre-constructed Linked List output buffers are not supported.
—	11	Output X Extension	Determines whether to jump to a chain of scatter/gather tables when referencing “Output X Pointer”. Codings: • 0x0 The “Output X Physical Pointer” field points to a contiguous data buffer. • 0x1 The “Output X Physical Pointer” field points to a chain of scatter/gather tables.
—	12–13	Output X Source	This coded field specifies where the DMA Engine is to get the data buffer used to store data for “Output X”. Codings: • 0x0 This Command entry. • 0x1 Channel Free Buffer List A • 0x2 Channel Free Buffer List B • 0x3 Null, no buffer is being specified. The Null code point should only be used in commands where there is no possibility of output being produced, such as NOPs.
—	14	Output X Format	This field indicates what descriptor format the buffer is to be described with. Codings: 0x0 Block Vector (BV) 0x1 Linked List (LL) Note that LL format is only supported when Output X Source is set to Free Buffer List A or B. Pre-constructed Linked List output buffers are not supported.
—	15–31	Output X Buffer Data Length	A number in the range 0 to 64KB that represents the number of bytes of output data that can be written at “Output X Physical Pointer” + “Output X Offset”. This field is only valid when “Output X Source” indicates that the output data bound for “Output X” is to be directed into the specified “Output X” buffer and not into buffers from the free buffer list. If this field is set to 0, then “Output X Source” must be set to “0x3”.

Table 17-146. Command FIFO Entry Fields (continued)

Offset	Bits	Name	Description
Offset + 48	0–127	Application Specific Pass Through	Use of this location is application specific. Unless a specific application specifies otherwise, the value populated here is echoed in the corresponding notification FIFO entry if this command causes a notification FIFO entry to be created.
Offset + 64	0–63	Output Y Physical Pointer	The physical memory address of the desired output buffer or head scatter/gather table. This field is only valid when “Output Y Source” indicates that the output data bound for “Output Y” is to be directed into the specified “Output Y” buffer and not into buffers from the free buffer list.
Offset + 72	0–15	—	Reserved
—	16–31	Output Y Offset	A number in the range 0 to 64KB-1 that represents the location of the first byte of output data to be written relative to “Output Y Physical Pointer”. This field is only valid when “Output Y Source” indicates that the output data bound for “Output Y” is to be directed into the specified “Output Y” buffer and not into buffers from the free buffer list.
Offset + 76	0	Output Y Extension	Determines whether to jump to a chain of scatter/gather tables when referencing “Output Y Pointer”. Codings: • 0x0 The “Output Y Physical Pointer” field points to a contiguous data buffer. • 0x1 The “Output Y Physical Pointer” field points to a chain of scatter/gather tables. This field is only valid when “Output Y Source” indicates that the output data bound for “Output Y” is to be directed into the specified “Output Y” buffer and not into buffers from the free buffer list.
—	1	Output Y IEEE	IEEE compatible. This bit, when set, forces the DMA Engine to ignore any set bits in the reserved region of IEEE 1212.1 format block descriptors when writing data to output scatter/gather buffers. This allows the DMA Engine to accept scatter/gather buffers of unknown origin and safely process them if the originator has set any bits in the reserved region.
—	2–14	—	Reserved
—	15–31	Output Y Buffer Data Length	A number in the range 0 to 64KB that represents the number of bytes of output data that can be written at “Output Y Physical Pointer” + “Output Y Offset”. This field is only valid when “Output Y Source” indicates that the output data bound for “Output Y” is to be directed into the specified “Output Y” buffer and not into buffers from the free buffer list. If this field is set to 0, then “Output Y Source” must be set to “0x3”.

Table 17-147. Command Code Enumeration

Value	Name	Description
0x0	NOP	No Operation. This command is intended for use as a fence by software. The users may want to know when their commands (issued with notifications suppressed) have been consumed by the Pattern Matcher. This is done by having a NOP command (without notifications suppressed) follow a set of commands issued with notifications suppressed. The NOP notification indicates that the Pattern Matcher has consumed all proceeding commands.
0x1	Stream Context Driven	Instructs the DMA Engine to read and interpret the stream context pointed to by “Stream ID”. This code point implies that a “higher layer” function has provisioned the stream context with sufficient information necessary to carry out the command and thus this lowest layer (the programming interface) can remain agnostic.
0x2	Stream Context Update	Instructs the DMA Engine to read and interpret the data buffer pointed to by “Data Pointer” as a set of {TYPE,VALUE} pairs containing values to be applied to the stream context indicated by “Stream ID”. Table 17-148 defines the TYPE code points and their meanings for the Pattern Matcher block.

Table 17-148. Stream Context Update Type and Value Descriptions

Type	Name	Bytes	Description
0x0104_0000	Activity Code	4	The Activity Code to be written to stream context. This is an enumerated type indicating the activity that is to be performed on the stream data unit. This field in the stream context is updated only at the request of software.
0x0104_0001	Session ID	4	The Session ID to be written to stream context. This field in the stream context is updated only at the request of software. Valid range is 0 to 131.071 million (27-bit value)
0x0100_0002	Residue Length	0	Causes the residue length field in the stream’s context to be set to zero. This field in the stream context is updated by the DMA Engine as work units are processed.
0x0101_0003	Pattern Set	1	The Pattern Set to be written to stream context. This field in the stream context is updated only at the request of software.
0x0102_0004	Pattern Subset	2	The Pattern Subset to be written to stream context. This field in the stream context is updated only at the request of software.
0x0100_0005	Sequence Number	0	Causes the sequence number field in the stream’s context to be set to zero. This field in the stream context is updated by the DMA Engine as work units are processed. The DMA Engine increments the sequence number by the count of bytes fully scanned per work unit. If residue applies, then the Sequence Number does not include bytes held in residue as they have not yet been fully scanned.
0x0104_0006	Result Stream ID High	4	The upper 32 bits of the Result stream ID that is written to Notification Entries. This field in the stream context is updated only at the request of software
0x0104_0007	Result Stream ID Low	4	The lower 32 bits of the Result stream ID value that is written to Notification FIFO Entries. This field in the stream context is updated only at the request of software.

Table 17-148. Stream Context Update Type and Value Descriptions (continued)

Type	Name	Bytes	Description
0x0104_0008	Residue Pointer High	4	The upper 32 bits of the pointer to the residue context for this stream. “Residue Length” indicates how many bytes of residue are present. If “Activity Code” indicates that residue is being used on this stream then the DMA Engine prefixes and update residue. Depending on the channel’s RES_CONFIG register settings, “Residue Pointer” represents either a pointer (memory address) or an index into a table of residues. This field in the stream context is updated only at the request of software.
0x0104_0009	Residue Pointer Low	4	The lower 32 bits of the pointer to the residue context for this stream. This field in the stream context is updated only at the request of software.
0x0104_000A	Deflate Stream ID High	4	The upper 32 bits of the Deflate stream ID value that is written to Notification FIFO Entries. This field in the stream context is updated only at the request of software.
0x0104_000B	Deflate Stream ID Low	4	The lower 32 bits of the Deflate stream ID value that is written to Notification FIFO Entries. This field in the stream context is updated only at the request of software.
0x0101_000C	End Of SUI Event Enable	1	This is a per-stream boolean control that determines whether or not the end of a scanned unit (SUI) is considered an event to the Stateful Rule Engine.
0x0101_0100	SRE Report Control Code	1	This field provides debug control over aspects of the Pattern Matcher’s operation. Valid codings: 0x0 SRE verbose rule reporting off, no special behavior. 0x1 SRE verbose rule reporting mode 1. 0x2 SRE verbose rule reporting mode 2. 0x3 SRE verbose rule reporting mode 3. 0x4-0xFF Reserved

Table 17-149. Stream Context Activity Code Field Definitions

Bits	Field Name	Valid Values	Description
0–7	Version	0x01	Pattern Matcher version. Must be set to 0x1 for this version of the Pattern Matcher block.
8–25	Reserved	0x0	Reserved for future use. Must be set to 0x0 for this version of the Pattern Matcher block.
26–27	Compression Type	0x0	DEFLATE. Scan units processed on this stream are to be decompressed using the DEFLATE algorithm.
		0x1	ZLIB. Scan units processed on this stream are to be decompressed using the ZLIB algorithm.
		0x2	GZIP Scan units processed on this stream are to be decompressed using the GZIP algorithm.
		0x3	PASSTHRU. Scan units processed on this stream are not to be decompressed.
28	Recover	0x0	Scan units processed on this stream, compressed or not, are not to be recovered into the indicated deflate output buffer.
		0x1	Scan units processed on this stream, compressed or not, are to be recovered into the indicated deflate output buffer.

Table 17-149. Stream Context Activity Code Field Definitions (continued)

Bits	Field Name	Valid Values	Description
29	Residue	0x0	Residue bytes are not to be kept by the DMA Engine for scan units processed on this stream.
		0x1	Residue bytes are to be kept by the DMA Engine for scan units processed on this stream.
30–31	Type	0x0	Control. Indicates that scan units processed on this stream are to be treated as batches of Pattern Matcher Control commands by the DMA Engine. All other Stream Context Activity Code fields except the Version field must be set to 0.
		0x1	Scan. Indicates that scan units processed on this stream are to be scanned for pattern matches. Scan units are decompressed according to the algorithm specified in the Compression Type fields prior to scanning.
		0x2	Decompress. Indicates that scan units processed on this stream are to be decompressed according to the algorithm specified in the Compression Type field, but are not to be scanned for pattern matches.
		0x3	Reserved. This code point is treated as being equivalent to Scan (0x1) for this version of the Pattern Matcher block.

Table 17-150. Read Table Entry Pattern Matcher Control Format

Offset	Bits	Name	Description
Offset + 0	0–7	Version	Identifies the version of the Pattern Matcher that this command is intended for. Set to 0x1 for this revision.
—	8–15	Type	An enumerated field indicating what type of Pattern Matcher Control command this is. Codings: • 0x00 Read Table Entry • 0x01 Write Table Entry • 0x08 Clear Session Context by Session ID
	16–31	Reserved	—
Offset + 4	0–31	Length	The length of the overall instruction including the type, length, and message identifier fields.
Offset + 8	0–63	Message ID	A token that allows software to map read responses to read commands. This value is echoed in the response.
Offset + 16	0–31	Table ID	This field identifies the Pattern Matcher table to be accessed. Codings: 0x0 1-Byte Trigger table 0x1 2-Byte Trigger table 0x2 Variable Length Trigger table 0x3 Confidence table 0x4 Pattern Description and Stateful Rule table 0x5 User-Defined Group Mapping table 0x6 Equivalent Byte Mapping table 0x7 SRE Context table 0x8 Special Trigger table
Offset + 20	0–31	Index	The index within the table of the entry that is to be read.

Table 17-151. Read Table Entry Response Format

Offset	Bits	Name	Description
Offset + 0	0–7	Version	Echoed from the Read Instruction.
—	8–15	Type	Echoed from the Read Instruction.
—	16–31	Reserved	—
Offset + 4	0–31	Length	The length of the overall response including the type, length, and message identifier fields.
Offset + 8	0–63	Message ID	Echoed from the Read Instruction.
Offset + 16	0–31	Table ID	Echoed from the Read Instruction.
Offset + 20	0–31	Index	Echoed from the Read Instruction.
Offset + 24	N x 32	Data	An integral number of 32 bit data words containing table read data. The value of N is table specific: • Table ID 0x0: N=8. The entire 256 bit table is read with a single control command. • Table ID 0x1: N=2. 36 bit table entries are right justified within 8 bytes. • Table ID 0x2: N=2. 36 bit table entries are right justified within 8 bytes. • Table ID 0x3: N=1, entries are 32 bits. • Table ID 0x4: N=32, only full 128 bytes entries may be read. • Table ID 0x5: N=1. 3-bit entry is right justified within the 32-bit response data word. • Table ID 0x6: N=1. 8-bit entry is right justified within the 32-bit response data word. • Table ID 0x7: N=8, only 32 byte entries may be read. • Table ID 0x8: N=8. The entire 256 bit table is read with a single control command.

Table 17-152. Write Table Entry Pattern Matcher Control Format

Offset	Bits	Name	Description
Offset + 0	0–7	Version	Identifies the version of the Pattern Matcher that this command is intended for. Set to 0x1 for this revision.
	8–15	Type	An enumerated field indicating what type of Pattern Matcher Control command this is. Codings: • 0x00 Read Table Entry • 0x01 Write Table Entry • 0x08 Clear Session Context by Session ID
—	16–31	Reserved	—
Offset + 4	0–31	Length	The length of the overall instruction including the type, length, and message identifier fields.
Offset + 8	0–63	Message ID	A token that allows software to map read responses to read commands. This value is echoed in the response.

Table 17-152. Write Table Entry Pattern Matcher Control Format (continued)

Offset	Bits	Name	Description
Offset + 16	0–31	Table ID	This field identifies the Pattern Matcher table to be accessed. Codings: This coded field specifies where the output data buffer was sourced from. Codings: 0x0 1-Byte Trigger table 0x1 2-Byte Trigger table 0x2 Variable Length Trigger table 0x3 Confidence table 0x4 Pattern Description and Stateful Rule table 0x5 User-Defined Group Mapping table 0x6 Equivalent Byte Mapping table 0x7 SRE Context table 0x8 Special Trigger table
Offset + 20	0–31	Index	The index within the table of the entry that is to be read or written.
Offset + 24	N x 32	Data	An integral number of 32 bit data words containing table write data. The value of N is table specific: • Table ID 0x0: N=8. The entire 256 bit table is written with a single control command. • Table ID 0x1: N=2. 36 bit table entries are right justified within 8 bytes. • Table ID 0x2: N=2. 36 bit table entries are right justified within 8 bytes. • Table ID 0x3: N=1, entries are 32 bits. • Table ID 0x4: N=8 to 32, entries are 32 to 128 bytes, in multiples of 32 bytes. If multiple entries are being written in a single instruction, however, then all entries must be 128 bytes. • Table ID 0x5: N=64. The entire 256 byte table is written with a single control command. • Table ID 0x6: N=64. The entire 256 byte table is written with a single control command. • Table ID 0x7: N=8. Entries are 32 bytes. • Table ID 0x8: N=8. The entire 256 bit table is written with a single control command. A single Write Table command can contain multiple Index/Data pairs to allow writing multiple entries within one table using a single command.

Table 17-153. Clear Session Context by Session ID Pattern Matcher Control Format

Offset	Bits	Name	Description
Offset + 0	0–7	Version	Identifies the version of the Pattern Matcher that this command is intended for. Set to 0x1 for this revision.
—	8–15	Type	An enumerated field indicating what type of Pattern Matcher Control command this is. Codings: • 0x00 Read Table Entry • 0x01 Write Table Entry • 0x08 Clear Session Context by Session ID

Table 17-153. Clear Session Context by Session ID Pattern Matcher Control Format (continued)

Offset	Bits	Name	Description
—	16–31	Reserved	—
Offset + 4	0–31	Length	The length of the overall instruction including the type, length, and message identifier fields.
Offset + 8	0–63	Message ID	A token that allows software to map read responses to read commands. This value is echoed in the response.
Offset + 16	0–31	Session ID	This field identifies the Session context to be cleared. Valid range is 0 to 131.071 million (27-bit value).
Offset + 20	0–23	Reserved	—
	24–31	Rule Capacity	This field specifies the number of rules per session to be used when processing this instruction. Valid coding values for this field are identical to those of the CNR field in the SREC register (Section 17.3.4.7 on page 38). Take note that this field is optional. That is, it is only used when the CNR field in the SREC register hasn't been configured.

Table 17-154. Notification FIFO Entry Fields (Format = Block Vector)

Offset	Bits	Name	Description
Offset + 0	0–63	Stream ID	Copied from the stream context pointed to by the Command FIFO entry. This is simply a per-stream token value that can be used by software. If this notification is the result of a NOP command, then the "Stream ID" value from the Command FIFO entry is echoed here.
Offset + 8	0–63	Output Physical Pointer	The physical memory address of the data buffer or chain of scatter/gather tables.
Offset + 16	0–63	Output Virtual Pointer	The virtual memory address of the data buffer or chain of scatter/gather tables. If the described output buffer was pre-constructed and supplied with the command then this field is not valid and contains the same value as Output Physical Pointer.
Offset + 24	0–63	—	Reserved
Offset + 32	0–15	Output Free Buffer Size	When the described data structure was created by the DMA Engine out of Free Buffers, this field reflects the size of Free Buffers that were used. A value of 0x0000 means a buffer size of 65,536 bytes.
	16–31	Output Buffer Count	The total number of buffers in the created data structure. This field is only valid if the structure was created by the DMA Engine out of free buffers. It is possible for this count to overflow if the amount of output data uses more than 65,535 buffers. The overflow can be avoided by setting the NRLL and NDLL registers to a non-zero value.
Offset + 36	0–31	Output Length	A number in the range 0 to 4GB-1. It represents the total number of data bytes contained in the described buffer.

Table 17-154. Notification FIFO Entry Fields (Format = Block Vector) (continued)

Offset	Bits	Name	Description
Offset + 40	0–15	Output Table Count	The total number of scatter/gather tables in the scatter/gather chain. This field is only valid for structures that the DMA Engine has created out of Free Buffers. It is possible for this count to overflow if the tables are using more than 65,535 buffers. The overflow can be avoided by setting the NRLL and NDLL registers to a non-zero value.
—	16–31	Output Offset	A number in the range 0 to 64KB-1 that represents the location of the first byte of output data relative to “Output Physical Pointer”.
Offset + 44	0–7	Notification FIFO Exception Code	Provides application specific exception code information.
—	8–9	—	Reserved
—	10	Output Truncated	Indicates that output data was truncated due to insufficient buffer space (Output Buffer Source = 0x0) or Free Buffer Exhaustion (Output Buffer Source = 0x1 or 0x2).
—	11	Output Extension	Determines whether to jump to a chain of scatter/gather tables whenever the “Output Physical Pointer” field in this same dword is used. Codings: 0x0 The “Output Physical Pointer” field points to contiguous data. 0x1 The”Output Physical Pointer” field points to a chain of scatter/gather tables.
—	12–13	Output Buffer Source	This coded field specifies where the output data buffer was sourced from. Codings: 0x0 Command entry. 0x1 Channel Free Buffer List A 0x2 Channel Free Buffer List B 0x3 Null, no buffer is present.
—	14	Output Buffer Format	This coded field indicates what descriptor format the data buffer is described with. Codings: 0x0 Block Vector 0x1 Linked List
—	15–31	Output Buffer Data Length	A number in the range 0 to 64KB that represents the number of bytes of output data at “Output Physical Pointer” + “Output Offset”.
Offset + 48	0–127	Application Specific Pass Through	Application Specific Pass Through.

Table 17-155. Notification FIFO Entry Fields (Format = Linked List)

Offset	Bits	Name	Description
Offset + 0	0–63	Stream ID	Copied from the stream context pointed to by the Command FIFO entry. This is simply a per-stream token value that can be used by software. If this notification is the result of a NOP command, then the “Stream ID” value from the Command FIFO entry is echoed here.
Offset + 8	0–63	Output Physical Head Pointer	The physical address of the first buffer in the linked list.
Offset + 16	0–63	Output Virtual Head Pointer	The virtual address of the first buffer in the linked list. This value is opaque to the pattern matcher and travels with the physical address as an assist to software.
Offset + 24	0–63	Output Virtual Tail Pointer	The virtual address of the last buffer in the linked list.
Offset + 32	0–15	Output Free Buffer Size	When the described data structure was created by the DMA Engine out of Free Buffers, this field reflects the size of Free Buffers that were used. A value of 0x0000 means a buffer size of 65,536 bytes.
—	16–31	Output Buffer Count	The total number of buffers in the linked list.
Offset + 36	0–31	Output Length	The total number of output bytes contained in the linked list of output buffers.
Offset + 40	0–15	—	Reserved
—	16–31	Output Offset	A number in the range 0 to 64KB-1 that represents the location of the first byte of output data relative to “Output Physical Pointer”.
Offset + 44	0–7	Notification FIFO Exception Code	Provides application specific exception code information.
—	8–9	—	Reserved
—	10	Output Truncated	Indicates that output data was truncated due to insufficient buffer space (Output Buffer Source = 0x0) or Free Buffer Exhaustion (Output Buffer Source = 0x1 or 0x2).
—	11	—	Reserved
—	12–13	Output Buffer Source	This coded field specifies where the output data buffer was sourced from. Codings: 0x0 Command entry. 0x1 Channel Free Buffer List A 0x2 Channel Free Buffer List B 0x3 Null, no buffer is present.
—	14	Output Buffer Format	This coded field indicates what descriptor format the data buffer is described with. Codings: 0x0 Block Vector 0x1 Linked List

Table 17-155. Notification FIFO Entry Fields (Format = Linked List) (continued)

Offset	Bits	Name	Description
—	15–31	Output Buffer Data Length	A number in the range 0 to 64KB that represents the number of bytes of output data beginning at “Output Physical Pointer” + “Output Offset”.
Offset + 48	0–127	Application Specific Pass Through	Application Specific Pass Through.

Table 17-156. Notification FIFO Exception Code Enumeration

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x00	No Exception	—	—	—	—	The work unit data was processed without encountering an exception.
0x20	Deflate Output Limited Failure	3	No	No-halt	Deflate Engine	This is the error that is returned if the quantity of data being output from a work unit exceeds the amount set in the DBPPWL register (see Section 17.3.5.4, “Deflate Bytes Produced Per Work Unit Limit (DBPPWL)”), if enabled.
0x21	Deflate Error Corrupted	3	No	No-halt	Deflate Engine	Indicates that the SIZE, as recorded in the GZIP trailer, did not match the expected values at the conclusion of decompression. It can also indicate that an invalid code was encountered within the Deflate data, indicating data corruption or a bad header.[4]
0x22	Deflate Error Truncated	3	No	No-halt	Deflate Engine	Indicates that the work unit data ended at an unexpected point indicating that the data was either corrupted or truncated. This is not related to the truncation indication bit in the Notification entry which indicates truncation due to insufficient buffer space or free buffer list exhaustion.
0x23	Deflate Error Header	3	No	No-halt	Deflate Engine	Indicates a problem with the GZIP, ZLIB, or Deflate headers was encountered. These would include corrupted data, as well as unsupported compression algorithms.
0x24	Deflate CRC Failure	3	No	No-halt	Deflate Engine	Indicates that the CRC (either Adler-32 or CRC-32), as recorded in the GZIP or ZLIB trailer, did not match the computed value at the conclusion of decompression.
0x25	Deflate Error Distance	3	No	No-halt	Deflate Engine	Indicates that when processing Deflate payload a distance/length pair was detected where the distance examined data further back in the history than currently exists. This data is returned as zeroes but is likely invalid. This error does not cause the Deflate operation to abort unless another error happens later in the work unit.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x40	KES Confidence Collision Limit	2	No	Common-control	Key Element Scanner	Key Element Scanner has detected a collision chain in the Confidence table that exceeds the maximum allowed entries. The maximum allowed entries is configured through the KEC register (see 17.3.2.10/17-27). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x41	KES Confidence Mask Error	2	No	Common-control	Key Element Scanner	Key Element Scanner has found a Confidence Mask code of value “0” in a Confidence table entry. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x48	DXE Invalid Repeat Error	2	No	Common-control	Data Examination Engine	Data Examination Engine (DXE) has encountered an instruction test line that includes an invalid combination of maximum and minimum repeat values. Example of an invalid combination would be a maximum repeat value that is smaller than the minimum repeat value. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x49	DXE Test Line Syntax Error	2	No	Common-control	Data Examination Engine	Data Examination Engine (DXE) has encountered an ill-formed test line. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x4b	DXE Instruction Limit Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has exceeded the maximum allowed test line executions per pattern. The maximum allowed test line executions is configured through the DEC register (see 17.3.3.7/17-33). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x4c	DXE Pattern Description and Stateful Rule Space Outrange Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has encountered a pointer to a test line block (128-byte block index) that exceeds the maximum allowed block index. The maximum allowed block index is configured through the DEC register (see 17.3.3.7/17-33). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x4d	DXE Stack Overflow Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) internal stack has overflowed while executing test line instructions. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x4e	DXE Alternate Link Same Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has encountered an instruction test line that contains an Alternate Link field with an instruction test line number pointing to itself. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x4f	DXE Subsequent Link Same Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has encountered an instruction test line that contains a Subsequent Link field with an instruction test line number pointing to itself. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x50	DXE Subsequent Link Reverse Error	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has encountered an instruction test line that contains a Subsequent Link field with an instruction test line number that is numerically lower than its instruction test line number. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x51	DXE Invalid Test Line Branch	2	No	Common-control	Data Examination Engine	The Data Examination Engine (DXE) has encountered an instruction test line that contains a Subsequent Link or an Alternate Link field with an instruction test line number that is numerically higher than the number of the last instruction test line for a pattern. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x59	SRE Invalid Reaction Head Block Number Instructions	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered a Reaction Head Block Number that contains an invalid value in its Instruction Word field. For instance a value of 1 would be invalid since a reaction must begin with a “reaction start” instruction” and end with an “end reaction” instruction. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x5a	SRE Reaction Limit	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has exceeded the maximum allowed number of Reaction Head blocks to traverse per pattern match event. The maximum allowed number of Reaction Head blocks is configured through the SEC1 register (see 17.3.4.14/17-43). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x5b	SRE Pattern Description and Stateful Rule Space Outrange Error	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered a pointer to a reaction block (128-byte block index) that exceeds the maximum allowed block index. The maximum allowed block index is configured through the SEC2 register (see 17.3.4.15/17-43). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x5c	SRE Session Context Outrange Error	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered an SRE instruction with a Context Offset field set to a value that exceeds the context size per session. The context size per session is configured through the SREC register (see 17.3.4.7/17-38). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x5d	SRE Context Table Outrange Error	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered a Session ID that would cause it to access the Session Context table with an index that exceeds the maximum allowed index. The maximum allowed index is configured through the SEC3 register (see 17.3.4.16/17-44). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x5e	SRE Instruction Limit	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has exceeded the maximum number of SRE instructions that are allowed to be executed per reaction. The maximum number of SRE instructions per reaction is configured through the SEC1 register (see 17.3.4.14/17-43). In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0x5f	SRE Invalid Instruction Jump	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered an SRE instruction that contains an invalid absolute jump location value. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x60	SRE Instruction Syntax Error	2	No	Common-control	Stateful Rule Engine	The Stateful Rule Engine (SRE) has encountered an ill-formed SRE instruction. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0x80	DE Pattern Matcher Control Command Error	1	Yes	Common-control	DMA Engine	This error code is generated if the DMA Engine has detected an improperly formed command while executing a batch of Pattern Matcher control commands. It is also possible to get this error code if KES, DXE or SRE encounter an error as a result of an ill-formed pattern matching data structure. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEPMCE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0xa0	DE Activity Code Error	1	No	Channel	DMA Engine	The DMA Engine has detected an unrecognized Activity Code in stream context during execution of a Command FIFO. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xa1	DE Command FIFO Entry Error	1	No	Channel	DMA Engine	The DMA Engine has detected an improperly formed Command FIFO entry. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0xc0	SRE Context System Memory Read Error	0	No	Channel	Stateful Rule Engine	A “context” system memory read transaction performed by the Stateful Rule Engine (SRE) has resulted in a system memory bus error. In addition to assigning this error code to notification (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.
0xc1	SRE Instruction System Memory Read error	0	No	Channel	Stateful Rule Engine	An “instruction/reaction” system memory read transaction performed by the Stateful Rule Engine (SRE) has resulted in a system memory bus error. In addition to assigning this error code to notification (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.
0xc2	DXE System Memory Read error	0	No	Channel	Data Examination Engine	A system memory read transaction performed by the Data Examination Engine (DXE) has resulted in a system memory bus error. In addition to assigning this error code to notification (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.
0xc3	FBM List B System Memory Read error	0	Yes	Channel	Free Block Manager	A “Free Buffer List B” system memory read transaction performed by the Free Block Manager (FBM) has resulted in a system memory bus error. In addition to assigning this error code to notification(s) (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.
0xc4	FBM List A System Memory Read error	0	Yes	Channel	Free Block Manager	A “Free Buffer List A” system memory read transaction performed by the Free Block Manager (FBM) has resulted in a system memory bus error. In addition to assigning this error code to notification(s) (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.
0xc5	DE System Memory Read Error	0	No	Channel	DMA Engine	A system memory read transaction performed by the DMA Engine (DE) has resulted in a system memory bus error. In addition to assigning this error code to notification (this code is also reflected in the channel’s CHERR register), the channel’s ISR[ERR] bit (if enable) is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0xc6	FBDF System Memory Read Error	0	No	Channel	DMA Engine	A system memory read transaction performed by the DMA Engine (DE) while performing Free Buffer Deallocate FIFO (FBDF) operations has resulted in a system memory bus error. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd0	DXE Memory Collision Error	0	No	Channel	Data Examination Engine	The Data Examination Engine (DXE) read a location in the SUI history memory in the same clock cycle in which the Key Element Scanner (KES) wrote to this same location. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd1	DXE SUI History Memory Parity Error	0	No	Channel	Data Examination Engine	A parity error has been detected by the Data Examination Engine (DXE) when reading from the SUI history memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd2	KES Confidence Memory Parity Error.	0	No	Channel	Key Element Scanner	A parity error has been detected by the Key Element Scanner (KES) when reading from the confidence memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd3	KES EUG Memory Parity Error	0	No	Channel	Key Element Scanner	A parity error has been detected by the Key Element Scanner (KES) when reading from the programmable Equivalent Byte Mapping and User Defined Group Mapping memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd4	KES 2-Byte Trigger Memory Parity Error	0	No	Channel	Key Element Scanner	A parity error has been detected by the Key Element Scanner (KES) when reading from the 2-Byte Trigger memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0xd5	KES Variable Length Trigger Memory Parity Error	0	No	Channel	Key Element Scanner	A parity error has been detected by the Key Element Scanner (KES) when reading from the Variable Length Trigger memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd6	Deflate History Memory Parity Error.	0	No	Channel	Deflate Engine	A parity error has been detected by the Deflate Engine when reading from the Decompression History Memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd7	DE Confidence Memory Parity Error.	0	No	Channel	DMA Engine	A parity error has been detected by the DMA Engine (DE) when reading from the confidence memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd8	DE EUG Memory Parity Error	0	No	Channel	DMA Engine	A parity error has been detected by the DMA Engine (DE) when reading from the programmable Equivalent Byte Mapping and User Defined Group Mapping memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xd9	DE 2-Byte Trigger Memory Parity Error	0	No	Channel	DMA Engine	A parity error has been detected by the DMA Engine (DE) when reading from the 2-Byte Trigger memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xda	DE Variable Length Trigger Memory Parity Error	0	No	Channel	DMA Engine	A parity error has been detected by the DMA Engine (DE) when reading from the Variable Length Trigger memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.

Table 17-156. Notification FIFO Exception Code Enumeration (continued)

Value	Name	Precedence[1]	Sticky[2]	Error Type[3]	Detecting Functional Unit	Description
0xdb	DE Internal Memory Parity Error	0	No	Common-control	DMA Engine	A parity error has been detected by the DMA Engine (DE) when reading from its internal memory. In addition to assigning this error code to notification (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEIMPE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0xdc	DXCM Internal Memory Parity Error	0	No	Channel	Data Examination Engine	A parity error has been detected by the Data Examination Engine (DXE) when reading from its internal Data Examination Context memory. In addition to assigning this error code to notification (this code is also reflected in the channel's CHERR register), the channel's ISR[ERR] bit (if enable) is set.
0xe0	SRE Context System Memory Write Error	0	Yes	Common-control	Stateful Rule Engine	A "context" system memory write transaction performed by the Stateful Rule Engine (SRE) has resulted in a system memory bus error. In addition to assigning this error code to notification(s) (this code is also reflected in CHERR[0..3] registers), the ISRCC[SREWMIE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0xe2	FBM System Memory Write Error	0	Yes	Common-control	Free Block Manager	A system memory write transaction performed by the Free Block Manager (FBM) has resulted in a system memory bus error. In addition to assigning this error code to notification(s) (this code is also reflected in CHERR[0..3] registers), the ISRCC[FBMWMIE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.
0xe3	DE System Memory Write Error	0	Yes	Common-control	DMA Engine	A system memory write transaction performed by the DMA Engine (DE) has resulted in a system memory bus error. In addition to assigning this error code to notification(s) (this code is also reflected in CHERR[0..3] registers), the ISRCC[DEWMIE] bit is set (if enable) and the ISR[CCERR] bit (if enable) of every channel is set.

1 0 corresponds to highest, 3 corresponds to lowest

2 Means that the condition persists across all subsequent notifications, as opposed to errors that can be isolated to a single work unit

3 See Section 17.4.5.6, "Error Handling," for a detailed description of the error types.

[4] This error can occur when a huffman code in the code tree is defined in the header but has no meaning as defined in the Deflate RFC (RFC1951). It could also indicate corrupted data if this corrupted data matched an undefined huffman code. This error is possible for fixed as well as dynamic type Deflate blocks. (See RFC 1951 for further details.)

Table 17-157. DMA Engine Channel Scheduling Algorithms

Name	Description
Weighted Round Robin By	Applies weighted round robin scheduling for servicing the four channels. weight value of 0..N is programmed for each channel. The scheduler “sticks” to the chosen channel until ((its programmed weight is reached) && (work is available on another channel)), or until ((it has been drained) && (work is available on another channel)). The scheduler also switches channels if the selected channel’s Notification FIFO is full or its free buffer list is depleted. If the selected channel has been drained but no work is available on another channel then its weight counter does not get reset. A channel’s weight counter is only reset when the selected channel changes. If multiple channels have work available and the selected channel has reached its weight or has been drained, then the scheduler selects the next channel in a round-robin manner following the sequence channel 0,1,2 and 3.
Weighted Round Robin 2 BY 2	Applies weighted round robin scheduling for servicing the four channels. weight value of 0..N is programmed for each channel. The scheduler “sticks” to the chosen channel until ((its programmed weight is reached) && (work is available on another channel)), or until ((it has been drained) && (work is available on another channel)). The scheduler also switches channels if the selected channel’s Notification FIFO is full or its free buffer list is depleted. If the selected channel has been drained but no work is available on another channel then its weight counter does not get reset. A channel’s weight counter is only reset when the selected channel changes. If multiple channels have work available and the selected channel has reached its weight or has been drained, then the scheduler selects the next channel in a round-robin manner following the sequence channel 0,2,1 and 3.

Table 17-158. Free Buffer Deallocate FIFO Linked List Entry Format

Offset	Bits	Name	Description
Offset + 0	0–63	Next Buffer Descriptor Physical Pointer	The physical address of the Next Buffer Descriptor of the next buffer in the linked list chain.
Offset + 8	0–63	Next Buffer Descriptor Virtual Pointer	The virtual address of the Next Buffer Descriptor of the next buffer in the linked list chain.
Offset + 16	0–15	Buffer Count	The number of free buffers in the linked list chain.
	16–31	—	Reserved
Offset + 20	0	Free Buffer List	Specifies which of the channel’s free buffer lists the buffers are to be deallocated to. Coding: • 0 Free Buffer List A. • 1 Free Buffer List B.
—	1	Format	Specifies the format of the Free Buffer Deallocate FIFO entry. Codings: • 0 Block Vector. • 1 Linked List.
	2–14	—	Reserved
	15–31	—	Reserved
Offset + 24	0–63	Physical Tail Pointer	The physical address of the tail buffer in the linked list chain.

Table 17-159. Free Buffer Deallocate FIFO Scatter/Gather Block Vector Entry Format

Offset	Bits	Name	Description
Offset + 0	0–63	Physical Pointer	The physical address of the first buffer in the scatter/gather chain.
Offset + 8	0–63	Virtual Pointer	The virtual address of the first buffer in the scatter/gather chain.
Offset + 16	0–15	Buffer Count	The number of free buffers in the scatter/gather chain, including those used as scatter/gather tables within the chain.
—	16	Extension	Meaning: • 0x0 Physical Pointer field points to a contiguous data buffer. • 0x1 Physical Pointer field points to a scatter/gather table.
—	17–31	—	Reserved
Offset + 20	0	Free Buffer List	Specifies which of the channel's free buffer lists the buffers are to be deallocated to. Coding: • 0 Free Buffer List A. • 1 Free Buffer List B.
—	1	Format	Specifies the format of the Free Buffer Deallocate FIFO entry. Codings: • 0 Block Vector. • 1 Linked List.
	2–31	—	Reserved
Offset + 24	0–63	—	Reserved

17.4.6 Free Buffer Manager

The Free Buffer Manager block manages a set of eight free buffer lists on behalf of the DMA Engine. The free buffer lists are flexibly assignable to one or more of the output data streams (result data and decompressed data) across DMA Engine channels. This scheme allows for flexible configuration of free buffer resources such that a single list can serve both sources of output on all channels, or each source of output on each channel can be served by a dedicated free buffer list. Any configuration combination in between those two extremes can be configured.

From a DMA channel's point of view, the list assignment results in an A/B pair of "virtual" free buffer lists being available for use. Per-command, software may specify which of the A/B Free Buffer Lists to use for each output stream. Mapping of physical free buffer lists to individual channels' free buffer lists is done at initialization via a set of FBM common control registers.

Each free buffer list consists of a linked-list of equal size memory buffers provisioned by software and managed by the FBM. Free buffers are sized as an integral number of 32 bytes up to a maximum size of 64 Kbytes.

A 64-byte link field within the buffer boundary is reserved per-buffer for storing linked-list links and other buffer context. This results in the useable buffer space being reduced by 64 bytes as each when the buffers are used to create linked-list data structures as output. When scatter/gather data structures are created using free buffer list buffers, the link field within the buffer is overwritten with buffer data or scatter/gather table data as the buffers are consumed from the free buffer list.

All free buffer deallocation onto free buffer lists is done via the DMA Engine's Free Buffer Deallocate FIFO.

FBM Internal Register space provides read-only visibility of the eight free buffer list structures. In the event that software needed to reap free buffers from a free buffer list, the linked list descriptors can be obtained from these registers so long as it is known that the DMA Engine channel(s) using the free buffer list are disabled or otherwise prevented from allocating free buffers.

17.4.6.1 Free Buffer Linked List Structural Specification

When free buffers are under the control of the FBM they are maintained in a linked list structure.

The FBM keeps head, tail, and buffer count registers for each free buffer list that it manages.

A 64 byte Next Buffer Descriptor field is present in the first 64 bytes of each buffer and contains buffer linking information. The 64 byte Next Buffer Descriptor also contains fields which describe the size of the buffer and the data within the buffer, however, these fields have no meaning when the buffer is on a free buffer list. Buffers on a free buffer list are presumed to be empty and to be of the programmed buffer size for that free buffer list.

Buffer linking information consists of a physical/virtual address pair. The physical address is used by the Free Buffer Manager to access buffers. The virtual address is an opaque value to the Free Buffer Manager and is maintained in the linked list structure for the benefit of software. Maintaining both a physical and virtual address for each buffer offloads software address translation operations when the linked list is being iterated by a software routine.

Table 17-144 details the general format of the Next Buffer Descriptor field within a linked list buffer. When a buffer is present on a free buffer list the "Offset" and "Buffer Data Length" fields are don't care values. When a buffer is consumed from a free buffer list the "Offset" field is reset 0x0 and the "Buffer Data Length" field is set to the programmed size value for that specific free buffer list. In addition, the balance of the 64 byte Next Buffer Descriptor is scrubbed to all 0's in case any stale software context is present in the Next Buffer Descriptor that could cause problems when driver or application software processes the linked list.

Figure 17-143 details the linked list structure used by the FBM for managing free buffers on a free buffer list.

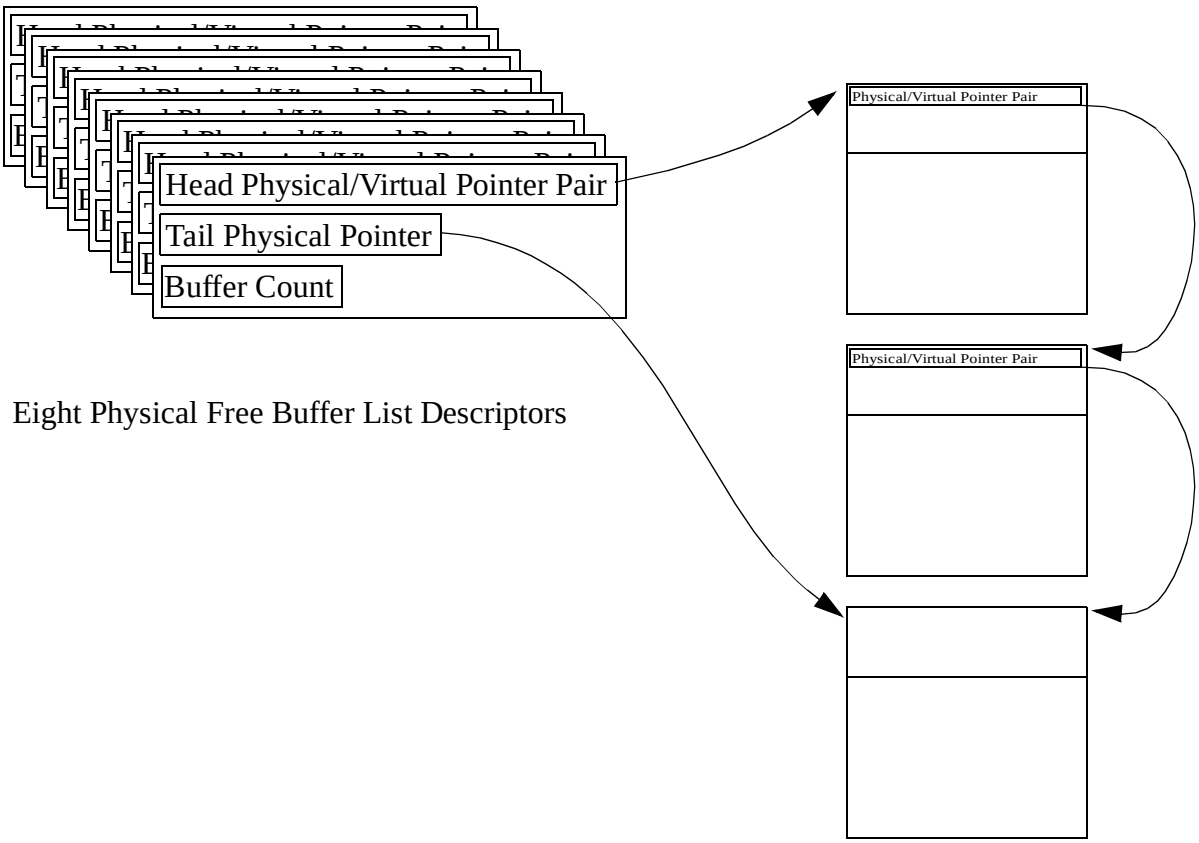


Figure 17-143. FBM Linked List Structure

17.4.6.2 Free Buffer List Depletion

In order to guard against loss of either Pattern Matcher result data or Deflate Engine decompressed output data due to free buffer list exhaustion, the DMA Engine and Free Buffer Manager implement the following list depletion detection and prevention features:

- A per-notification result length limit is programmed in the DMA Engine such that reports produced for a single notification entry are truncated when the limit is reached. This cap prevents a single large result from unexpectedly exhausting a free buffer. See Section 17.3.8.24, “Notification Deflate Length Limit Register Channel 0 (NDLL0) and Section 17.3.8.25, “Notification Result Length Limit Register Channel 0 (NRLL0).
- An interrupt threshold is set such that if the number of available free buffers in the free buffer list falls below the threshold, the DMA Engine generates an interrupt. See Section 17.3.9.1, “Free Buffer List Threshold A Register Channel 0 (FBL_THRES_A0).
- A stop service threshold is set such that if the number of available buffers in the free buffer list falls below the threshold, the DMA Engine stops servicing the channel’s Command FIFO. In order to be effective, the threshold must be at least as large as (the notification result length limit) * (the number of commands that can be in flight). See Section 17.3.9.1, “Free Buffer List Threshold A Register Channel 0 (FBL_THRES_A0).

17.4.6.3 FBM Cache Awareness and Memory Transaction Priority

The FBM_CR register (see Section 17.3.6.4, “Free Buffer Manager Control Register (FBM_CR)) provides a bit that allows control over the use of snoop during FBM memory transactions.

The FBM_CR register also provides a setting that determines the priority level applied to FBM memory transactions on the system bus.

17.4.7 Memory Interface Arbiter

The Memory Interface Arbiter (MIA) block provides access to system memory for all of the functional units within the PM, allowing them to initiate read and write transactions over the system bus. The MIA consists of the following major sub-blocks:

- Memory Read Port(s)
  - Used by a client functional unit to initiate a read from system memory
- Memory Write Port(s)
  - Used by a client functional unit to initiate a write to system memory
- Arbiter
  - Arbitrates among the client read and write ports for access to the system bus.

17.4.7.1 MIA Read and Write Ports

The MIA’s read and write ports provide buffering for read and write requests from a client to system memory.

For a write port, the address, byte_count, and write data are all buffered within the port, such that a client can post an entire write transaction into the port with or without gaps (wait states) in the data transfer. Once the write request and all write data for a transaction have been received, the transaction is eligible for posting on the system bus. The MIA arbitrates for the system bus, post the write transaction request, and transfer the write data from its buffer, possibly with wait states inserted by the target, without intervention from the client. Each write port contains a 4 deep write transaction queue, allowing the client to post up to 4 writes even while the system bus is busy.

For a read port, the address and byte_count are buffered within the port, and a client posts a read transaction in a single cycle. The MIA then takes care of arbitrating for the system bus and requesting read data from the target of the read transaction. The MIA read port contains a 4 deep queue of read requests from the client. This allows the MIA read client to take advantage of the read pipelining capability of the system bus, enabling up to 4 read transactions to be posted before any read data has returned from the target.

The largest max transaction size permitted is 512 bytes, this is limited by the max transaction size supported by the system bus gasket. For the MIA’s write ports, the max transaction size for each port should be made as small as is practical, to reduce the size of the write data buffers in the ports. A summary of the MIA port assignments in shown in Table 17-160.

Table 17-160. MIA Read and Write Port Assignments

Client	R or W	Queue depth	Max transaction size (bytes)	Description
DE	Read	4	512	On the Consumer side, DE reads the following: Command FIFO entries, stream context buffers, scan data buffer gather lists, scan data buffers, residue buffers. On the Producer side, DE reads the following: nothing.
DE	Write	4	32	On the Consumer side, DE writes the following: residue buffers, stream context (for config). On the Producer side, DE writes the following: Notification FIFO entries, Result buffer scatter lists, result data buffers.
DXE	Read	4	512	DXE reads to Pattern Description and Stateful Rule table in system memory.
SRE	Read	4	512	SRE reads to Pattern Description and Stateful Rule table in system memory.
SRE	Read	4	512	SRE reads to SRE Context table in system memory. Using 2 read ports for the SRE allows it to prefetch Stateful Rule for one pattern match while processing the rules for another.
SRE	Write	4	32	SRE writes to SRE Context table in system memory.
FBM	Read	4	512	FBM reads the Next Buffer Descriptor (pointer to next buffer) of the buffer currently at the head of the free buffer list, when allocating a buffer from the free buffer list.
FBM	Write	4	16	FBM writes the Next Buffer Descriptor (pointer to next buffer) of the buffer currently at the tail of the free buffer list, when deallocating a buffer or a linked list of buffers to the free buffer list. Fields to be written are the physical (8 bytes) and virtual (8 bytes) addresses.

17.4.7.2 MIA Read/Write Port Arbitration

The MIA performs arbitration for access to the system bus among its attached read and write ports. The attached read and write ports are classified into two groups, associated read/write port pairs, or read-only ports. In the PM block MIA services 3 associated port pairs: DE write with DE read port, SRE write with SRE session context read port, and FBM write with FBM read port. MIA also services two read-only ports, one from the DXE for reading pattern descriptions and one from the SRE for reading stateful rules (such as reactions).

The arbitration scheme used is two-tiered. At the upper tier, the 5 items to be serviced (3 associated read/write port pairs and 2 read-only ports) are serviced in round-robin fashion to select the next transaction to be sent to the system bus. At the lower tier, arbitration between reads and writes in each of the associated port pairs can be configured into one of two modes via the MIA_CR register, serialized mode or read priority weighted mode. After a reset, serialized mode is the default arbitration mode, with both local snoop and write cancellation disabled on all port pairs.

17.4.7.2.1 Serialized Mode

When serialized mode arbitration is used, reads and writes on an associated port pair are presented to the system bus in the order in which they arrived at the client read and write port. Each associated port pair uses an 8-deep SRQ (Serialized Request Queue) which sits in parallel with the 4 deep write and read transaction queues of these ports. Reads and writes are placed in order in the SRQ, simultaneous read and

write (which is not an error as long as the read and write are to two different pieces of data), puts read ahead of write.

17.4.7.2.2 Read Priority Weighted Mode

In this mode, the arbiter selects a read transaction over a write by default when there are pending reads to be serviced in this port pair. If there are no pending reads, or if the number of consecutively serviced reads while a write is pending exceeds the read weight programmed for this port pair, then a write transaction is selected. This mode is intended to improve performance by favoring reads over writes. Write starvation is avoided by appropriate programming of the read weight in MIA_ACR register.

When read and write transactions are both pending in a port pair, the arbiter services RW + 1 consecutive reads (RW is the read weight programmed for each port pair in MIA_ACR register), then services one write transaction. Therefore with RW = 0, one write is serviced for every read from this port pair. With RW at its max value (127), one write is serviced for every 128 consecutive reads.

Note that read priority weighted mode is only supported by the SRE context port pair. The DE and FBM port pairs do not support read priority weighted mode, they must always be programmed for the default serialized mode, with local snoop disabled.

17.4.7.3 MIA Local Snoop and Write Cancellation

When an associated read/write port pair on MIA is operated in read priority weighted arbitration mode (this mode is only supported for the SRE context port pair), writes to the system bus that are pending in MIA's write queues may not proceed until several pending reads have been serviced. Therefore reads can, and likely will, bypass writes, causing a potential hazard if a client attempts to read a piece of data that was recently written, but is still pending in the write queue. To resolve this issue, the client has an MIA read port and an associated "locally snoopable" write port, and all read transactions (address and byte_count) are snooped in the associated write queue if local snoop is enabled on the port pair in the MIA_CR register. If a local snoop hit occurs on a pending write transaction in the queue, the write is cancelled (does not go to the system bus) if MIA_CR[WCEn] is set for this port pair, the read request is not passed on to the system bus, and the requested read data is taken from the write queue's data buffer and returned to the client over a dedicated local snoop data interface. The term "local snoop" is used for this mechanism to differentiate from snooping operations in the processor's L1 and/or L2 cache, which is unrelated to the local snoop operation performed internally by MIA between its associated read and write queues.

Write cancellation on a local snoop hit of a pending write queue entry is a performance enhancement, and it relies on the fact that a client sequencer which reads a piece of context data intends to write it back to memory at some later time, therefore the write transaction that is currently pending is unnecessary. However there are cases when the client performing the read does not intend to write back the data, or intends to write it back only if modified. These cases are handled on the interface between MIA and its client sequencers, by dynamically disabling write cancellation for certain read transactions. Write cancellation can also be enabled or disabled for all transactions on each associated port pair via MIA_CR.

After a pending write is sent to the system bus, the state of its entry in the write queue is changed from pending to committed, while waiting for the data phase of the transaction to complete. A committed entry in the write queue remains valid for local snooping, although it cannot be cancelled by a local snoop hit even if write cancellation is enabled. If a read transaction snoop hits on a write queue entry that is in the

committed state (for example, the write has been posted to the system bus, but the data phase has not yet completed), the write still completes to system memory, but the read request is not passed on to the system bus, and the requested read data is returned to the client over the aforementioned dedicated local snoop data interface.

Local snoop operation can be enabled or disabled via the MIA_CR register. When a port pair is operated in read priority weighted mode arbitration, local snoop must also be enabled for that port pair, to correctly handle reads which bypass writes as described above. It is also valid to enable local snoop as a performance enhancement with serialized mode arbitration, in which case reads which snoop hit in the associated write queue dpes not go to the system bus, and the requested read data is returned to the client over the dedicated local snoop data interface. This allows a recently written piece of context to be retrieved by the sequencer without incurring the latency of a read from system memory.

However, serialized mode arbitration does not support write transaction cancellation, so when serialized_mode arbitration is used on an associated port pair, write_cancellation is automatically disabled on that port pair, such as setting serialized_mode to ON overrides the write_cancellation enable setting in MIA_CR[WCEn].

17.4.8 Internal Register Interface (IRI)

The Internal Register Interface block converts read and write requests into transactions on the PM's internal register (IR) bus, providing access to all registers within the PM.

Each functional unit within PM contains a number of registers, which are accessed internally via the IR bus. The register space is described Section 17.3, "Memory Map and Register Definition."

17.4.9 Performance Monitor Event Interface

Pattern Matcher provides 32 events that can be counted by the central Performance Monitor. Table 17-161 lists these events. Note that the central Performance Monitor operates at twice the frequency of the Pattern Matcher, therefore all pm_pmon_events are double-counted.

Table 17-161. Pattern Matcher Performance Monitor Events

Pattern Matcher Event	PMC Counter Number	Description of Event Counted
pm_pmon_events[2:0]	C1:tbd C2:tbd C3:tbd	Number of read transactions (from 0 to 4) currently pending on the system bus.
pm_pmon_events[3]	C4:tbd	Number of read transactions issued on the system bus. Asserts high for one cycle for every read transaction issued.
pm_pmon_events[6:4]	C5:tbd C6:tbd C7:tbd	Number of write transactions (from 0 to 4) currently pending on the system bus.
pm_pmon_events[7]	C8:tbd	Number of write transactions issued on the system bus. Asserts for one cycle for every write transaction issued.

Table 17-161. Pattern Matcher Performance Monitor Events (continued)

Pattern Matcher Event	PMC Counter Number	Description of Event Counted
pm_pmon_events[8]	C9:tbd	Number of cycles in which the Memory Interface Arbiter (MIA) is busy. Asserted if there are any outstanding read or write transactions remaining in MIA's internal queues.
pm_pmon_events[9]	C1:tbd	Number of cycles in which the DMA Engine (DE) is at busy1 activity level.
pm_pmon_events[10]	C2:tbd	Number of cycles in which the DMA Engine (DE) is at busy2 activity level.
pm_pmon_events[11]	C3:tbd	Number of cycles in which the DMA Engine (DE) is at busy3 activity level.
pm_pmon_events[12]	C4:tbd	Number of cycles in which the Huffman Decoding Engine (HDE) is at busy1 activity level.
pm_pmon_events[13]	C5:tbd	Number of cycles in which the Huffman Decoding Engine (HDE) is at busy2 activity level.
pm_pmon_events[14]	C6:tbd	Number of cycles in which the Data Expansion Engine (DEE) is at busy1 activity level.
pm_pmon_events[15]	C7:tbd	Number of cycles in which the Data Expansion Engine (DEE) is at busy2 activity level.
pm_pmon_events[16]	C8:tbd	Number of cycles in which the Key Element Scanner (KES) is at busy1 activity level.
pm_pmon_events[17]	C9:tbd	Number of cycles in which the Key Element Scanner (KES) is at busy2 activity level.
pm_pmon_events[18]	C7:tbd	Number of cycles in which the Free Buffer Manager (FBM) is at busy1 activity level.
pm_pmon_events[19]	C8:tbd	Number of cycles in which the Free Buffer Manager (FBM) is at busy2 activity level.
pm_pmon_events[20]	C9:tbd	Number of cycles in which the Free Buffer Manager (FBM) is at busy3 activity level.
pm_pmon_events[21]	C4:tbd	Number of cycles in which the SRE Prefetch Sequencer is at busy1 activity level.
pm_pmon_events[22]	C5:tbd	Number of cycles in which the SRE Prefetch Sequencer is at busy2 activity level.
pm_pmon_events[23]	C6:tbd	Number of cycles in which the SRE Prefetch Sequencer is at busy3 activity level.
pm_pmon_events[24]	C4:tbd	Number of cycles in which the SRE Execute Sequencer is at busy1 activity level.
pm_pmon_events[25]	C5:tbd	Number of cycles in which the SRE Execute Sequencer is at busy2 activity level.
pm_pmon_events[26]	C6:tbd	Number of cycles in which the SRE Execute Sequencer is at busy3 activity level.
pm_pmon_events[27]	C1:tbd	Number of cycles in which the DXE Prefetch Sequencer is at busy1 activity level.
pm_pmon_events[28]	C2:tbd	Number of cycles in which the DXE Prefetch Sequencer is at busy2 activity level.
pm_pmon_events[29]	C3:tbd	Number of cycles in which the DXE Prefetch Sequencer is at busy3 activity level.
pm_pmon_events[30]	C4:tbd	Number of cycles in which the DXE Execute Sequencer is at busy1 activity level.
pm_pmon_events[31]	C5:tbd	Number of cycles in which the DXE Execute Sequencer is at busy2 activity level.

17.4.9.1 System Memory Access Performance Monitoring

The Memory Interface Arbiter can issue up to 4 transactions on the system bus in pipelined fashion. The pm_pmon_events[2:0] and pm_pmon_events[6:4] signals provide an output indicating how many read and/or write transactions are pending (from 0 to 4) in each clock cycle. With a performance monitor counter attached to each of these signals and to pm_pmon_events[3] and pm_pmon_events[7], the average latency per read or write transaction can be calculated as follows:

average number of cycles of latency per read =

((pm_pmon_events[2] count * 4) + (pm_pmon_events[1] count * 2) + (pm_pmon_events[0] count * 1)) / pm_pmon_event[3] count

average number of cycles of latency per write =

((pm_pmon_events[6] count * 4) + (pm_pmon_events[5] count * 2) + (pm_pmon_events[4] count * 1)) / pm_pmon_event[7] count

17.4.9.2 Internal Sequencer Performance Monitoring

For each block within PM that contains a sequencer, 2 or 3 performance monitor events are provided. Together with the clock cycle counter in the central performance monitor block, these allow a user to measure how busy or idle each sub-block is, and identify potential bottlenecks. Three possible activity levels are identified for each sequencer to indicate how busy the sequencer is.

- Busy1 activity level:
  - asserted when the sequencer is busy (for example, not idle), OR has some input available to be processed.
  - negated when the sequencer is idle and there is no input to process.
- Busy2 activity level:
  - asserted when the busy1 activity level is asserted AND the sequencer's output channel is available to receive some output.
- Busy3 activity level:
  - asserted when the busy 2 activity level is asserted, AND the sequencer has not waited for system memory for longer than the excess memory latency threshold.

The Busy3 activity level is used only with sequencers that contain a memory read and/or write interface. Waiting for system memory consists of three parts, waiting to post a memory write request, waiting to post a memory read request, and waiting for read data to return once the read request is posted. In each case, the sequencer asserts an internal signal when it is stalled waiting for one of these three conditions.

For memory reads, sequencers launch one or more reads, then perhaps do other work, then check to see if some or all of the read data has returned, and then begin processing the data if it has arrived. The sequencer is considered to be waiting for memory read data to return when it is ready to process the data (such as not launching more reads and not doing other work) and is waiting for some or all of the read data to arrive.

There is likely a minimum number of cycles for each read transaction in which the sequencer is always waiting for memory, due to minimum memory read latency. So it may be interesting to see the number of cycles above a certain minimum per transaction in which waiting for memory occurs, this is the intent of the excess memory latency threshold. The excess memory latency threshold is programmable by the user, see . When a sequencer enters a state in which it is waiting for memory (for any of the 3 conditions listed above), a counter starts. When count >= programmed threshold, Busy3 activity level is negated. Count is reset and Busy3 activity level is asserted when the sequencer becomes unblocked, either by the memory read or write port becoming available to accept a request, or by receiving the read data that it has been waiting for. Implementing it this way means that a single counter can be used even when a sequencer launches multiple reads in pipelined fashion.

17.5 Initialization Information

17.5.1 Initialization Sequence

The common-control register space must be initialized prior to any channel specific registers, but thereafter channels can be initialized and used independently.

17.5.1.1 Common-Control Register Space

Initialization of the common-control register space can be generalized to the following steps;

- **SRE and DXE resource allocation:** system memory regions need to be allocated and corresponding registers initialized.
- **free buffer list sizes and mappings**: there are 8 physical free buffer lists that can be defined, each configured for buffers of a certain size. Each channel sees up to 2 virtual free buffer lists which are mapped to physical free buffer lists using FBLAAR and FBLABR registers - these same mappings define which virtual free buffer lists are enabled and which are act as the corresponding free buffer list's master. Each free buffer list should have a master, as this is used as a hint to software driving the channel that it should take responsibility for seeding the free buffer list with buffers (and cleaning up on teardown) - there are channel registers that reflect each channel's "virtual" view of these mappings.
- **miscellaneous configuration:** a variety of other common control registers need to be initialized, there is no particular ordering requirement (for example, channel-scheduling, cache-awareness, etc.)
- **interrupt initialization**: (see below)

17.5.1.2 Channel Register Space

Initialization of the channel register space can be generalized to the following steps;

- **resource allocation**: stream contexts and/or pattern-matching residues can either address-based or tabular. In the address-based mode contexts/residues are referred to by their memory location. In tabular mode, a region of memory is set aside and contexts/residues are referred to by index into the table. Command FIFO, Notification FIFO and Free Buffer Deallocate FIFO must be allocated from system memory. All these configurations must be programmed into the corresponding channel registers.
- **miscellaneous configuration**: a variety of other channel registers need to be initialized, there is no particular ordering requirement (for example, cache-awareness, etc.).
- **free buffer list initialization**: if the common-control registers have assigned one or two virtual free buffer list mappings for the channel, corresponding thresholds may be set. A service threshold can be set to prevent servicing of this channel's Command FIFO if a free buffer list has too few buffers (to avert truncation issues). If the channel is designated master for either or both virtual free buffer lists, it should enable the on deck behavior if it had been disabled during a previous reset. A low-water threshold can also be set to interrupt the channel when free buffer list length reaches it, typically when mastering a free buffer list (triggering the allocation of new buffers to the free buffer list).

- **interrupt initialization**: (see below)
- **Command FIFO enable**: the CRCR[0..3] (Section 17.3.8.17, “Channel Reset and Control Register Channel 0 (CRCR0)”) register(s) allow the channel's Command FIFO to be enabled or disabled, in addition to the virtual buffer free lists (ignored if they were not mapped/assigned by the common-control registers) and the Free Buffer Deallocate FIFO.

17.5.2 Interrupt Initialization

When initializing common-control or channel register space, it is wise to clear any stray interrupt bits whose sources might have been asserted during initial is at ion by writing-to-clear the IS (Interrupt Status) register. As long as the IE (Interrupt Enable) register is zero (the default/reset value) no interrupts occur—but clearing interrupt bits can prevent stray interrupts firing when interrupts are enabled.

17.5.2.1 Interrupt Registers

The common-control and channel register spaces have analogous interrupt registers and mechanisms, the differences between those in the common-control and channel spaces relate to the placement and significance of the interrupt bits themselves.

The set of interrupt registers allows for a variety of software models for managing interrupts and the corresponding interrupt status, in particular adaptive interrupt coalescing is possible as well.

- **IS (Interrupt Status) register**: this register contains those interrupt sources (or “reasons”) that have been asserted by the DMA Engine (DE). The only register that can alter the bits that are or are not set is the ISD register, see below. Once a bit is set in the IS register, it remains set until a write of 1 is performed to that bit of the IS register to clear it. If the source condition (level triggered) is still asserted, the IS bit is reasserted by the DE immediately after the write-to-clear.
- **IE (Interrupt Enable) register**: this register controls which bits in the IS register, when asserted by the DE, can trigger the channel's (or Common Control's) interrupt line to assert. This register does not influence which bits in the IS register are set, it purely controls which bits can trigger interrupts. On startup (or reset), this register is zero meaning that no interrupts occur until it is set non-zero, even if interrupt sources become asserted.
- **ISD (Interrupt Status Disable) register**: unlike IE, this register does influence which bits in the IS register are set. The ISD register suppresses the source conditions that assert bits in the IS register. As such, bits set in the ISD register prevents corresponding bits being asserted in the IS register and therefore no interrupt is asserted either, irrespective of whether the bit is enabled in the IE register or not.
- **II (Interrupt Inhibit) register**: this register doesn't share the same bit-mask definition as the previous registers - it is a boolean. When II is set, no interrupts are asserted even if matching bits are set in IS and IE that are not set in ISD. Unsetting the II register causes an interrupt to trigger if it had been suppressing such an interrupt beforehand. This is useful for disabling interrupts between an ISR (Interrupt Service Routine) and a corresponding dispatched handler. The ISR can set the II register to inhibit interrupts and the dispatched handler can unset it once finished. In this way, interrupt floods can be avoided and an adaptive interrupt-coalescing behavior observed without requiring the use of throttling measures that introduce unnecessary latency.

- **IIC (Interrupt Interval Control) register (not in common-control)**: this register can be used to set a minimum interval between successive assertions of the per-channel interrupt line for Command FIFO, Notification FIFO or Free Buffer Deallocate FIFO interrupt sources. If any of these interrupt source (set in IS and IE, not set in ISD, and II is off) asserts prior to this interval having expired since the previous interrupt assertion, the DE does not raise the interrupt line until the interval expires. Other interrupt sources assert immediately and no such interval-throttling occurs. It is possible to overwrite this mechanism (see Section 17.5.2.2.1, “Command FIFO,” for a more detailed description of the interval-throttling mechanism).

17.5.2.2 Channel FIFO Interrupts

Among the interrupt sources (bits) in a channel's register space are threshold-based triggers on the fill levels for the channel's Command FIFO, Notification FIFO, and Free Buffer Deallocate FIFO. These thresholds and interrupts are based on FIFO indices, each of which is either written by the DMA Engine (DE) for software or written by software for the DE.

The following briefly describes how the FIFO interrupt bits are managed.

17.5.2.2.1 Command FIFO

The Command FIFO has 3 associated index registers, CPI (Commands Produced Index), CCI (Commands Consumed Index), and CEI (Commands Expired index). CPI is written (for example, incremented) by software when the FIFO entry(ies) it was referring to has been populated with a command (for example, CPI refers to the FIFO entry immediately following the most recent command written by software). CCI is incremented by the DE beyond FIFO entries it has consumed, so CCI is equal to CPI if there are no unconsumed commands in the FIFO. CEI is incremented by software to acknowledge commands that have been consumed by the DE. CEI is equal to CCI if software is “aware” of all command-consumption that the DE has completed. Software can use CEI to determine when there is no space left to write commands—if CPI and CEI refer to the same FIFO entry but have different values, the FIFO is full.

The source condition for the Command FIFO interrupt is asserted if CEI and CCI differ. If the IIC register is set non-zero, then the corresponding interrupt does not asserted until the IIC interval has expired since the last interrupt assertion unless the difference between CEI and CCI exceeds the command FIFO threshold register (CFIFO_THRES[0..3]), in which case the interrupt-throttling interval is overridden and the interrupt is asserted immediately. If IIC is not set, the FIFO threshold register is ignored and the interrupt is asserted if and only if CEI and CCI differ. Note that command FIFO entries may set bits that override interval-throttling per-command and/or for the command's corresponding notification(s).

Note that the interrupt source asserts as soon as the hardware has consumed commands that the software hasn't yet expired. By throttling this interrupt source (clearing it in IE or toggling II) until command-expiries have completed, adaptive interrupt coalescing occurs—under light loads, the consumption of a single command by hardware triggers an interrupt, whereas under heavy load interrupts are only reenabled once the software has processed its existing work and this may allow multiple command-expiries to occur for each interrupt.

17.5.2.2.2 Notification FIFO

The Notification FIFO has 2 index registers, NPI (Notifications Produced Index) and NCI (Notifications Consumed Index). NPI is incremented by the DE when a FIFO entry has been populated with a notification. NCI is incremented by software to acknowledge notifications that have been consumed by software. If NCI is equal to NPI then software has consumed all the notifications produced thus far. If NCI and NPI refer to the same FIFO entry but differ in value, the Notification FIFO is full.

The Notification FIFO interrupt source (and any throttling of the associated interrupt-assertion) is consistent with the behavior described above for the Command FIFO interrupt source.

17.5.2.2.3 Free Buffer Deallocate FIFO

The Free Buffer Deallocate FIFO is used to “deallocate” free buffer list buffers from software to the hardware-managed free buffer lists in FBM (each FIFO entry can release multiple buffers at a time, pre-formatted in linked-list or scatter-gather form). This FIFO has 2 index registers, PI and CI that follow the same descriptions as per the Command FIFO.

The Free Buffer Deallocate FIFO interrupt source does not follow the semantics of the Command/Notification FIFO interrupt sources. The Free Buffer Deallocate FIFO interrupt source is asserted if the difference between CI and PI is less than the Free Buffer Deallocate FIFO threshold register (FBFIFO_THRES[0..3]) if it is set non-zero. If this threshold register is zero, this interrupt source is disabled. If the IIC register is set non-zero, then the corresponding interrupt is not asserted until the IIC interval has expired

17.5.2.3 Free Buffer List Depletion

The Free Buffer List Depletion interrupt source is asserted when the number of buffers on the free buffer list is less than or equal to a programmed threshold.

17.5.2.4 Channel Error Interrupts

The DE makes certain assurances before asserting channel error interrupt sources to simplify software handling. In particular, error interrupt sources are only asserted for errors that cause a channel to be brought down - errors that allow the channel to continue consuming commands do not assert any interrupt sources, their error information is contained entirely in the corresponding notification(s).

When an error is detected by the DE that asserts an error interrupt source, the DE ceases consuming commands from the channel's command FIFO, assigns error codes to notifications as appropriate for in-flight work and produces notifications as expected for all commands that have been consumed from the command FIFO. The DE also updates the appropriate CHERR[0..3] register with the error code.

The error interrupt source is not asserted until all such notifications have been produced and any command and/or notification FIFO interrupt sources are also asserted. Due to interrupt-coalescing, interrupt latency, or software throttling of interrupts (through selective toggling of IE register bits or the use of the II register), it is possible that command and/or notification interrupt sources coalesce with the error source in the IS register once the interrupt itself triggered the software's ISR—there may be outstanding expiry work remaining for software to complete as it sees the error interrupt source. However, any such command and/or notification expiry work can be completed at this point, the presence of the error interrupt source

itself is a guarantee that all outstanding expiry work is available—the FIFO interrupt sources are set, the FIFO indices are updated, and the DE has completed all work of this nature. Apart from running any outstanding expiry work as a first priority, error handling can therefore begin immediately upon detecting an error interrupt source - there is no need to poll for outstanding operations to complete, this was already performed by the DE prior to setting the error interrupt source.

17.5.2.5 Common-Control Error Interrupts

Certain errors have global consequences, or at least have the potential to be so (for example, detection of ill-formed pattern matching data structure). In such cases, the DE asserts a special per-channel error bit (Common Control Error bit) on all enabled channels as well as asserting an error bit in the common-control register space that defines which common-control error occurred. In this way, each channel can follow its conventional error-handling logic—the common-control error bit merely acts as a hint that the problem is not local to the channel itself. A channel is not able to go into reset while an error bit is asserted in the common-control interrupt status register. In this way, the common-control register space retains control over the system in reaction to the global error and only allows channels to be reset and reused once it clears the error bit(s). If a channel was already in reset when the common-control error was asserted, the channel error is asserted immediately upon exiting reset if the common-control error hadn't yet been cleared.

17.5.3 Bring Down Procedure

The asynchronous nature of the DMA Engine channel and the pipeline of work units that it supports generally require that any kind of cleanup (for teardown or reset) be asynchronous. With this in mind, it is worth thinking of the first step as “syncing”—ensuring that all work for that channel has ceased and that all resources associated with the channel are returned. The second stop is the teardown or reset, as required.

Note that the common-control register space has no concept of a reset, it is generally assumed that it is initialized prior to any channels and resources associated with the common-control space are released as a final step.

17.5.3.1 Syncing

Syncing would occur when the user has requested that a channel teardown or reset begin, or when an error interrupt has been received. Channel-syncing can be generalized to the following steps;

- **Command FIFO disable**: disable the Command FIFO via the CRCR[0..3] registers. This causes the DE to stop consuming commands from the Command FIFO. If an error interrupt has already occurred, then this step is not strictly necessary.
- **reap unconsumed commands**: there may be commands placed in the Command FIFO that haven't been consumed by the DMA Engine - these commands are between the CCI and CPI indices.
- **disable free buffer list interrupt threshold**: for any virtual free buffer list mastering, no more allocation should be attempted. So disabling the interrupt threshold (set the threshold to 0) prevents any interrupts that might provoke this.
- **wait**: syncing is complete when outstanding resources have been released and the FB Deallocate FIFO is idle. For any virtual free buffer list mastering, any other channel that is mapped to the same free buffer list must be fully synced before moving to the next step.

17.5.3.1.1 Teardown

Teardown assumes a channel has been synced. The following steps are involved;

- **disable interrupts**: this can be achieved by zeroing the IE (Interrupt Enable) register, setting bits in the ID (Interrupt Disable) register, or setting the II (Interrupt Inhibit) register.
- **deallocate channel resources**: as the channel has been synced, it is safe to deallocate FIFOs, as well as context and/or residue tables if tabular mode was used.
- **deallocate free buffer list resources**: if the channel was acting as master for either of its virtual free buffer lists, then upon teardown it should examine the on deck, head, and tail registers for the corresponding free buffer list(s) and iterate the physical free buffer list deallocating buffers. If the on deck buffer is filled, then it is the head of the free buffer list and can be iterated until the tail is found - in this case the list length register should be incremented by one to obtain the real list length. If the on deck buffer is not filled, head/tail provides the list extremities and the length register is accurate.
- **free buffer list reset**: if the channel was acting as master for either of its virtual free buffer lists, it should reset (through (FBL_ODD_A[0..3] registers) the free buffer list after deallocating its buffers to ensure that reinitialization of this channel (or other channels who share the free buffer list) do not use stale list information. Take note that care must be taken when performing teardown of a channel that is mastering a free buffer list - any other channel that is mapped to the same free buffer list must be fully synced before this teardown is possible.

17.5.3.2 Reset

As with teardown, a channel reset presumes that the channel has been synced. If a reset is attempted before all resources have been returned then leaks are possible, and if a reset is attempted before FIFO activity is idle, then the DMA Engine attempts to perform memory transactions to addresses that are being reset. The basic steps for a reset are;

- **disable interrupts**: as per teardown
- **channel reset**: setting then unsetting the channel reset bit resets the channel's register space to default settings.
- **reprogram channel registers**: the same steps apply here as for channel initialization except that resource allocation should not be required, even when mastering freelists.

The differences between a reset and a full teardown-initialization cycle are that (i) resources do not need to be deallocated so nor do they need to be reallocated, and (ii) as a result, physical free buffer lists can persist across a reset of the free buffer list's master and so no ordering is required in the resetting of channels with regard to which is acting as the master.

17.6 Application Information

This is additional information intended for use by the device or SoC customer, and is generally not included in functional verification.

Chapter 18
Table Lookup Unit

18.1 Introduction

The table lookup units (TLUs) provide access to application-defined routing topology, control, and statistics tables in external memory. This device offers two instantiations of the TLU described in this chapter. The TLU accesses external memory arrays attached to either the device DDR memory controller or the local bus controller (LBC). Communication between the CPU and the TLU occurs via messages passed through the TLU memory-mapped configuration and status registers. The TLU uses a 64-bit wide data path for such register accesses.

The TLU supports several types of table lookup algorithms and provides resources for efficient generation of table entry addresses in memory, hash generation of addresses, and binary table searching algorithms for both exact-match and longest-prefix match strategies.

Figure 18-1 shows a block diagram of the TLU.

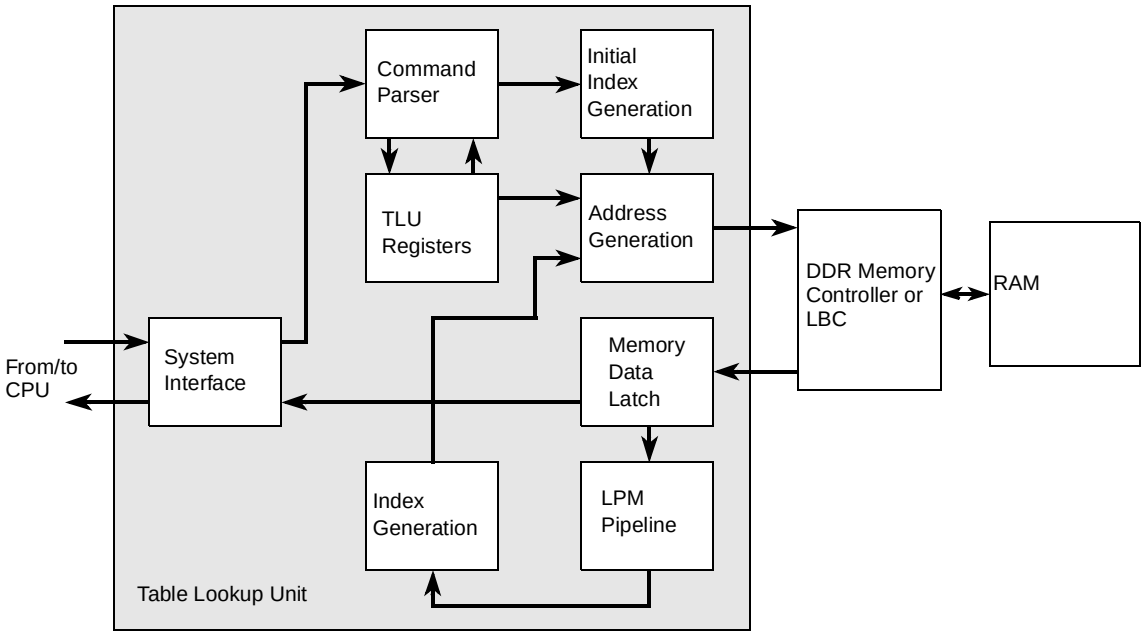


Figure 18-1. Table Lookup Unit Block Diagram

The blocks of the TLU allows the implementation of a variety of table lookup algorithms to meet different application needs. In general, its functional blocks are organized in a basic loop that performs the following functions:

1. Parses a TLU command issued by software via registers
2. Calculates the initial index based on a key (for example, an initial hash value)
3. Evaluates the current table node as follows:
   a) Fetches memory data at the current index
   b) Fetches a portion of the key
   c) If the lookup algorithm is complete, goes to step 5; otherwise goes to step 4
4. Calculates a new index, and then goes back to step 3
5. Fetches the data at the current index
6. Returns the data to the CPU via key/data buffer registers

18.2 Features

The TLU has these distinctive features:

- Dedicated, low-latency interface to the local bus controller (LBC)
  - All local bus memory technologies supported at up to 167 MHz and 32 bits, with optional parity checking
  - TLU table accesses occur without degradation of DDR bandwidth to the CPU and remaining system
  - Table memory organized into four independent banks, on arbitrary 256-Mbyte boundaries, allowing access to 4 Gbytes of local bus memory
- Per-bank selection of interface to general system memory
  - Multiple RAM technologies supported via DDR memory controller
  - Table memory banks can access 64 Gbytes of address space
  - Each of the four table memory banks can be configured to access LBC or DDR controller targets independently, allowing use of mixed memory
- Four TLU complex table types supported
  - Hash-trie-key table for hash-based exact-match algorithms
  - Chained hash table for partially indexed and hashed exact-match algorithms
  - Variable prefix-expansion trie table for longest prefix match algorithm
  - Flat data table for retrieving search results and simple indexed algorithms
- Flexible command set
  - Direct table entry reads and writes for safe in-place updates
  - Key find operations on 32, 64, 96, and 128-bit keys, with don’t care bits masked to zero
  - Key find with post write or post add operations
  - TLU registers are aligned to allow 64-bit accesses to replace a pair of 32-bit accesses for improved performance.

- High-performance hash capability
  - Ultra-robust hash function features reversible mixing for funneling immunity and avalanche of 1/3 to 2/3 of index bits per bit changed in the key
  - Hash function minimizes collisions on both real-world data and pathological patterns, and offers superior randomization over cyclic-redundancy codes
  - Incremental hash algorithm allows hashes from 64 bits to 2048 bits to be constructed in 64-bit increments
  - Hashes of 128-bit keys computed in 6 system clock cycles
- Individually configurable table structures
  - Up to 32 physical tables available, held in any of the 4 memory banks
  - Sub-tables of varying kinds may be nested inside physical tables as required by applications
  - 8, 16, 32, and 64-byte table entry sizes supported

18.3 Modes of Operation

The TLU supports thirty-two (32) physical tables, numbered PTBL0–PTBL31, each of which has an associated configuration register. Within the configuration register for each physical table is a base table address configured by software. Also, in the configuration registers is the initial table format to expect, the method of generating the initial index, and how an index value maps to a physical memory address by selection of one of four external memory banks. This means that all table entries can be accessed by an index value, and individual bytes by an offset value. The entries can range from 8 bytes to 64 bytes in length.

In general, the TLU performs a lookup using the nine steps listed below:

1. Parses the command issued via the CMDOP (and possibly CMDIX) register as follows:
   - Extracts all of the information it needs to perform its functions from the command registers.
   - Determines type of command (for example, find, findr, findw).
   - Determines table ID and key value.
   - Determines which data is expected as arguments and which data should be returned upon completion.
   - Based on the address of the command, determines to which buffer data should be returned.
2. Uses the TBL number from the CMDOP register as an index into the 32 physical tables.
3. Reads the configuration registers. The TLU then uses the table ID and its associated configuration registers to determine:
   - The initial data format to expect.
   - The memory bank in which the table resides.
   - The base address of the table in the associated memory bank.
   - How table addressing works for this table.
4. Uses the current format as a state and computes:
   - Next index to read = function of (key value, key size, data format, configuration registers).
5. Converts the index to an address using:

— Address = (bank base address memory page number * 256 M) +
(table base address memory page number * 4096) + next index * (size of entry).

6. Reads the memory data.
7. Computes the next index to read using:
   — Next index to read = function of (key value, memory data, current format).
8. Computes next format based on: key value, key size, current format, and memory data.
9. Performs the required find response:
   — Find, returns current index value in CSTAT[INDEX].
   — Findr, read from current index value and returns that data with the index.
   — Findw, write the doubleword of data from the key/data buffer at the current index value.

NOTE

This occurs when the table format and TLU state is one of data.

TLU operations are driven by TLU registers. These are detailed in the following sections.

18.4 Memory Map/Register Definition

The TLU registers must be accessed with aligned 32-bit or 64-bit accesses. In the case of 64-bit accesses an adjacent pair of registers may be written and read simultaneously. Writes to reserved register bits must always store 0, as writing 1 to reserved bits may have unintended side-effects. Reads from unmapped register addresses return zero. Unless otherwise specified, the read value of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

18.4.1 Top-Level Module Memory Map

4 Kbytes of memory-mapped space is allocated for each TLU register space, starting at offset 0x2_F000 from CCSRBAR for TLU1 and offset 0x1_5000 from CCSRBAR for TLU2. This space is further divided as indicated in Table 18-1.

Table 18-1. Module Memory Map Summary

Address Offset	Function
000–0FF	TLU general control/status registers
100–1FF	TLU physical table configuration registers
200–2FF	—
300–3FF	—
400–4FF	—
500–5FF	TLU statistics counters
600–6FF	TLU command/response registers
700–7FF	—

Table 18-1. Module Memory Map Summary (continued)

Address Offset	Function
800–8FF	TLU key/data registers
900–FFF	—

18.4.2 Detailed Memory Map—Control/Status Registers

Table 18-2 lists the address, name, and a cross-reference to the complete description of each register.

Table 18-2. Module Memory Map

Offset	Name	Access [1]	Reset	Section/Page
TLU1 block base address: 0x2_F000				
TLU General Control/status Registers				
0x2_F000	TLU_ID1—TLU Identifier1 register	R	0x002F_0100	18.4.3.1.1/18-8
0x2_F004	TLU_ID2—TLU Identifier2 register	R	0x0000_0000	18.4.3.1.2/18-9
0x2_F008–0x2_F00C	Reserved	R	0x0000_0000	—
0x2_F010	IEVENT—Interrupt event register	R/W	0x0000_0000	18.4.3.1.3/18-9
0x2_F014	IMASK—Interrupt mask register	R/W	0x0000_0000	18.4.3.1.4/18-10
0x2_F018	IEATR—Interrupt error attributes register	R/W	0x0000_0000	18.4.3.1.5/18-11
0x2_F01C	IEADD—Interrupt error address register	R/W	0x0000_0000	18.4.3.1.6/18-12
0x2_F020	IEDIS—Interrupt error disable register	R/W	0x0000_0000	18.4.3.1.7/18-12
0x2_F024–0x2_F03C	Reserved	R	0x0000_0000	—
0x2_F040	MBANK0—Memory bank 0 base register	R/W	0x0000_0000	18.4.3.1.8/18-13
0x2_F044	MBANK1—Memory bank 1 base register	R/W	0x0000_0000	18.4.3.1.8/18-13
0x2_F048	MBANK2—Memory bank 2 base register	R/W	0x0000_0000	18.4.3.1.8/18-13
0x2_F04C	MBANK3—Memory bank 3 base register	R/W	0x0000_0000	18.4.3.1.8/18-13
0x2_F050–0x2_F0FC	Reserved	R	0x0000_0000	—
TLU Physical Table Configuration Registers				
0x2_F100	PTBL0—Physical table 0 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F104	PTBL1—Physical table 1 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F108	PTBL2—Physical table 2 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F10C	PTBL3—Physical table 3 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F110	PTBL4—Physical table 4 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F114	PTBL5—Physical table 5 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14

Table 18-2. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_F118	PTBL6—Physical table 6 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F11C	PTBL7—Physical table 7 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F120	PTBL8—Physical table 8 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F124	PTBL9—Physical table 9 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F128	PTBL10—Physical table 10 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F12C	PTBL11—Physical table 11 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F130	PTBL12—Physical table 12 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F134	PTBL13—Physical table 13 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F138	PTBL14—Physical table 14 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F13C	PTBL15—Physical table 15 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F140	PTBL16—Physical table 16 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F144	PTBL17—Physical table 17 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F148	PTBL18—Physical table 18 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F14C	PTBL19—Physical table 19 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F150	PTBL20—Physical table 20 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F154	PTBL21—Physical table 21 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F158	PTBL22—Physical table 22 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F15C	PTBL23—Physical table 23 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F160	PTBL24—Physical table 24 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F164	PTBL25—Physical table 25 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F168	PTBL26—Physical table 26 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F16C	PTBL27—Physical table 27 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F170	PTBL28—Physical table 28 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F174	PTBL29—Physical table 29 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F178	PTBL30—Physical table 30 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F17C	PTBL31—Physical table 31 configuration register	R/W	0x0000_0000	18.4.3.2.1/18-14
0x2_F180–0x2_F4FC	Reserved	R	0x0000_0000	—
TLU Statistics Counters				
0x2_F500	CRAMR—Memory reads count register	R/W	0x0000_0000	18.4.3.3.1/18-16
0x2_F504	CRAMW—Memory writes count register	R/W	0x0000_0000	18.4.3.3.2/18-16
0x2_F508	CFIND—Total find count register	R/W	0x0000_0000	18.4.3.3.3/18-17
0x2_F50C	Reserved	R	0x0000_0000	—

Table 18-2. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_F510	CTHTK—Hash/index-trie-key table find count register	R/W	0x0000_0000	18.4.3.3.4/18-17
0x2_F514	CTCRT—CRT table find count register	R/W	0x0000_0000	18.4.3.3.5/18-18
0x2_F518	CTCHS—Chained hash table find count register	R/W	0x0000_0000	18.4.3.3.6/18-18
0x2_F51C	CTDAT—Flat data table lookup count register	R/W	0x0000_0000	18.4.3.3.7/18-19
0x2_F520–0x2_F52C	Reserved	R	0x0000_0000	—
0x2_F530	CHITS—Successful find count register	R/W	0x0000_0000	18.4.3.3.8/18-19
0x2_F534	CMISS—Failed find count register	R/W	0x0000_0000	18.4.3.3.9/18-20
0x2_F538	CHCOL—Hash collision count register	R/W	0x0000_0000	18.4.3.3.10/18-20
0x2_F53C	CCRTL—CRT level count register	R/W	0x0000_0000	18.4.3.3.11/18-21
0x2_F540–0x2_F5EC	Reserved	R	0x0000_0000	—
0x2_F5F0	CARO—Counter carries-out register	R/W	0x0000_0000	18.4.3.3.12/18-21
0x2_F5F4	CARM—Counter carry mask register	R/W	0xFFF0_0000	18.4.3.3.13/18-22
0x2_F5F8–0x2_F5FC	Reserved	R	0x0000_0000	—
TLU Command/Response Registers				
0x2_F600	CMDOP—TLU command operation register	R/W	0x0000_0000	18.4.3.4.1/18-23
0x2_F604	CMDIX—TLU command index register	R/W	0x0000_0000	18.4.3.4.2/18-25
0x2_F608	Reserved	R	0x0000_0000	—
0x2_F60C	CSTAT—TLU command status and response register	R/W	0x8000_0000	18.4.3.4.3/18-25
0x2_F610–0x2_F7FC	Reserved	R	0x0000_0000	—
TLU Key/Data Registers				
0x2_F800	KD0B—Key/data word 0 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F804	KD1B—Key/data word 1 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F808	KD2B—Key/data word 2 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F80C	KD3B—Key/data word 3 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F810	KD4B—Key/data word 4 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F814	KD5B—Key/data word 5 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F818	KD6B—Key/data word 6 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F81C	KD7B—Key/data word 7 buffer register	R/W	0x0000_0000	18.4.3.5.1/18-26
0x2_F820–0x2_FFFC	Reserved	R	0x0000_0000	—
TLU2 offers the same registers but beginning at block base address: 0x1_5000				

[1] R = means read-only, R/W = read and write.

18.4.3 TLU Register Descriptions

This section provides a detailed description of all the TLU registers.

18.4.3.1 TLU General Control/Status Registers

This section describes the general control and status registers used for controlling the TLU. All of the registers are 32 bits wide, but pairs of registers may be accessed as 64-bit double words.

The TLU provides a flexible error reporting and interrupt mechanism. Non-error events, such counter overflow (COV), directly set the event register, IEVENT, and affect no other error registers. Error events, however, are initially gated by the settings in the error disable register, IEDIS. If an error event is not disabled, it sets the corresponding event in IEVENT, and at the same time locks the error attributes into IEATR and the failing index into IEADD. After an error is locked in this manner, no further errors can update the IEVENT, IEATR, and IEADD registers until the IEATR[V] flag is cleared by software. An interrupt is generated only in the case where events in IEVENT are also enabled in the IMASK register. Normally software should first write 1's to clear pending IEVENT events, and then clear IEATR[V] before returning from a TLU interrupt service routine. Should software choose not to service interrupts, all bits in IMASK may be cleared, and IEVENT can be polled periodically.

18.4.3.1.1 TLU Identifier1 Register (TLU_ID1)

The TLU identifier1 register (TLU_ID1) is a read-only register. The TLU_ID1 register is used to identify the TLU block and revision. Figure 18-2 describes the definition for the TLU_ID1 register.

Offset TLU1: 0x2_F000
TLU2: 0x1_5000

Access: Read only

	0															15	16							23	24							31
R	TLU_ID																TLU_REV_MJ								TLU_REV_MN							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 18-2. TLU_ID1 Register Format

Table 18-3 describes the fields of the TLU_ID1 register.

Table 18-3. TLU_ID1 Field Descriptions

Bits	Name	Description
0–15	TLU_ID	Value identifies the table lookup unit (TLU) 0x002F Unique identifier for TLU.
16–23	TLU_REV_MJ	Value identifies the major revision number of the TLU. 0x01 Initial revision
24–31	TLU_REV_MN	Value identifies the minor revision number of the TLU. 0x00 Initial revision

18.4.3.1.2 TLU Identifier2 Register (TLU_ID2)

The TLU identifier2 register (TLU_ID2) is a read-only register. The TLU_ID2 register is used to identify TLU capabilities and the TLU environment.

Figure 18-3 describes the definition for the TLU_ID2 register.

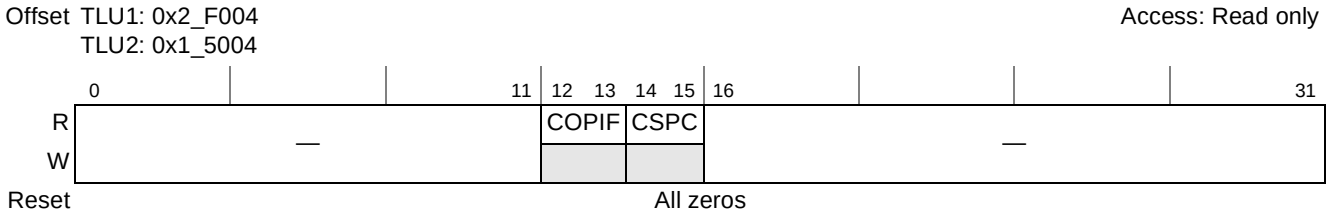


Figure 18-3. TLU_ID2 Register Format

Table 18-4. TLU_ID2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–13	COPIF	Co-processor interfaces attached. 00 No coprocessors—software access only. 01 Reserved. 10 Reserved. 11 Reserved.
14–15	CSPCS	Number of command spaces for software access. 00 1 space. 01 Reserved. 10 Reserved. 11 Reserved.
16–31	—	Reserved

18.4.3.1.3 Interrupt Event Register (IEVENT)

Interrupt events cause bits in the IEVENT register to be set. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the interrupt mask register (IMASK), the event also causes an interrupt at the PIC. A bit in the interrupt event register is cleared by writing a 1 to that bit position. A write of 0 has no ffect.

Figure 18-4 describes the definition for the IEVENT register.

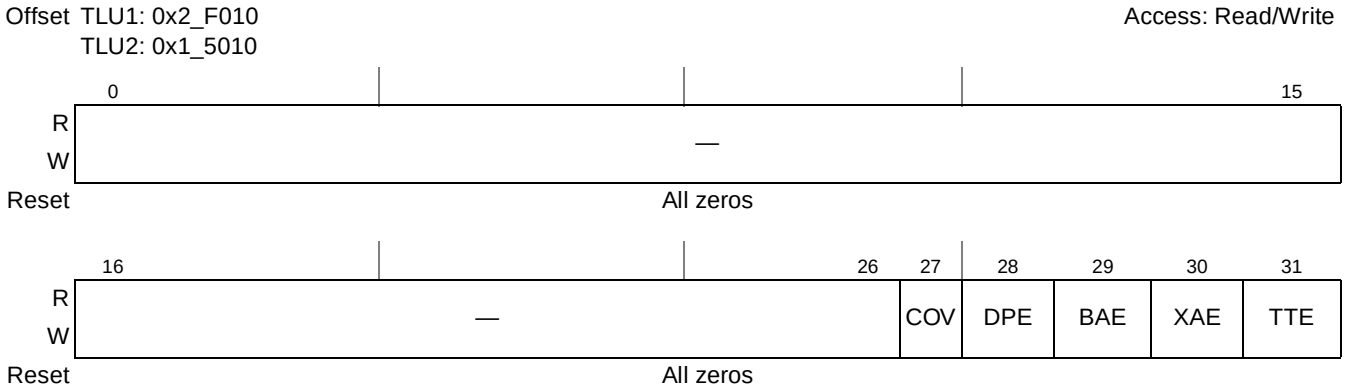


Figure 18-4. IEVENT Register Format

Table 18-5 describes the fields of the IEVENT register.

Table 18-5. IEVENT Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	COV	Counter overflow. 0 No statistics counter overflowed. 1 One or more of the statistics counters overflowed. The corresponding bits in the CARO register indicate which counters caused this event.
28	DPE	Data parity error on table read. 0 No error. 1 Data parity error occurred on a memory read.
29	BAE	Bad access error on table read or write. 0 No error. 1 An error occurred on a memory read or write due to address out of range or bus time-out.
30	XAE	Excessive number of memory accesses error. 0 No error. 1 More than 255 memory accesses were attempted to satisfy a command.
31	TTE	Unsupported physical table type error. 0 No error. 1 The table type, TYPE, of the selected PTBL was not defined as a known type for a command that accessed the table. This can occur if the PTBL is not initialized.

18.4.3.1.4 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events in the IEVENT register are permitted to participate in generating interrupts to the PIC. All implemented bits in this register are read/write and cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, the PIC receives an interrupt from the TLU. The internal interrupt signal remains asserted until either the IEVENT bit is cleared, by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Figure 18-5 describes the definition for the IMASK register.

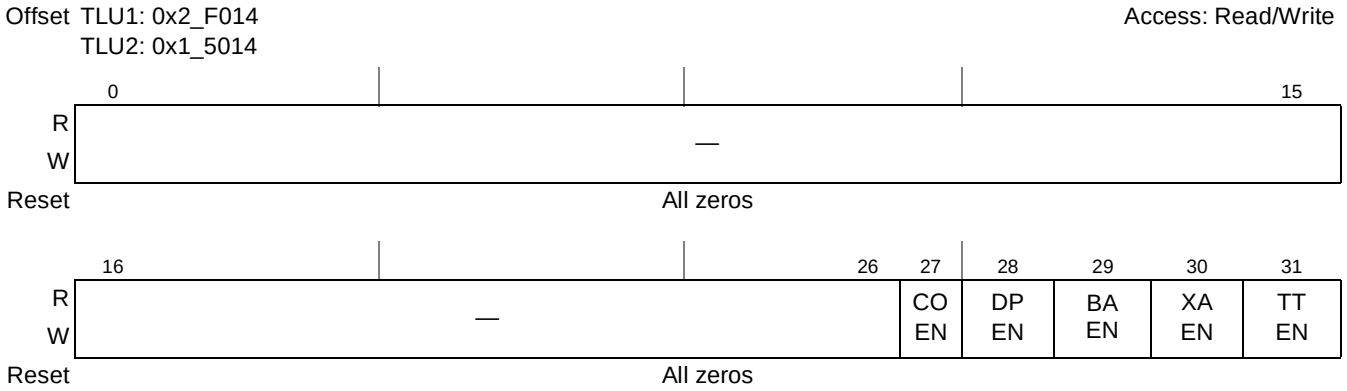


Figure 18-5. IMASK Register Format

Table 18-6 describes the fields of the IMASK register.

Table 18-6. IMASK Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	COEN	Statistics counter overflow interrupt enable.
28	DPEN	Data parity error on table read interrupt enable.
29	BAEN	Bad access error on table read or write interrupt enable.
30	XAEN	Excessive number of memory accesses error interrupt enable.
31	TTEN	Unsupported physical table type error interrupt enable.

18.4.3.1.5 Interrupt Error Attributes Register (IEATR)

The interrupt error attributes register (IEATR) is a read/write register, which indicates the status and attributes of error events in IEVENT. An error event is one that causes IEATR[V] to be set. In the TLU, all events in Table 18-5 other than IEVENT[COV] are considered error events. The fields in both the IEATR and IEADD registers are defined and remain frozen in relation to the last error until IEATR[V] is cleared by software. Errors that occur following the first setting of IEATR[V] are not queued and thus not captured by any register. It is not necessary to enable interrupts in IMASK in order to capture error events, but if an event is to be captured it must not be disabled in the IEDIS register.

Figure 18-6 describes the definition for the IEATR register.

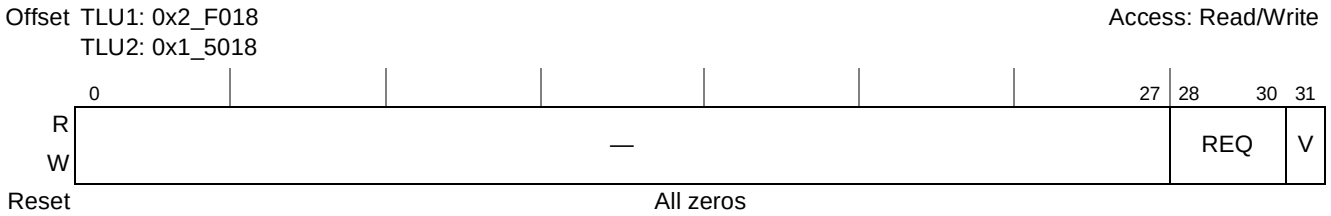


Figure 18-6. IEATR Register Format

Table 18-7 describes the fields of the IEATR register.

Table 18-7. IEATR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–30	REQ	Requestor code. Determines which initiating entity or command set was associated with the error event. 000 An access through command set 0 caused the error. 001–111 Reserved
31	V	Error attribute capture is valid. Indicates that the captured error information is valid. Clear this bit to re-enable capture of error events (in IEVENT) and associated attributes. 0 Error attributes are not valid. 1 Captured error attributes are valid and frozen until bit V is cleared by software.

18.4.3.1.6 Interrupt Error Address Register (IEADD)

The interrupt error address register (IEADD) is a read/write register which indicates which physical table and access location gave rise to the error events in IEVENT. The value of this register is valid only if IEATR[V] is set, and its value remains frozen in relation to the last error until IEATR[V] is cleared by software.

Figure 18-7 describes the definition for the IEADD register.

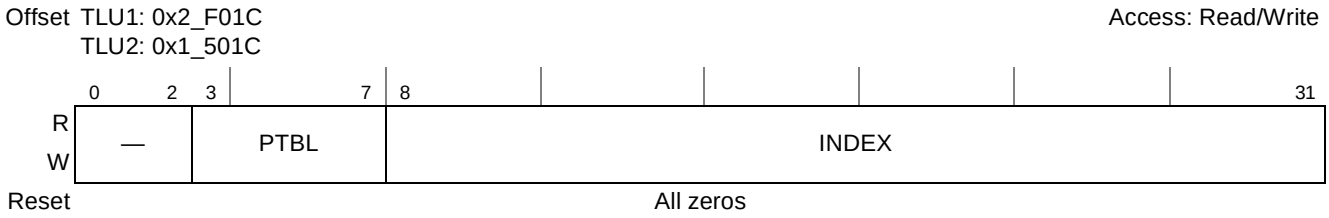


Figure 18-7. IEADD Register Format

Table 18-8 describes the fields of the IEADD register.

Table 18-8. IEADD Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	PTBL	Physical table number associated with the current error condition or access. Valid only when IEATR[V] is set.
8–31	INDEX	Index of double-word (8-byte) location accessed in table PTBL, even if the table entries are larger than 8 bytes. The table base address must be added to INDEX in order to form a full address for the error. INDEX is defined only in the case of BAE or DPE events, and INDEX is valid only when IEATR[V] is set.

18.4.3.1.7 Interrupt Error Disable Register (IEDIS)

The interrupt error disable register provides control over which possible error events are recognized by the TLU when they occur. All implemented bits in this register are read/write and cleared upon a hardware

reset. If an error condition occurs and the corresponding bit in IEDIS is set, the TLU does not recognize the event, and neither IEVENT, IEATR, nor IEADD are affected. Software should clear IEDIS to enable all error detection.

Figure 18-8 describes the definition for the IEDIS register.

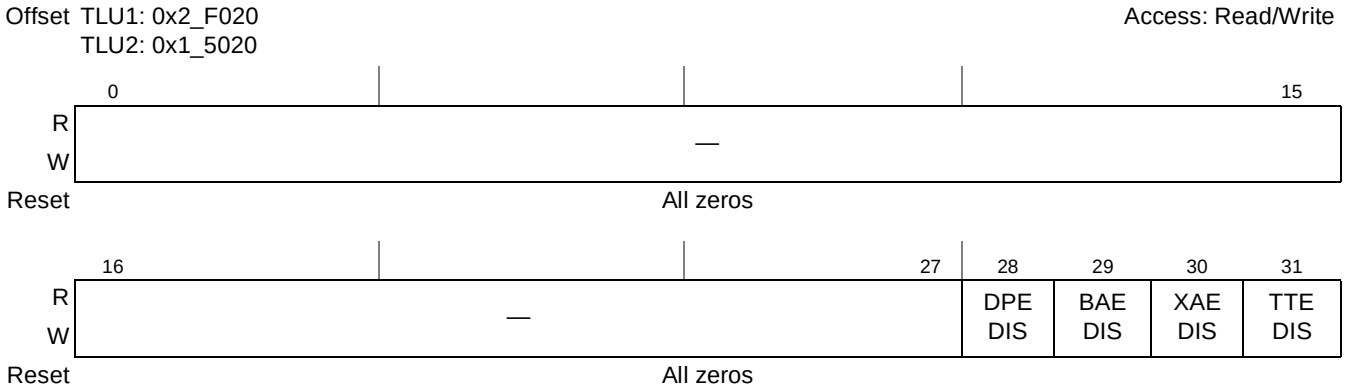


Figure 18-8. IEDIS Register Format

Table 18-6 describes the fields of the IEDIS register.

Table 18-9. IEDIS Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	DPEDIS	Data parity error on table read disable.0Error detection enabled for data parity errors on table reads 1 Error detection disabled for data parity errors on table reads
29	BAEDIS	Bad access error on table read or write error disable.0Error detection enabled for bad access errors on table reads or writes 1 Error detection disabled for bad access errors on table reads or writes
30	XAEDIS	Excessive number of memory accesses error disable.0Error detection enabled for excessive number of memory accesses errors 1 Error detection disabled for excessive number of memory accesses errors
31	TTEDIS	Unsupported physical table type error disable.0Error detection enabled for unsupported physical table type errors 1 Error detection disabled for unsupported physical table type errors

18.4.3.1.8 Memory Bank 0–3 Base Registers (MBANK0–MBANK3)

The memory bank 0–3 base registers (MBANK0–MBANK3) are read/write registers that define the base address of each bank used for physical table storage. Banks may be located on any 256-Mbyte boundary in physical memory with the exception of on-chip SRAM space, and should correspond with memory regions mapped by the LBC when TGT = 0, or the DDR controller when TGT = 1. For example, a bank may map directly to a local bus chip select.

Figure 18-9 describes the definition for the MBANK*n* registers.

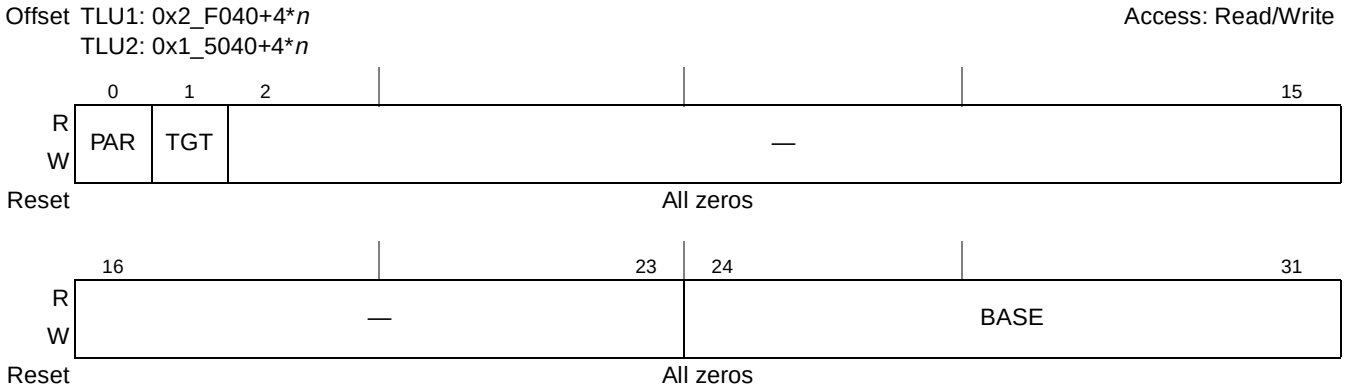


Figure 18-9. MBANK*n* Register Format

Table 18-10 describes the fields of each MBANK*n* register.

Table 18-10. MBANK*n* Field Descriptions

Bits	Name	Description
0	PAR	Parity checking on memory reads from this bank. 0 Disables checks for memory parity on reads, and disables IEVENT[DPE]. 1 Enables checks for memory parity on reads, which can result in IEVENT[DPE] events in the case of parity errors.
1	TGT	Memory target assignment for this bank. 0 All memory reads and writes to this bank occur via the LBC. Use this setting for highest performance. 1 All memory reads and writes to this bank occur via the system DDR memory controller.
2–23	—	Reserved
24–31	BASE	The base address of this memory bank is BASE * 256 Mbytes. That is, the 28 least significant bits of the 36-bit bank base address are assumed to be zero.

18.4.3.2 TLU Physical Table Configuration Registers

Each memory-mapped physical table accessible by the TLU is configured by registers in this section. Physical tables are established by software, typically prior to using the TLU for lookups. All of the registers are 32 bits wide, but pairs of registers may be accessed as 64-bit double words.

18.4.3.2.1 Physical Table 0–31 Configuration Registers (PTBL0–PTBL31)

The physical table 0–31 configuration registers (PTBL0–PTBL31) are read/write registers. Each of the 32 PTBL*n* registers defines the location and attributes of physical table *x*. Every physical table configured through these registers must be initialized prior to find commands being issued.

Figure 18-10 describes the definition for each PTBL*n* register.

Offset TLU1: 0x2_F100+4*n*
TLU2: 0x1_5100+4*n*

Access: Read/Write

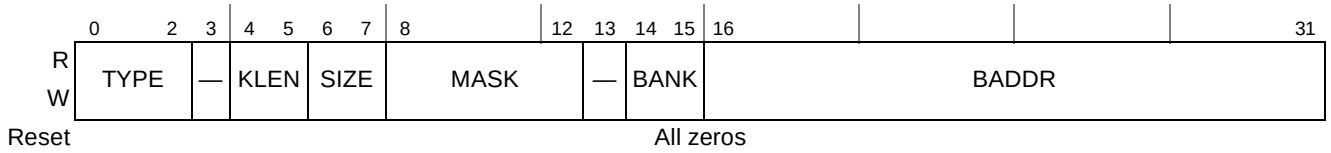


Figure 18-10. PTBL*n* Register Format

Table 18-11 describes the fields of the PTBL*n* registers.

Table 18-11. PTBL*n* Field Descriptions

Bits	Name	Description
0–2	TYPE	Table type for this table. 000 Uninitialized table. 001 Flat data table. 010 CRT (longest-prefix match or chained hash) table. 011 Reserved 100 Hash table. 101 Reserved 110 Reserved. 111 Reserved
3	—	Reserved
4–5	KLEN	Key length for searching this table. Keys shorter than the indicated length should be padded to the right with zero-bits. KLEN is ignored in the case of TYPE = flat data, as keys are always assumed to be 32 bits for such tables. 00 32-bit keys. 01 64-bit keys. 10 96-bit keys. 11 128-bit keys.
6–7	SIZE	Size of individual table entries when the TLU is accessing data or key/data simple tables. Each index into a data table addresses an entry of the size given. This field is ignored for addressing other simple tables used in lookups. 00 8 bytes. 01 16 bytes. 10 32 bytes. 11 64 bytes.
8–12	MASK	Mask to be applied to the key. • For a CRT table, this field corresponds to the KEYSHL field in the CRT data entry format; allowable range = 0 to 15. That is, every level of CRT table traversed consumes another MASK+1 left-most bits of the key and shifts it left by that amount. • For a hash table, the size of the table, in 32-bit hash entries, is determined by 2^{MASK}; allowable range of MASK = 0 to 16, permitting the initial table to be as large as 65,536 entries or 32,768 doublewords. • For a data table, the size of the table is determined by 2^{MASK}; allowable range of MASK = 0 to 24, permitting the initial table to be as large as 16,777,216 entries. • This field is unused for external tables.
13	—	Reserved

Table 18-11. PTBL*n* Field Descriptions (continued)

Bits	Name	Description
14–15	BANK	Memory bank selector for locating this table. See Section 18.4.3.1.8, “Memory Bank 0–3 Base Registers (MBANK0–MBANK3),” on page 18-13 for definitions of the banks. 00 The table is located in memory bank 0, whose address starts at MBANK0. 01 The table is located in memory bank 1, whose address starts at MBANK1. 10 The table is located in memory bank 2, whose address starts at MBANK2. 11 The table is located in memory bank 3, whose address starts at MBANK3.
16–31	BADDR	Defines the base address of the table in memory. The base address is defined as BADDR * 4 Kbytes from the start of the memory bank selected by BANK. The upper bounds of the table are not defined, hence tables can potentially wrap around the end of the bank.

18.4.3.3 TLU Statistics Counters

The TLU collects statistics of its memory accesses and table lookup performance. Software may use these statistics for the purpose of fine-tuning the performance of TLU-assisted applications. All of the registers are 32 bits wide, but pairs of registers may be accessed as 64-bit double words.

18.4.3.3.1 Memory Reads Count Register (CRAMR)

The memory reads count register (CRAMR) is a read/write register that counts the number of memory read transactions that occurred in response to any TLU command. Over a time interval, this count can be used to measure the TLU’s memory utilization for lookups.

Figure 18-11 describes the definition for the CRAMR register.

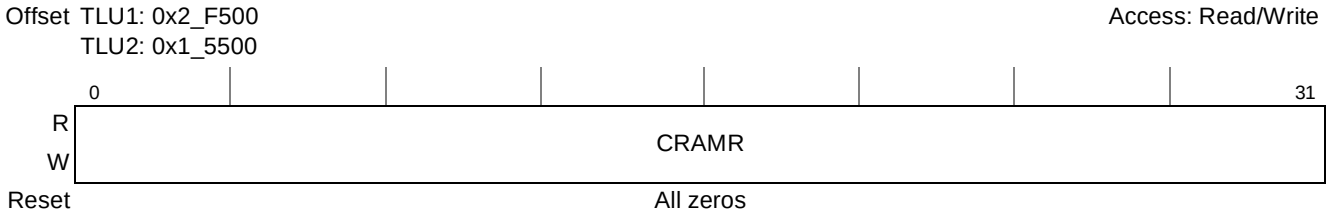


Figure 18-11. CRAMR Register Format

Table 18-12 describes the fields of the CRAMR register.

Table 18-12. CRAMR Field Descriptions

Bits	Name	Description
0–31	CRAMR	Memory read counter. Increments each time a memory read is issued by the TLU.

18.4.3.3.2 Memory Writes Count Register (CRAMW)

The memory writes count register (CRAMW) is a read/write register that counts the number of memory write transactions that occurred in response to any TLU command.

Figure 18-12 describes the definition for the CRAMW register.

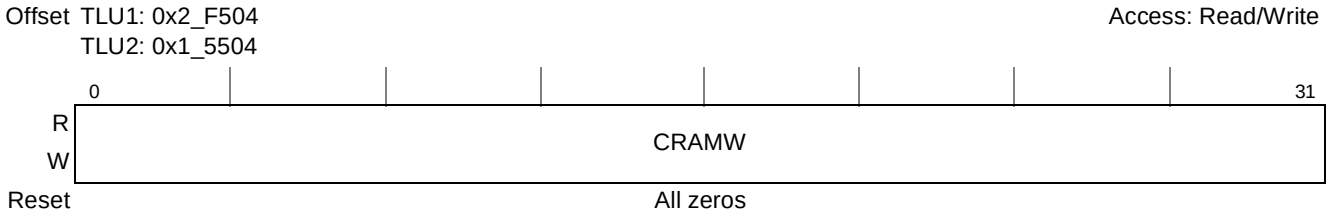


Figure 18-12. CRAMW Register Format

Table 18-13 describes the fields of the CRAMW register.

Table 18-13. CRAMW Field Descriptions

Bits	Name	Description
0–31	CRAMW	Memory write counter. Increments each time a memory write is issued by the TLU.

18.4.3.3.3 Total Find Count Register (CFIND)

The total find count register (CFIND) is a read/write register that counts the number of find/r/w commands issued to the TLU.

Figure 18-13 describes the definition for the CFIND register.

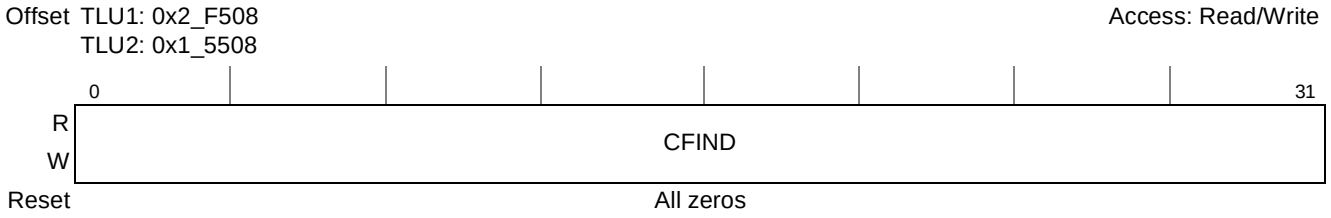


Figure 18-13. CFIND Register Format

Table 18-14 describes the fields of the CFIND register.

Table 18-14. CFIND Field Descriptions

Bits	Name	Description
0–31	CFIND	Find command counter. Increments each time a find/r/w command is issued.

18.4.3.3.4 Hash-Trie-Key Table Find Count Register (CTHTK)

The hash-trie-key table find count register (CTHTK) is a read-write register that counts the number of find/r/w commands that access an initial table type of hash-trie-key.

Figure 18-14 describes the definition for the CTHTK register.

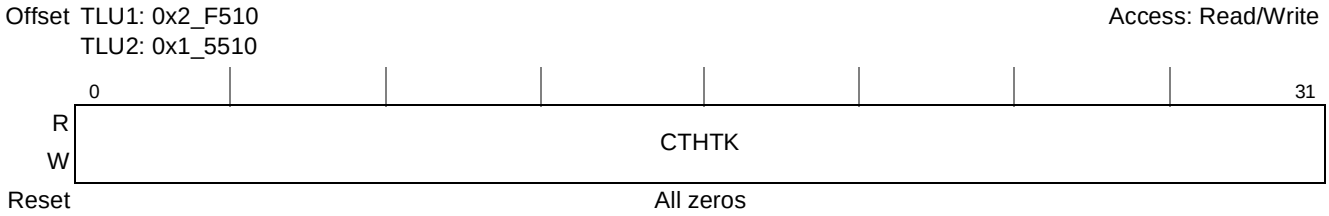


Figure 18-14. CTHTK Register Format

Table 18-15 describes the fields of the CTHTK register.

Table 18-15. CTHTK Field Descriptions

Bits	Name	Description
0–31	CTHTK	Hash-trie-key table find command counter. Increments each time a find/r/w command is issued on a table of type hash-trie-key.

18.4.3.3.5 CRT Table Find Count Register (CTCRT)

The CRT table find count register (CTCRT) is a read/write register that counts the number of find/r/w commands that access an initial table type of CRT and do not transition to a hash table before completion.

Figure 18-15 describes the definition for the CTCRT register.

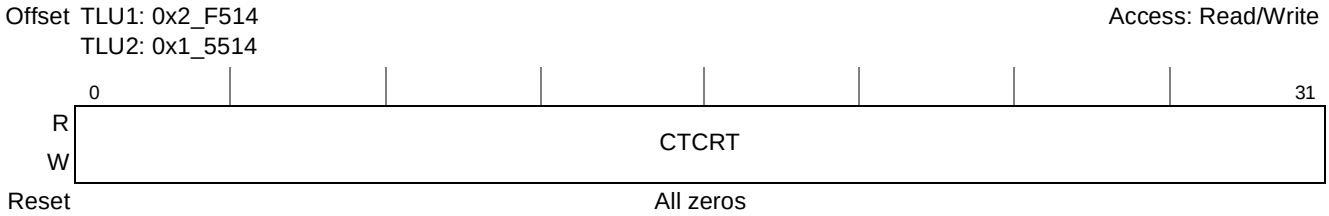


Figure 18-15. CTCRT Register Format

Table 18-16 describes the fields of the CTCRT register.

Table 18-16. CTCRT Field Descriptions

Bits	Name	Description
0–31	CTCRT	CRT table find command counter. Increments each time a find/r/w command is issued on a table of type CRT, where that table does not link to a table of type hash.

18.4.3.3.6 Chained Hash Table Find Count Register (CTCHS)

The chained hash table find count register (CTCHS) is a read/write register that counts the number of find/r/w commands that access an initial table type of CRT and then transition to a hash table before completion.

Figure 18-16 describes the definition for the CTCHS register.

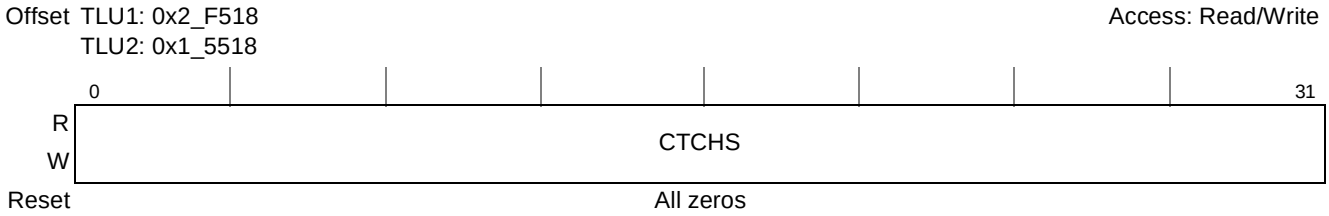


Figure 18-16. CTCHS Register Format

Table 18-17 describes the fields of the CTCHS register.

Table 18-17. CTCHS Field Descriptions

Bits	Name	Description
0–31	CTCHS	Chained-hash table find command counter. Increments each time a find/r/w command is issued on a table of type CRT, where that table or a subsidiary table links to a table of type hash.

18.4.3.3.7 Flat Data Table Lookup Count Register (CTDAT)

The flat data table lookup count register (CTDAT) is a read-write register that counts the number of find/r/w commands that access an initial table type of flat-data.

Figure 18-17 describes the definition for the CTDAT register.

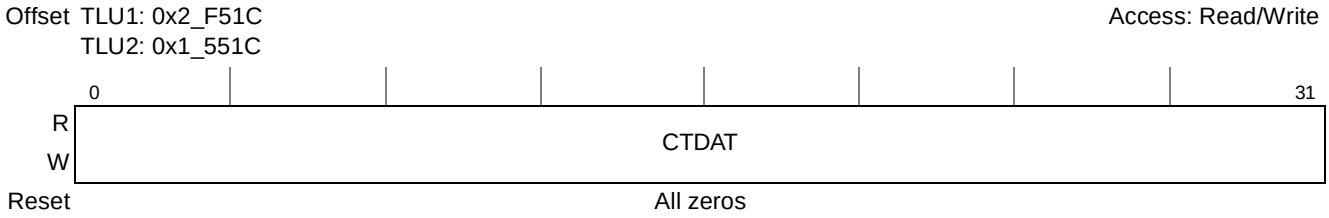


Figure 18-17. CTDAT Register Format

Table 18-18 describes the fields of the CTDAT register.

Table 18-18. CTDAT Field Descriptions

Bits	Name	Description
0–31	CTDAT	Flat-data table find command counter. Increments each time a find/r/w command is issued on a table of type flat-data.

18.4.3.3.8 Successful Find Count register (CHITS)

The successful find count register (CHITS) is a read/write register that counts the number of find/r/w commands that completed successfully and returned a search result.

Figure 18-18 describes the definition for the CHITS register.

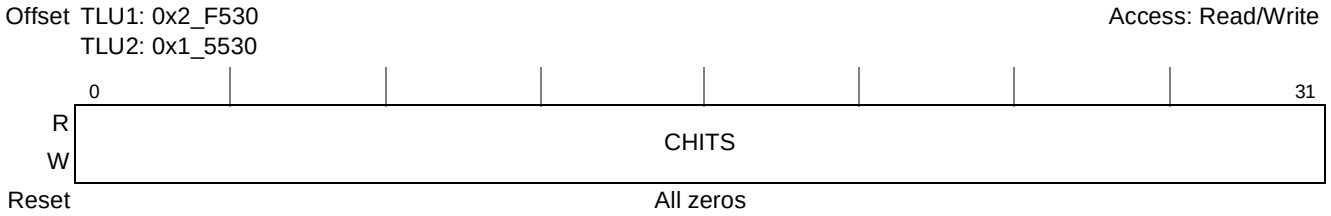


Figure 18-18. CHITS Register Format

Table 18-19 describes the fields of the CHITS register.

Table 18-19. CHITS Field Descriptions

Bits	Name	Description
0–31	CHITS	Find hits counter. Increments each time a find/r/w command completes with success.

18.4.3.3.9 Failed Find Count Register (CMISS)

The failed find count register (CMISS) is a read/write register that counts the number of find/r/w commands that completed with a failed search result, possibly due to errors.

Figure 18-19 describes the definition for the CMISS register.

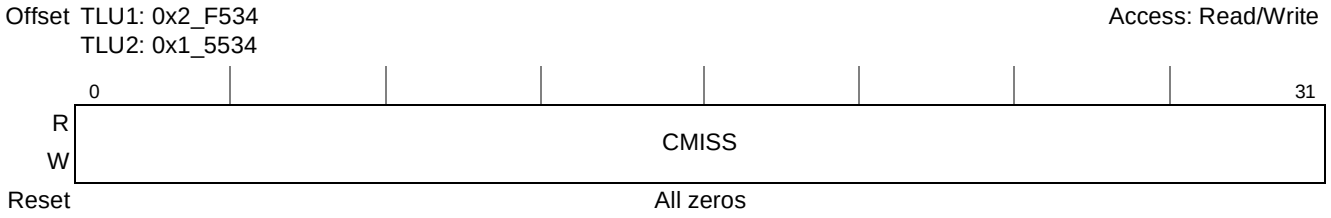


Figure 18-19. CMISS Register Format

Table 18-20 describes the fields of the CMISS register.

Table 18-20. CMISS Field Descriptions

Bits	Name	Description
0–31	CMISS	Find fails counter. Increments each time a find/r/w command completes with failure.

18.4.3.3.10 Hash Collision Count Register (CHCOL)

The hash collision count register (CHCOL) is a read/write register that counts the number of trie entries traversed during a find/r/w command applied to hash-trie-key or chained-hash tables. The presence of trie entries in a hash table indicates hash collisions across multiple keys. Depending on the balance and sparseness of the binary collision tree, at most 2^{NT} keys hashed to the initial table index, where *NT* is the maximum number of trie entries that must be traversed to resolve the collisions.

Figure 18-20 describes the definition for the CHCOL register.

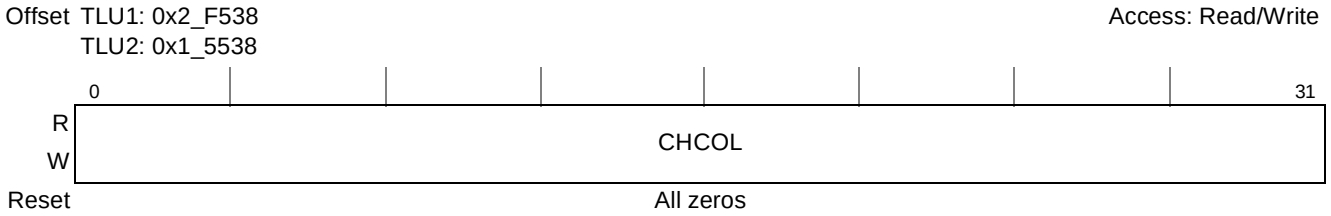


Figure 18-20. CHCOL Register Format

Table 18-21 describes the fields of the CHCOL register.

Table 18-21. CHCOL Field Descriptions

Bits	Name	Description
0–31	CHCOL	Hash collisions counter. Increments upon each access to a trie entry during a find/r/w command on a hash-trie-key or chained-hash table. Trie entries are accessed only to resolve hash collisions.

18.4.3.3.11 CRT Level Count Register (CCRTL)

The CRT level count register (CCRTL) is a read/write register that counts the number of CRT entries traversed during a find/r/w command of a CRT table. This counter records the efficiency of longest-prefix match (LPM) searches. LPM searches that resolve to long prefixes (small subnets in CIDR IP routing) require more CRT levels than searches resolving shorter prefixes. Failing LPM searches are also counted.

Figure 18-21 describes the definition for the CCRTL register.

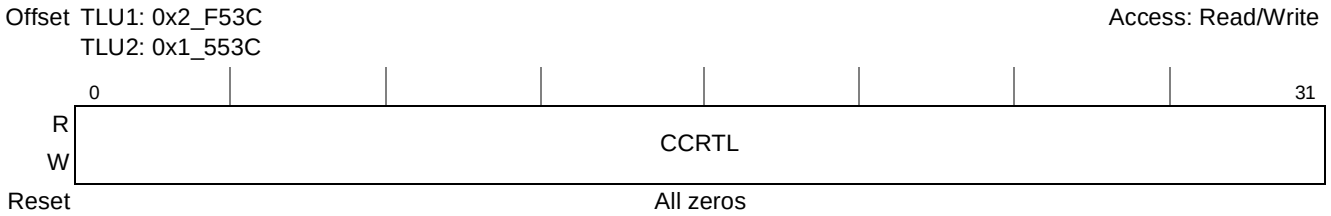


Figure 18-21. CCRTL Register Format

Table 18-22 describes the fields of the CCRTL register.

Table 18-22. CCRTL Field Descriptions

Bits	Name	Description
0–31	CCRTL	CRT level counter. Increments upon each access to a CRT non-hash entry during a find/r/w command on a CRT table. Initial, leaf, and fail CRT sub-tables are counted.

18.4.3.3.12 Counter Carries-Out Register (CARO)

The counter carries-out register (CARO) is a read/write register. Each bit in the CARO register indicates that the associated statistics counter rolled over (bit set) or has not produced a carry-out (bit clear). Bits in the CARO register are cleared by writing 1 to them.

Figure 18-22 describes the definition for the CARO register.

Offset TLU1: 0x2_F5F0
TLU2: 0x1_55F0

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12–15
R	CRR	CRW	CFI	CHT	CRT	CCH	CFD	—	CHI	CMS	CCO	CTL	—
W													
Reset	All zeros												

	16–31
R	—
W	
Reset	All zeros

Figure 18-22. CARO Register Format

Table 18-23 describes the fields of the CARO register.

Table 18-23. CARO Field Descriptions

Bits	Name	Description
0	CRR	Counter of memory reads carry out indication.
1	CRW	Counter of memory writes carry out indication.
2	CFI	Counter of find/r/w commands carry out indication.
3	CHT	Counter of hash-trie-key table searches carry out indication.
4	CRT	Counter of CRT table searches carry out indication.
5	CCH	Counter of chained-hash table searches carry out indication.
6	CFD	Counter of flat-data table searches carry out indication.
7	—	Reserved
8	CHI	Counter of successful find/r/w commands carry out indication.
9	CMS	Counter of failed find/r/w commands carry out indication.
10	CCO	Counter of hash collision resolution accesses carry out indication.
11	CTL	Counter of CRT (non-hash) accesses carry out indication.
12–31	—	Reserved

18.4.3.3.13 Counter Carry Mask Register (CARM)

The counter carry mask register (CARM) is a read/write register. Each bit in the CARM register that is clear allows the associated bit in the CARO register to cause an IEVENT[COV] event. By default, all defined bits of CARM are set, therefore no carry-out can cause a counter overflow interrupt until software clears the masks.

Figure 18-23 describes the definition for the CARM register.

Offset TLU1: 0x2_F5F4
TLU2: 0x1_55F4

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12			15
R W	MRR	MRW	MFI	MHT	MRT	MCH	MFD	—	MHI	MMS	MCO	MTL	—			
Reset	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

	16			31
R W	—			
Reset	All zeros			

Figure 18-23. CARM Register Format

Table 18-24 describes the fields of the CARM register.

Table 18-24. CARM Field Descriptions

Bits	Name	Description
0	MRR	Mask of memory reads carry bit.
1	MRW	Mask of memory writes carry bit.
2	MFI	Mask of find/r/w commands carry bit.
3	MHT	Mask of hash-trie-key table searches carry bit.
4	MRT	Mask of CRT table searches carry bit.
5	MCH	Mask of chained-hash table searches carry bit.
6	MFD	Mask of flat-data table searches carry bit.
7	—	Reserved.
8	MHI	Mask of successful find/r/w commands carry bit.
9	MMS	Mask of failed find/r/w commands carry bit.
10	MCO	Mask of hash collision resolution accesses carry bit.
11	MTL	Mask of CRT (non-hash) accesses carry bit.
12–31	—	Reserved

18.4.3.4 TLU Command/Response Registers

Software issues commands to the TLU and retrieves responses via the registers in this section. All of the registers are 32 bits wide, but pairs of registers may be accessed as 64-bit double words.

18.4.3.4.1 TLU Command Operation Register (CMDOP)

The TLU command operation register (CMDOP) is a read-write register used to launch commands to the TLU. For commands that require operands, it is necessary for software to initialize the appropriate KD*n*B (see Section 18.4.3.5.1, “Key/Data Words 0–7 Buffer Registers (KD0B–KD7B).”) and CMDIX registers

prior to writing the command to CMDOP as the last step. Each write to CMDOP automatically issues the TLU command which was written. A 64-bit write of the CMDOP/CMDIX pair can be used to both initialize the registers and issue the corresponding TLU command.

Figure 18-24 describes the definition for the CMDOP register.

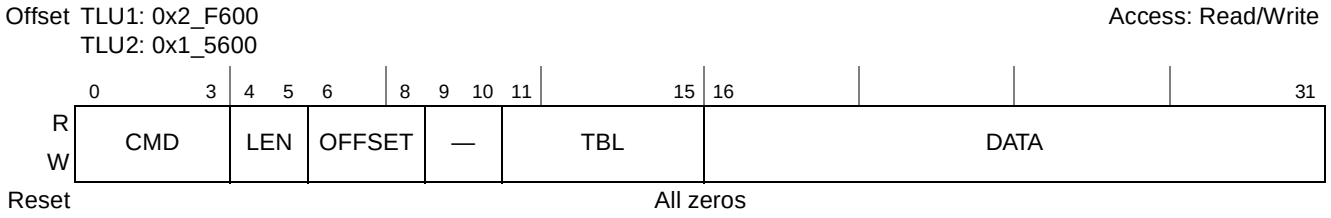


Figure 18-24. CMDOP Register Format

Table 18-25 describes the fields of the CMDOP register.

Table 18-25. CMDOP Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. The behavior of the TLU for reserved values of CMD is unpredictable. 0000 Reserved 0001 Write—Table write 0010 Reserved 0011 Add—Add data to table entry 0100 Read—Table read 0101 Reserved 0110 Acchash—Accumulate key to running hash value 0111 Reserved 1000 Find—Find key 1001 Reserved 1010 Findr—Find key and read table 1011 Reserved 1100 Findw—Find key and write table 1101 Reserved 1110 Reserved 1111 Reserved
4–5	LEN	Number of double words to write or read. Applicable only to write, read, acchash, and findr commands. 00 1 double word (in KD0B–KD1B) 01 2 double words (in KD0B–KD3B) 10 3 double words (in KD0B–KD5B) 11 4 double words (in KD0B–KD7B)
6–8	OFFSET	Offset in doublewords, 0–7, to be added to table index in register CMDIX[INDEX]. Applicable only to write, read, findr, and findw commands.
9–10	—	Reserved
11–15	TBL	Index, 0–31, of table to access for the command, except for acchash command. The type of the physical table indexed by TBL must support the command in CMD, otherwise a TTE event will occur.
16–31	DATA	Optional 16-bit data for add command.

18.4.3.4.2 TLU Command Index Register (CMDIX)

The TLU command index register (CMDIX) is a read/write register. The CMDIX register is optionally used by TLU commands that require a table index argument, such as write, add, read. For write, add, and read commands the TLU treats the accessed table as if it were a data table, regardless of its table type for lookups; hence CMDIX[INDEX] is multiplied by PLTB*x*[SIZE] before being used to address the table.

Figure 18-25 describes the definition for the CMDIX register.

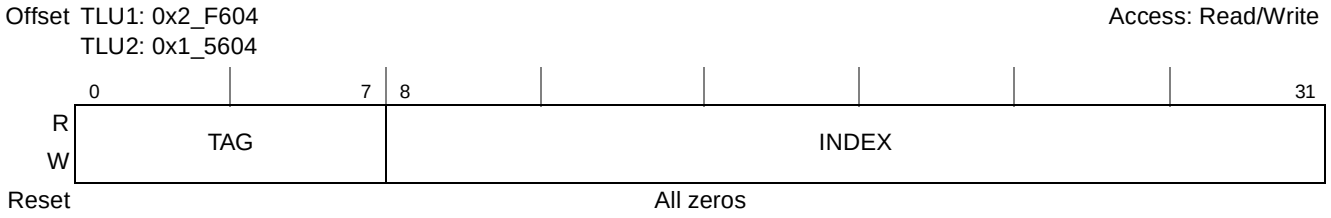


Figure 18-25. CMDIX Register Format

Table 18-26 describes the fields of the CMDIX register.

Table 18-26. CMDIX Field Descriptions

Bits	Name	Description
0–7	TAG	Tag used to identify write data byte lanes (for write and add commands); TAG should be set to 0xFF for writes of more than one double word. Bit 0 corresponds with the most significant byte of the double word in the table entry, whereas bit 7 corresponds with the least significant byte of the double word in the table entry.
8–31	INDEX	Index of entry for the table identified by CMDOP[TBL] for write, add, and read commands.

18.4.3.4.3 TLU Command Status and Response Register (CSTAT)

The TLU command status and response register (CSTAT) is a read/write register that returns the status of the last command issued via the CMDOP register.

Figure 18-26 describes the definition for the CSTAT register.

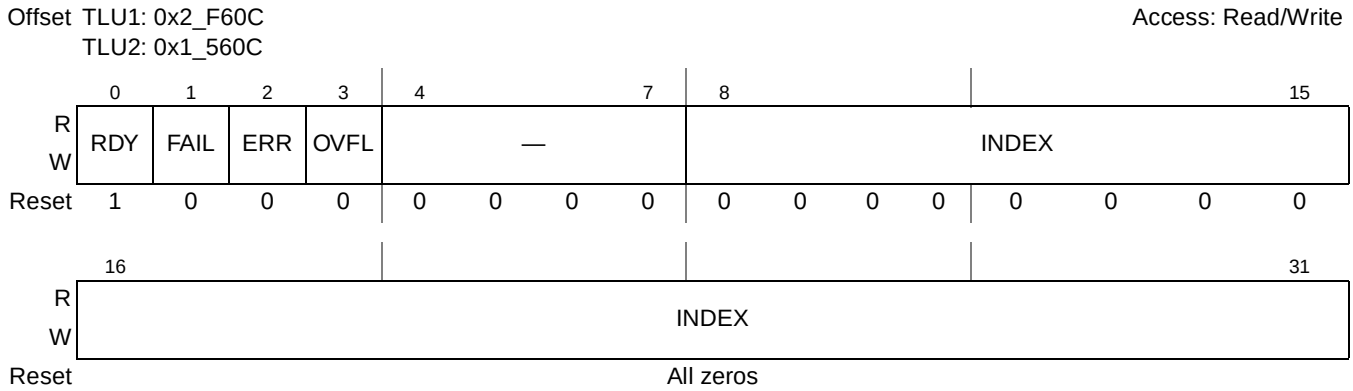


Figure 18-26. CSTAT Register Format

Table 18-27 describes the fields of the CSTAT register.

Table 18-27. CSTAT Field Descriptions

Bits	Name	Description
0	RDY	Command response ready flag. This flag is automatically cleared by the TLU following any write to the CMDOP register, and later set by hardware when the issued command completes. 0 Command has not completed and no other fields are valid. 1 Command has completed and other fields may be checked.
1	FAIL	Command failure flag. Valid only if RDY was set by the TLU. 0 Command has completed successfully. For commands returning a response, INDEX contains the index of the retrieved table entry. 1 Command has completed with failure, either due to errors (ERR = 1) or a table search terminating on a failing key match.
2	ERR	Command error flag. Valid only if RDY was set by the TLU. 0 Command has completed successfully without errors. 1 Command has completed with failure due to errors. The setting of IEDIS is not taken into account when setting ERR. Enabled errors are diagnosed in the IEVENT, IEATR, and IEADD registers.
3	OVFL	Add command overflow error flag. Valid only for add commands and if RDY was set by the TLU. 0 32-bit result of add command did not generate a carry out of the most significant bit. 1 32-bit result of add command generated a carry out of the most significant bit, which may indicate that a counter held in a table has overflowed.
4–7	—	Reserved
8–31	INDEX	The index of the retrieved table data or data/key entry for commands find, findr, and findw that completed successfully. The initial index accessed in the case FAIL = 1, ERR = 0. The index of the double word accessed in error in the case FAIL = 1, ERR = 1. This value is undefined for all other commands.

18.4.3.5 TLU Key/Data Registers

The registers in this section are used with any TLU commands that accept arguments and/or return data. Key/data registers must be set-up prior to a command being issued; only those words that are relevant to the command and its table structure need to be initialized. Commands that return data overwrite the arguments, allowing table lookups to be chained across tables without copying of data. All of the registers are 32 bits wide, but pairs of registers may be accessed as 64-bit double words.

18.4.3.5.1 Key/Data Words 0–7 Buffer Registers (KD0B–KD7B)

The key/data word 0 buffer registers (KD0B–KD7B) are read/write registers used for communicating command arguments and results to and from the TLU under software control. The length of written search keys is expected to match the size given by a table's configuration register (PTBL*n*[KLEN]). The length of result data is determined by the command and its parameters supplied to register CMDOP.

Figure 18-27 describes the definition for the KD0B–KD7B registers.

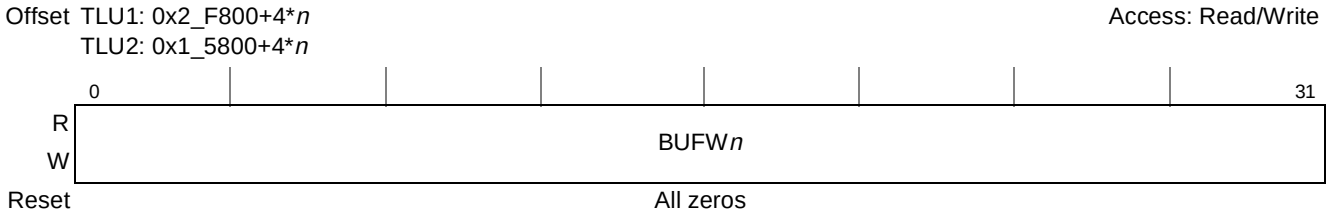


Figure 18-27. KD*n*B Register Format

Table 18-28 describes the fields of the KD0B–KD7B registers.

Table 18-28. KD0B–KD7B Field Descriptions

Bits	Name	Description
0–31	BUFW*n*	Word *n* of the key/data buffer. Written by software prior to issuing a TLU command. Hardware overwrites these words with command results before setting CSTAT[RDY].

18.5 Functional Description

18.5.1 Background

The table lookup unit is used by network protocol stacks to store and retrieve data in tables. To retrieve data, the TLU searches tables using a search key which is normally derived from one or more fields of a packet being processed, and returns the data associated with that key.

The TLU is normally used for such things as searching an IP route table, using the destination IP address as the search key, to determine the egress port and queue for an IP packet, or searching an ATM connection table using the VPI/VCI fields of an ATM cell to determine how it should be forwarded.

The TLU can also be used for more complex packet classification operations. Classification is used to determine such things as whether to forward or discard an incoming packet, what class of service to provide, or how to charge for it. Classification requires searching a table of classification rules with a search key formed by concatenating multiple fields of interest in the packet, such as the source and destination IP addresses and the source and destination TCP/UDP port numbers.

Tables in the TLU are held in contiguous areas of external memory. Up to 32 tables may be configured individually. Tables can be considered sparse arrays, where each table entry comprises a key and some data associated with that key. When the CPU launches a lookup of a table with a specific search key, the elements of the table are searched until an entry is found with a key that matches the search key, and the data of the entry is returned as the requester as the result of the lookup. If no matching key is found (a lookup miss), an error is signalled back to the requester in CSTAT[FAIL]. The table search operation is shown in Figure 18-28.

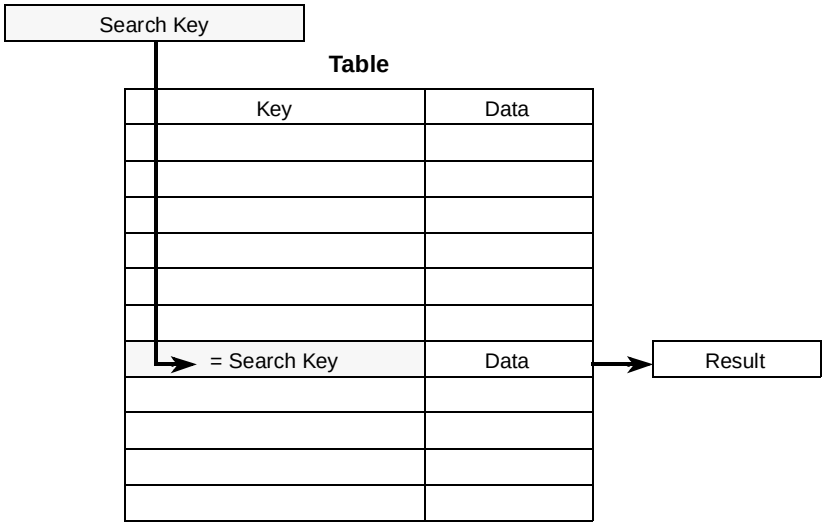


Figure 18-28. Table Lookup Operation

It would be impractical to search a table by comparing each key with the search key until a match was found. Instead, a number of different table structures and search algorithms are available which minimize the number of memory accesses required to locate a matching key. For all table types except flat data, the key can be 32, 64, 96 or 128 bits long. For all table types, the amount of data associated with the key is 8, 16, 32, or 64 bytes, which includes the necessary space for the key itself in the hash tables.

Search algorithms and table structures are provided to implement two search strategies. Exact match (EM) lookups search a table for a key which is bit for bit identical to the search key. Longest prefix match (LPM) lookups search a table for a key which matches the longest sequence of bits starting at the most significant end of the search key, especially when more than one prefix of bits would match the most significant bits of the key. LPM searches are used to implement IPv4 and IPv6 routing tables and can also be used to implement wild-carded classification lookups. EM searches are used in all other cases.

18.5.2 TLU Command Set

TLU commands are issued by writing an opcode into the CMDOP register. However, prior to issuing a command, software must ensure that:

- Table memory banks are allocated on the local bus, and located with the MBANK0–MBANK3 registers.
- Physical tables are configured and located via the PTBL0–PTBL31 registers.
- Table contents are initialized and valid with respect to the table configuration given in the associated PTBL register.
- For commands that require a key and/or data, the data is written to the necessary subset of the KD0B–KD7B registers, with any short keys being padded to the right with zero bits.
- For commands that require an index argument, the index is written to the CMDIX register.
- Any fields in CMDOP correctly reflect the requirements of the command and the characteristics of the physical table being accessed.

Immediately following the write to register CMDOP, the CSTAT[RDY] flag is cleared and remains clear for the duration of the command. While CSTAT[RDY] remains clear, software must not overwrite any of the CMDOP, CMDIX, or KD0B–KD7B registers used by the pending command; otherwise the TLU behaves unpredictably. Software may poll CSTAT until CSTAT[RDY] is read set, at which point any results can be recovered from the CSTAT and KD0B–KD7B registers. In particular, the FAIL status of a command is not valid until CSTAT[RDY] is set. If interrupts occur during the command execution, the corresponding interrupt bits in IEVENT are set prior to command completion. The PIC may receive the interrupt from the TLU prior to command completion.

NOTE

Access to the TLU memory-mapped registers requires that they reside in non-cacheable regions. Ensure that the memory management unit has cache-inhibit set for any pages that map the TLU register space.

A summary of all TLU commands is given in Table 18-29.

Table 18-29. Summary of TLU Commands

Command	CMD	Use of CMDIX	Arguments in KD0B–KD7B	Response in CSTAT[1]	Returned in KD0B–KD7B	Function[2]
write	0x1	index to write	data to be written	RDY, FAIL, ERR	—	Write 1–4 double words to data table at index + offset.
add	0x3	index to read/write	—	RDY, FAIL, ERR	—	Adds up to 16 bits to data table entry at index + offset.
read	0x4	index to read	—	RDY, FAIL, ERR	data read from table	Read 1–4 double words from data table at index + offset.
acchash	0x6	—	key, padded right with zeros	RDY	—	Accumulate 1–4 double words of key to running hash.
find	0x8	—	key, padded right with zeros	RDY, FAIL, ERR, INDEX found	—	Search for key and return index of key/data entry in table.
findr	0xA	—	key, padded right with zeros	RDY, FAIL, ERR, INDEX found	data read from found entry	Search for key and return index of key/data entry with read data.
findw	0xC	—	key, padded, followed by data	RDY, FAIL, ERR, INDEX found	—	Search for key and write a double word to found data entry.

1 All commands return RDY = 1 upon completion. FAIL = 1 indicates failure to find required index.

2 Number of key/data words is given by CMDOP[LEN] or PTBL[KLEN]. Offset is given by CMDOP[OFFSET].

18.5.2.1 Write—Write Data to Table Index Command

The write command is used to write data to the TLU's external memory. Two kinds of writes are available:

1. The first type, when CMDOP[LEN] = 0, is a masked write from one to eight bytes in length. The bytes are selected using the CMDIX[TAG] field bits. Bytes must be contiguous and completely contained within double word boundaries (that is, masks of 0x05 or 0x00 are illegal, while 0x03 and 0xF8 are both legal mask values). The TLU ignores all bits set right of the leftmost mask for non-contiguous masks, and ignore the write command if no bytes are selected. The write data is contained in the KD0B and KD1B registers.

2. The second type, when CMDOP[LEN] > 0, can write up to four consecutive memory double word locations. The TLU uses the value of CMDOP[LEN] to determine the actual number of memory locations to write, and copies double words from the KD0B–KB7B registers. The CMDIX[TAG] field bits must be set to 0xFF for this type of write, or an unpredictable number of bytes is written.

NOTE

If software chooses to bypass the TLU for table writes by accessing the local bus directly, care should be taken in modifying tables that may be in use. Use of the Table Services API is recommended, as the software safely updates tables in place.

18.5.2.1.1 Write Command Format

The registers used for issuing the write command are shown in Figure 18-29.

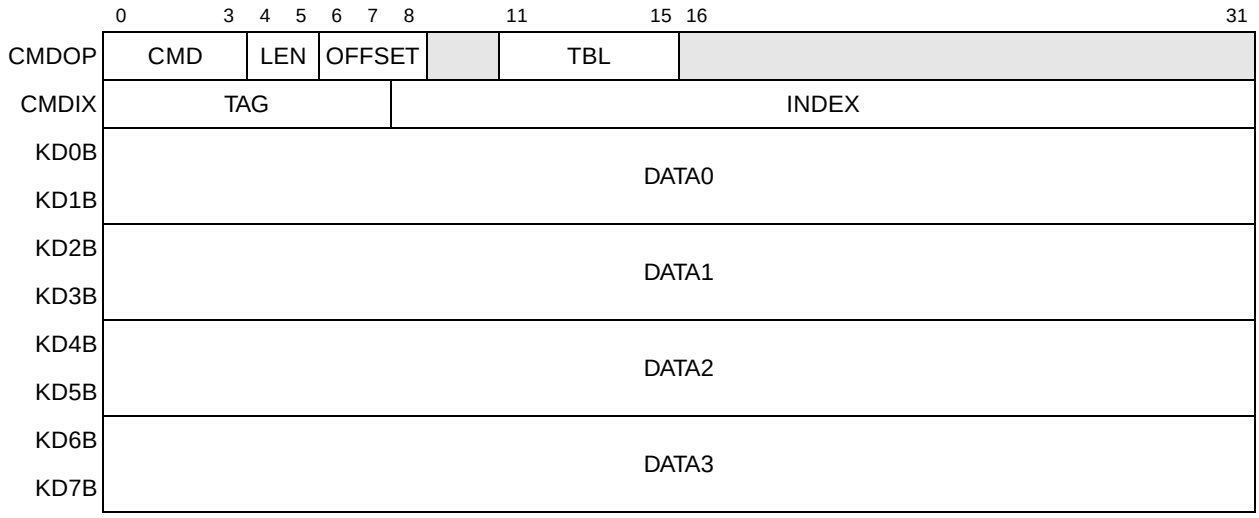


Figure 18-29. Write Command Format

Each write command field is defined in Table 18-30.

Table 18-30. Write Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 0001 Write
4–5	LEN	Number of double words to write. 00 1 double word or 1–8 bytes under TAG control (in KD0B–KD1B) 01 2 double words (in KD0B–KD3B) 10 3 double words (in KD0B–KD5B) 11 4 double words (in KD0B–KD7B)
6–8	OFFSET	Offset in double words, 0–7, to be added to INDEX.
11–15	TBL	Index 0–31 of table to access for the write.

Table 18-30. Write Command Field Descriptions (continued)

Bits	Name	Description
0–7	TAG	Tag used to identify write data byte lanes; TAG should be set to 0xFF for writes of more than one double word. Bit 0 corresponds with the most significant byte of the double word in the table entry, whereas bit 7 corresponds with the least significant byte of the double word in the table entry. At least one bit of TAG must be set to perform a write operation.
8–31	INDEX	Index of data entry for the table identified by TBL. Regardless of the table type, INDEX is scaled by the table's SIZE attribute in forming the write address.
0–31 (KD*n*B), 0–31 (KD*n+1* B)	DATA0–DATA3	Table data to write. The lowest numbered KD buffer register holds the most significant byte of the double word. The relevant fields that must be initialized depend on LEN: 0 DATA0, bytes 0–7 1 DATA0–DATA1, bytes 0–15 2 DATA0–DATA2, bytes 0–23 3 DATA0–DATA3, bytes 0–31

18.5.2.1.2 Write Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table or supplied index are inaccessible, both CSTAT[FAIL] and CSTAT[ERR] are set. No data is returned, and therefore the data in KD0B–KD7B remains unmodified.

18.5.2.2 Add—Add Data to Table Index Command

The add command behaves similarly to the write command, except that 16 bits of data from CMDOP[DATA] are added to 32 bits of existing data in the table. Carries out of the 32-bit sum are returned as CSTAT[OVFL]. The four bytes to modify are selected using the CMDIX[TAG] field bits. Bytes must be contiguous and aligned on a word boundary—that is, only mask values of 0xF0 or 0x0F are legal. The TLU ignores the add command if legal fields are not selected. The 16 bits of CMDOP[DATA] are zero-extended to 32 bits, added to the word read from memory, and the sum is written back to replace the four bytes read.

18.5.2.2.1 Add Command Format

The registers used for issuing the add command are shown in Figure 18-30.

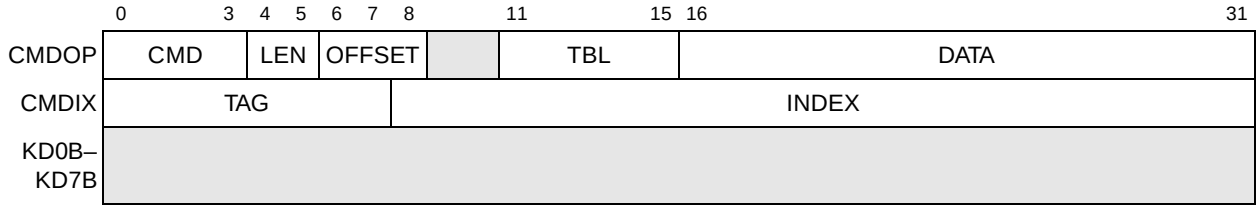


Figure 18-30. Add Command Format

Each add command field is defined in Table 18-31.

Table 18-31. Add Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 0011 Add
4–5	LEN	Number of double words to modify. Must be cleared to 0.
6–8	OFFSET	Offset in double words, 0–7, to be added to INDEX.
11–15	TBL	Index 0–31 of table to access for the add.
16–31	DATA	Data to be added. In forming the 32-bit sum this is either the 16 least significant bits or the 16 most significant bits of the addend, depending upon the TAG field.
0–7	TAG	Tag used to identify 4 write data byte lanes. Bit 0 corresponds with the most significant byte of the double word in the table entry, whereas bit 7 corresponds with the least significant byte of the double word in the table entry. Exactly 4 adjacent bits should be set, aligned to word boundaries. Bits 0–3 select the most significant word to be added to DATA, while bits 4–7 select the least significant word to be added to DATA.
8–31	INDEX	Index of data entry for the table identified by TBL. Regardless of the table type, INDEX is scaled by the table's SIZE attribute in forming the access address.

18.5.2.2.2 Add Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table or supplied index are inaccessible, both CSTAT[FAIL] and CSTAT[ERR] are set. A carry out of the 32-bit sum is returned as CSTAT[OVFL], which indicates a possible overflow condition. No data is returned.

18.5.2.3 Read—Read Data from Table Index Command

The read command is used to read data from the TLU's external memory.

18.5.2.3.1 Read Command Format

The registers used for issuing the read command are shown in Figure 18-31.

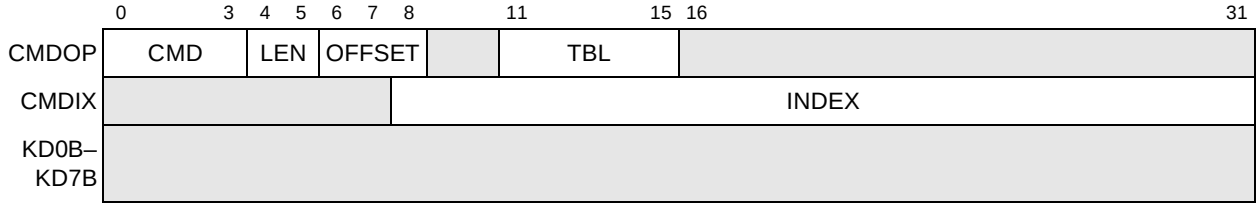


Figure 18-31. Read Command Format

Each read command field is defined in Table 18-32.

Table 18-32. Read Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 0100 Read
4–5	LEN	Number of double words to read. 00 1 double word (returned in KD0B–KD1B) 01 2 double words (returned in KD0B–KD3B) 10 3 double words (returned in KD0B–KD5B) 11 4 double words (returned in KD0B–KD7B)
6–8	OFFSET	Offset in doublewords, 0–7, to be added to INDEX.
11–15	TBL	Index 0–31 of table to access for the read.
8–31	INDEX	Index of data entry for the table identified by TBL. Regardless of the table type, INDEX is scaled by the table's SIZE attribute in forming the read address.

18.5.2.3.2 Read Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table or supplied index are inaccessible, both CSTAT[FAIL] and CSTAT[ERR] are set. Read data is returned in the KD0B–KD7B registers, with the most significant word of the lowest numbered double word being returned in KD0B. In the case of an error, some or all values of KD0B–KD7B may be undefined.

18.5.2.4 Acchash—Accumulate to Hash Command

The acchash command is used to perform a hash over very long keys by incrementally accumulating the hash result for parts of the key. Acchash can hash only multiples of 64-bits. The hash index so accumulated forms the initial TLU hash state on the next find/r/w command. For example, to hash a 48-byte (or 384-bit) key, acchash is used initially to hash the first 32 bytes of the key, followed by a find command on the remaining 16 bytes of the key to complete the entire hash.

In order for the accumulated hash to be useful, the final find/r/w command must lookup a hash-trie-key table. If the entire key is not a multiple of 64-bits in length, padding with zeros should be applied to only the final part of the key issued to the closing find/r/w command. Every find/r/w command resets the TLU hash state, allowing fresh invocations of acchash. Note that since a final find command checks only the most recently entered key for exact match with a stored key, software must separately match previous parts of a long key in order to establish overall search success with acchash. That is, acchash is intended primarily for signature recognition applications rather than an extension to exact match key lengths.

18.5.2.4.1 Acchash Command Format

The registers used for issuing the acchash command are shown in Figure 18-32.

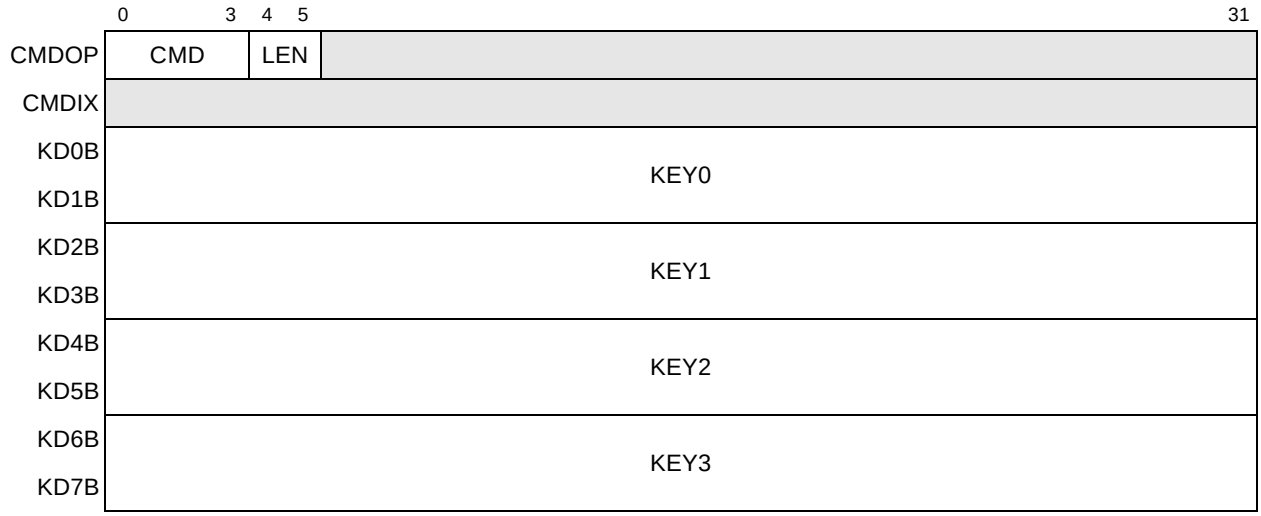


Figure 18-32. Acchash Command Format

Each acchash command field is defined in Table 18-33.

Table 18-33. Acchash Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 0110 Acchash
4–5	LEN	Number of double words of key to accumulate into the hash. Padding of unused bits should occur on the final hash as part of the final Find command. 00 1 double word (in KD0B–KD1B) 01 2 double words (in KD0B–KD3B) 10 3 double words (in KD0B–KD5B) 11 4 double words (in KD0B–KD7B)
0–31 (KD*n*B), 0–31 (KD*n+1*B)	KEY0–KEY3	Key portion to hash. The lowest numbered KD buffer register holds the most significant byte of the doubleword. The relevant fields that must be initialized depends on LEN: 0 KEY0, bytes 0–7 1 KEY0–KEY1, bytes 0–15 2 KEY0–KEY2, bytes 0–23 3 KEY0–KEY3, bytes 0–31

18.5.2.4.2 Acchash Command Response

The CSTAT[RDY] bit is set upon completion of the command. No data is returned, and therefore the data in KD0B–KD7B remains unmodified.

Note that the fail status of a hash-trie-key find command following acchash is determined by matching only the last part of the multipart key against a key table entry. Therefore, if FAIL = 0, software should separately compare the earlier parts of the key against retrieved data. However, if FAIL = 1 an exact match lookup has failed.

18.5.2.5 Find—Find Key Command

The find command attempts to locate a key in a table using a preprogrammed table as specified by CMDOP[TBL]. This command returns a key/data entry index upon successfully finding the key. Software must initialize the table identified by CMDOP[TBL] before the find command is issued. The length of key expected by this command is determined by the mapped table configuration register.

18.5.2.5.1 Find Command Format

The registers used for issuing the find command are shown in Figure 18-33.

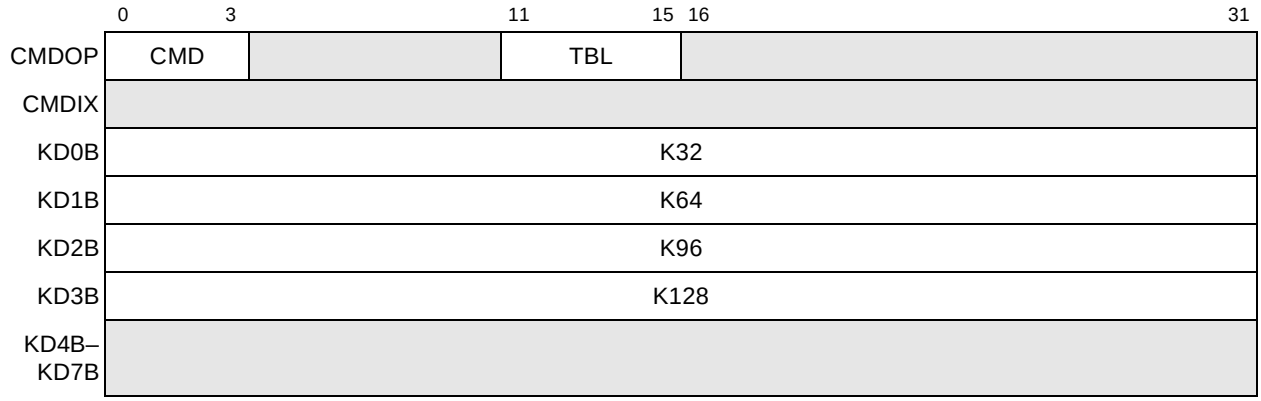


Figure 18-33. Find Command Format

Each find command field is defined in Table 18-34.

Table 18-34. Find Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 1000 Find
11–15	TBL	Index 0–31 of table to access for the lookup.
0–31	K32	• All 32 bits of a 32-bit key. • Upper 32 bits of 64-bit, 96-bit, or 128-bit keys.
0–31	K64	• Lower 32 bits of a 64-bit key. • 32 bits of 96-bit or 128-bit keys. • Unused (don’t care) for 32-bit keys.
0–31	K96	• Lower 32 bits of a 96-bit key. • 32 bits of a 128-bit key. • Unused (don’t care) for 32-bit and 64-bit keys.
0–31	K128	• Lower 32 bits of a 128-bit key. • Unused (don’t care) for 32-bit, 64-bit, and 96-bit keys.

18.5.2.5.2 Find Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table is inaccessible or the key cannot be located in the table, CSTAT[FAIL] is set. In the case of an error causing the failure, CSTAT[ERR] is also set. Otherwise, a successful find returns the index of the matching key or data table entry in

CSTAT[INDEX]. The size of the key/data entry is specified by PTBL*n*[SIZE]. No other data is returned, and therefore the key in KD0B–KD7B remains unmodified.

18.5.2.6 Findr—Find Key and Read Table Command

The findr command attempts to locate a key in a table using a preprogrammed table as specified by CMDOP[TBL]. Upon successfully finding the key, this command performs a read operation at the matching index, and returns both the read data and key/data index. That is, for hash-trie-key tables, CMDOP[OFFSET] = 0 refers to the high-order double word of the key part of the key entry; CRT entries have no key part. In this way, the findr command provides for arbitrary mappings from keys to data, but at the expense of an additional data entry read. Software must initialize the table identified by CMDOP[TBL] before the findr command is issued. The length of key expected by this command is determined by the mapped table configuration register.

18.5.2.6.1 Findr Command Format

The registers used for issuing the findr command are shown in Figure 18-34.

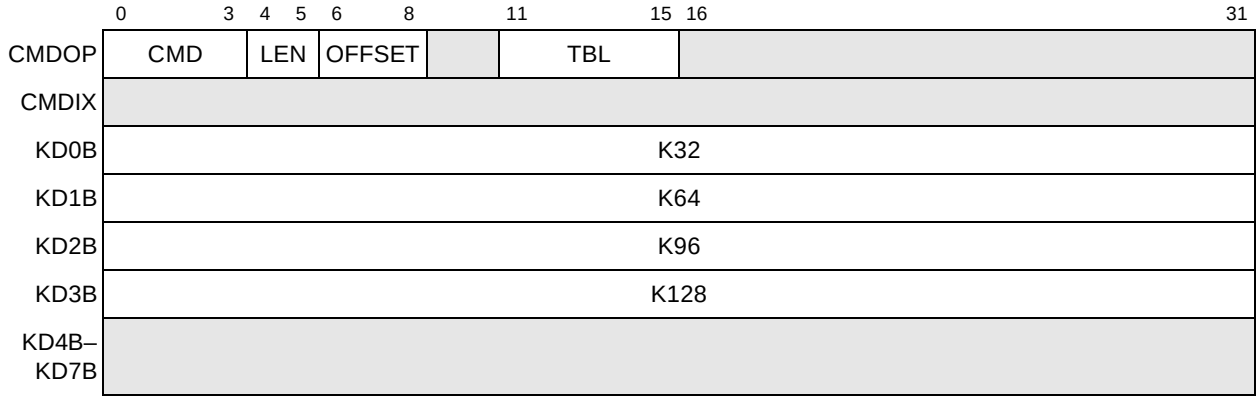


Figure 18-34. Findr Command Format

Each findr command field is defined in Table 18-35.

Table 18-35. Findr Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 1010 Findr
4–5	LEN	Number of double words to read. 00 1 double word (returned in KD0B–KD1B) 01 2 double words (returned in KD0B–KD3B) 10 3 double words (returned in KD0B–KD5B) 11 4 double words (returned in KD0B–KD7B)
6–8	OFFSET	Offset in doublewords, 0–7, to be added to the found index before performing the read.
11–15	TBL	Index 0–31 of table to access for the lookup.
0–31	K32	All 32 bits of a 32-bit key. Upper 32 bits of 64-bit, 96-bit, or 128-bit keys.

Table 18-35. Findr Command Field Descriptions (continued)

Bits	Name	Description
0–31	K64	Lower 32 bits of a 64-bit key. 32 bits of 96-bit or 128-bit keys. Unused (don't care) for 32-bit keys.
0–31	K96	Lower 32 bits of a 96-bit key. 32 bits of a 128-bit key. Unused (don't care) for 32-bit and 64-bit keys.
0–31	K128	Lower 32 bits of a 128-bit key. Unused (don't care) for 32-bit, 64-bit, and 96-bit keys.

18.5.2.6.2 Findr Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table is inaccessible or the key cannot be located in the table, CSTAT[FAIL] is set. In the case of an error causing the failure, CSTAT[ERR] is also set, and the contents of KD0B–KD7B are undefined. Otherwise, a successful findr returns the index of the matching key or data table entry in CSTAT[INDEX], and the read data from the table in as many KD0B–KD7B registers as determined by CMDOP[LEN]. The size of the key/data entry is specified by PTBL*n*[SIZE].

18.5.2.7 Findw—Find Key and Write Table Command

The findw command attempts to locate a key in a table using a preprogrammed table as specified by CMDOP[TBL]. Upon successfully finding the key, this command performs a write operation of one double word at the matching index plus a specified offset, and returns the key/data table index found. Software must initialize the table identified by CMDOP[TBL] before the findw command is issued. The length of key expected by this command is determined by the mapped table configuration register. The findw command has a similar format to write, except it can only write 8 bytes of data, and does not support write masks.

18.5.2.7.1 Findw Command Format

The registers used for issuing the findw command are shown in Figure 18-35.

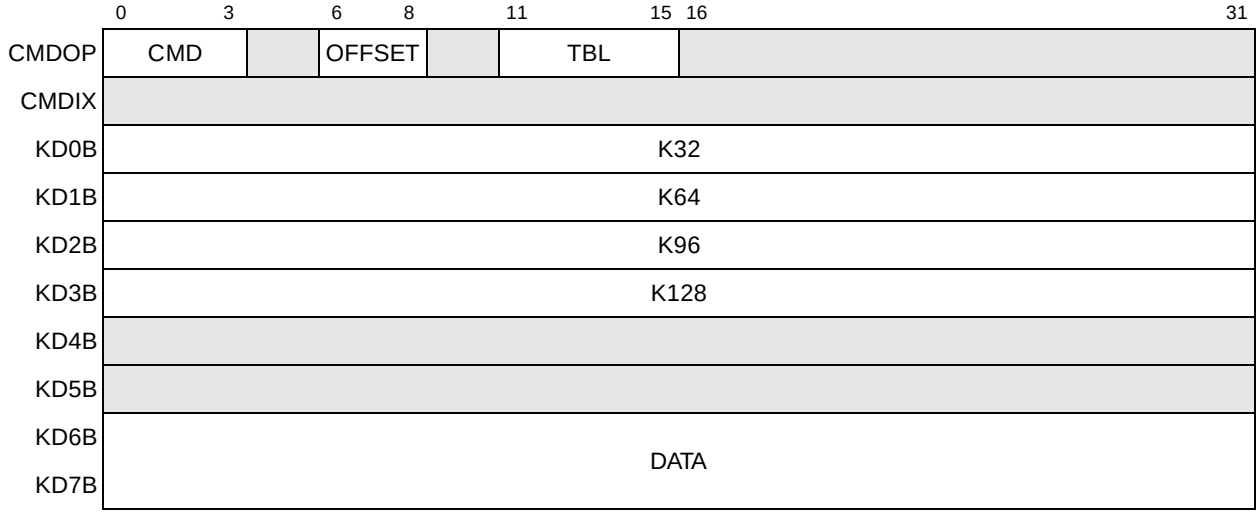


Figure 18-35. Findw Command Format

Each findw command field is defined in Table 18-36.

Table 18-36. Findw Command Field Descriptions

Bits	Name	Description
0–3	CMD	TLU command code. 1100 Findw
6–8	OFFSET	Offset in double words, 0–7, to be added to the found index before performing the write.
11–15	TBL	Index 0–31 of table to access for the lookup.
0–31	K32	All 32 bits of a 32-bit key. Upper 32 bits of 64-bit, 96-bit, or 128-bit keys.
0–31	K64	Lower 32 bits of a 64-bit key. 32 bits of 96-bit or 128-bit keys. Unused (don't care) for 32-bit keys.
0–31	K96	Lower 32 bits of a 96-bit key. 32 bits of a 128-bit key. Unused (don't care) for 32-bit and 64-bit keys.
0–31	K128	Lower 32 bits of a 128-bit key. Unused (don't care) for 32-bit, 64-bit, and 96-bit keys.
0–31 (KD6B), 0–31 (KD7B)	DATA	Table data to write if the lookup succeeds. KD6B holds the most significant word of the doubleword.

18.5.2.7.2 Findw Command Response

The CSTAT[RDY] bit is set upon completion of the command. If the table is inaccessible or the key cannot be located in the table, CSTAT[FAIL] is set. In the case of an error causing the failure, CSTAT[ERR] is

also set. Otherwise, a successful findw returns the index of the matching key or data table entry in CSTAT[INDEX]. The size of the key/data entry is specified by PTBL*n*[SIZE]. No other data is returned, and therefore the key and data in KD0B–KD7B remain unmodified.

18.5.2.8 Error Handling

TLU lookups may fail due to keys not being found, or errors occurring during table traversal. The CSTAT flags should be interpreted as shown in Table 18-37. Note that FAIL is always set when the lookup response is invalid, and ERR distinguishes the reason for the failure.

Table 18-37. Command Status in Relation to Errors

RDY	FAIL	ERR	Meaning
0	X	X	Operation status pending
1	0	0	Successful operation
1	0	1	Reserved (not possible)
1	1	0	Lookup failed due to missing key
1	1	1	Lookup failed due to table access errors

The error cause can be found in the IEVENT register following a lookup command provided that the event was not disabled in IEDIS. Regardless of IEDIS, errors that occur during a lookup cause CSTAT[ERR] to be set. Each error is described in more detail in Table 18-38.

Table 18-38. TLU Error Conditions

IEVENT Flag	Description
DPE	Data parity error occurred on a memory read. This error condition cannot occur unless MBANK*n*[PAR] is set and the local bus is configured to generate/check data parity.
BAE	An error occurred on a memory read or write due to address out of range or bus time-out. This can occur if: • The memory banks configured in MBANK0–MBANK3 do not hit a local bus base address; • A memory device on the local bus fails to complete an access due to a handshake time-out.
XAE	More than 255 memory accesses were attempted to satisfy a command. This is a fail-safe that ensures that badly formed tables or uninitialized tables do not continue to be traversed in an unending loop. No realistic tables should require 256 accesses to complete a lookup.
TTE	The table type, TYPE, of the indexed PTBL was not defined as a known type. This can occur if the PTBL is not initialized, or the TYPE field was set to a reserved value. Note that any TLU command—even read or write—that accesses a physical table will check TYPE for validity.

18.5.3 TLU Tables and Operation

Applications typically need to implement complex data structures, such as tries, for search and lookup activities. A table is the designation applied to the TLU for instances of such complex data structures. Each PTBL*n* register configures exactly one such table. The TLU command set operates over such tables as if they were opaque data structures for searching.

However, the underlying structure of tables such as CRT and chained-hash is a collection of one or more simple tables, each linked by pointers. A pointer is the index of the table entry counted from the base of

the complex table. The TLU recognizes five distinctive types of simple table, each with an inherent entry size:

- Data—entry size given by PTBL*n*[SIZE] from its enclosing table
- Key—entry size given by PTBL*n*[SIZE] from its enclosing table
- Hash—entry size of 8 bytes
- Trie—entry size of 8 bytes
- Index-variable prefix trie-data (IPTD)—entry size of 8 bytes

With the exception of flat data tables, these simple tables are always used in combinations (known as algorithms) of up to four different table types to form the complex tables used for exact match and longest prefix match searches.

NOTE

The maximum size of any TLU table is 16M double words or 128 Mbytes. Regardless of the table base address, no index provided to a command or calculated by the TLU during a lookup can exceed the 128-Mbyte bounds from the base of the table. Index calculations that are inadvertently scaled or incremented beyond the 128-Mbyte boundary will wrap around in the hardware towards zero.

At the core of the TLU's operation is a state machine for executing the algorithms. The state machine uses a key, and the data contents of the initial node to determine which state to expect. It can determine which data format to expect next; therefore enabling it to generate the next node address. The TLU operates in the following manner: it reads the initial data format, checks memory for the format data to act upon, generates the next address, and traverses the table to get to the associated data for a given key.

The ten allowed TLU state transitions are listed in Table 18-39. IPDT states/tables are the most general, permitting transitions to either hash or data tables. Hash tables always transition to collision resolution tables (trie) or terminate at keys if not failed. Data and key tables are terminal nodes in any lookup.

Table 18-39. Allowed TLU State Transitions

Start State → Next State
Hash → Trie
Hash → Key
Hash → Fail
Trie → Trie
Trie → Key
Trie → Fail
IPTD → IPTD
IPTD → Hash
IPTD → Data
IPTD → Fail

The relationship between complex table type (as given by PTBL*n*[TYPE]) and TLU states is listed in Table 18-40. Note that the complex type simply determines the initial simple table type, and hence initial TLU state, with subsequent state transitions occurring according to the rules in Table 18-39.

Table 18-40. TLU Table State versus Table Type

Algorithm	PTBL*n*[TYPE]	Initial Simple Table	Subsequent Tables/States
Flat Data	001	Data	—
Chained Hash	010	IPTD	IPTD, Data, Hash, Trie, Key, Fail
Compressed Radix Trie	010	IPTD	IPTD, Data, Hash, Trie, Key, Fail
Hash-Trie-Key	100	Hash	Trie, Key, Fail

18.5.3.1 Data Simple Table

Simple data tables are linear arrays of entries. Each entry can be 1, 2, 4, or 8 double words in length, as given by the PTBL*n*[SIZE] field associated with the complex table. The format of an entry, as shown in Figure 18-36, is entirely user-defined, as entries are not interpreted by the TLU.

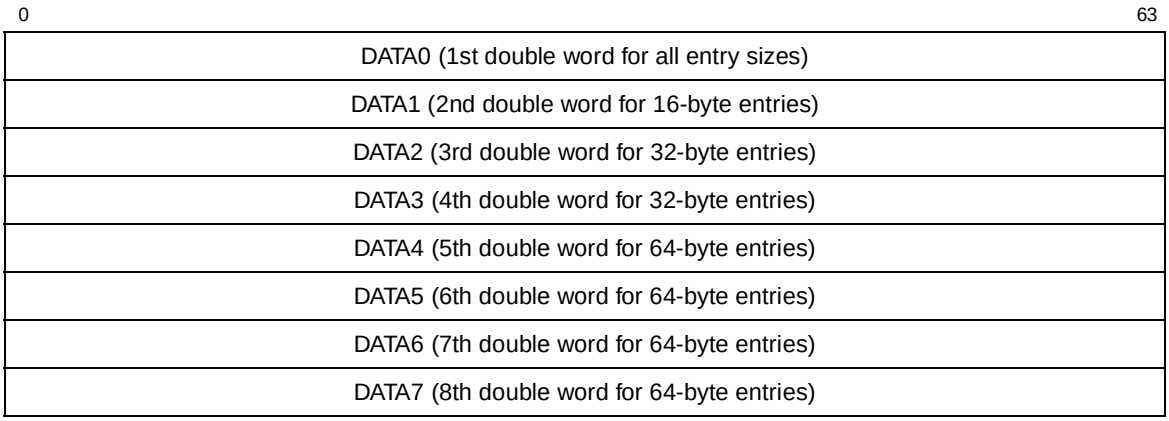


Figure 18-36. Data Simple Table Entry Format

18.5.3.2 Hash Simple Table

Hash simple tables are linear arrays of hash entries, indexed by an initial hash of the search key. The format of a hash table entry is shown in Figure 18-37. Note that one double word table location comprises, in fact, a pair of adjacent hash entries designated the left (L) and right (R) entries. Accordingly, if the hash function produces a hash value *H*, the Hash entry at index *H*/2 is accessed and the LSB of *H* determines whether the left (LSB = 0) or right (LSB = 1) portion is used. Hash collisions are not resolved by this table, but by trie tables.

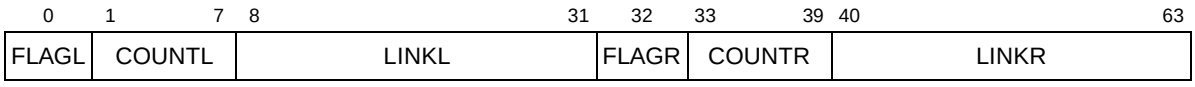


Figure 18-37. Hash Simple Table Entry Format

The hash entry is defined in Table 18-41.

Table 18-41. Hash Entry Field Descriptions

Bits	Name	Description
0	FLAGL	Left hash entry flag. Indicates next entry format. 0 Next entry is a trie entry used to resolve a collision. 1 Next entry is a key entry used to determine success of the lookup.
1–7	COUNTL	Left hash entry bit count, with 0 denoting the MSB of the key. The total number of bits that match traversing down the branch. The TLU uses the bit in the key associated with the COUNTL value to determine: • If the left (bit=0) high-order 32 bits of the next trie data is to be used or • If the right (bit=1) low-order 32 bits of the next trie data is to be used.
8–31	LINKL	Left hash entry link pointer. If LINKL = 0, the hash lookup is deemed failed, and no more entries are examined regardless of FLAG. However, in the case of FLAGL = 0, LINKL is the index of the doubleword trie table entry to access next, whereas in the case of FLAGL = 1, LINKL is the index of the configured-size key table entry to access next.
32	FLAGR	Right hash entry flag. Indicates next entry format. 0 Next entry is a trie entry used to resolve a collision. 1 Next entry is a key entry used to determine success of the lookup.
33–39	COUNTR	Right hash entry bit count, with 0 denoting the MSB of the key. The total number of bits that match traversing down the branch. The TLU uses the bit in the key associated with the COUNTR value to determine: • If the left (bit = 0) high-order 32bits of the next trie data is to be used or • If the right (bit = 1) low-order 32bits of the next trie data is to be used.
40–63	LINKR	Right hash entry link pointer. If LINKR = 0, the hash lookup is deemed failed, and no more entries are examined regardless of FLAG. However, in the case of FLAGR = 0, LINKR is the index of the double word trie table entry to access next, whereas in the case of FLAGR = 1, LINKR is the index of the configured-size key table entry to access next.

18.5.3.3 Trie Simple Table

Trie simple tables are nodes of a binary tree used to resolve collisions between multiple keys hashing to a given hash table entry. A benefit of using trie tables for collision resolution in conjunction with a hash table is that for N colliding keys that collide in the table, usually only $\log_2(N)$ entries need to be checked to resolve the collisions. For collision resolution, only the bits that differ in the collided keys are checked. The format of a trie table entry is shown in Figure 18-38.

Note that one double word table location constitutes a complete binary node with both left (L) and right (R) links. A trie table entry resolves 1-bit of collision at a time. As the table is traversed, the COUNTL and COUNTR fields in the previous node are used to specify which bit of the key is to be evaluated in the current node. The value (0 or 1) of the bit being evaluated indicates whether to take the left branch or the right branch (0 = left branch, 1 = right branch) of the tree.

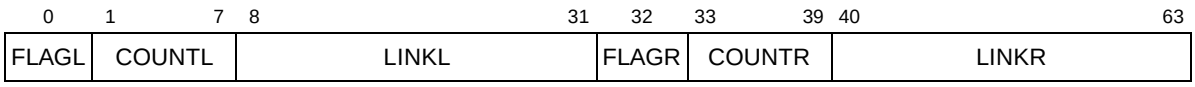


Figure 18-38. Trie Simple Table Entry Format

In the case where the TLU commences its lookup on a trie table (as in an index-trie-data data structure) with an initial index value *I*, the trie entry at index *I*/2 is accessed and the LSB of *I* determines whether the left (LSB = 0) or right (LSB = 1) portion is used.

The trie entry is defined in Table 18-42.

Table 18-42. Trie Entry Field Descriptions

Bits	Name	Description
0	FLAGL	Left chain flag. Indicates next entry format. 0 Next entry is a trie entry used to resolve a collision at another level. 1 Next entry is a key entry used to determine success of the lookup.
1–7	COUNTL	Left key bit count, with 0 denoting the MSB of the key. Specifies the position of the bit in the key that is used to determine whether to take the left or right branch in the next node pointed to by LINKL. The TLU uses the bit in the key to determine: • If the left (bit = 0) high-order 32 bits of the next trie data is to be used or • If the right (bit= 1) low-order 32 bits of the next trie data is to be used.
8–31	LINKL	Left binary tree link pointer. If LINKL = 0, the lookup is deemed failed, and no more entries are examined regardless of FLAG. However, in the case of FLAGL = 0, LINKL is the index of the doubleword trie table entry to access next, whereas in the case of FLAGL = 1, LINKL is the index of the configured-size key table entry to access next.
32	FLAGR	Right chain flag. Indicates next entry format. 0 Next entry is a trie entry used to resolve a collision at another level. 1 Next entry is a key entry used to determine success of the lookup.
33–39	COUNTR	Right key bit count, with 0 denoting the MSB of the key. Specifies the position of the bit in the key that is used to determine whether to take the left or right branch in the next node pointed to by LINKR. The TLU uses the bit in the key to determine: • if the left (bit = 0) high-order 32 bits of the next trie data is to be used or • if the right (bit = 1) low-order 32 bits of the next trie data is to be used.
40–63	LINKR	Right binary tree link pointer. If LINKR = 0, the lookup is deemed failed, and no more entries are examined regardless of FLAG. However, in the case of FLAGR = 0, LINKR is the index of the double word trie table entry to access next, whereas in the case of FLAGR = 1, LINKR is the index of the configured-size key table entry to access next.

The collision resolution mechanism is illustrated by the example in Figure 18-39. Here, three keys, A (starting with 00010), B (00101), and C (00110), all hash to the same hash entry, the right half of entry H. Since a collision needs to be resolved, H has FLAGR = 0 and LINKR points to the root node of a binary trie starting at R. A FLAG of 0 indicates that a further Trie is being linked, whereas a FLAG of 1 is used to link to a Key entry. In H, COUNTR = 2 indicates that the TLU needs to skip the first 2 bits of the key to differentiate keys in R. The value of the key bit selects either the left half of the linked entry (bit = 0) or the right half (bit = 1).

At bit position 2, key A has a 0 bit, therefore the left half of entry R is used for A, whereas the right half of entry R must distinguish B and C. Given that only A can be matched on the left of R, FLAGL = 1, and LINKL of R points to a key entry, KA, that can be used to match key A entirely. However, in order to match B and C, FLAGR = 0, and LINKR of R must point to another level of trie, T. COUNTR = 3 in entry R indicates that now bit 3 of the key is used to distinguish B from C in T.

At bit position 3, key B has a 0 bit, therefore the left half of entry T points to a key entry at KB, and FLAGL = 1. But at bit 3 key C has a 1 bit, hence the right half of entry T points to a key entry at KC, and FLAGR = 1.

Note that since three keys collide, the TLU needed to access at least two trie entries. In general, TLU needs to access $\log_2 C$ trie entries to efficiently resolve a collision of C keys.

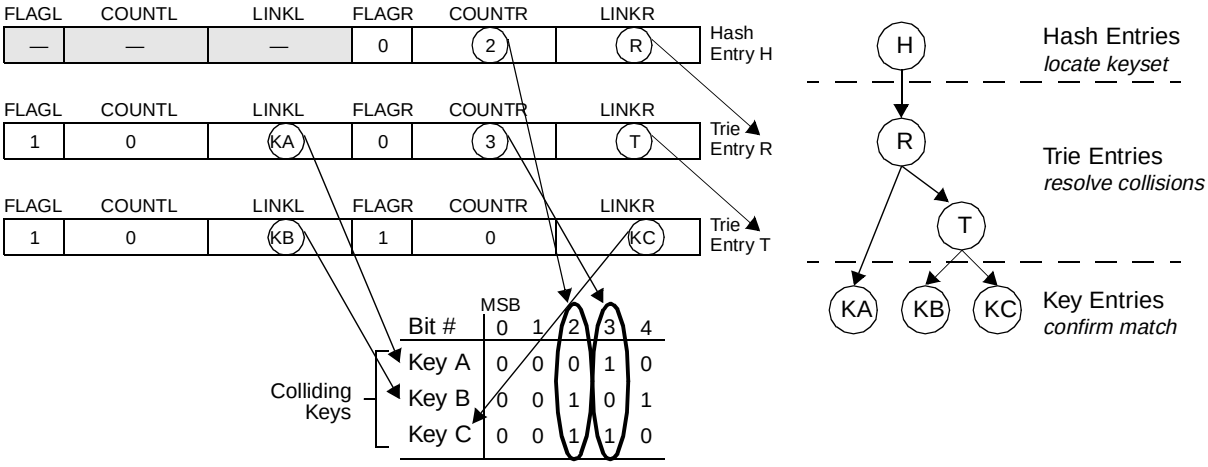


Figure 18-39. Use of COUNT Fields to Resolve Collisions with Trie Entries

18.5.3.4 Key Simple Table

Key tables are used in exact match algorithms, and contain both the key of an entry and the associated data. The last step of an exact match algorithm compares the lookup key with the key from the entry in the key sub-table. The TLU supports key sizes of 32, 64, 96, and 128 bits. The associated data portion of the entry in a key sub-table may be returned by a findr command with an OFFSET value set to skip the key portion. For example, for a 128-bit key, setting OFFSET = 2 skips the first two double words of the key portion. A key data structure occupies the first one or two double words of a data entry (see Section 18.5.3.1, “Data Simple Table”). It is typically the termination of trie and hash tables. For key sizes up to 64 bits, only one double word is used for the key, whereas for larger key sizes, both double words are used. The format of a key table entry is shown in Figure 18-40 for 32-bit and 64-bit keys, and in Figure 18-41 for 96-bit and 128-bit keys.

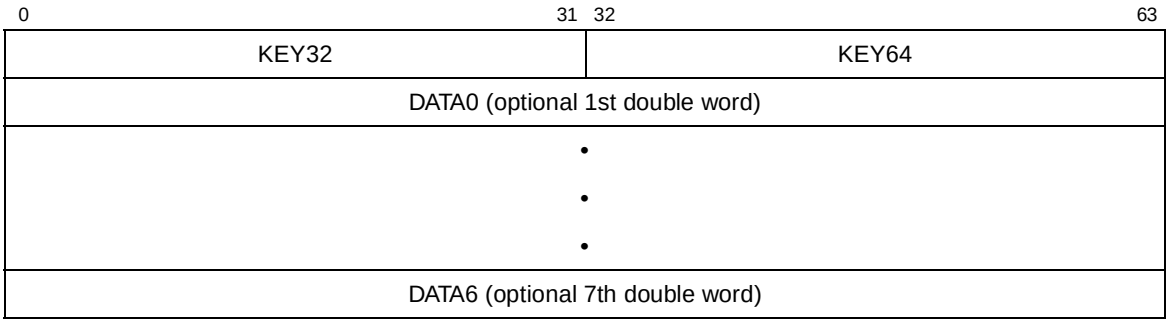


Figure 18-40. Key Simple Table Entry Format for 32-Byte/64-Byte Keys

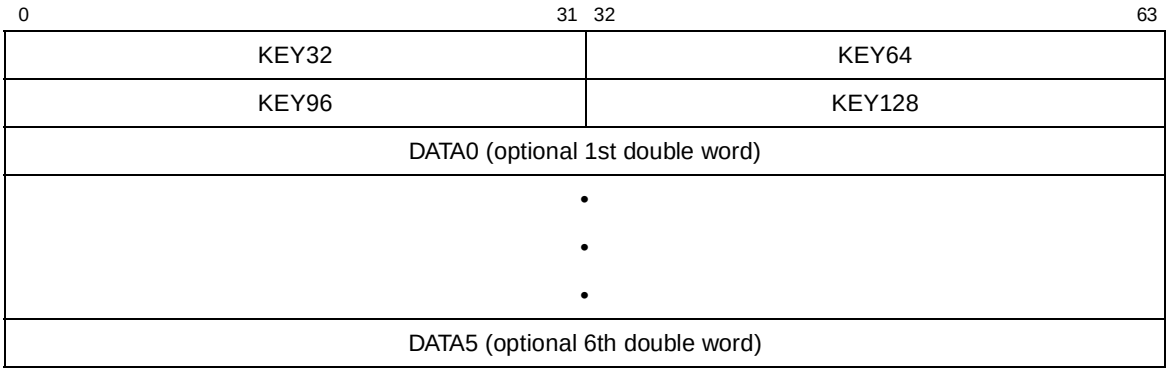


Figure 18-41. Key Simple Table Entry Format for 96-Byte/128-Byte Keys

The key entry is defined in Table 18-43.

Table 18-43. Key Entry Field Descriptions

Bits	Name	Description
0–31	K32	Key to match against lookup key. • All 32 bits of a 32-bit key. • Upper 32 bits of 64-bit, 96-bit, or 128-bit keys.
32–63	K64	Key to match against lookup key. • Lower 32 bits of a 64-bit key. • 32 bits of 96-bit or 128-bit keys. • Unused (don't care) for 32-bit keys.
0–31	K96	Key to match against lookup key. • Lower 32 bits of a 96-bit key. • 32 bits of a 128-bit key. • Unused for 32-bit and 64-bit keys.
32–63	K128	Key to match against lookup key. • Lower 32 bits of a 128-bit key. • Unused for 32-bit, 64-bit, and 96-bit keys.
0–63	DATA0–DATA7	Optional user-defined data associated with key. • For 32-bit and 64-bit keys DATA0–DATA6 or a maximum of 7 double words may be associated with the key. • For 96-bit and 128-bit keys DATA0–DATA5 or a maximum of 6 double words may be associated with the key.

18.5.3.5 Index-VPT-Data Simple Table

Index-variable prefix trie-data tables are the underlying data structure of the compressed radix trie (CRT) longest-prefix and indexed search algorithms. Each such sub-table is a trie node comprising a contiguous array of IPTD entries. Sub-table entries link recursively to subsidiary sub-tables via base pointers to form *n*-way tries composed of multiple sub-tables. The given search key is consumed by the TLU from left to right, up to 16 bits per sub-table. Each portion of the key indexes a sub-table of *n* entries to read an IPTD entry and act on its entry type (ETYPE) field. According to the entry type, the TLU determines either failure (no further searching), a successful match (the longest-prefix has matched entirely), or additional trie traversal (the longest-prefix has matched this far, but without conclusion). As an optimization, a trie

compression algorithm may be invoked to minimize the size of sub-tables containing largely redundant data, as is common with expanded prefix tables. A description of the CRT longest-prefix match algorithms appears in Section 18.5.5.1, “Compressed Radix Trie (CRT).” The format of an IPTD table entry is shown in Figure 18-42.

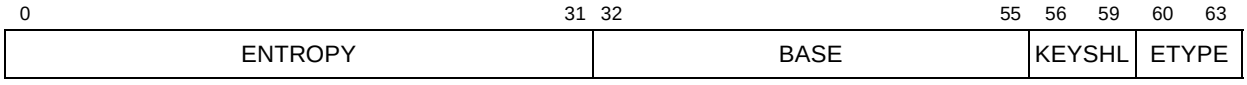


Figure 18-42. IPTD Simple Table Entry Format

The IPTD entry is defined in Table 18-44.

Table 18-44. IPTD Entry Field Descriptions

Bits	Name	Description
0–31	ENTROPY	Entropy encoding bits for compressing the linked sub-table. This allows the compressed sub-table contents to be decompressed without loss. It also allows the search procedure to convert any desired offset in the uncompressed sub-table into the appropriate offset in the compressed sub-table without having to read any of the compressed sub-table contents. The interpretation of ENTROPY is dependent on ETYPE.
32–55	BASE	Sub-table base address. This is the beginning doubleword address where the sub-table is stored, or the index of the Data entry in the case where next hop data is accessed. Sub-tables are stored in contiguous doubleword locations starting from here.
56–59	KEYSHL	Key shift left. This is the number of key bits, minus one, consumed by the CRT step. Thus, the encoded value ranges from 0–15 to denote consumption of another 1–16 bits of the key. Since this field indicates how many positions to shift-left the key following this CRT stage, it is equal to $\log_2$ of the uncompressed size of the sub-table.
60–63	ETYPE	Entry type. Defines the type of compression used to encode the nature of the data linked to by BASE. See descriptions of each type following in this section. 0000 FAIL—Fail entry. 0001 DATA—Next hop pointer pointing to a Data entry at index given by BASE. 0010 SIMPLE—Next sub-table is a simple, uncompressed table. 0011 HASH—Pointer to a hash table for implementing the chained hash structure. 0100 RUNDELTA—Run-length and delta coded compressed trie for CRT. 0101–1111 Reserved

The basic CRT search algorithm follows. In the code, `entry` is the IPTD entry being examined at the current level, `ptbl` is the PTBL*n* register associated with the lookup, `searchKey` is the search key, and `tableSpace` is the array in memory occupied by the IPTD table. Function `high_order_bits(K, N)` returns a bit vector of the `N` most significant bits of `K`, while function `replace_high_bits(N, K, L)` replaces the `N` most significant bits of bit vector `K` by the bits in vector `L`. Initially, `entry` is set to point to the initial SIMPLE sub-table, which is always assumed to start at index 0, and `inithash` is set to the result of the hash function applied to `searchKey` (with its `ptbl.MASK+1 MSBs` cleared).

```
// Compressed Radix Trie search algorithm: TLU searching IPTD tables
// Initialization from PTBLn
entry.ETYPE                                 = SIMPLE;                   // SIMPLE
table type to begin with always
entry.BASE                                  = 0;                        // Initial
table starts at index 0 of tableSpace
entry.KEYSHL                                = ptbl.MASK;
```

```
// Main CRT traversal loop
while (entry.ETYPE == SIMPLE || entry.ETYPE == RUNDELTA) {
        shl                     = entry.KEYSHL + 1;
        // extract shl MSBs from searchKey
        keyBits                 = high_order_bits(searchKey, shl);
        // index next sub-table according to entry data and high order key bits
        if (entry.ETYPE == RUNDELTA)
                index           = entry.BASE + offset_RUNDELTA(entry.ENTROPY, keyBits,
shl);
        else
                index           = entry.BASE + offset_SIMPLE(entry.ENTROPY, keyBits, shl);
        // get next entry according to final index
        entry                   = tableSpace[index];
        // shift key left for LPM searches only
        if (entry.ETYPE != HASH)
                searchKey                       = searchKey << shl;
}

// Check reason for loop to terminate: return result or fail otherwise
if (entry.ETYPE == DATA) {
        return tableSpace[entry.BASE << ptbl.SIZE];
} else if (entry.ETYPE == HASH) {
        // shl MSBs of entry.ENTROPY replace shl MSBs of searchKey
        newKeyBits                      = high_order_bits(entry.ENTROPY, shl);
        replace_high_bits(shl, searchKey, newKeyBits);
        // modify original hash for new mapping
        new_hash                        = inithash + entry.ENTROPY.hashadj;
        // execute hash-trie-key lookup on table of size found, with new hash
        return hash_trie_key(entry.BASE, entry.KEYSHL + 1, new_hash);
} else
        return keyNotFound;
```

The following sections describe each IPTD sub-table type and their associated index calculations.

18.5.3.5.1 FAIL ETYPE

The FAIL ETYPE indicates that a null leaf (no matching prefix) has been reached. The data word is ignored, and the CRT loop terminates.

18.5.3.5.2 DATA ETYPE

The DATA ETYPE indicates that a valid leaf (best matching prefix) has been reached. The BASE field is the index of the data pool entry that contains the next hop data. Note that the data index is generally pre-scaled by PTBL*n*[SIZE], the size of a data entry, which may be larger than a double word. This index is returned by the TLU in CSTAT[INDEX].

18.5.3.5.3 SIMPLE ETYPE

The SIMPLE sub-table type indicates that the sub-table at the next level is uncompressed. The ENTROPY field is ignored. Normally, SIMPLE is only used for the initial sub-table of a CRT structure, although a simple CRT table can comprise SIMPLE sub-tables at all levels of the trie if it is densely populated.

The code for calculating the index offset for SIMPLE sub-tables is as follows.

```
int
```

```
offset_SIMPLE(unsigned int entropy, unsigned int keyBits, int keyShl) {
        return keyBits;
}
```

18.5.3.5.4 HASH ETYPE

The HASH sub-table type is used only to implement chained hash tables. The initial IPTD table lookup is indexed by the KEYSHL MSBs of the search key. The result is an ENTROPY field from the table entry that maps the existing high key bits to a new key, newkey, which is used to replace the KEYSHL MSBs of the search key before commencing a regular hash-trie-key lookup sequence. As a result, in order for an eventual key entry to match, its key contents must include newkey in the KEYSHL MSBs. If each newkey is set to simply the same value as the existing high key bits (or the IPTD index) then no mapping takes place in effect, and key entries match given search keys. However, in the case where existing keys map *N* to 1 into a new set, ENTROPY can be used to make this grouping.

In the case where the TLU commences with an IPTD table type it assumes that either key hashing is not required (for LPM lookups), or that a chained hash structure will appear. To cater for the latter, the TLU substitutes zero-bits for the KEYSHL MSBs of the search key before computing a hash on the key; the search key itself is not changed however. Call this initial hash value `inithash`. Clearing the high bits of the key for hashing ensures that the hash search is not affected by any possible remappings via the initial IPTD lookup. Once a ETYPE of HASH is found, the initial hash can be modified by adding the value hashadj to it, which has the effect of adjusting the hash for the change in key.

The ENTROPY format for the HASH ETYPE is shown in Figure 18-43.

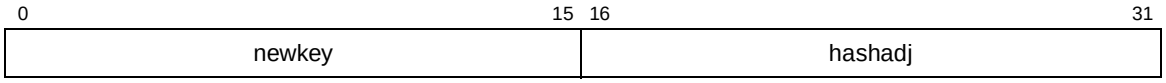


Figure 18-43. ENTROPY Format for HASH ETYPE

18.5.3.5.5 RUNDELTA ETYPE

The RUNDELTA ETYPE combines delta and run length encodings to compress the sub-table. As usual, the `shl` value indicates the uncompressed sub-table size, which must be at least 32. Thus, `shl` ≥ 5.

The uncompressed sub-table is partitioned into 16 blocks, BLK0–BLK15, each of size 2^{shl-4} entries. Each pair of bits—or dibit—in the ENTROPY field corresponds to one block, as shown in Figure 18-44.

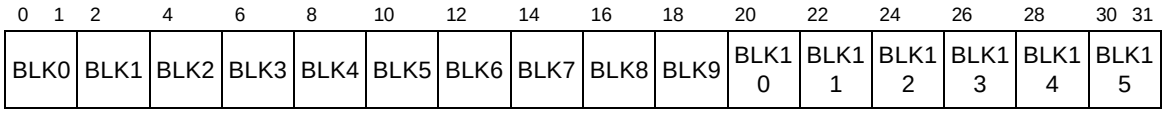


Figure 18-44. ENTROPY Format for RUNDELTA ETYPE

The sixteen ENTROPY dibits are encoded as follows:

- For BLK0, encode:
  - A 0b01 dibit if the block is stored compressed to a single entry when all 2^{shl-4} uncompressed entries are identical;
  - A 0b11 dibit if the block is stored fully uncompressed;

  - Dibits 0b00 and 0b10 are reserved and should not be used for BLK0, or the TLU will behave unpredictably.
- For BLK*i*, *i* > 0, encode:
  - A 0b00 dibit if every entry in block *i* is identical to the last entry of block (*i* – 1), and no entries are stored in the sub-table for block *i*;
  - A 0b01 dibit if block *i* differs from the last entry of block (*i* – 1), and one entry of block *i* is stored in the sub-table since there is exactly one distinct value in the block;
  - A 0b10 dibit if the first 2^{shl-5} entries of block *i* are identical to the last entry of block (*i* – 1), and the second 2^{shl-5} entries of block *i* are stored in the sub-table;
  - A 0b11 dibit if block *i* differs from the last entry of block (*i* – 1), and all entries of block *i* are stored in the sub-table since there is more than one distinct value in the block.

The high-order two bits of ENTROPY hold the dibit for BLK0, while BLK15 uses the low-order two bits of ENTROPY. Only BLK0 and the blocks that correspond to non-zero dibits are stored in the compressed sub-table. All other blocks that correspond to 0b00 dibits are omitted. To decompress back to the original sub-table, each 0b00 dibit causes the most recently encountered entry to be replicated 2^{shl-4} times. The function of uncompression, however, is to emulate a given index into the uncompressed sub-table by mapping it to the equivalent index in the compressed sub-table.

The offset calculation is done as follows for the RUNDELTA case:

1. The high order 4 bits of the `keyBits` determine which of the 16 blocks the access falls into.
2. This 4-bit block number is used to index one of BLK0–BLK15 in the 32-bit ENTROPY field, starting from the left for BLK0.
3. For each dibit before that indexed by the block number, count the number of entries that were stored according to the run-length encoding—1, 2^{shl-5} entries, or 2^{shl-4} entries. This is the number of entries that were stored in the sub-table before the desired block. Ignore the 0b00 dibits, as these correspond to blocks that were omitted from the sub-table.
4. The count from step 3 counts how many blocks to skip over in the sub-table—call this count `entries_skipped`. The dibit indexed by the block number determines whether the low-order bits of `keyBits` are needed to complete the index offset calculation, as follows:
   - If the current dibit indicates 0b00, then `entries_skipped-1` indexes the last repeated entry;
   - If the current dibit indicates 0b01, then `entries_skipped` indexes the stored entry;
   - If the current dibit indicates 0b10 and the `shl-4` LSBs of `keyBits` $\check{S} \geq 2^{shl-5}$, then the `shl-5` LSBs of `keyBits` are added to `entries_skipped` so as to index into the second half of the current block, otherwise `entries_skipped-1` indexes the last repeated entry;
   - If the current dibit indicates 0b11, then the `shl-4` LSBs of `keyBits` are added to `entries_skipped` so as to index into the current block.

The code for calculating the index offset for RUNDELTA sub-tables is as follows. The function `popcount(N)` returns the number of bits in `N` set to “1”.

```
int
offset_RUNDELTA(unsigned int entropy, unsigned int keyBits, int keyShl) {
        unsigned int                                    blknum_by2, entropy_mask, dibit_highbits;
        int                                             entries_skipped;
```

```
        // blknum_by2 = 2 * current block index, which skips bits of ENTROPY
        blknum_by2                                          = (keyBits >> (keyShl–4)) << 1;
        // duplicate high-order bit of each dibit to distinguish dibit type
        dibit_highbits                                      = entropy & 0xaaaaaaaa;// mask
high bits
        dibit_highbits                                      |= dibit_highbits >> 1;// shift
and duplicate
        // mask out all ENTROPY bits except those left of current block
        entropy_mask                                        = ~((unsigned int)0xffffffff >>
blknum_by2);
        // we've skipped both 1 and half/full entries per block cases: count total
        entries_skipped                                     =
                 popcount(entropy & ~dibit_highbits & entropy_mask) +
                (popcount(entropy &  dibit_highbits & entropy_mask) << (keyShl–5));

        // index last stored entry or current block according to current dibit
        switch ((entropy >> (30 – blknum_by2)) & 3) {
                case 0:                     return entries_skipped – 1;
                case 1:                     return entries_skipped;
                case 2:                     return (keyBits & (1 << (keyShl–5)))?
                                                        entries_skipped +
(keyBits & ~(~0 << (keyShl–5))) :
                                                        entries_skipped – 1;
                case 3:                     return entries_skipped + (keyBits & ~(~0 <<
(keyShl–4)));
        }
}
```

The following examples show how RUNDELTA compression and decompression operate.

- *Example 1:* Fully-compressed and uncompressed blocks with runs:

  `[AAAA AAAA BBBB BBBB CCDD DDDD DDDD EFGH HHHH HHHH BBBB BBBB JJJJ JJJJ JJJJ JJJJ]`

  SHL = 6 (uncompressed size 64; block size = 2^{SHL-4} = 4)

  ENTROPY = “01.00.01.00.11.00.00.11.00.00.01.00.01.00.00.00”

  stored sub-table = `[A B CCDD EFGH B J]`

  (12 entries encoding 64)

- *Example 2:* Partially-compressed and uncompressed blocks with runs:

  `[ABCD DDDD DDEE EEFF FFFF GHHI IIJJ JJJJ JJJJ KKKK KKKK KKLL LLLL LLMM MMMM MMMM]`

  SHL = 6 (uncompressed size 64; block size = 2^{SHL-4} = 4)

  ENTROPY = “11.00.10.10.00.11.10.00.00.01.00.10.00.10.00.00”

  stored sub-table = `[ABCD EE FF GHHI JJ K LL MM]`

  (19 entries encoding 64)

- *Example 3:* Blocks without runs, but still compressible:

  `[AABB CDDD EEEE FFFF GGGH IIII JJJJ KKKK LLLL LMMM NNNN OOOO PPPP QQQR SSSS TTTT]`

  SHL = 6 (uncompressed size 64; block size = 2^{SHL-4} = 4)

  ENTROPY = “11.11.01.01.11.01.01.01.01.11.01.01.01.11.01.01”

  stored sub-table = `[AABB CDDD E F GGGH I J K L LMMM N O P QQQR S T]`

  (31 entries encoding 64)

- *Example 4:* Decompression of Example 2 when `keyBits = 27` = “0110.11”:

[ABCD DDDD DDEE EEFF FFFF GHHI IIJJ JJJJ JJJJ KKKK KKKK KKLL LLLL LLMM MMMM MMMM]

SHL = 6 (uncompressed size 64; block size = 2^{SHL-4} = 4)

ENTROPY = "**11.00.10.10.00.11**.*10*.00.00.01.00.10.00.10.00.00"

stored sub-table = [ABCD EE FF GHHI JJ K LL MM]

block number = 27 >> (6–4) = 6, blknum_by2 = 12, entropy_mask = 0xFFF0000

dibit_highbits = "11.**00.11.11.00.11**.11.00.00.00.00.11.00.11.00.00"

~dibit_highbits = "**00.11.00.00.11.00**.00.11.11.11.11.00.11.00.11.11"

entries_skipped = popcount(0xCA300000)*2 + popcount(0x0) = 12

indexed dibit = "10" (skip half, store half uncompressed), keyBits & "11" = 3 ≥ 2^1

offset_RUNDELTA = entries_skipped + 1 = 13 (points to second entry J).

18.5.4 Exact Match Algorithms

18.5.4.1 Flat Data

Flat data tables are the simplest tables for exact match operations. The table consists of a linear array in TLU memory, and the search key is used directly as an index into the data table. In this case, the search key is limited to a maximum of 32 bits (for all other tables it can be up to 128 bits long). However, only the PTBL*n*[MASK] least significant bits of the key are used to index the table. For keys denoted longer than 32 bits, the TLU will assume that a 32-bit key has been provided regardless. The masked key value is multiplied by the data table entry size (8, 16, 32, or 64 bytes) and then added to the table base address in order to access an entry in the table. The TLU returns the table index in CSTAT, and reads the entry in the case of a findr command. The operation of the flat data algorithm is depicted in Figure 18-45.

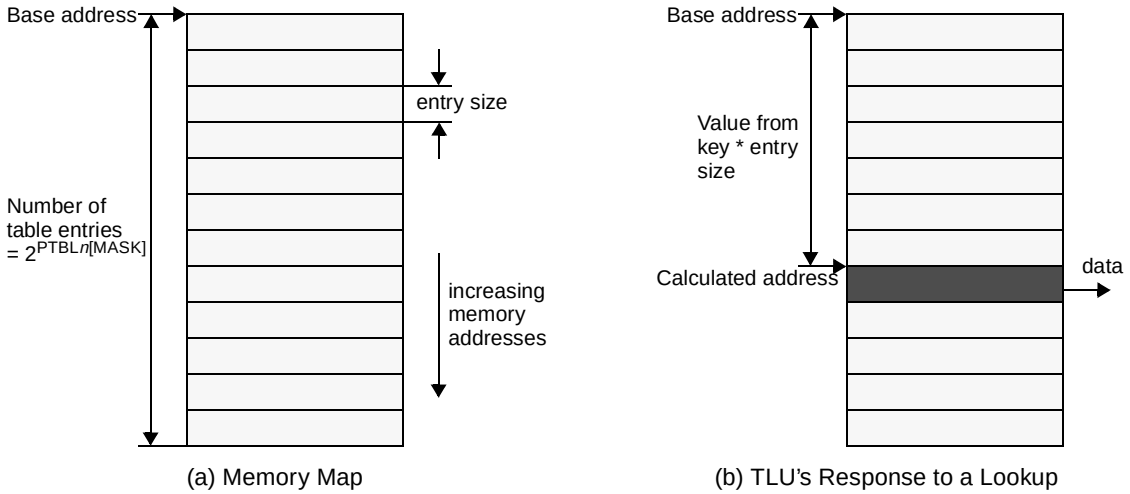


Figure 18-45. Flat Data Recommended Memory Layout

Simple data tables are recommended only if the set of keys is dense (ideally contiguous) so that space in the table is not wasted by entries that do not contain valid data. No attempt is made to track whether entries contain valid data, therefore CSTAT[FAIL] = 0 unless an error occurs. If this is an issue, it is recommended that the table be initialized with all entries set to a distinctive invalid pattern that will not occur in valid

data. The search will always succeed, but the application can then test the data returned from the lookup for the invalid pattern.

18.5.4.2 Hash-Trie-Key

The hash-trie-key algorithm is formed by linking together a hash table, a trie table and a key table as shown in Figure 18-47. Hash table entries may contain an index into the trie table, an index into the key table, or a failure indicator (a zero index). Trie table nodes comprise a left link and a right link. The left and right links can be either an index of the next node in the trie table, an index into the key table, or a fail indicator. Key table entries contain the key and the data associated with that key.

The example structure shown in Figure 18-46 hashes both keys A and B to hash result 2, which corresponds with the left half of hash entry 1; keys C, D, E, F and G all hash to 11, which corresponds with the right half of hash entry 5; and key H hashes to 14, which corresponds with the left half of hash entry 7. Hash table entries 1 and 5 contain indices into the trie table to resolve the collisions on these hashes, whereas hash table entry 7 points directly to the key table.

Hash-trie-key tables are recommended for use in all cases where an exact match lookup is required, except for cases where a flat data table is appropriate.

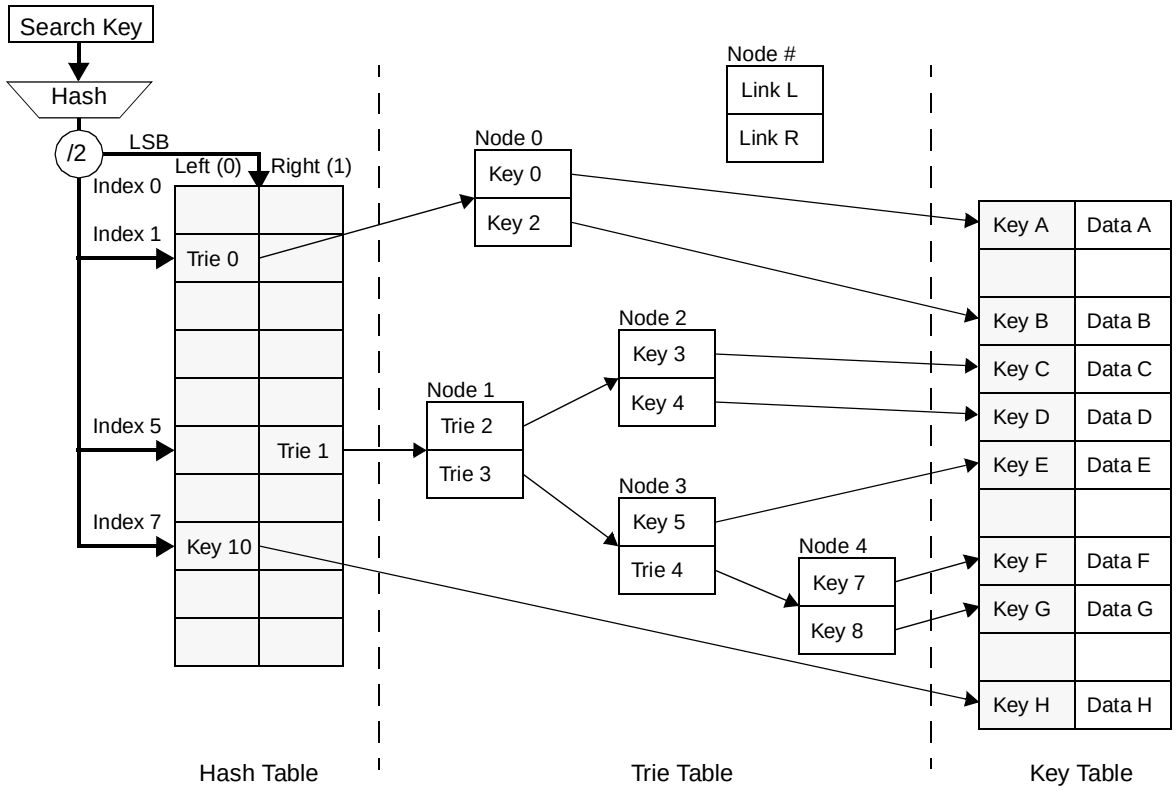


Figure 18-46. Hash-Trie-Key Data Structure

The layout of the hash-trie-key data structure, flattened, in memory is illustrated in Figure 18-47. Note that the hash simple table always coincides with the base of the complex table.

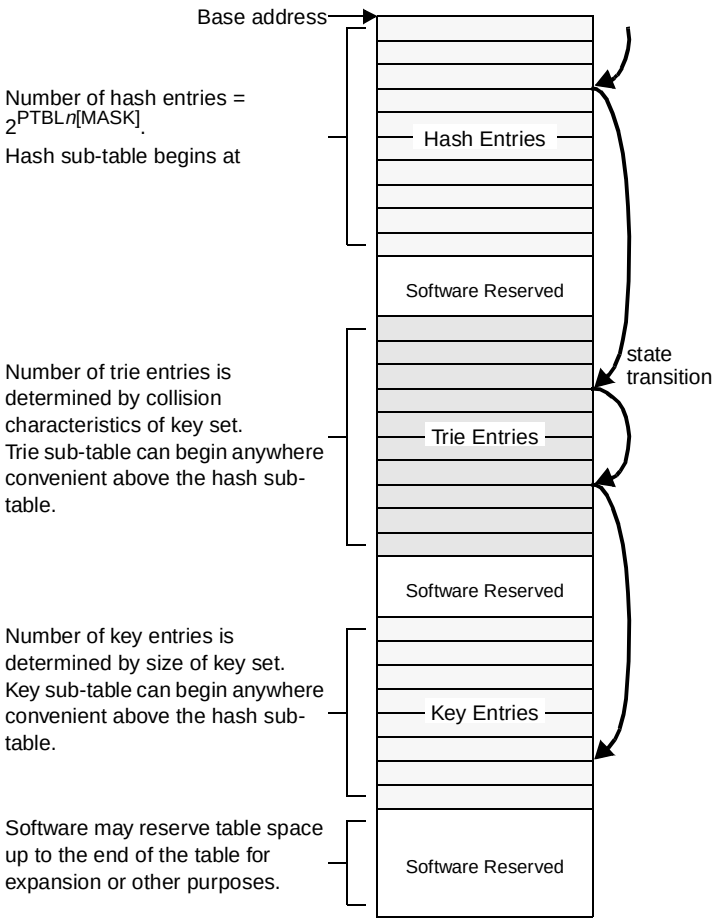


Figure 18-47. Typical Memory Layout of Hash-Trie-Key Tables

The hash-trie-key algorithm proceeds as follows, with state transitions shown in Figure 18-48:

1. The search key is hashed using a fixed hash function (see Section 18.5.6, “TLU Hash Function,” on page 18-61) and the value returned by the hash function is used as an index into the hash table.
2. The hash table may contain the index of a node in the key table, or the index of an entry in the trie table, or may be a fail indicator.
   a) If the search key yields a unique hash value, the hash table link is an index into the key table, the trie table is bypassed and TLU goes directly to step 3.
   b) If more than one key value yielded the same hash value, the hash table link is an index into the trie table, which is used to resolve hash collisions. The trie table is walked using successive bits of the search key to determine whether to take the left (bit = 0) or right (bit = 1) branches at each node until either a link to the key table is found, in which case TLU goes to step 3, or a fail indicator is encountered, in which case the search fails.
   c) If the hash table entry contains a fail indicator, the search fails

3. In case 2a where a key table index was found, or case 2b, TLU now has an index into the key table. This table contains both a key and the data associated with the key. The search key is compared with the key stored in the key table entry.
   a) If the search key matches the key in the key table, then TLU has found the data required, and the associated data can be read from the key table and returned as the result
   b) If the search key and the key from the key table do not match, the search fails.

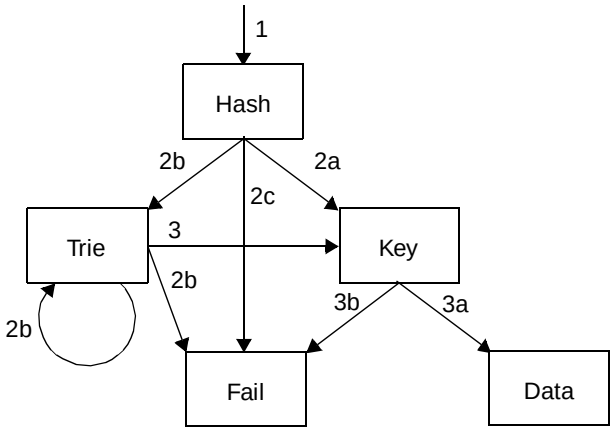


Figure 18-48. Hash-Trie-Key State Transitions

18.5.4.3 Chained-Hash

The chained-hash algorithm is formed by linking together an IPTD table with a regular hash-trie-key table as shown in Figure 18-49. IPTD table entries may either fail the lookup immediately, or vector to a hash sub-table to resolve multiple keys mapping to the same IPTD entry. Hash entries may contain an index into the trie table, an index into the key table, or a failure indicator (a zero index). Trie table nodes comprise a left link and a right link. The left and right links can be either an index of the next node in the trie table, an index into the key table, or a fail indicator. Key table entries contain the key and the data associated with that key. However, unlike pure hash-trie-key tables, key entries for the chained-hash algorithm hold a modified key where the PTBL*n*[MASK]+1 most significant bits of the original key are replaced by bits taken from the IPTD entry's ENTROPY field. Such key replacement allows the IPTD lookup to compress a set of keys to a group sharing a common prefix.

The example structure shown in Figure 18-49 maps the PTBL*n*[MASK]+1 high order bits of the search key to a new key, K2, for high order bits of value 5, 8, or 10. Other values map to other hash tables, or fail immediately. In the case where the high order bits equal 5, the TLU looks up the base of the hash table, as Tab2, replaces the high order bits of the key by K2, and modifies the initial hash (computed on the low order bits of the key) by adding H2 to it to produce a final hash. The final hash is an offset into a regular hash-trie-key data structure, hence the hash is divided by 2 before indexing into the selected hash table. Supposing that the sum (initial hash + H2) = 6, the left half of entry 3 in hash table 2 points directly to a key entry that matches key G. In order for the TLU to successfully match G, however, G must be a concatenation of partial key K2 with the low order bits of the original search key. Even though this example depicts multiple hash tables, one for each keyset K0, K1, and K2, it is typically practical to merge all lookups over a single, large hash table, and rely on key replacement to distinguish one keyset from another.

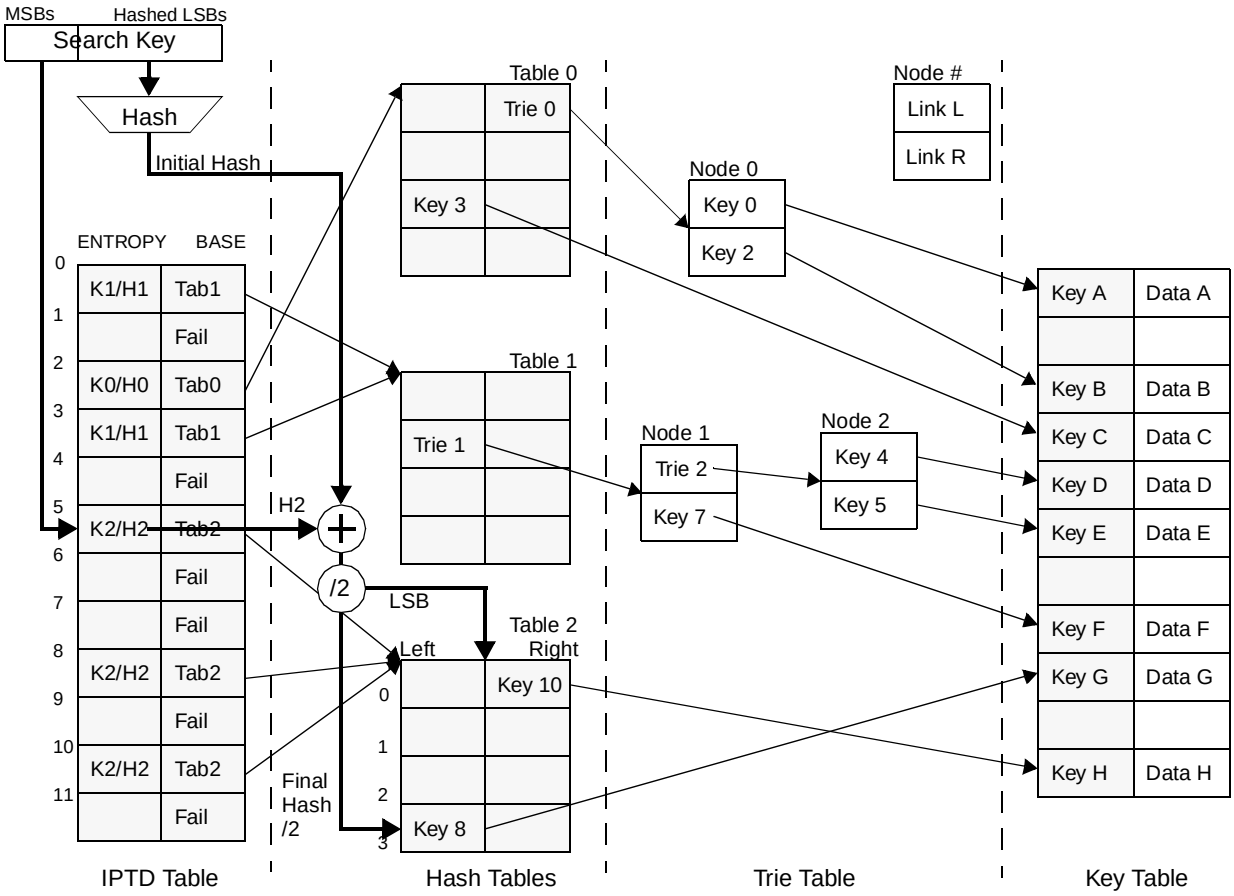


Figure 18-49. Chained-Hash Data Structure

The layout of the chained-hash data structure, flattened, in memory is illustrated in Figure 18-50. Note that the IPTD simple table always coincides with the base of the complex table.

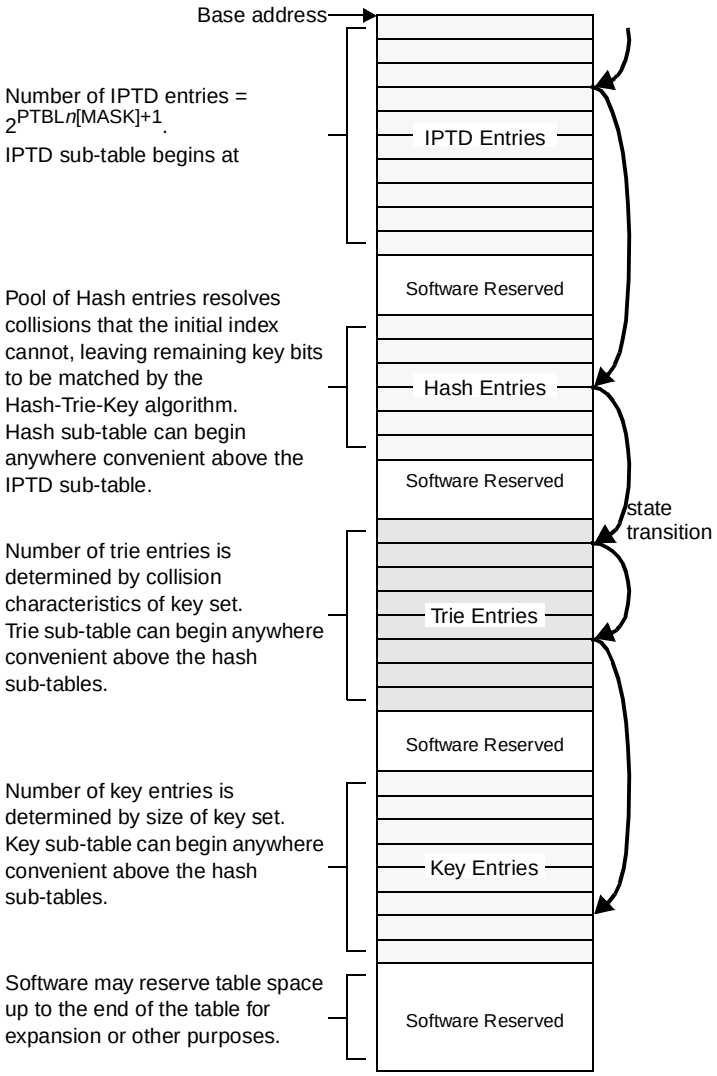


Figure 18-50. Typical Memory Layout of Chained-Hash Tables

The algorithm proceeds as follows, with state transitions shown in Figure 18-51:

1. Initial state is set to IPTD based on the PTBL*n* register used. In the IPTD state, the TLU performs a memory read, then transitions to either the hash or failure states. Specifically, the TLU:
   a) Computes the first address to read as the first PTBL*n*[MASK]+1 bits of the key indexed to the base address of the table.
   b) Hashes the remaining bits of the key (the first PTBL*n*[MASK]+1 bits are taken as zero) with a fixed hash function (see Section 18.5.6, “TLU Hash Function,” on page 18-61) to create an initial hash value.
   c) Parses the IPTD index formatted data. On reading the entry, the entry type is checked. If ETYPE = FAIL, the lookup has no corresponding hash table to transition to, so the TLU lookup fails. If ETYPE = HASH, the sub-table BASE points to the start of a hash table, and the TLU continues as if beginning a hash-trie-key lookup with the hash size and start of table being programmed into the IPTD entry.

2. The search key is modified and a new hash value is computed. Specifically, the TLU:
   a) Replaces the first PTBL*n*[MASK]+1 bits of the key with a value found in the IPTD entry ENTROPY field. See Section 18.5.3.5.4, “HASH ETYPE,” on page 18-48 for details of HASH ENTROPY layout.
   b) Modifies the initial hash computed in step 1b by adding a value found in the IPTD entry ENTROPY.
   c) Uses the modified hash value as an index into the hash table starting from the base given by the previous IPTD entry.
3. The hash table may contain the index of a node in the key table, or the index of an entry in the trie table, or may be a fail indicator.
   a) If the search key yields a unique hash value, the hash table link is an index into the key table, the trie table is bypassed and TLU goes directly to step 4.
   b) If more than one key value yielded the same hash value, the hash table link is an index into the trie table, which is used to resolve hash collisions. The trie table is walked using successive bits of the search key to determine whether to take the left (bit = 0) or right (bit = 1) branches at each node until either a link to the key table is found, in which case the TLU goes to step 4, or a fail indicator is encountered, in which case the search fails.
   c) If the hash table entry contains a fail indicator, the search fails
4. In case 3a where a key table index was found, or case 3b, TLU now has an index into the key table. This table contains both a key and the data associated with the key. The modified search key formed at step 2a is compared with the key stored in the key table entry.
   a) If the modified search key matches the key in the key table, then TLU has found the data required, and the associated data can be read from the key table and returned as the result
   b) If the modified search key and the key from the key table do not match, the search fails.

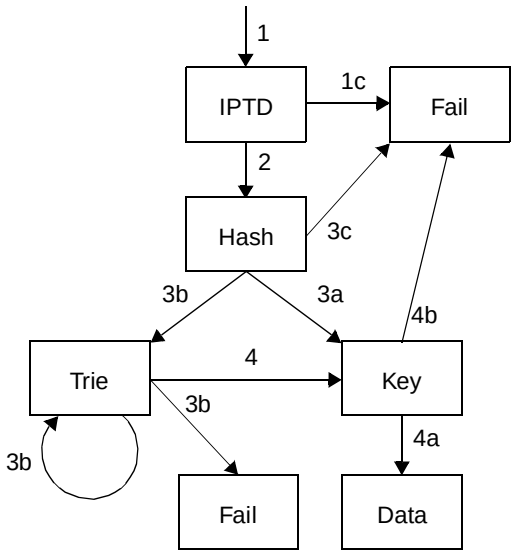


Figure 18-51. Chained-Hash State Transitions

18.5.5 Longest-Prefix Match Algorithms

18.5.5.1 Compressed Radix Trie (CRT)

The CRT data structure is formed by linking together an initial IPTD index table, followed by trie sub-tables, possibly compressed, as shown in Figure 18-52. As the search key is consumed from left to right, each part of the key is used to index into the next level of the trie—either by simple addition, as shown in the example for SIMPLE entry types, or by a compressed index calculation as described in Section 18.5.3.5, “Index-VPT-Data Simple Table,” on page 18-45. CRT belongs to the family of variable-expansion, compressed radix-*n* trie algorithms commonly used for longest-prefix match lookups.

In general the CRT data structure comprises a large number of linked IPTD sub-tables, although the example in Figure 18-52 shows only the initial sub-table (of which there is one) and two particular subsidiary sub-tables that satisfy the specific search path given. The first (PTBL*n*[MASK]+1) 16 bits of IP address 192.10.72.9 forms an index 192.10 into the initial sub-table. In the example, the result is an L2 sub-table base pointer which defines the start of the L2 sub-table. The next 8 bits of the address (72) are added to the base pointer to form an index into the L2 sub-table.

The L2 sub-table entry is a base pointer, this time to the start of an L3 sub-table. The last 8 bits of the address (9) are added to the base pointer to form an index into the L3 sub-table. The L3 entry contains the data index for route 192.10.72.9/32. The final step is to use the data index to access the data pool and retrieve the next hop data.

Other examples are shown in the figure. The SIMPLE sub-table lookup for an address of the form 192.8.X.X would return the data index for the route 192.8/16 immediately. An address of the form 192.11.X.X would result in a FAIL entry type being read and the TLU would return CTSAT[FAIL] set. Finally, a shorter address of the form 192.10.72.X would return the data index for the route 192.10.72/24 from the lookup of entry 0 in the L3 sub-table.

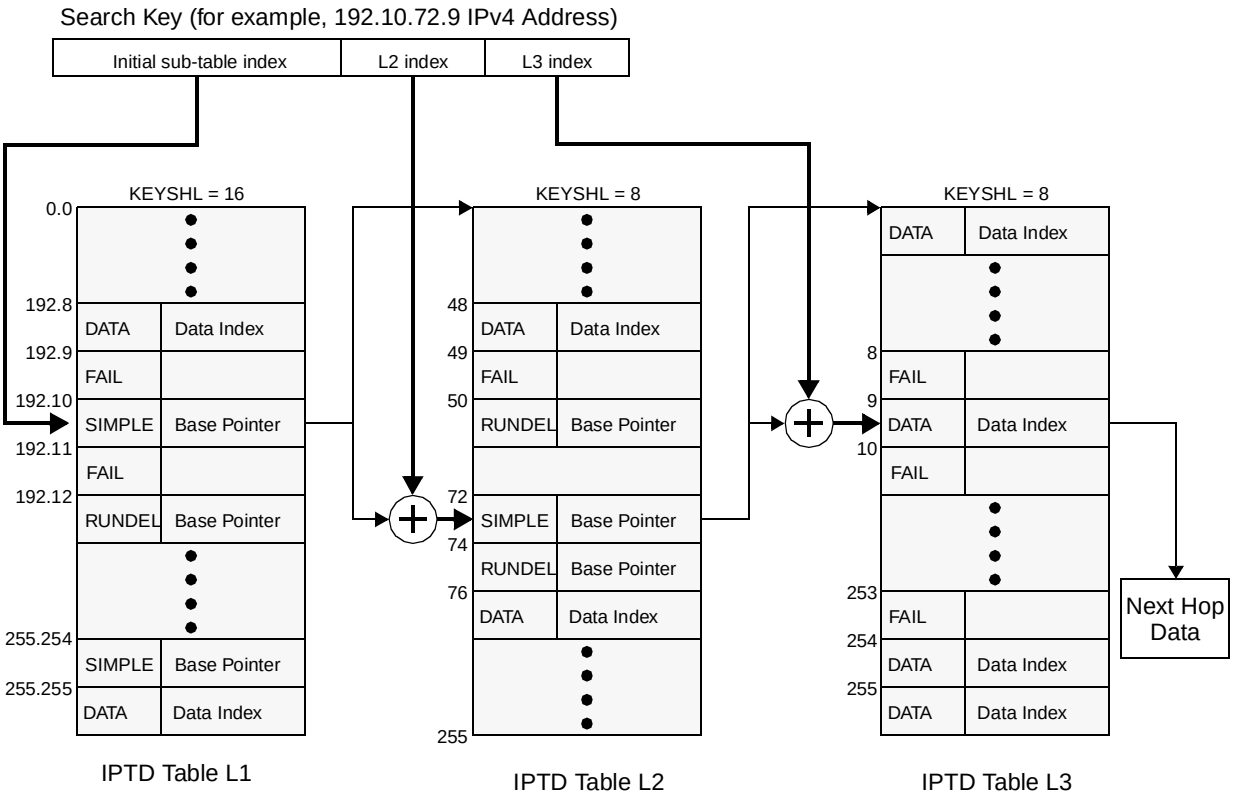


Figure 18-52. CRT Data Structure

The layout of the CRT data structure, flattened, in memory is illustrated in Figure 18-53. Note that the initial sub-table, the IPTD simple table indexed by the most significant key bits, always coincides with the base of the complex table.

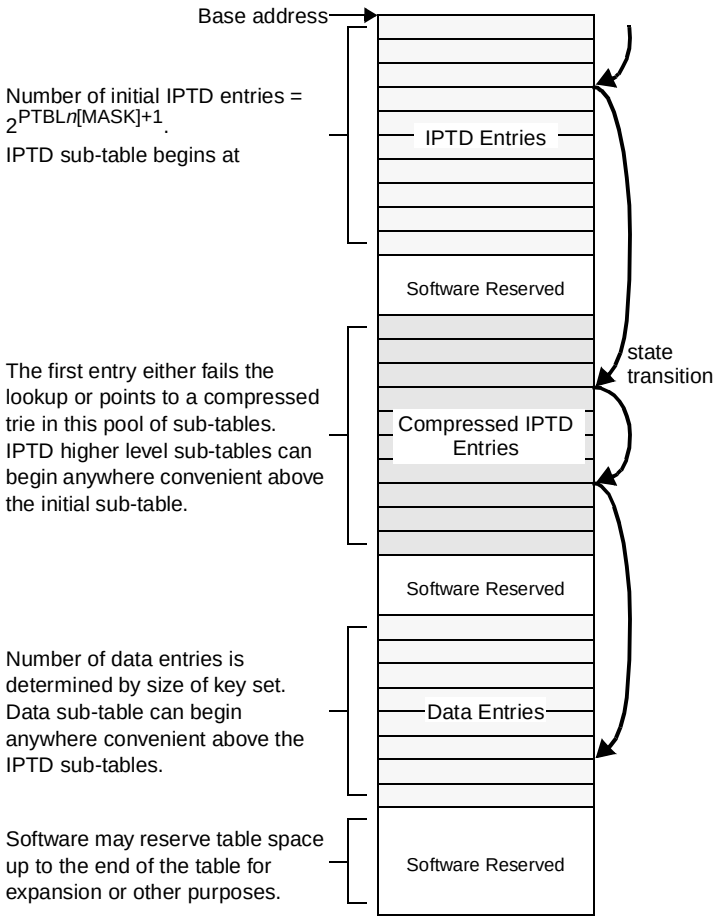


Figure 18-53. Typical Memory Layout of CRT Tables

The algorithm proceeds as follows, with state transitions shown in Figure 18-54:

1. The TLU indexes into the IPTD sub-table, using PTBL*n*[MASK]+1 MSBs of the key. The results of this access is either a pointer to associated data, or another IPTD table entry. The CRT table entry consists of:
   - The base address of the next sub-table (BASE) that holds all the prefixes that are extensions of the prefix from step 1.
   - The number of bits of the key decoded to select an entry from the next sub-table (that is, $\log_2$ of the uncompressed sub-table size), (KEYSHL+1).
   - The compression format of the sub-table (ETYPE).
   - The entropy encoding (ENTROPY) used to compute a random-access into the compressed sub-table.
2. If the result of the previous step is:
   a) ETYPE 0 (FAIL), then there is no match and a miss is returned.
   b) ETYPE 1 (DATA), then BASE points to the key's associated data—go to step 3.
   c) For all other ETYPEs, decode the next address, retrieve a new encoded entry, and repeat step 2.

3. Finally, de-reference the associated data pointer. Note: there is no compare step in the CRT trie search as the key is embodied in the path traversed in arriving at the data.

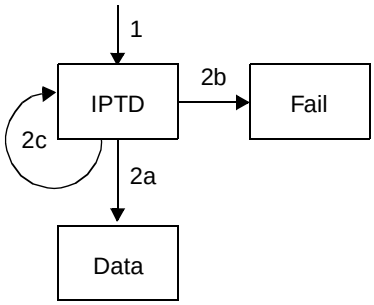


Figure 18-54. CRT State Transitions

18.5.6 TLU Hash Function

The TLU subjects the search key to a randomizing hash function whenever it needs to index simple tables of type hash. A listing, in C, of the function is shown in Figure 18-55. This function is replicated in the TLU hardware, and evaluates typically an order of magnitude faster in the TLU than in software. The design of the function avoids many of the pathologies typically encountered by simple hash functions, such as cyclic-redundancy codes.

```
/*      TLU Hash Function
        Freescale Semiconductor, Inc. */
#define BYTE_MASK(bnum) (0xff << ((bnum)<<3))

static unsigned int remix8(unsigned int a, unsigned int b, int j)
{
        unsigned int c;
        static int shift_rights[9] =
                //  j=0  j=0   j=0   j=3   j=3  j=3  j=6   j=6   j=6
                {    13,   8,   13,   12,   16,   5,   3,   10,   15 };
        static int shift_lefts[9] =
                { 32-13,32-8,32-13,32-12,32-16,32-5,32-3,32-10,32-15 };

        // mix a and b, byte by byte
        a =             (((a & BYTE_MASK(0)) - (b & BYTE_MASK(0))) & BYTE_MASK(0)) |
                        (((a & BYTE_MASK(1)) - (b & BYTE_MASK(1))) & BYTE_MASK(1)) |
                        (((a & BYTE_MASK(2)) + (b & BYTE_MASK(2))) & BYTE_MASK(2)) |
                        (((a & BYTE_MASK(3)) + (b & BYTE_MASK(3))) & BYTE_MASK(3));
        // XOR a by b rotated right random number of bits
        a ^=            (b>>shift_rights[j]) | (b<<shift_lefts[j]);
        // mix b and new a, byte by byte
        b =             (((b & BYTE_MASK(0)) + (a & BYTE_MASK(0))) & BYTE_MASK(0)) |
                        (((b & BYTE_MASK(1)) + (a & BYTE_MASK(1))) & BYTE_MASK(1)) |
                        (((b & BYTE_MASK(2)) - (a & BYTE_MASK(2))) & BYTE_MASK(2)) |
                        (((b & BYTE_MASK(3)) - (a & BYTE_MASK(3))) & BYTE_MASK(3));
        // XOR b by a rotated right random number of bits
        b ^=            (a>>shift_rights[j+1]) | (a<<shift_lefts[j+1]);
        // mix a and b again, byte by byte
        c =             (((a & BYTE_MASK(0)) - (b & BYTE_MASK(0))) & BYTE_MASK(0)) |
                        (((a & BYTE_MASK(1)) - (b & BYTE_MASK(1))) & BYTE_MASK(1)) |
                        (((a & BYTE_MASK(2)) + (b & BYTE_MASK(2))) & BYTE_MASK(2)) |
                        (((a & BYTE_MASK(3)) + (b & BYTE_MASK(3))) & BYTE_MASK(3));
```

```
        // XOR c by b rotated right random number of bits
        c ^=            (b>>shift_rights[j+2]) | (b<<shift_lefts[j+2]);
        // return a mixed with b three times
        return c;
}

/* Generalized version of hash function for arbitrary number of 32-bit
 * words. Uses previous result as part of new key input.
 * key points to an array of 32-bit words of key,
 * length = number of 32-bit words pointed to by key. */
static unsigned int hash_by32(unsigned int *key, int length)
{
        int i;
        unsigned int a, b, c;

        // if odd number of words, zero last word
        if (length & 1)
                key[length++] = 0;

        c = 0x9e3779b9;  // start by arbitrary random seed: golden ratio
        // hash two words at a time, combining in current hash value with each pair
        for (i = 0; i < length; i += 2) {
                a = remix8(key[i], key[i+1], 0);
                b = remix8(a,      c,        3);
                c = remix8(c,      b,        0);
        }
        return c;
}
```

Figure 18-55. TLU Hash Function in C

18.5.7 Initializing and Maintaining Tables

18.5.7.1 Configuration of Local Bus

The TLU has a dedicated connection to the local bus controller (LBC), which handles access to external table memory when MBANK*n*[TGT] = 0. If external memory is not attached to the local bus for use by the TLU, then MBANK*n*[TGT] must be set to 1 for all banks, and each MBANK*n*[BASE] must map to a region controlled by the system DDR controller. For best performance it is recommended that TLU tables map to local bus chipselects controlling zero-bus turnaround (ZBT) SRAM or equivalent fast SRAM technology. These devices are typically controlled by one of the UPM controllers in local bus.

Prior to launching lookups with the TLU, local bus configuration registers must be initialized correctly. Unless properly initialized, the local bus is likely to cause BAE events in IEVENT. A suggested set of initializations are as follows:

1. For each bank of the local bus connected to external table memory (such as ZBT), initialize the BR*x* (base address) registers. Points to note:
   a) The bank base addresses used must not conflict with those of banks used for system boot or non-TLU purposes.

b) Ensure that each TLU MBANK*n*[BASE] maps to a local bus bank. For example, set bits 0–3 of local bus BR*x*[BA] to the 4 LSBs of MBANK*n*[BASE], and clear bits 4–16 of BR*x*[BA]. Clear MBANK*n*[TGT] to assign each bank to the local bus.

c) Set the port size in BR*x*[PS] to match the memory width; PS = 11 sets a 32-bit wide port.

d) If MBANK*n*[PAR] is to be set, enabling parity checking, it will also be needed on the local bus, by setting BR*x*[DECC] to 01. Set local bus register LBCR[EPAR] = 0 for odd parity or LBCR[EPAR] = 1 for even parity.

e) Select the local bus UPM controller C for ZBT by setting BR*x*[PS] to 110. Ensure that BR*x*[V] is set before use.

2. For each bank of the local bus connected to external table memory (such as ZBT), initialize the OR*x* (option register, UPM mode) registers. Points to note:

   a) Set the bank size to match the size of the external memory. If the bank size is equal to or less than 256 Mbytes, follow the local bus directions for setting OR*x*[AM]. Otherwise, multiple TLU banks map to one local bus bank, and the high order bits of OR*x*[AM] should mask out the non-common address bits across the set of MBANK*n*[BASE] values.

   b) For maximum performance with UPM C, do not enable any relaxed timing features, and ensure that OR*x*[BI] = 0 to enable burst transfers over the local bus.

3. Assuming that UPM controller C is being used for TLU accesses, initialize the local bus MCMR mode register. Points to note:

   a) Initially UPM C will need to have a ZBT control program loaded. Do this via control of the local bus MCMR[OP] = 01 opcode to write control words in register MDR to UPM array locations set by MCMR[MAD]. The UPM programs are specific to the manufacturer of the memory; however, include both single access and burst patterns.

   b) Make sure that normal UPM operation is enabled by setting MCMR[OP] = 00 before continuing.

4. Set the local bus clocks in register LCRR. Points to note:

   a) To operate at high bus clock frequencies, enable the skew-correcting PLL by clearing bit LCRR[DBPY].

   b) Set the LALE signal assertion timing to one bus cycle by setting LCRR[EADC] = 01.

   c) For bus clock frequencies in the range 133 MHz to 167 MHz, set the clock divider LCRR[CLKDIV] = 0010 (divide by 2).

5. Ensure that local bus is enabled by clearing LBCR[LDIS].

Chapter 19
DMA Controllers

This chapter describes the DMA controllers of the MPC8572E. This chapter describes a single DMA controller which is instantiated twice on this device. As such, all functionality is identical between the two controllers with the exception of the register offsets, as noted in Table 19-4, and the external signals, as noted in Table 19-3.

19.1 Introduction

The DMA controller transfers blocks of data between the serial RapidIO controller, PCI Express, the local bus controller (LBC) interface, and the local address space, independent of the e500 core or external hosts.

19.1.1 Block Diagram

Figure 19-1 shows the block diagram of the DMA controller.

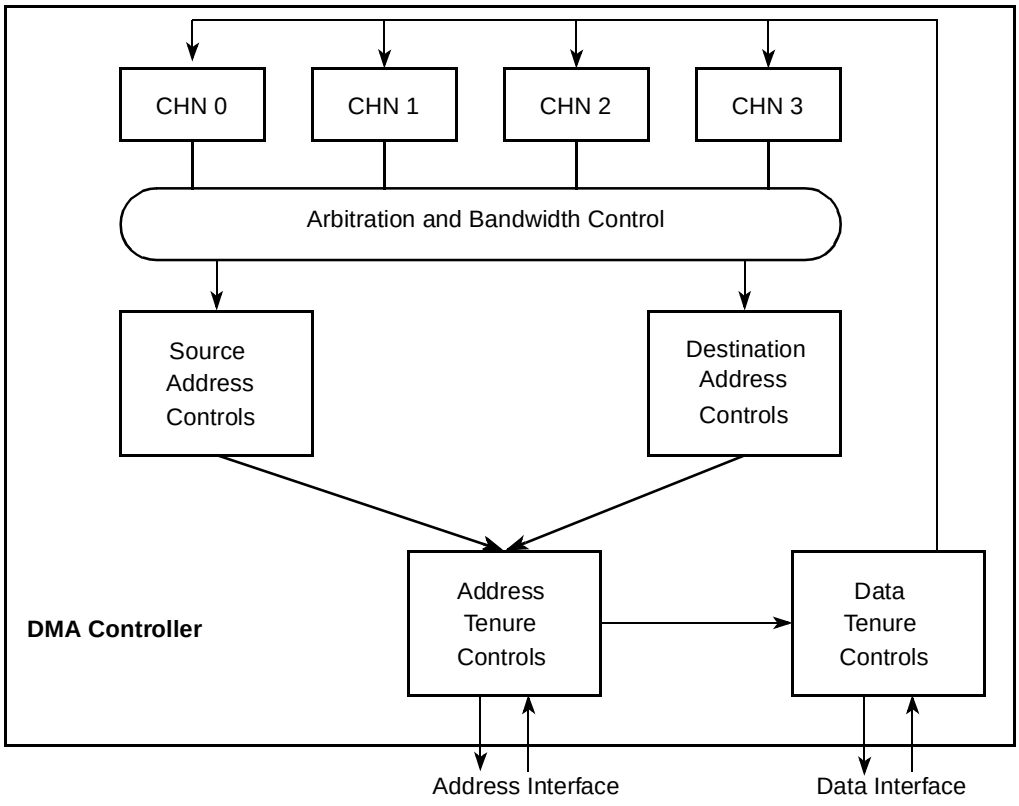


Figure 19-1. DMA Block Diagram

19.1.2 Overview

The DMA controller has four high-speed DMA channels. Both the e500 core and external devices can initiate DMA transfers. All channels are capable of complex data movement and advanced transaction chaining. Figure 19-1 is a high-level block diagram of the DMA controller. 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.

19.1.3 Features

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
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$

19.1.4 Modes of Operation

The MPC8572E has two modes of operation: basic and extended. Basic mode is the DMA legacy mode. It does not support advanced features. Extended mode supports advanced features like striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. Table 19-1 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 19-1 before starting a transfer.
- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 19-1 before starting a transfer.
- Single-write start mode. The DMA process can be started by 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 which 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 19-1. 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)

Table 19-2 describes bit settings required for each DMA mode of operation.

Table 19-2. DMA Mode Bit Settings

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[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

Table 19-2. DMA Mode Bit Settings (continued)

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
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

Refer to Section 19.4, “Functional Description,” for details on these modes.

Figure 19-2 shows the general DMA operational flow chart.

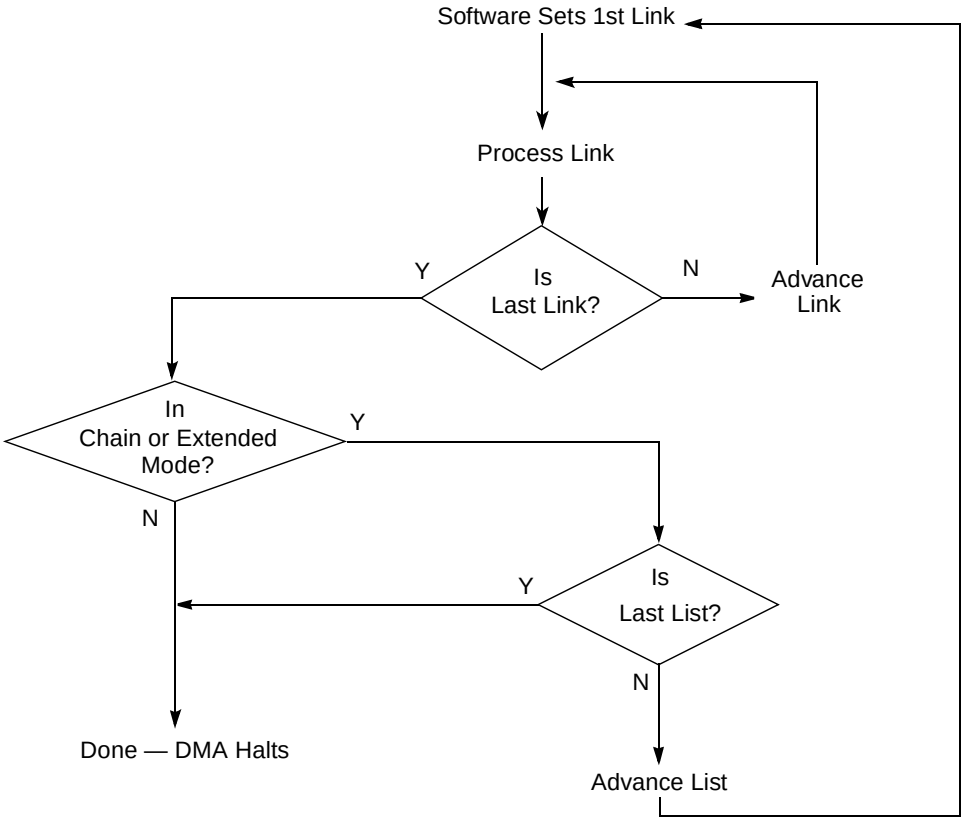


Figure 19-2. DMA Operational Flow Chart

19.2 External Signal Description

This section describes the DMA signals.

19.2.1 Signal Overview

Figure 19-3 summarizes the DMA controller signals.

Note that this device does not support external control for every DMA channel.

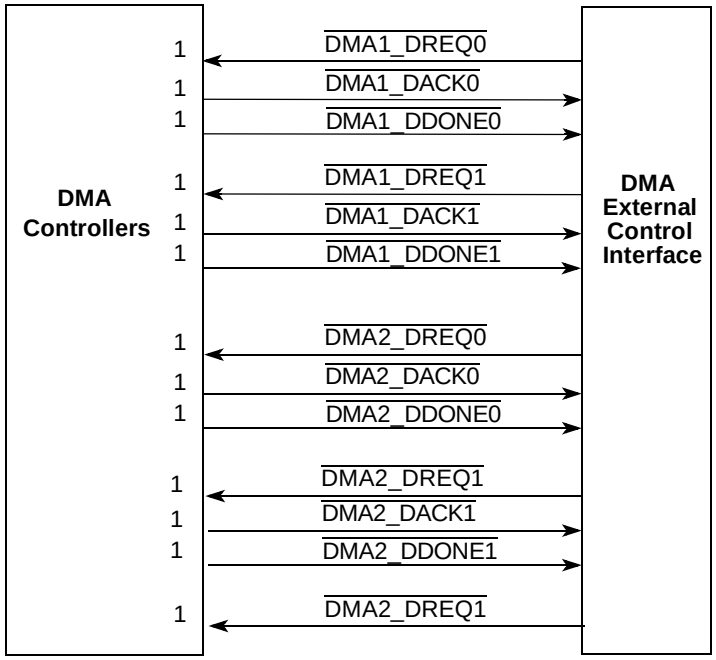


Figure 19-3. DMA Signal Summary

19.2.2 Detailed Signal Descriptions

Table 19-3 describes the DMA signals.

Table 19-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DMA\_DREQ}}n$ DMA request	I	DMA request. The DMA request signal indicates the start of a DMA transfer or a restart from a paused request. Assertion of $\overline{\text{DMA\_DREQ}}n$ causes MR*n*[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DMA\_DREQ}}n$ while $\overline{\text{DMA\_DACK}}n$ is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while $\overline{\text{DMA\_DACK}}n$ is asserted results in an illegal condition. Negated—Negation while $\overline{\text{DMA\_DACK}}n$ is asserted has no effect. Negation before the assertion of $\overline{\text{DMA\_DACK}}n$ results in an illegal condition.
		Timing	Assertion—Can be asserted asynchronously Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA\_DACK}}n$

Table 19-3. DMA Signals—Detailed Signal Description (continued)

Signal	I/O	Description	
DMA_DACK*n*	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*n* to indicate the start of a transfer Negated—Negated after finishing a complete transfer or after entering a paused state if MR*n*[EMP_EN] is set
		Timing	Assertion—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
DMA_DDONE*n*	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*n* to indicate completion of a transfer. For a paused transfer, DMA_DDONE*n* is asserted asynchronously after the negation of the final DMA_DACK*n*. Negation—Negated asynchronously after the assertion of DMA_DREQ*n* for the next transfer

19.3 Memory Map/Register Definition

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 offset 0x000–0xFFF are reserved.

Table 19-4 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 19-4.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 19-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
DMA Controller 1 Block Base Address: 0x2_1000				
0x100	MR0—DMA 0 mode register	R/W	0x0000_0000	19.3.1.1/19-9
0x104	SR0—DMA 0 status register	Mixed	0x0000_0000	19.3.1.2/19-12
0x108	ECLNDAR0—DMA 0 current link descriptor extended address register	R/W	0x0000_0000	19.3.1.3/19-13

Table 19-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x10C	CLNDAR0—DMA 0 current link descriptor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x110	SATR0—DMA 0 source attributes register	R/W	0x0000_0000	19.3.1.4/19-15
0x114	SAR0—DMA 0 source address register	R/W	0x0000_0000	19.3.1.5/19-16
0x118	DATR0—DMA 0 destination attributes register	R/W	0x0000_0000	19.3.1.6/19-17
0x11C	DAR0—DMA 0 destination address register	R/W	0x0000_0000	19.3.1.7/19-19
0x120	BCR0—DMA 0 byte count register	R/W	0x0000_0000	19.3.1.8/19-20
0x124	ENLNDAR0—DMA 0 next link descriptor extended address register	R/W	0x0000_0000	19.3.1.9/19-21
0x128	NLNDAR0—DMA 0 next link descriptor address register	R/W	0x0000_0000	19.3.1.9/19-21
0x130	ECLSDAR0—DMA 0 current list descriptor extended address register	R/W	0x0000_0000	19.3.1.10/19-22
0x134	CLSDAR0—DMA 0 current list descriptor address register	R/W	0x0000_0000	19.3.1.10/19-22
0x138	ENLSDAR0—DMA 0 next list descriptor extended address register	R/W	0x0000_0000	19.3.1.11/19-23
0x13C	NLSDAR0—DMA 0 next list descriptor address register	Mixed	0x0000_0000	19.3.1.11/19-23
0x140	SSR0—DMA 0 source stride register	R/W	0x0000_0000	19.3.1.12/19-24
0x144	DSR0—DMA 0 destination stride register	R/W	0x0000_0000	19.3.1.13/19-24
0x148–0x17C	Reserved	—	—	—
0x180	MR1—DMA 1 mode register	R/W	0x0000_0000	19.3.1.1/19-9
0x184	SR1—DMA 1 status register	Mixed	0x0000_0000	19.3.1.2/19-12
0x188	ECLNDAR1—DMA 1 current link descriptor extended address register	R/W	0x0000_0000	19.3.1.3/19-13
0x18C	CLNDAR1—DMA 1 current link descriptor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x190	SATR1—DMA 1 source attributes register	R/W	0x0000_0000	19.3.1.4/19-15
0x194	SAR1—DMA 1 source address register	R/W	0x0000_0000	19.3.1.5/19-16
0x198	DATR1—DMA 1 destination attributes register	R/W	0x0000_0000	19.3.1.6/19-17
0x19C	DAR1—DMA 1 destination address register	R/W	0x0000_0000	19.3.1.7/19-19
0x1A0	BCR1—DMA 1 byte count register	R/W	0x0000_0000	19.3.1.8/19-20
0x1A4	ENLNDAR1—DMA 1 next link descriptor extended address register	R/W	0x0000_0000	19.3.1.9/19-21
0x1A8	NLNDAR1—DMA 1 next link descriptor address register	R/W	0x0000_0000	19.3.1.9/19-21
0x1B0	ECLSDAR1—DMA 1 current list descriptor extended address register	R/W	0x0000_0000	19.3.1.10/19-22
0x1B4	CLSDAR1—DMA 1 current list descriptor address register	R/W	0x0000_0000	19.3.1.10/19-22
0x1B8	ENLSDAR1—DMA 1 next list descriptor extended address register	R/W	0x0000_0000	19.3.1.11/19-23
0x1BC	NLSDAR1—DMA 1 next list descriptor address register	R/W	0x0000_0000	19.3.1.11/19-23

Table 19-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x1C0	SSR1—DMA 1 source stride register	R/W	0x0000_0000	19.3.1.12/19-24
0x1C4	DSR1—DMA 1 destination stride register	R/W	0x0000_0000	19.3.1.13/19-24
0x1C8–0x1FC	Reserved	—	—	—
0x200	MR2—DMA 2 mode register	R/W	0x0000_0000	19.3.1.1/19-9
0x204	SR2—DMA 2 status register	Mixed	0x0000_0000	19.3.1.2/19-12
0x208	ECLNDAR2—DMA 2 current link descriptor extended address register	R/W	0x0000_0000	19.3.1.3/19-13
0x20C	CLNDAR2—DMA 2 current link descriptor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x210	SATR2—DMA 2 source attributes register	R/W	0x0000_0000	19.3.1.4/19-15
0x214	SAR2—DMA 2 source address register	R/W	0x0000_0000	19.3.1.5/19-16
0x218	DATR2—DMA 2 destination attributes register	R/W	0x0000_0000	19.3.1.6/19-17
0x21C	DAR2—DMA 2 destination address register	R/W	0x0000_0000	19.3.1.7/19-19
0x220	BCR2—DMA 2 byte count register	R/W	0x0000_0000	19.3.1.8/19-20
0x224	ENLNDAR2—DMA 2 next link descriptor extended address register	R/W	0x0000_0000	19.3.1.9/19-21
0x228	NLNDAR2—DMA 2 next link descriptor address register	R/W	0x0000_0000	19.3.1.9/19-21
0x230	ECLSDAR2—DMA 2 current list descriptor extended address register	R/W	0x0000_0000	19.3.1.10/19-22
0x234	CLSDAR2—DMA 2 current list descriptor address register	R/W	0x0000_0000	19.3.1.10/19-22
0x238	ENLSDAR2—DMA 2 next list descriptor extended address register	R/W	0x0000_0000	19.3.1.11/19-23
0x23C	NLSDAR2—DMA 2 next list descriptor address register	R/W	0x0000_0000	19.3.1.11/19-23
0x240	SSR2—DMA 2 source stride register	R/W	0x0000_0000	19.3.1.12/19-24
0x244	DSR2—DMA 2 destination stride register	R/W	0x0000_0000	19.3.1.13/19-24
0x248–0x27C	Reserved	—	—	—
0x280	MR3—DMA 3 mode register	R/W	0x0000_0000	19.3.1.1/19-9
0x284	SR3—DMA 3 status register	Mixed	0x0000_0000	19.3.1.2/19-12
0x288	ECLNDAR3—DMA 3 current link descriptor extended address register	R/W	0x0000_0000	19.3.1.3/19-13
0x28C	CLNDAR3—DMA 3 current link descriptor address register	R/W	0x0000_0000	19.3.1.3/19-13
0x290	SATR3—DMA 3 source attributes register	R/W	0x0000_0000	19.3.1.4/19-15
0x294	SAR3—DMA 3 source address register	R/W	0x0000_0000	19.3.1.5/19-16
0x298	DATR3—DMA 3 destination attributes register	R/W	0x0000_0000	19.3.1.6/19-17
0x29C	DAR3—DMA 3 destination address register	R/W	0x0000_0000	19.3.1.7/19-19
0x2A0	BCR3—DMA 3 byte count register	R/W	0x0000_0000	19.3.1.8/19-20

Table 19-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2A4	ENLNDAR3—DMA 3 next link descriptor extended address register	R/W	0x0000_0000	19.3.1.9/19-21
0x2A8	NLNDAR3—DMA 3 next link descriptor address register	R/W	0x0000_0000	19.3.1.9/19-21
0x2B0	ECLSDAR3—DMA 3 current list descriptor extended address register	R/W	0x0000_0000	19.3.1.10/19-22
0x2B4	CLSDAR3—DMA 3 current list descriptor address register	R/W	0x0000_0000	19.3.1.10/19-22
0x2B8	ENLSDAR3—DMA 3 next list descriptor extended address register	R/W	0x0000_0000	19.3.1.11/19-23
0x2BC	NLSDAR3—DMA 3 next list descriptor address register	R/W	0x0000_0000	19.3.1.11/19-23
0x2C0	SSR3—DMA 3 source stride register	R/W	0x0000_0000	19.3.1.12/19-24
0x2C4	DSR3—DMA 3 destination stride register	R/W	0x0000_0000	19.3.1.13/19-24
0x2C8–0x2FC	Reserved	—	—	—
0x300	DGSR—DMA general status register	R	0x0000_0000	19.3.1.14/19-25
DMA Controller 2 Block Base Address: 0x0_C000				
0x000–0xFFC	DMA 2 registers **Note:** All registers defined for DMA controller 1 are also defined for DMA controller 2; the offsets of DMA controller 2 registers are the same except they have a different block base address.			

19.3.1 DMA Register Descriptions

The following sections describe the DMA registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register.

19.3.1.1 Mode Registers (MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. Figure 19-4 describes the MR*n*.

Offset 0x100, 0x180, 0x200, 0x280 — Access: Read/Write

	0–3	4–7	8–9	10	11–12	13	14	15
R W	—	BWC	—	EMP_EN	—	EMS_EN	DAHTS	
Reset	All zeros							

	16–17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SAHTS	DAHE	SAHE	—	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM/SWSM	CA	CTM	CC	CS
Reset	All zeros														

Figure 19-4. DMA Mode Registers (MR*n*)

Table 19-5 describes the MR*n* fields.

Table 19-5. MR*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. If multiple channels are executing transfers concurrently the value of MR*n*[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and switches to the next channel. If only one channel is executing transfers the value of MR*n*[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of $\overline{\text{DREQ}}$ resumes channel operation. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 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.
8–9	—	Reserved
10	EMP_EN	External master pause enable. Valid only if MR*n*[EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MR*n*[BWC].
11–12	—	Reserved
13	EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 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 MR*n*[CS].
14–15	DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[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 MR*n*[DAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17	SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[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 MR*n*[SAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
18	DAHE	Destination address hold enable 0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MR*n*[DAHTS]. Hardware only supports aligned transfers for this feature.

Table 19-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
19	SAHE	Source address hold enable 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.
20	—	Reserved
21	SRW	Single register write (Direct mode only; reserved for chaining mode.) 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.
22	EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDAR*n*[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 CLNDAR*n*[EOSIE].
23	EOLNIE	End-of-links interrupt enable 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, NLNDAR*n*[EOLND] is set).
24	EOLSIE	End-of-lists interrupt enable 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, NLNDAR*n*[EOLND] and NLSDAR*n*[EOLSD] are set).
25	EIE	Error interrupt enable 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.
26	XFE	Extended features enable 0 Disable the new chaining features. 1 Enable the new chaining features.
27	CDSM/ SWSM	• In chaining mode: current descriptor start mode/single-write start mode — 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. • 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.
28	CA	Channel abort 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.
29	CTM	Channel transfer mode 0 Configure the channel in chaining mode. 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.

Table 19-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
30	CC	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. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31	CS	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 $\overline{\text{DMA\_DREQ}}$ does NOT clear this bit. 0 Halt the DMA process if channel is busy (SR*n*[CB] is set). No effect if the channel is not busy. 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.

19.3.1.2 Status Registers (SR*n*)

The status registers, shown in Figure 19-5, report various DMA conditions during and after a DMA transfer.

Offset 0x104, 0x184, 0x204, 0x284 — Access: w1c

	0–23	24	25	26	27	28	29	30	31
R	—	TE	—	CH	PE	EOLNI	CB	EOSI	EOLSI
W		w1c			w1c	w1c		w1c	w1c
Reset	All zeros								

Figure 19-5. Status Registers (SR*n*)

Table 19-6 describes the bits of the SR*n*.

Table 19-6. SR*n* Field Descriptions

Bits	Name	Description
0–23	—	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 Section 19.4.3, “DMA Errors,” for additional information.
25	—	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 (SR*n*[CB] is cleared), this bit will remain 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 MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

Table 19-6. SR*n* Field Descriptions (continued)

Bits	Name	Description
29	CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 A DMA transfer is currently in progress.
30	EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MR*n*[EOSIE] is set or if CLNDAR*n*[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 MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

19.3.1.3 Current Link Descriptor Address Registers (CLNDAR*n* and ECLNDAR*n*)

Current link descriptor address registers contain the address of the current link descriptor. In basic chaining mode, shown in Figure 19-6, software must initialize these registers to point to the first link descriptors in memory.

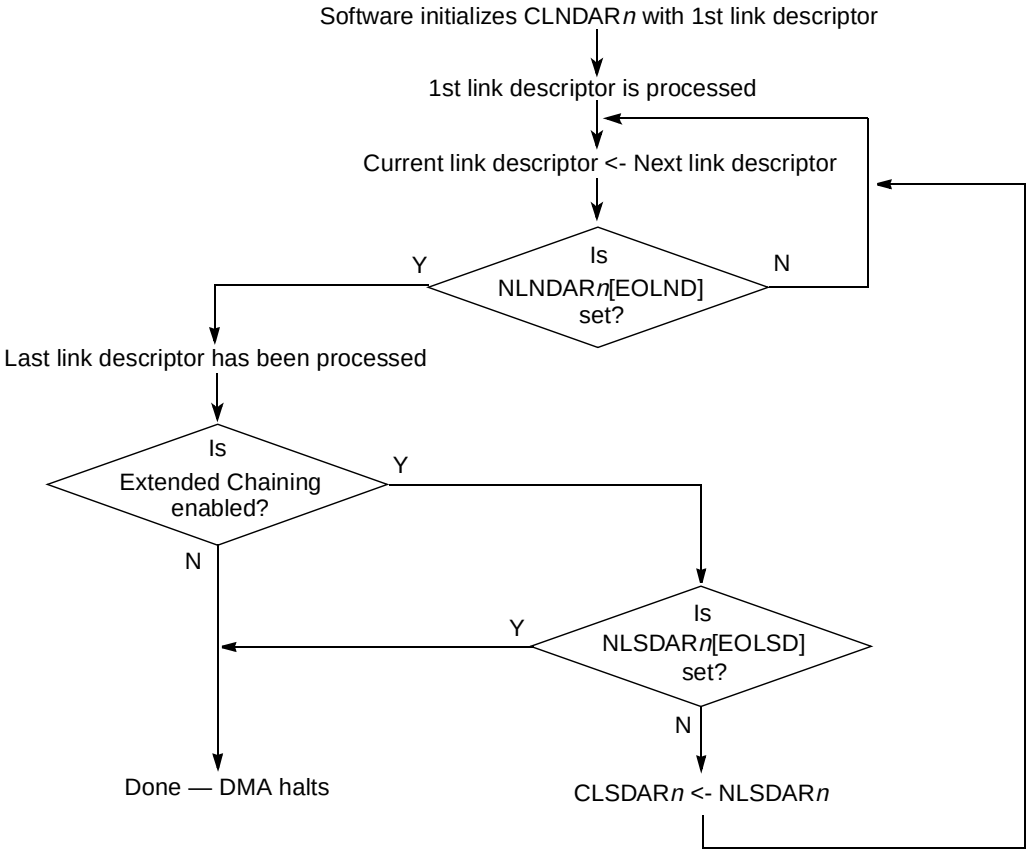


Figure 19-6. Basic Chaining Mode Flow Chart

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 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 NLSDAR*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.

Figure 19-7 shows ECLNDAR*n*.

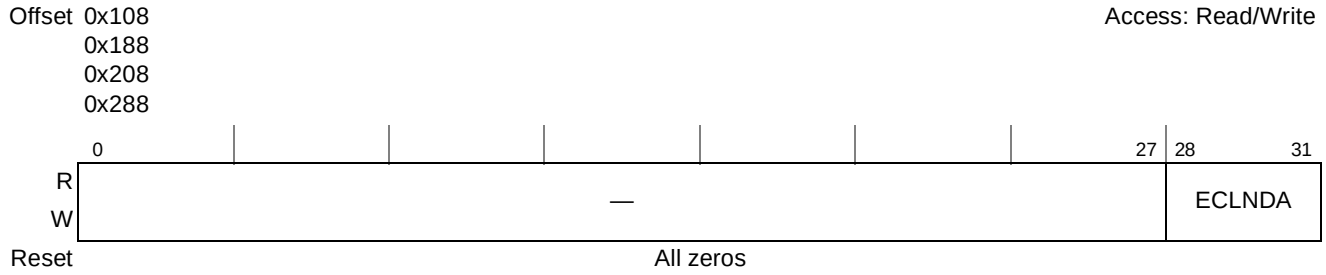


Figure 19-7. Extended Current Link Descriptor Address Registers (ECLNDAR*n*)

Table 19-7. ECLNDAR*n* Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	ECLNDA	Current link descriptor extended address (upper 4 bits of 36-bit address)

Figure 19-8 shows CLNDAR*n*.

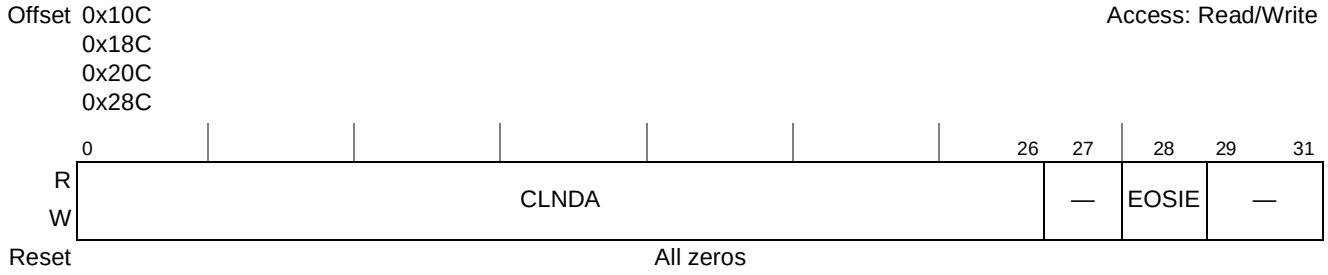


Figure 19-8. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 19-8 describes the fields of the CLNDAR*n*.

Table 19-8. CLNDAR*n* Field Descriptions

Bits	Name	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 CLNDAR*n*[CLNDA] and ECLNDAR*n*[ECLNDA].)
27	—	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	—	Reserved

19.3.1.4 Source Attributes Registers (SATR*n*)

The source attributes registers, shown in Figure 19-9, contain the transaction attributes to be used for the DMA operation. Stride mode is enabled by setting SATR*n*[SSME].

If SATR*n*[SBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in SATR*n*[SREADTTYPE] using the local address space category.

Offset 0x110, 0x190, 0x210, 0x290 — Access: Read/Write

	0–3	4–5	6	7	8–11	12–15	16–21	22–31
R W	—	STFLOWLVL	SPCIORDER	SSME	STRANSINT	SREADTTYPE	—	ESAD
Reset	All zeros							

Figure 19-9. Source Attributes Registers (SATR*n*)

Table 19-9 describes the fields of the SATR*n*.

Table 19-9. SATR*n* Field Descriptions

Bits	Name	Description
0–12	—	Reserved
3	—	Reserved
4–5	STFLOWLVL	RapidIO transaction flow level 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority level transaction flow 11 Reserved Applicable only to RapidIO interface, while SATR*n*[SBPATMU] is set.
6	SPCIORDER	Follow PCI transaction ordering rules (elevate write priority one level over reads). Applicable only while SATR*n*[SBPATMU] is set.

Table 19-9. SATR*n* Field Descriptions (continued)

Bits	Name	Description
7	SSME	Source stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the source address can be accomplished by enabling SATR*n*[SSME] and setting the desired stride size and distance in the SSR*n*.
8–11	STRANSINT	DMA source transaction interface 0000 PCI Express interface 1 0001 PCI Express interface 2 0010 PCI Express interface 3 0011 Reserved 1100 RapidIO interface 1101 Reserved 1110 TLU1 1111 TLU2 Applicable only to RapidIO interface, while SATR*n*[SBPATMU] is set.
12–15	SREADTTYPE	DMA source transaction type. Reserved values will result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space 0000–0001 Reserved 0011 Reserved 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0111 Read, unlock L2 cache line 1000–1111 Reserved
16–21	—	Reserved
22–31	ESAD	Extended source address. ESAD[6–9] represents the four high-order bits of the 36-bit source address. When used for RapidIO interface, ESAD[0–7] represents the target ID and ESAD[8–9] represent the two high-order bits of the local device offset over the RapidIO interface.

19.3.1.5 Source Address Registers (SAR*n*)

The source address registers, shown in Figure 19-10, contain the address from which the DMA controller reads data. In direct mode, if MR*n*[CDSM/SWSM] and MR*n*[SRW] are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

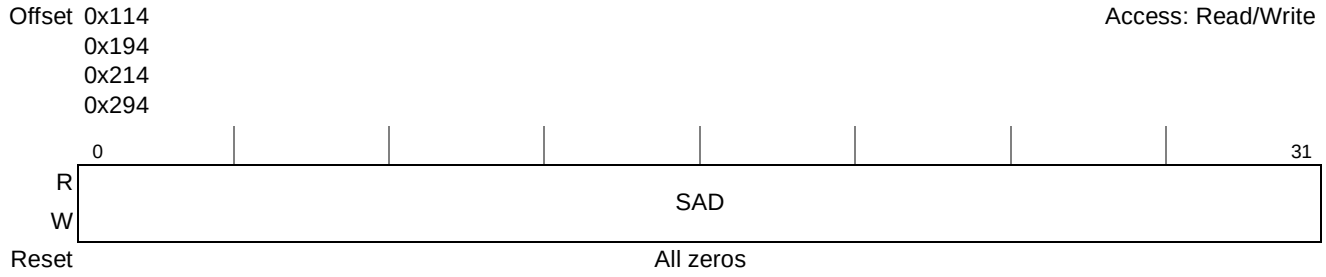


Figure 19-10. Source Address Registers (SAR*n*)

Table 19-10 describes the field of the SAR*n*.

Table 19-10. SAR*n* Field Descriptions

Bits	Name	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.

19.3.1.5.1 Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

If RapidIO is the source of a transaction, the SAR*n* registers are redefined as shown below. Several options exist for the packet type that can be specified. A maintenance read is one of many possible reads. Maintenance packets have an offset instead of an address. Figure 19-11 describes the SAR*n*.

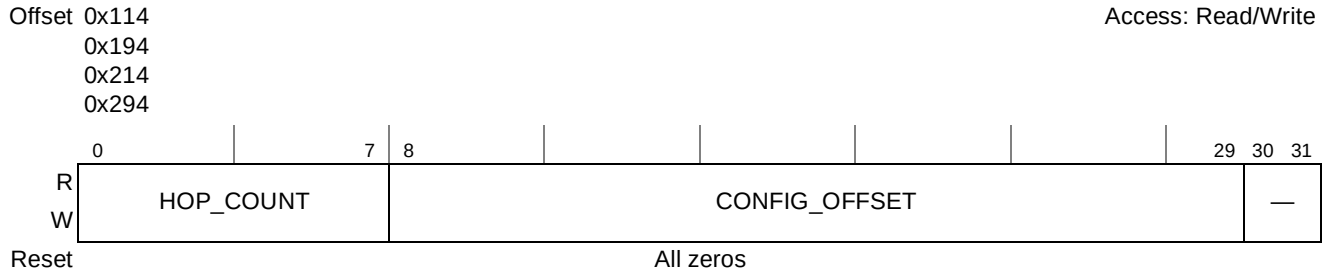


Figure 19-11. Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

Table 19-11 describes the fields of the SAR*n*.

Table 19-11. SAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop_count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

19.3.1.6 Destination Attributes Registers (DATR*n*)

The destination attributes registers, shown in Figure 19-12, contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATR*n*[DSME].

If DATR*n*[DBPATMU] is cleared, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DATR*n*[DWRITETTYPE] using the local address space category.

Offset 0x118 Access: Read/Write
0x198
0x218
0x298

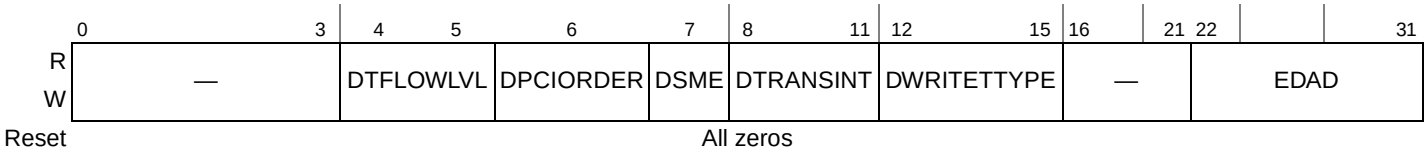


Figure 19-12. Destination Attributes Registers (DATR*n*)

Table 19-12 describes the fields of the DATR*n*.

Table 19-12. DATR*n* Field Descriptions

Bits	Name	Description
0–13	—	Reserved
4–5	DTFLOWLVL	RapidIO transaction flow level 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved Applicable only to RapidIO interface, while DATR*n*[DBPATMU] is set.
6	DPCI_ORDER	PCI ordering rules enable. Applicable only while DATR*n*[DBPATMU] is set. 0 Retain original transaction ordering 1 Follow PCI transaction ordering rules on RapidIO (elevate write priority one level over reads).
7	DSME	Destination stride mode enable 0 Stride mode disabled 1 Stride mode enabled Ignored in basic mode (MR*n*[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSR*n*.
8–11	DTRANSINT	DMA destination transaction interface. Applicable only while DATR*n*[DBPATMU] is set. 0000 PCI Express interface 1 0001 PCI Express interface 2 0010 PCI Express interface 3 0011 Reserved 0100 Reserved 1100 Serial RapidIO interface 1101–1110Reserved 1111 Local address space
12–15	DWRITETTYPE	DMA destination transaction type. Reserved values will result in a programming error being detected and logged in SR[PE].Transaction type to run on local address space 0000–0011 Reserved 0100 Write, don’t snoop local processor 0101 Write, snoop local processor 0110 Write, allocate L2 cache line 0111 Write, allocate and lock L2 cache line 1000–1111 Reserved

Table 19-12. DATR*n* Field Descriptions (continued)

Bits	Name	Description
16–21	—	Reserved
22–31	EDAD	Extended destination address. EDAD[6–9] represents the four high-order bits of the 36-bit destination address. When used for RapidIO interface, EDAD[0–7] represents the target ID and EDAD[8–9] is defined as follows, subject to the transaction type: Message: EDAD[8–9] represents the value for the mbox field in the message packet. Other: EDAD[8–9] represents the two high-order bits of the local device offset.

19.3.1.7 Destination Address Registers (DAR*n*)

The destination address registers, shown in Figure 19-13, contain the addresses to which the DMA controller writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM/SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

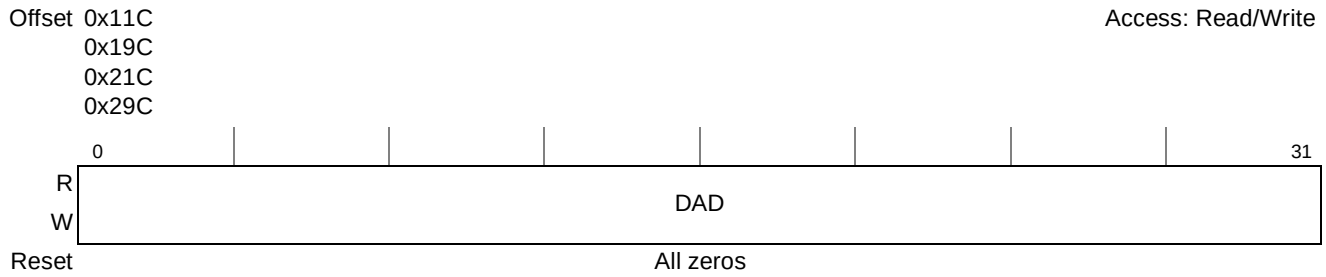


Figure 19-13. Destination Address Registers (DAR*n*)

Table 19-13 describes the field of the DAR*n*.

Table 19-13. DAR*n* Field Descriptions

Bits	Name	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.

19.3.1.7.1 Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

If RapidIO is the destination of a transaction, the DAR*n* registers are redefined as shown below. Several options exist for the transaction type that can be specified. There are a number of noncoherent write and flush types for address-based write transactions, and message types for port-based write transactions.

Maintenance packets have an offset instead of an address. Figure 19-14 shows the DAR*n*.

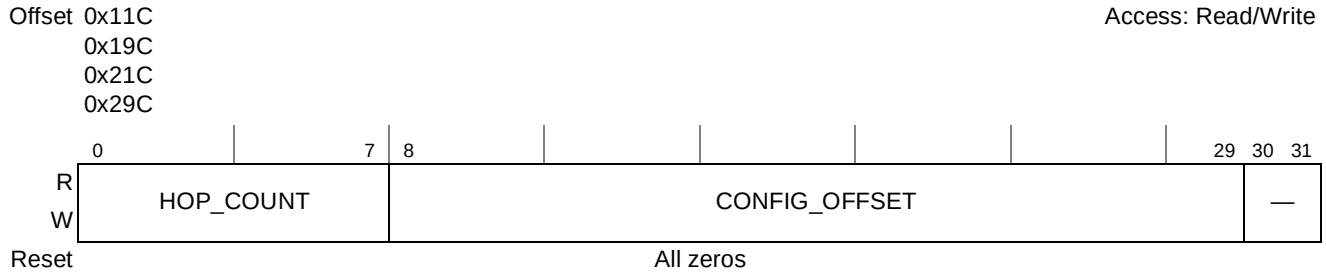


Figure 19-14. Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

Table 19-14 describes the fields of the DAR*n*.

Table 19-14. DAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

19.3.1.8 Byte Count Registers (BCR*n*)

The byte count register, shown in Figure 19-15, contains the number of bytes to transfer.

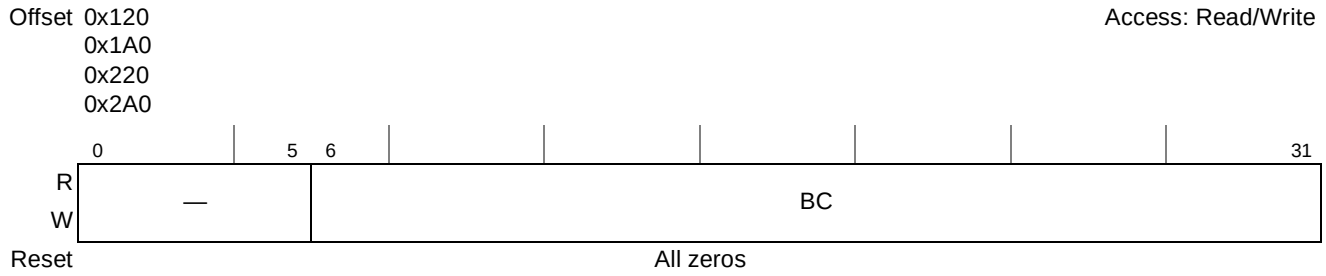


Figure 19-15. Byte Count Registers (BCR*n*)

Table 19-15 describes the fields of the BCR*n*.

Table 19-15. BCR*n* Field Descriptions

Bits	Name	Description
0–5	—	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.

19.3.1.9 Next Link Descriptor Address Registers (NLNDAR*n* and ENLNDAR*n*)

The next link descriptor address registers, shown in Figure 19-16 and Figure 19-17, contain the address for the next link descriptor in memory. Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

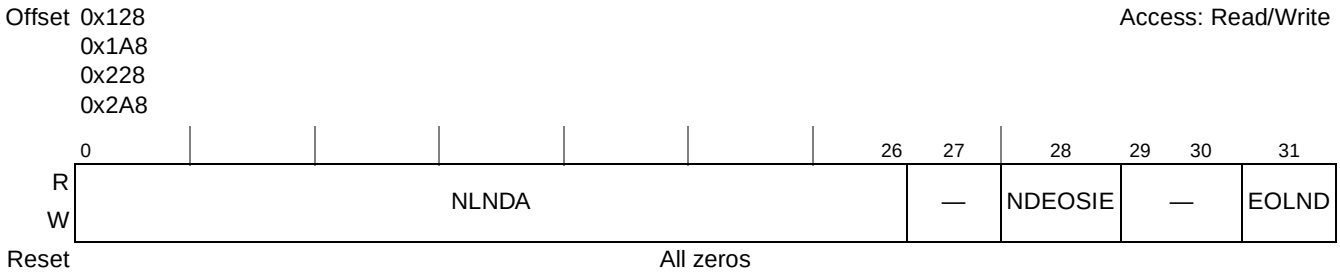


Figure 19-16. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 19-16 describes the fields of the NLNDAR*n* registers.

Table 19-16. NLNDAR*n* Field Descriptions

Bits	Name	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.
27	—	Reserved
28	NDEOSIE	Next descriptor end-of-segment interrupt enable 0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished. 1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.
29–30	—	Reserved
31	EOLND	End-of-links descriptor. This bit is ignored in direct mode. 0 This descriptor is not the last link descriptor in memory for this list. 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 NLSDAR*n*[EOLSD] is also set in extended mode.

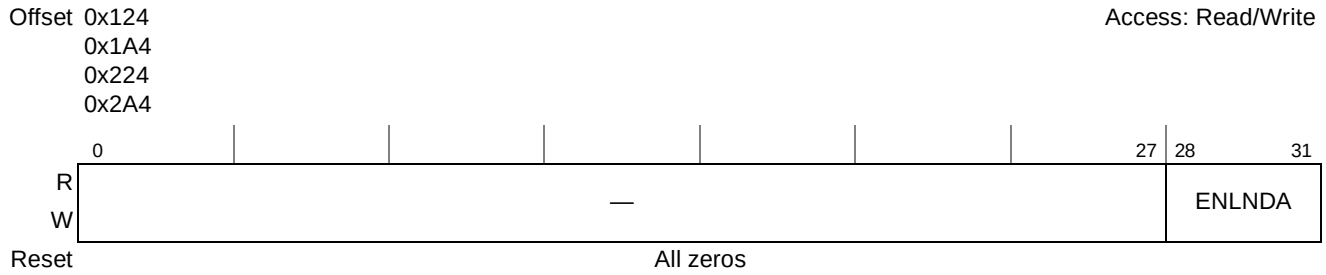


Figure 19-17. Extended Next Link Descriptor Address Registers (ENLNDAR*n*)

Table 19-17 describes the fields of the ENLNDAR*n* registers.

Table 19-17. ENLNDAR*n* Field Descriptions

Bit	Name	Description
0–27	—	Reserved
28–31	ENLNDA	Next link descriptor extended address bits (upper 4 bits of 36-bit address)

19.3.1.10 Current List Descriptor Address Registers (CLSDAR*n* and ECLSDAR*n*)

The current list descriptor address registers, shown in Figure 19-19 and Table 19-19, contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*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.

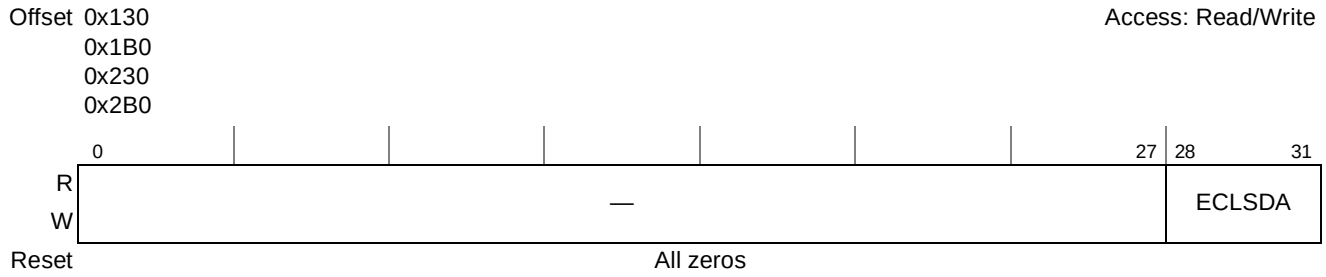


Figure 19-18. Extended Current List Descriptor Address Registers (ECLSDAR*n*)

Table 19-18. ECLSDAR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ECLSDA	Current list descriptor extended address bits (upper 4 bits of 36-bit address)

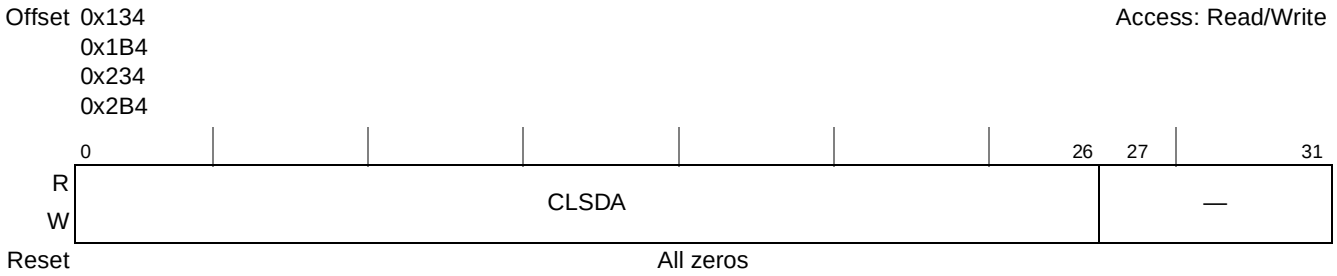


Figure 19-19. Current List Descriptor Address Registers (CLSDAR*n*)

Table 19-19 describes the fields of the CLSDAR*n*.

Table 19-19. CLSDAR*n* Field Descriptions

Bits	Name	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	—	Reserved

19.3.1.11 Next List Descriptor Address Registers (NLSDAR*n* and ENLSDAR*n*)

The next list descriptor address registers, shown in Figure 19-20 and Figure 19-21, contain the address for the next list descriptor in memory. If the contents are transferred to the current list descriptor address register they become effective for the current transfer in extended chaining mode.

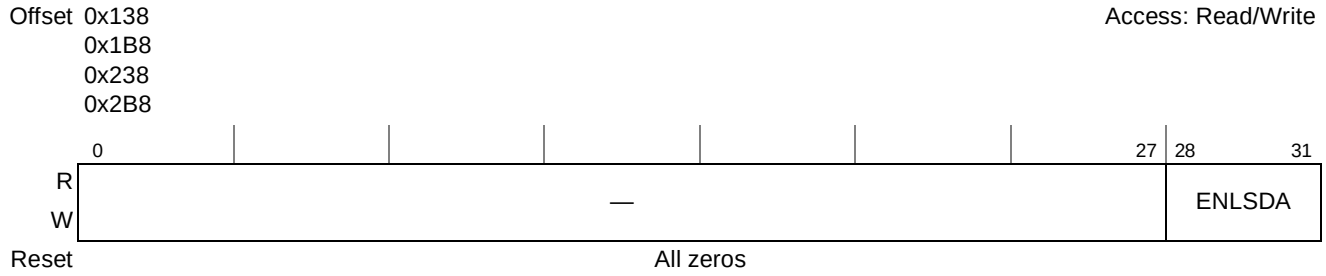


Figure 19-20. Extended Next List Descriptor Address Registers (ENLSDAR*n*)

Table 19-20. ENLSDAR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	ENLSDA	Next list descriptor extended address bits (upper 4 bits of 36-bit address)

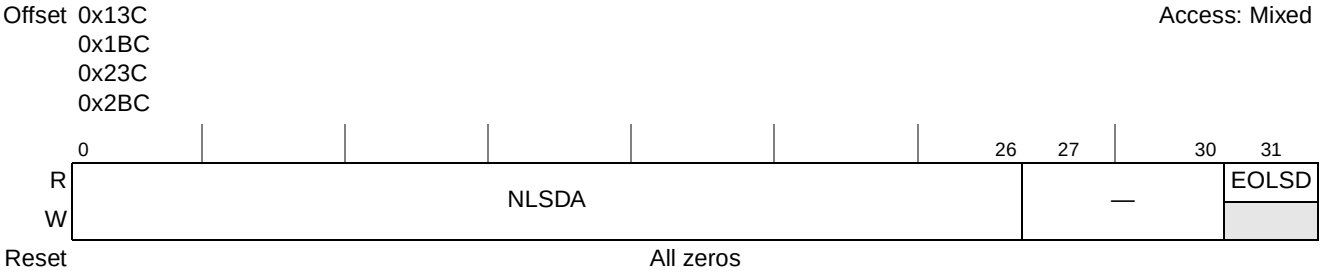


Figure 19-21. Next List Descriptor Address Registers (NLSDAR*n*)

Table 19-21 describes the fields of the NLSDAR*n*.

Table 19-21. NLSDAR*n* Field Descriptions

Bits	Name	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	—	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.

19.3.1.12 Source Stride Registers (SSR*n*)

The source stride register, shown in Figure 19-22, 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.

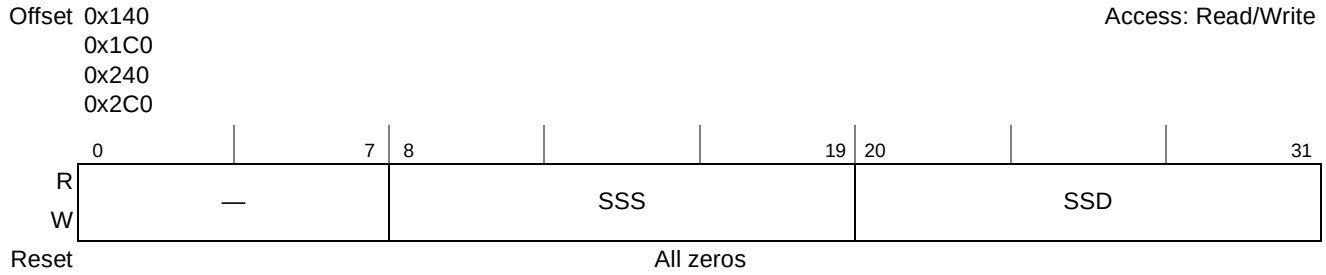


Figure 19-22. Source Stride Registers (SSR*n*)

Table 19-22 describes the fields of the SSR*n*.

Table 19-22. SSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
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.

19.3.1.13 Destination Stride Registers (DSR*n*)

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. Figure 19-23 describes the DSR*n*.

Offset 0x144
0x1C4
0x244
0x2C4

Access: Read/Write

	0	7	8	19	20	31
R	—		DSS		DSD	
W						
Reset	All zeros					

Figure 19-23. Destination Stride Registers (DSR*n*)

Table 19-23 describes the fields of the DSR*n*.

Table 19-23. DSR*n* Field Descriptions

Bits	Name	Description
0–7	—	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.

19.3.1.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register. This register is read-only. Figure 19-24 describes the DGSR.

Offset 0x300

Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE0	—	CH0	PE0	EOLNI0	CB0	EOSI0	EOLSI0	TE1	—	CH1	PE1	EOLNI1	CB1	EOSI1	EOLSI1
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TE2	—	CH2	PE2	EOLNI2	CB2	EOSI2	EOLSI2	TE3	—	CH3	PE3	EOLNI3	CB3	EOSI3	EOLSI3
W																
Reset	All zeros															

Figure 19-24. DMA General Status Register (DGSR)

Table 19-24 describes the fields of the DGSR.

Table 19-24. DGSR Field Descriptions

Bits	Name	Description
0	TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1	—	Reserved
2	CH0	Channel halted, channel 0

Table 19-24. DGSR Field Descriptions (continued)

Bits	Name	Description
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
8	TE1	Transfer error, channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9	—	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	—	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	—	Reserved
26	CH3	Channel halted, channel 3
27	PE3	Programming error, channel 3
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

19.4 Functional Description

This section describes the function of the DMA controller.

19.4.1 DMA Channel Operation

All DMA channels support two different modes of operation: a basic mode (MR*n*[XFE] is cleared) and an extended mode (MR*n*[XFE] is set). In both modes, a channel can be activated by clearing and setting MR*n*[CS], or through the single-write start mode using MR*n*[CDSM/SWSM] and MR*n*[SRW], or through an external control mode using MR*n*[ECS_EN].

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 align the source and destination addresses to a 64-byte boundary. 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 controller is designed to support RapidIO transaction types, including various priority level support. The DMA controller offers additional features from previous generations in which the reads can be mapped to globally coherent (IO_READ), non-coherent (NREAD), or maintenance reads. In addition, the writes can be mapped to coherent (flush with data) and non-coherent (NWRITE, NWRITE_R) writes, messages, and maintenance writes.

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 19-25 for more complete descriptions of the channel states and state transitions.

19.4.1.1 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.

19.4.1.1.1 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 19-25), 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.

19.4.1.1.2 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 19-25), 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, 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.

19.4.1.1.3 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 19-25), 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, 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, 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.

19.4.1.1.4 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 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 19-25), 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, 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.

19.4.1.2 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 MR*n*[XFE].

19.4.1.2.1 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 MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

19.4.1.2.2 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 MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

19.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then CLSDAR*n* and ECLSDAR*n* 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 19-25), 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, 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, 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.

19.4.1.2.4 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], MR*n*[CTM], and MR*n*[XFE] 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 19-25), 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.

19.4.1.3 External Control Mode Transfer

An external control can be used to control all DMA channels by setting MR*n*[EMS_EN]. The external control can control the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

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 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.

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 $\overline{\text{DREQ}}$ 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.

The following signals are defined for the external control interface:

- $\overline{\text{DMA\_DREQ}}$ - Asserting edge triggers a DMA transfer start or restart from a pause request. Sets MR*n*[CS]. (Note that negating $\overline{\text{DMA\_DREQ}}$ does NOT clear MR*n*[CS].)
- $\overline{\text{DMA\_DACK}}$ - Indicates a DMA transfer currently in progress. SR*n*[CB] is set.
- $\overline{\text{DMA\_DDONE}}$ - 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 19-3. The timing diagram of the external control interface is shown in Figure 19-25.

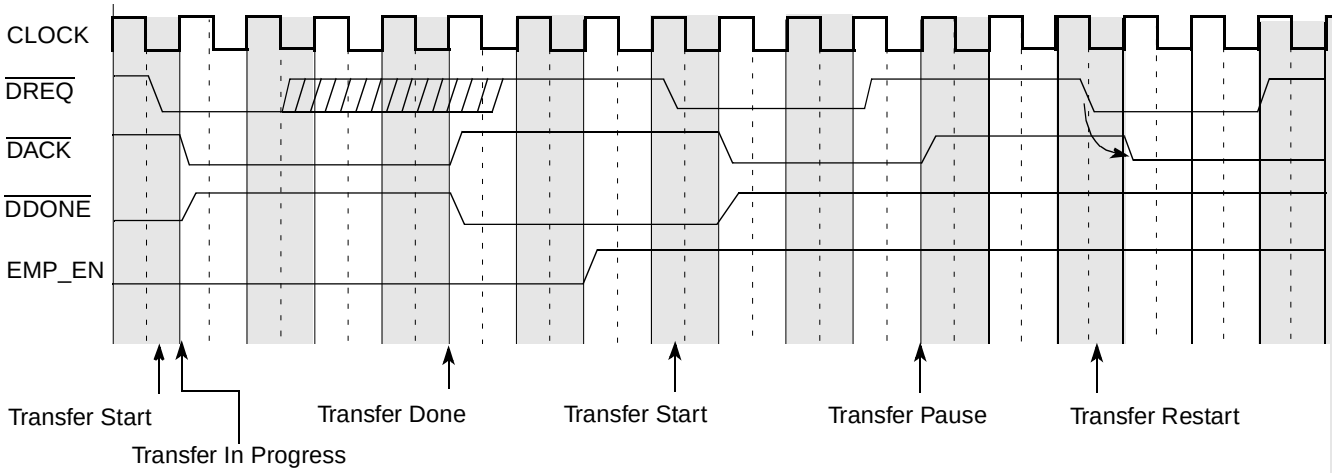


Figure 19-25. External Control Interface Timing

19.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode (enabled when MR*n*[CC] is set) offers software the flexibility of having the DMA controller get started 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 (SR*n*[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.

19.4.1.4.1 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.

19.4.1.4.2 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.

19.4.1.5 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.

19.4.1.6 Bandwidth Control

MR*n*[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.

19.4.1.7 Channel State

Table 19-25 defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 19-25. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[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 MR*n*[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

19.4.1.8 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. Figure 19-26 illustrates the stride size and distance parameters. 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.

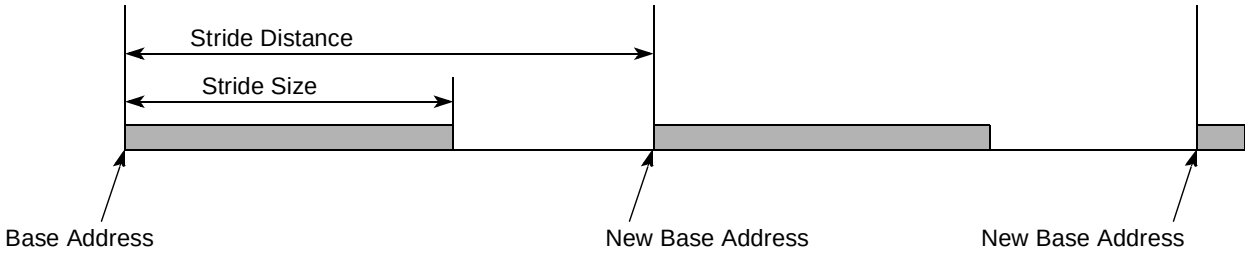


Figure 19-26. Stride Size and Stride Distance

19.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

19.4.3 DMA Errors

On a transfer error (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, address mapping errors, for example), 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
- Transfer started with a priority of three
- Illegal type, defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE], used for the transfer.

19.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link 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.

Table 19-26 summarizes the DMA list descriptors.

Table 19-26. 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.

Table 19-26. List DMA Descriptor Summary (continued)

Descriptor Field	Description
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

Table 19-27 summarizes the DMA link descriptors.

Table 19-27. 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
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.

Figure 19-27 describes the DMA transaction flow.

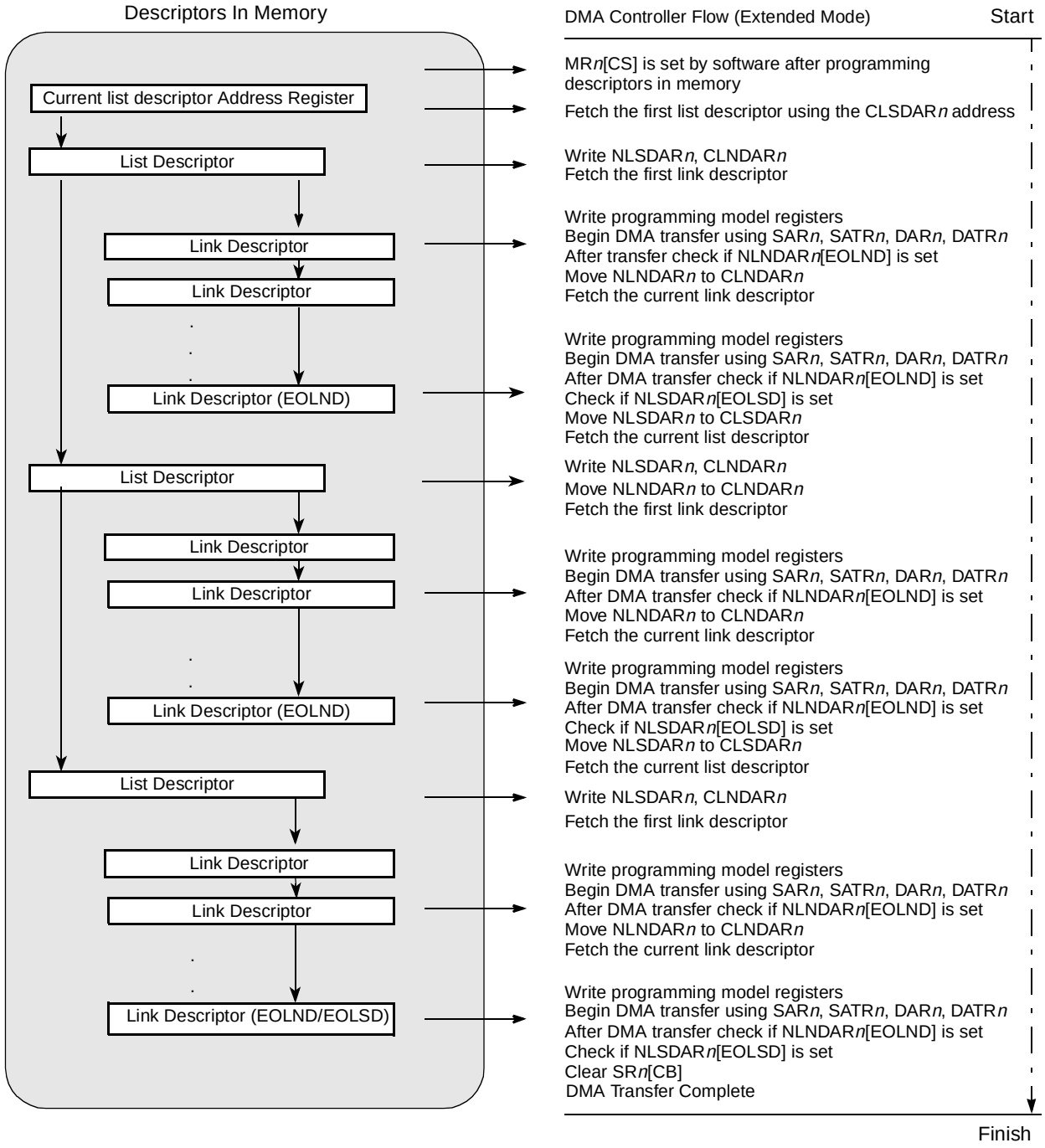


Figure 19-27. DMA Transaction Flow with DMA Descriptors

Figure 19-28 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 19-28. List Descriptor Format

Figure 19-29 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 19-29. Link Descriptor Format

19.4.5 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the DMA controller and 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, stride sizes less than 64 bytes should be avoided. 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:

- Setting the source or destination priority level (STFLOWLVL or DTFLOWLVL) to a value of three (0b11) is considered a programming error.
- 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.
- If the DMA is programmed to send SWRITEs over RapidIO, the programmer must ensure that the destination address is double-word aligned and that the byte count is a double-word multiple.
- If the DMA is programmed to send messages over RapidIO, the programmer must ensure that the message length (BCR*n*[BC]) is 8, 16, 32, 64, 128 or 256 bytes. This can be achieved by setting the byte count register (BCR) to a power of 2 value equal to 8 or greater.
- Striding does not work if the destination transaction type is MESSAGE (DATR[DWRITETTYPE] = 0x0110) because messages have no memory addresses. As well, destination address hold should be disabled (MR*n*[DAHE] is cleared) unless the destination address hold transfer size indicates an 8-byte message (MR*n*[DAHTS] = 0x11). Software is responsible for disabling striding and DAHE, in this case, and for ensuring that the bandwidth control is large enough to support the desired message size. Failure to adhere to these restrictions results in boundedly undefined behavior.

19.5 DMA System Considerations

This section provides information about how to make most effective use of the DMA channels.

Figure 19-30 shows the most important data paths within the device. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller.

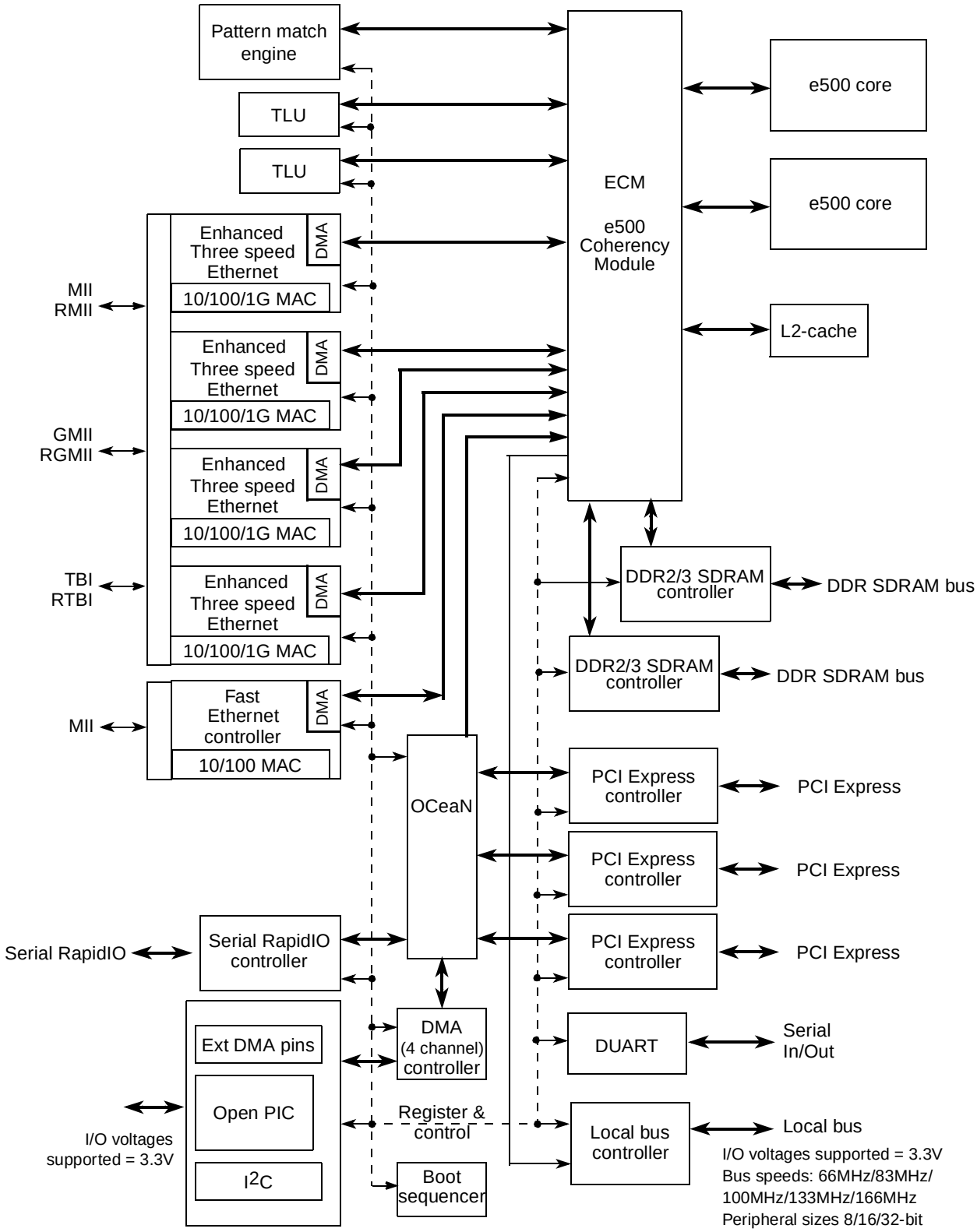


Figure 19-30. DMA Data Paths

Note: On-chip target configuration registers include I^2C data register.

Note: On-chip Ethernet captive resource. Not available to external masters.

Note: On-chip 4-channel controller can serve external masters.

19.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks 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.

19.5.1.1 DMA to e500 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.

19.5.1.2 DMA to Ethernet

The Ethernet controllers cannot serve as DMA targets because they have no suitable internal memory for this purpose. The Ethernet controllers have dedicated DMA channels to move data between the external transmit and receive buffers and the internal packet buffer. This dedicated channel is the only DMA service to the internal packet buffers.

However, Ethernet ports can serve as DMA targets by using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

19.5.1.3 DMA to Configuration, Control, and Status Registers

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 I^2C 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.

19.5.1.4 DMA to I²C

The I²C controller is not transparent to DMA transfers. Observe the caveats listed in Section 19.5.1.3, "DMA to Configuration, Control, and Status Registers," when accessing any I²C register, including the data register (I2CDR).

19.5.1.5 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in Chapter 13, "DUART," specifically, Section 13.4.5, "FIFO Mode."

Chapter 20
Serial RapidIO Interface

The serial RapidIO controller consists of a RapidIO endpoint and the RapidIO messaging unit (RMU). Both portions are compliant with the *RapidIO Interconnect Specification, Revision 1.2*.

20.1 Overview

The serial RapidIO interface provides a RapidIO port and message unit to communicate with other RapidIO devices. Figure 20-1 shows the RapidIO port and message unit as they interface with OCeaN and with each other.

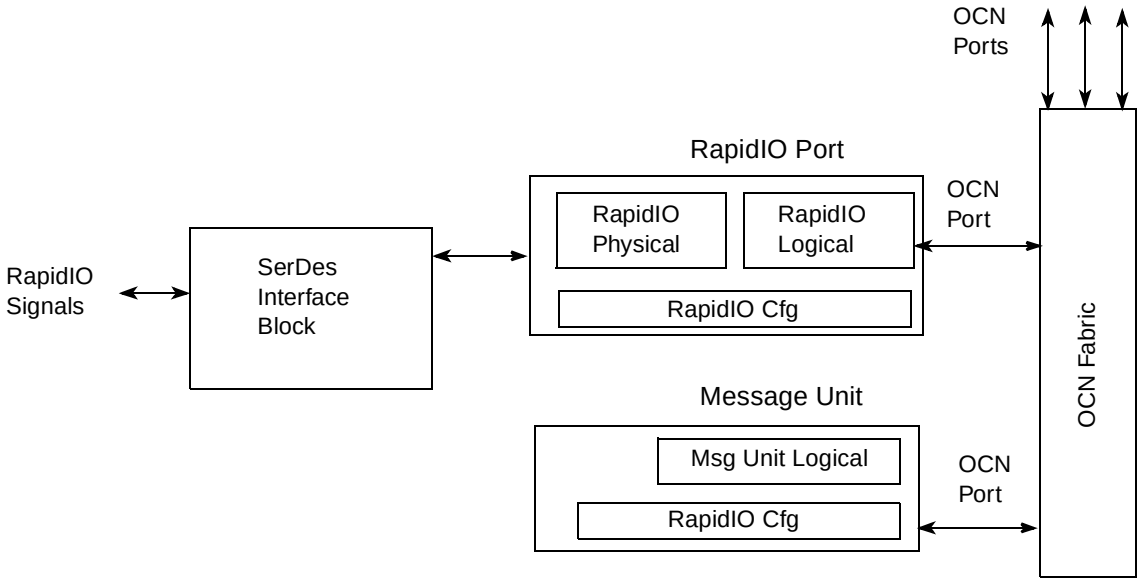


Figure 20-1. RapidIO Endpoint and RMU

20.2 Features

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

- Critical request flow (CRF) as described in the *RapidIO Interconnect Specification, Revision 1.3*, Part 6: 1x/rx LP-Serial Physical Layer Specification.
- 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 w/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 controllers
- 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 the default window
- 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 w/data transactions
- No support for transmitter-controlled flow control

- No decrementing of a maintenance packet hop count (pass-through support does not imply switch functionality)
- No support for multicast event control symbols

RapidIO endpoint supports the following features of RapidIO 1x/4x LP-Serial:

- Both 1x and 4x LP-Serial link interfaces
- Transmission rates of 1.25, 2.5, and 3.125 Gbaud (data rates of 1.0, 2.0, and 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 1x/4x LP-Serial:

- RapidIO endpoint cannot be configured as four 1x ports

20.3 Modes of Operation

20.3.1 RapidIO Port

The RapidIO port's primary operating modes are the following:

- 1x or 4x LP-Serial link interfaces
- Transmission rates of 1.25, 2.5, or 3.125 Gbaud (data rates of 1.0, 2.0, and 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

20.3.2 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.

20.4 1x/4x 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 Section 4.4.3.6, "I/O Port Selection," for specific configuration details.

See Section 4.4.4.3, "SGMII Clocks," for proper selection of CCB/platform clock frequency with regard to serial RapidIO link width and frequency selection.

The following sections describe the serial RapidIO signal functionality. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x LP-Serial Specifications, Chapter 8, Electrical Specifications,* for electrical characteristic details.

20.4.1 Serial Rapid I/O Interface Overview

The serial Rapid I/O (SRIO) interface is compatible with the *RapidIO Interconnect Specification, Revision 1.2, Part VI: Physical Layer 1x/4x LP-Serial Specifications.*

20.4.2 Serial Rapid I/O Interface Detailed Signal Descriptions

20.4.2.1 SD1_TX[4:7]/$\overline{\text{SD1\_TX}}$[4:7]—Outputs

These are the serial data outputs. They are differential pairs, one for each lane in 4x mode of operation. In the case of 1x mode of operation on a 1x/4x-compatible serial RapidIO interface, only pairs SD1_TX[4]/$\overline{\text{SD1\_TX}}$[4] and/or SD1_TX[6]/$\overline{\text{SD1\_TX}}$[6] may be used. 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 1.25, 2.5, and 3.125 Gbaud.

20.4.2.2 SD1_RX[4:7]/$\overline{\text{SD1\_RX}}$[4:7]—Inputs

These are the serial data input pads. They are differential pairs, one for each lane in 4x mode of operation. In the case of 1x mode of operation on a 1x/4x-compatible serial RapidIO interface, only pair SD1_RX[4]/$\overline{\text{SD1\_RX}}$[4] is used. 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 1.25, 2.5, and 3.125 Gbaud.

20.5 Memory Map/Register Definition

Table 20-1 is the memory map of the RapidIO configuration registers.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 20-1. RapidIO Memory Map

Offset	Register	Access	Reset[1]	Section/Page
Architectural				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0040_0002 = MPC8572E 0x0041_0002 = MPC8572	20.6.1.1/20-10
0xC_0004	Device information capability register (DICAR)	R	0x*nnnn_nnnn*	20.6.1.2/20-11
0xC_0008	Assembly identity capability register (AIDCAR)	R/W	0x*nnnn_nnnn*	20.6.1.3/20-12
0xC_000C	Assembly information capability register (AICAR)	R/W	0x0000_0000	20.6.1.4/20-12
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE0F8_00*n*9	20.6.1.5/20-13
0xC_0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	20.6.1.6/20-14
0xC_001C	Destination operations capability register (DOCAR)	R	0x0000_FCF4	20.6.1.7/20-15
0xC_0040	Mailbox command and status register (MCSR)	R	0x2020_0000	20.6.1.8/20-16
0xC_0044	Port -Write and doorbell command and status register (PWDCSR)	R	0x2000_0020	20.6.1.9/20-18
0xC_004C	Processing element logical layer control command and status register (PELLCCSR)	R	0x0000_0001	20.6.1.10/20-19
0xC_005C	Local configuration space base address 1 command and status register (LCSBA1CSR)	R/W	0x0000_0000	20.6.1.11/20-19
0xC_0060	Base device ID command and status register (BDIDCSR)	R/W	0x00*nn_nnnn*	20.6.1.12/20-20
0xC_0068	Host base device ID lock command and status register (HBDIDLCSR)	Special	0x0000_FFFF	20.6.1.13/20-21
0xC_006C	Component tag command and status register (CTCSR)	R/W	0x0000_0000	20.6.1.14/20-21
Extended Features Space				
1x/4x LP-Serial				
0xC_0100	Port maintenance block header 0 (PMBH0)	R	0x0600_0001	20.6.2.1/20-22
0xC_0120	Port link time-out control command and status register (PLTOCCSR)	R/W	0xFFFF_FF00	20.6.2.2/20-22
0xC_0124	Port response time-out control command and status register (PRTOCCSR)	R/W	0xFFFF_FF00	20.6.2.3/20-23
0xC_013C	General control command and status register (GCCSR)	R/W	0x*n*000_0000	20.6.2.4/20-23
0xC_0140	Link maintenance request command and status register (LMREQCSR)	R/W	0x0000_0000	20.6.2.5/20-24
0xC_0144	Link maintenance response command and status register (LMRESPCSR)	R	0x0000_0000	20.6.2.6/20-25
0xC_0148	Local ackID status command and status register (LASCSR)	Mixed	0x0000_0000	20.6.2.7/20-25
0xC_0158	Error and status command and status register (ESCSR)	Mixed	0x0000_0001	20.6.2.8/20-26
0xC_015C	Control command and status register (CCSR)	R/W	0x5060_0001	20.6.2.9/20-28
Error Reporting, Logical				
0xC_0600	Error reporting block header (ERBH)	R	0x0000_0007	20.6.3.1/20-30

Table 20-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xC_0608	Logical/Transport layer error detect command and status register (LTLEDCSR)	R/W	0x0000_0000	20.6.3.2/20-30
0xC_060C	Logical/Transport layer error enable command and status register (LTLEECSR)	R/W	0x0000_0000	20.6.3.3/20-32
0xC_0614	Logical/Transport layer address capture command and status register (LTLACCSR)	R/W	0x0000_0000	20.6.3.4/20-33
0xC_0618	Logical/Transport layer device ID capture command and status register (LTLDIDCCSR)	R/W	0x0000_0000	20.6.3.5/20-34
0xC_061C	Logical/Transport layer control capture command and status register (LTLCCCSR)	R/W	0x0000_0000	20.6.3.6/20-35
Error Reporting, Physical				
0xC_0640	Error detect command and status register (EDCSR)	R/W	0x0000_0000	20.6.4.1/20-36
0xC_0644	Error rate enable command and status register (ERECSR)	R/W	0x0000_0000	20.6.4.2/20-36
0xC_0648	Error capture attributes command and status register (ECACSR)	R/W	0x0000_0000	20.6.4.3/20-37
0xC_064C	Packet/control symbol error capture command and status register 0 (PCSECCSR0)	R/W	0x0000_0000	20.6.4.4/20-38
0xC_0650	Packet error capture command and status register 1 (PECCSR1)	R/W	0x0000_0000	20.6.4.5/20-39
0xC_0654	Packet error capture command and status register 2 (PECCSR2)	R/W	0x0000_0000	20.6.4.6/20-39
0xC_0658	Packet error capture command and status register 3 (PECCSR3)	R/W	0x0000_0000	20.6.4.7/20-40
0xC_0668	Error rate command and status register (ERCSR)	R/W	0x8000_0000	20.6.4.8/20-40
0xC_066C	Error rate threshold command and status register (ERTCSR)	R/W	0xFFFF_0000	20.6.4.9/20-41
Implementation Space				
General				
0xD_0004	Logical layer configuration register (LLCR)	R/W	0x0000_0000	20.6.5.1/20-42
0xD_0010	Error / port-write interrupt status register (EPWISR)	R	0x0000_0000	20.6.5.2/20-43
0xD_0020	Logical retry error threshold configuration register (LRETCR)	R/W	0x0000_00FF	20.6.5.3/20-43
0xD_0080	Physical retry error threshold configuration register (PRETCR)	R/W	0x0000_00FF	20.6.5.4/20-44
0xD_0100	Alternate device ID command and status register (ADIDCSR)	R/W	0x0000_0000	20.6.5.5/20-44
0xD_0120	Accept-all configuration register (AACR)	R/W	0x*n*000_000*n*	20.6.5.6/20-45
0xD_0124	Logical Outbound Packet time-to-live configuration register (LOPTTLCR)	R/W	0x0000_0000	20.6.5.7/20-45
0xD_0130	Implementation error command and status register (IECSR)	w1c	0x0000_0000	20.6.5.8/20-46
0xD_0140	Physical configuration register (PCR)	R/W	0x0000_8010	20.6.5.9/20-47
0xD_0158	Serial link command and status register (SLCSR)	w1c	0x0000_0000	20.6.5.10/20-47

Table 20-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_0160	Serial link error injection configuration register (SLEICR)	R/W	0x0000_0000	20.6.5.11/20-48
Revision Control Register				
0xD_0BF8	IP Block Revision Register 1 (IPBRR1)	R	0x01C0_0000	20.6.6.1/20-49
0xD_0BFC	IP Block Revision Register 2 (IPBRR2)	R	0x0000_0000	20.6.6.2/20-49
ATMU				
0xD_0C00	RapidIO outbound window translation address register 0 (ROWTAR0)	R/W	0xFF80_0000	20.6.7.2/20-52
0xD_0C04	RapidIO outbound window translation extended address register 0 (ROWTEAR0)	R/W	0x0000_003F	20.6.7.3/20-53
0xD_0C10	RapidIO outbound window attributes register 0 (ROWAR0)	R/W	0x8004_4023	20.6.7.5/20-54
0xD_0C20	RapidIO outbound window translation address register 1 (ROWTAR1)	R/W	0x0000_0000	20.6.7.2/20-52
0xD_0C24	RapidIO outbound window translation extended address register 1 (ROWTEAR1)	R/W	0x0000_0000	20.6.7.3/20-53
0xD_0C28	RapidIO outbound window base address register 1 (ROWBAR1)	R/W	0x0000_0000	20.6.7.4/20-54
0xD_0C30	RapidIO outbound window attributes register 1 (ROWAR1)	R/W	0x0004_4023	20.6.7.5/20-54
0xD_0C34	RapidIO outbound window segment 1 register 1 (ROWS1R1)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0C38	RapidIO outbound window segment 2 register 1 (ROWS2R1)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0C3C	RapidIO outbound window segment 3 register 1 (ROWS3R1)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0C40 – 0xD_0CFC	RapidIO outbound window 2 through outbound window 7	—	—	—
0xD_0D00	RapidIO outbound window translation address register 8 (ROWTAR8)	R/W	0x0000_0000	20.6.7.2/20-52
0xD_0D04	RapidIO outbound window translation extended address register 8 (ROWTEAR8)	R/W	0x0000_0000	20.6.7.3/20-53
0xD_0D08	RapidIO outbound window base address register 8 (ROWBAR8)	R/W	0x0000_0000	20.6.7.4/20-54
0xD_0D10	RapidIO outbound window attributes register 8 (ROWAR8)	R/W	0x0004_4023	20.6.7.5/20-54
0xD_0D14	RapidIO outbound window segment 1 register 8 (ROWS1R8)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0D18	RapidIO outbound window segment 2 register 8 (ROWS2R8)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0D1C	RapidIO outbound window segment 3 register 8 (ROWS3R8)	R/W	0x0044_0000	20.6.7.6/20-56
0xD_0D60	RapidIO Inbound window translation address register 4 (RIWTAR4)	R/W	0x0000_0000	20.6.7.7/20-58
0xD_0D68	RapidIO Inbound window base address register 4 (RIWBAR4)	R/W	0x0000_0000	20.6.7.8/20-58
0xD_0D70	RapidIO inbound window attributes register 4 (RIWAR4)	R/W	0x0004_4021	20.6.7.9/20-59
0xD_0D80– 0xD_0DBC	RapidIO inbound window 3 through inbound window 2			
0xD_0DC0	RapidIO inbound window translation address register 1 (RIWTAR1)	R/W	0x0000_0000	20.6.7.7/20-58

Table 20-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_0DC8	RapidIO inbound window base address register 1 (RIWBAR1)	R/W	0x0000_0000	20.6.7.8/20-58
0xD_0DD0	RapidIO inbound window attributes register 1 (RIWAR1)	R/W	0x0004_4021	20.6.7.9/20-59
0xD_0DE0	RapidIO inbound window translation address register 0 (RIWTAR0)	R/W	0x0000_0000	20.6.7.7/20-58
0xD_0DF0	RapidIO inbound window attributes register 0 (RIWAR0)	Mixed	0x8004_4021	20.6.7.9/20-59
RapidIO Message Unit				
RapidIO Outbound Message Unit 0 Registers				
0xD_3000	Outbound message 0 mode register (OM0MR)	R/W	0x0000_0000	20.7.1.1/20-61
0xD_3004	Outbound message 0 status register (OM0SR)	Mixed	0x0000_0000	20.7.1.2/20-63
0xD_3008	Extended outbound message 0 descriptor queue dequeue pointer address register (EOM0DQDPAR)	R/W	0x0000_0000	20.7.1.3/20-64
0xD_300C	Outbound message 0 descriptor queue dequeue pointer address register (OM0DQDPAR)	R/W	0x0000_0000	20.7.1.3/20-64
0xD_3010	Extended outbound message 0 source address register (EOM0SAR)	R/W	0x0000_0000	20.7.1.5/20-67
0xD_3014	Outbound message 0 source address register (OM0SAR)	R/W	0x0000_0000	20.7.1.5/20-67
0xD_3018	Outbound message 0 destination port register (OM0DPR)	R/W	0x0000_0000	20.7.1.6/20-68
0xD_301C	Outbound message 0 destination attributes Register (OM0DATR)	R/W	0x0000_0000	20.7.1.7/20-69
0xD_3020	Outbound message 0 double-word count register (OM0DCR)	R/W	0x0000_0000	20.7.1.8/20-69
0xD_3024	Extended outbound message 0 descriptor queue enqueue pointer address register (EOM0DQEPAR)	R/W	0x0000_0000	20.7.1.4/20-66
0xD_3028	Outbound message 0 descriptor queue enqueue pointer address register (OM0DQEPAR)	R/W	0x0000_0000	20.7.1.4/20-66
0xD_302C	Outbound message 0 retry error threshold configuration register (OM0RETCR)	R/W	0x0000_0000	20.7.1.9/20-70
0xD_3030	Outbound message 0 multicast group register (OM0MGR)	R/W	0x0000_0000	20.7.1.10/20-71
0xD_3034	Outbound message 0 multicast list register (OM0MLR)	R/W	0x0000_0000	20.7.1.11/20-72
RapidIO Inbound Message Unit 0 Registers				
0xD_3060	Inbound message 0 mode register (IM0MR)	R/W	0x0000_0000	20.7.2.1/20-72
0xD_3064	Inbound message 0 status register (IM0SR)	Mixed	0x0000_0002	20.7.2.2/20-74
0xD_3068	Extended inbound message 0 frame queue dequeue pointer address register (EIM0FQDPAR)	R/W	0x0000_0000	20.7.2.3/20-75
0xD_306C	Inbound message 0 frame queue dequeue pointer address register (IM0FQDPAR)	R/W	0x0000_0000	20.7.2.3/20-75
0xD_3070	Extended inbound message 0 frame queue enqueue pointer address register (EIM0FQEPAR)	R/W	0x0000_0000	20.7.2.4/20-76
0xD_3074	Inbound message 0 frame queue enqueue pointer address register (IM0FQEPAR)	R/W	0x0000_0000	20.7.2.4/20-76

Table 20-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_3078	Inbound message 0 maximum interrupt report interval register (IM0MIRIR)	R/W	0xFFFF_FF00	20.7.2.5/20-77
RapidIO Outbound Message Unit 1 Registers				
0xD_3100	Outbound message 1 mode register (OM1MR)	R/W	0x0000_0000	20.7.1.1/20-61
0xD_3104	Outbound message 1 status register (OM1SR)	Mixed	0x0000_0000	20.7.1.2/20-63
0xD_3108	Extended outbound message 1 descriptor queue dequeue pointer address register (EOM1DQDPAR)	R/W	0x0000_0000	20.7.1.3/20-64
0xD_310C	Outbound message 1 descriptor queue dequeue pointer address register (OM1DQDPAR)	R/W	0x0000_0000	20.7.1.3/20-64
0xD_3110	Extended outbound message 1 source address register (EOM1SAR)	R/W	0x0000_0000	20.7.1.5/20-67
0xD_3114	Outbound message 1 source address register (OM1SAR)	R/W	0x0000_0000	20.7.1.5/20-67
0xD_3118	Outbound message 1 destination port register (OM1DPR)	R/W	0x0000_0000	20.7.1.6/20-68
0xD_311C	Outbound message 1 destination attributes register (OM1DATR)	R/W	0x0006_0000	20.7.1.7/20-69
0xD_3120	Outbound message 1 double-word count register (OM1DCR)	R/W	0x0000_0000	20.7.1.8/20-69
0xD_3124	Extended outbound message 1 descriptor queue enqueue pointer address register (EOM1DQEPAR)	R/W	0x0000_0000	20.7.1.4/20-66
0xD_3128	Outbound message 1 descriptor queue enqueue pointer address register (OM1DQEPAR)	R/W	0x0000_0000	20.7.1.4/20-66
0xD_312C	Outbound message 1 retry error threshold configuration register (OM1RETCR)	R/W	0x0000_0000	20.7.1.9/20-70
0xD_3130	Outbound message 1 multicast group register (OM1MGR)	R/W	0x0000_0000	20.7.1.10/20-71
0xD_3134	Outbound message 1 multicast list register (OM1MLR)	R/W	0x0000_0000	20.7.1.11/20-72
RapidIO Inbound Message Unit 1 Registers				
0xD_3160	Inbound message 1 mode register (IM1MR)	R/W	0x0000_0000	20.7.2.1/20-72
0xD_3164	Inbound message 1 status register (IM1SR)	Mixed	0x0000_0002	20.7.2.2/20-74
0xD_3168	Extended inbound message 1 frame queue dequeue pointer address register (EIM1FQDPAR)	R/W	0x0000_0000	20.7.2.3/20-75
0xD_316C	Inbound message 1 frame queue dequeue pointer address register (IM1FQDPAR)	R/W	0x0000_0000	20.7.2.3/20-75
0xD_3170	Extended inbound message 1 frame queue enqueue pointer address register (EIM1FQEPAR)	R/W	0x0000_0000	20.7.2.4/20-76
0xD_3174	Inbound message 1 frame queue enqueue pointer address register (IM1FQEPAR)	R/W	0x0000_0000	20.7.2.4/20-76
0xD_3178	Inbound message 1 maximum interrupt report interval register (IM1MIRIR)	R/W	0xFFFF_FF00	20.7.2.5/20-77
RapidIO Doorbell Registers				
0xD_3400	Outbound doorbell mode register (ODMR)	R/W	0x0000_0000	20.7.3.1/20-78
0xD_3404	Outbound doorbell status register (ODSR)	Mixed	0x0000_0000	20.7.3.2/20-79

Table 20-1. RapidIO Memory Map (continued)

Offset	Register	Access	Reset[1]	Section/Page
0xD_3408–0xD_3414	Reserved			
0xD_3418	Outbound doorbell destination port register (ODDPR)	R/W	0x0000_0000	20.7.3.3/20-79
0xD_341C	Outbound doorbell destination attributes register (ODDATR)	R/W	0x0000_0000	20.7.3.4/20-80
0xD_3420–0xD_3428	Reserved			
0xD_342C	Outbound doorbell retry error threshold configuration register (ODRETCR)	R/W	0x0000_0000	20.7.3.5/20-81
0xD_3430–0xD_345C	Reserved			
0xD_3460	Inbound doorbell mode register (IDMR)	R/W	0x0000_0000	20.7.4.1/20-81
0xD_3464	Inbound doorbell status register (IDSR)	Mixed	0x0000_0002	20.7.4.2/20-83
0xD_3468	Extended inbound doorbell queue dequeue pointer address register (EIDQDPAR)	R/W	0x0000_0000	20.7.4.3/20-84
0xD_346C	Inbound doorbell queue dequeue Pointer address register (IDQDPAR)	R/W	0x0000_0000	20.7.4.3/20-84
0xD_3470	Extended inbound doorbell queue enqueue pointer address register (EIDQEPAR)	R/W	0x0000_0000	20.7.4.4/20-85
0xD_3474	Inbound doorbell Queue enqueue pointer address register (IDQEPAR)	R/W	0x0000_0000	20.7.4.4/20-85
0xD_3478	Inbound doorbell maximum interrupt report interval register (IDMIRIR)	R/W	0xFFFF_FF00	20.7.4.5/20-86
0xD_347C	Reserved			
RapidIO Port-Write Registers				
0xD_34E0	Inbound port-write mode register (IPWMR)	R/W	0x0000_0000	20.7.5.1/20-87
0xD_34E4	Inbound port-write status register (IPWSR)	Mixed	0x0000_0000	20.7.5.2/20-88
0xD_34E8	Extended inbound port-write queue base address register (EIPWQBAR)	R/W	0x0000_0000	20.7.5.3/20-88
0xD_34EC	Inbound port-write queue base address register (IPWQBAR)	R/W	0x0000_0000	20.7.5.3/20-88

1 Values indicated with *n* are read from configuration signals at reset; see register description for details.

20.6 RapidIO Endpoint Configuration Register Definitions

The RapidIO endpoint registers are described in detail below.

20.6.1 RapidIO Architectural Registers

20.6.1.1 Device Identity Capability Register (DIDCAR)

Figure 20-2 shows the fields of the device identity capability register (DIDCAR). 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.

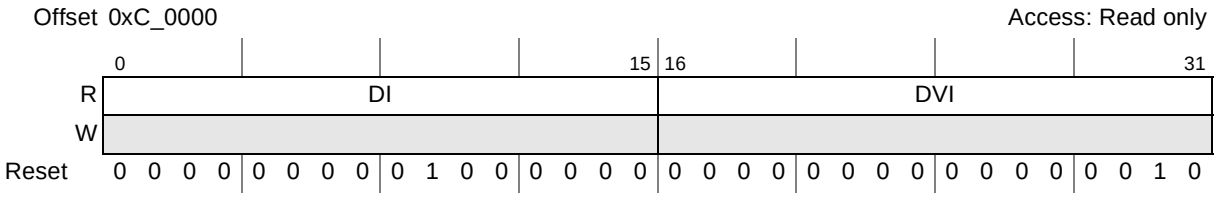


Figure 20-2. Device Identity Capability Register (DIDCAR)

Table 20-2 lists DIDCAR fields.

Table 20-2. DIDCAR Field Descriptions

Bits	Name	Description
0–15	DI	Device identity 0x0040 = MPC8572E 0x0041 = MPC8572
16–31	DVI	Device vendor identity (Freescale = 0x0002)

20.6.1.2 Device Information Capability Register (DICAR)

Figure 20-3 shows the fields of the read-only device information capability register (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 Section 5.2, "e500 Processor and System Version Numbers," and Section 6.5.4, "System Version Register (SVR)," for more information.

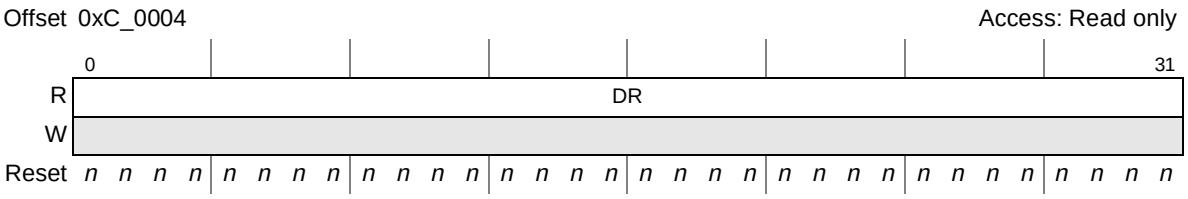


Figure 20-3. Device Information Capability Register (DICAR)

Table 20-3 lists DICAR fields.

Table 20-3. DICAR Field Descriptions

Bits	Name	Description
0–31	DR	Device revision. This is a copy of the device's system version register (SVR). See Section 23.4.1.16, "System Version Register (SVR)," for additional information. 0x80E8_0011 MPC8572E with security 0x80E0_0011 MPC8572 without security

20.6.1.3 Assembly Identity Capability Register (AIDCAR)

Figure 20-4 shows the fields of the assembly identity capability register (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.

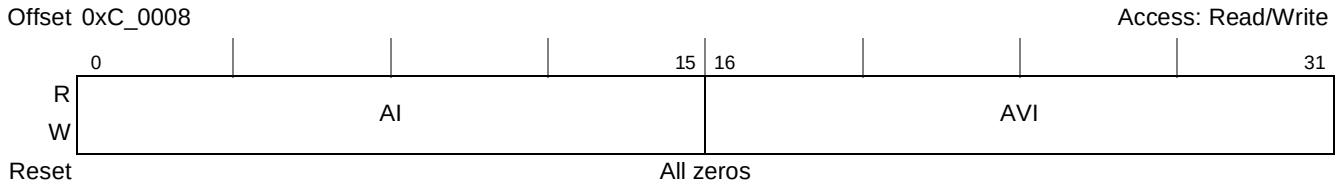


Figure 20-4. Assembly Identity Capability Register (AIDCAR)

Table 20-4 lists the fields of the AIDCAR.

Table 20-4. AIDCAR Field Descriptions

Bits	Name	Description
0–15	AI	Assembly identity (all zeros)
16–31	AVI	Assembly vendor identity (all zeros)

20.6.1.4 Assembly Information Capability Register (AICAR)

Figure 20-5 shows the fields of the assembly information capability register (AICAR). AICAR contains additional information about the assembly and the pointer to the first entry in the extended features list.

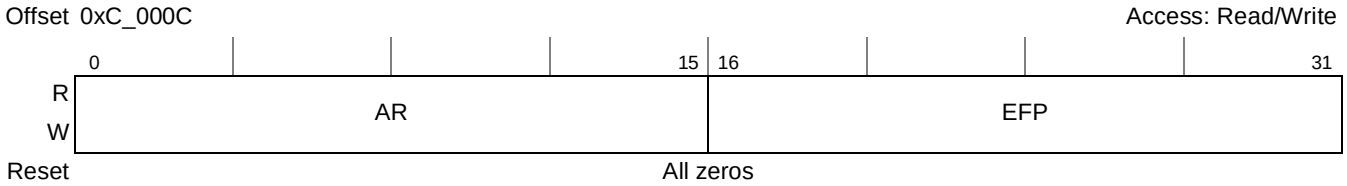


Figure 20-5. Assembly Information Capability Register (AICAR)

Table 20-5 lists AICAR fields.

Table 20-5. AICAR Field Descriptions

Bits	Name	Description
0–15	AR	AssyRev field (all zeros)
16–31	EFP	ExtendedFeaturesPtr field (all zeros)

20.6.1.5 Processing Element Features Capability Register (PEFCAR)

The processing element features capability register (PEFCAR), shown in Figure 20-6, identifies the major functionality provided by the processing element. PEFCAR is a read-only register.

Offset 0xC_0010 Access: Read only

	0	1	2	3	4 7	8 11	12	13 25	26	27	28	29 31
R	BR	MEM	PROC	SW	—	MB	DB	—	CRF	CTLS	EF	EAS
W												
Reset	1	1	1	0	0 0 0 0	1 1 1 1	1	0 0 0 0 0 0 0 0 0 0 0 0 0	1	*n*	1	0 0 1

Figure 20-6. Processing Element Features Capability Register (PEFCAR)

Table 20-6 lists fields of the PEFCAR.

Table 20-6. PEFCAR Field Descriptions

Bits	Name	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)
4–7	—	Reserved
8–11	MB	Mailbox 0–3. Bit 0 indicates PE supports inbound mailbox 0. Bit 1 indicates PE supports inbound mailbox 1. Bit 2 indicates PE supports inbound mailbox 2. Bit 3 indicates PE supports inbound mailbox 3. (MB = 1111)
12	DB	Doorbell. The RapidIO controller supports inbound doorbells. (DB = 1)
13–25	—	Reserved
26	CRF	Critical request flow. PE supports the critical request flow (CRF) function. (CRF = 1)
27	CTLS	This bit reflects the selected RapidIO common transport system size. The RapidIO common transport system size is determined at power-on reset. See Section 4.4.3.20, “RapidIO System Size,” for POR configuration details. 0 PE only supports common transport small systems (up to 256 devices). 1 PE supports common transport large systems (up to 65,536 devices).

Table 20-6. PEFCAR Field Descriptions (continued)

Bits	Name	Description
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)

20.6.1.6 Source Operations Capability Register (SOCAR)

The source operations capability register (SOCAR), shown in Figure 20-7, defines the set of RapidIO I/O logical operations that can be issued by this processing element. SOCAR is a read-only register.

Offset 0xC_0018 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10					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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
W																
Reset	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0

Figure 20-7. Source Operations Capability Register (SOCAR)

Table 20-7 lists SOCAR fields.

Table 20-7. SOCAR Field Descriptions

Bits	Name	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)
9	TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15	—	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)

Table 20-7. SOCAR Field Descriptions (continued)

Bits	Name	Description
20	M	PE supports a Message operation. (M = 1)
21	D	PE supports a Doorbell operation. (D = 1)
22	—	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	—	Reserved
29	PW	PE does not support a Port-Write operation. (PW = 0)
30–31	—	Reserved

20.6.1.7 Destination Operations Capability Register (DOCAR)

The destination operations capability register (DOCAR), shown in Figure 20-8, defines the set of RapidIO I/O operations that can be supported by this processing element. DOCAR is a read-only register.

Offset 0xC_001C — Access: Read only

	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	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NR	NW	SW	NWR	M	D	—	ATS	AI	AD	AS	AC	—	PW	—	
Reset	1	1	1	1	1	1	0	0	1	1	1	1	0	1	0	0

Figure 20-8. Destination Operations Capability Register (DOCAR)

Table 20-8 lists DOCAR fields.

Table 20-8. DOCAR Field Descriptions

Bits	Name	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)

Table 20-8. DOCAR Field Descriptions (continued)

Bits	Name	Description
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)
9	TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–15	—	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	—	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	—	Reserved
29	PW	PE supports a Port-Write operation. (PW = 1)
30–31	—	Reserved

20.6.1.8 Mailbox Command and Status Register (MCSR)

The MCSR, shown in Figure 20-9, reflects the status of the RapidIO message controllers on this device.

Offset 0xC_0040 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14				31
R	A0	FU0	EM0	B0	FA0	ERR0	—		A1	FU1	EM1	B1	FA1	ERR1	—				
W																			
Reset	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Figure 20-9. Mailbox Command and Status Register (MCSR)

Table 20-9 describes the fields of the MCSR.

Table 20-9. MCSR Field Definitions

Bits	Name	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	—	Reserved
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	—	Reserved

20.6.1.9 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 20-10, 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.”

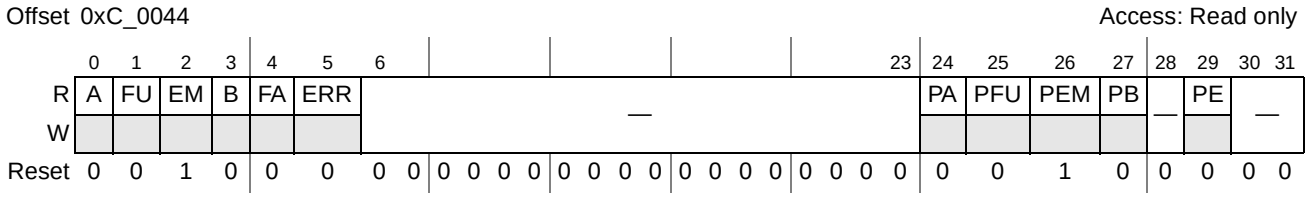


Figure 20-10. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 20-10 describes the fields of the PWDCSR.

Table 20-10. PWDCSR Field Descriptions

Bits	Name	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 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	—	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.

Table 20-10. PWDCSR Field Descriptions (continued)

Bits	Name	Description
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	—	Reserved
29	PE	Error This field always returns a 0.
30–31	—	Reserved

20.6.1.10 Processing Element Logical Layer Control Command and Status Register (PELLCCSR)

The processing element logical layer control command and status register (PELLCCSR), shown in Figure 20-11, controls the extended addressing abilities. The RapidIO endpoint will only supports 34-bit addressing. PELLCCSR is a read-only register.

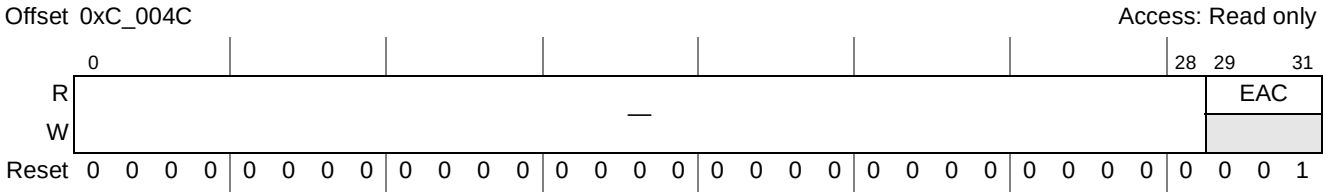


Figure 20-11. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 20-11 lists PELLCCSR fields.

Table 20-11. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. Read-only value is 0b001.

20.6.1.11 Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

The local configuration space base address 1 command and status register (LCSBA1CSR), shown in Figure 20-12, 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.

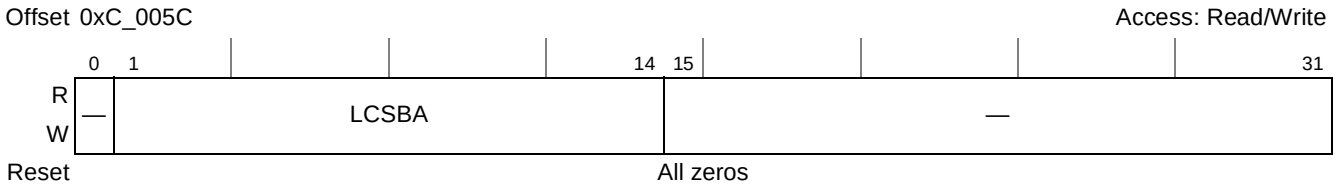


Figure 20-12. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 20-12 lists LCSBA1CSR fields.

Table 20-12. LCSBA1CSR Field Descriptions

Bits	Name	Description
0	—	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	—	Reserved

20.6.1.12 Base Device ID Command and Status Register (BDIDCSR)

BDIDCSR, shown in Figure 20-13, contains the base device ID values for the processing element.

Figure 20-13. Base Device ID Command and Status Register (BDIDCSR)

1 Values indicated with *n* are determined by configuration signals at reset.

Table 20-13 lists BDIDCSR fields.

Table 20-13. BDIDCSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	BDID	Base device ID of the device in a small common transport system (RapidIO device ID). If RapidIO is configured as a host, BDID = {0b0000_0, cfg_device_ID[5:7]}. If RapidIO is configured as an agent, BDID = {0b1111_111, cfg_device_ID[7]}. (See Section 4.4.3.5, “Host/Agent Configuration,” and Section 4.4.3.19, “RapidIO Device ID,” for details.)
16–31	LBDID	Large base device ID of the device in a large common transport system. This field is valid only if PEFCAR[CTLS] is set. If RapidIO is configured as a host, LBDID = 0b0000_0000. If RapidIO is configured as an agent, LBDID = 0b1111_1111. (See Section 4.4.3.5, “Host/Agent Configuration,” for details.)

20.6.1.13 Host Base Device ID Lock Command and Status Register (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. HBDIDLCSR is shown in Figure 20-14.

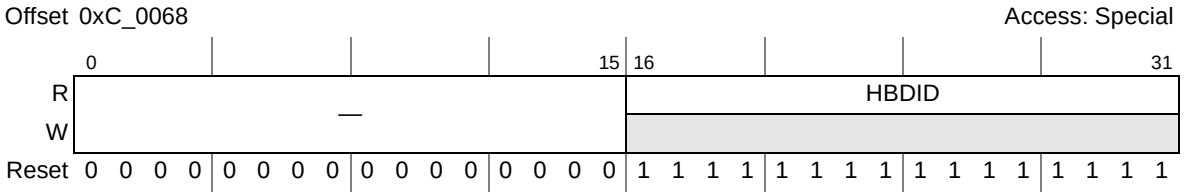


Figure 20-14. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

Table 20-14 lists HBDIDLCSR fields.

Table 20-14. HBDIDLCSR Field Descriptions

Bits	Name	Description
0–15	—	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.

20.6.1.14 Component Tag Command and Status Register (CTCSR)

The component tag command and status register (CTCSR), shown in Figure 20-15, 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.

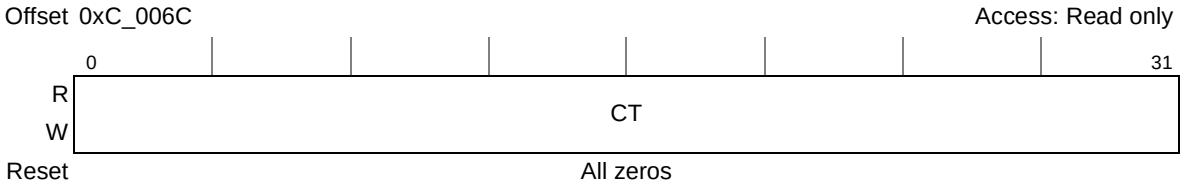


Figure 20-15. Component Tag Command and Status Register (CTCSR)

Table 20-15 lists CTCSR fields.

Table 20-15. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag

20.6.2 RapidIO Extended Features Space, 1x/4x LP-Serial Registers

20.6.2.1 Port Maintenance Block Header 0 Register (PMBH0)

The port maintenance block header 0 register (PMBH0), shown in Figure 20-16, contains a pointer 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.

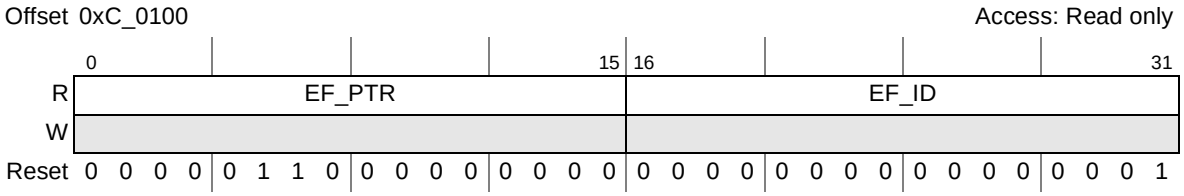


Figure 20-16. Port Maintenance Block Header 0 (PMBH0)

Table 20-16 lists PMBH0 fields.

Table 20-16. PMBH0 Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

20.6.2.2 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The port link time-out control command and status register (PLTOCCSR), shown in Figure 20-17, 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 value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

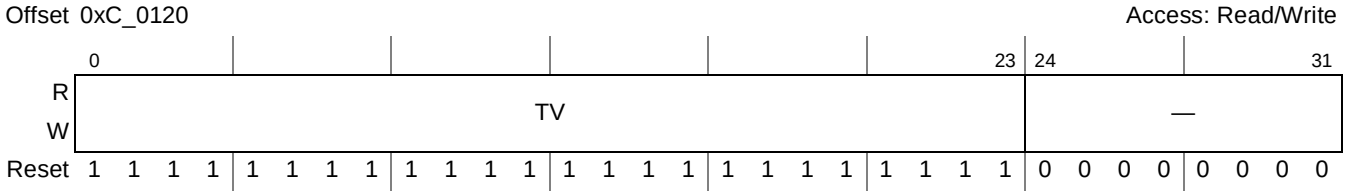


Figure 20-17. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

Table 20-17 lists PLTOCCSR fields.

Table 20-17. PLTOCCSR Field Descriptions

Bits	Name	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	—	Reserved

20.6.2.3 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

The port response time-out control command and status register (PRTOCCSR), shown in Figure 20-18, 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.The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms. Note that this applies to the RapidIO endpoint and the messaging unit.

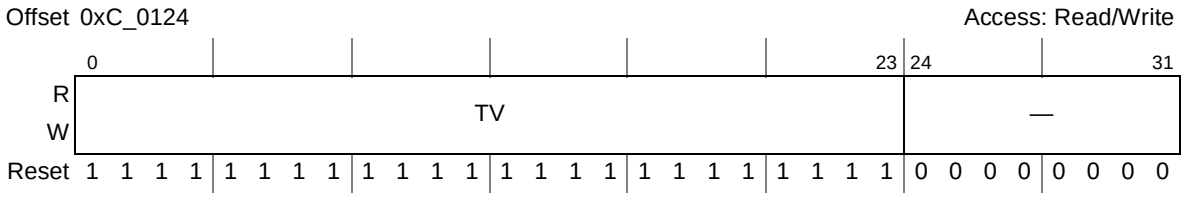


Figure 20-18. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

Table 20-18 lists PRTOCCSR fields.

Table 20-18. PRTOCCSR Field Descriptions

Bits	Name	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	—	Reserved

20.6.2.4 General Control Command and Status Register (GCCSR)

The general control command and status register (GCCSR), shown in Figure 20-19, contains control register bits applicable to the RapidIO interface.

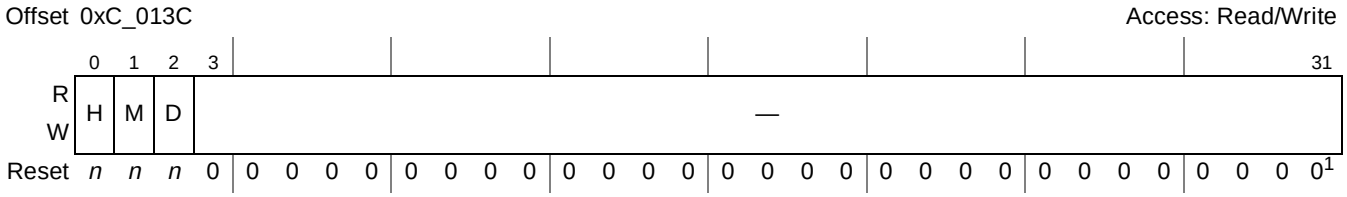


Figure 20-19. General Control Command and Status Register (GCCSR)

[1] Values indicated with *n* are read from configuration signals at reset.

Table 20-19 lists GCCSR fields.

Table 20-19. GCCSR Field Descriptions

Bits	Name	Description
0	H	Host. The value of this bit is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.5, "Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Agent device 1 Host device
1	M	Master. The value of this bit is identical to GCCSR[H] which is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.5, "Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Device is not enabled to issue requests into the system. 1 Device is enabled to issue requests into the system. M is ignored by the RapidIO endpoint. It is expected that if M = 0, software will not send packets out of outbound. If packets are sent by OCN to outbound, they are sent out of the RapidIO endpoint, regardless of the value of M.
2	D	Discovered. The value of this bit is identical to GCCSR[H] which is assigned by power-on reset configuration signals, $\overline{\text{LWE}}n/\overline{\text{LBS}}n$, which determine host/agent configuration for the device. See Section 4.4.3.5, "Host/Agent Configuration," for more information. (Note that although this status bit is R/W, manually changing its value does not affect logical operation.) 0 Device has not been discovered by system host. 1 Device has been discovered by system host.
3–31	—	Reserved

20.6.2.5 Link Maintenance Request Command and Status Register (LMREQCSR)

The link maintenance request command and status register (LMREQCSR), shown in Figure 20-20, is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the RapidIO endpoint port interface. Care should be taken when writing this register that it is only used for hot swap and not for software-assisted error recovery (which is not supported).

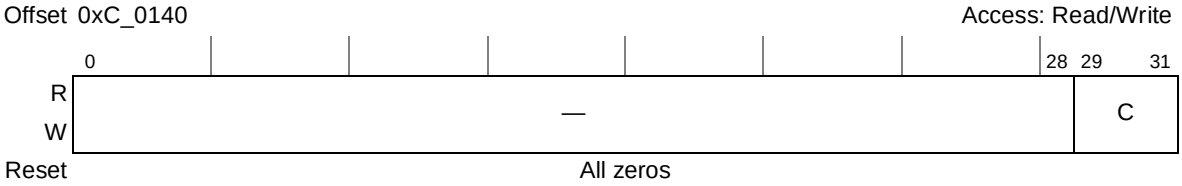


Figure 20-20. Link Maintenance Request Command and Status Register (LMREQCSR)

Table 20-20 lists LMREQCSR fields.

Table 20-20. LMREQCSR Field Descriptions

Bits	Name	Description
0–28	—	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.

20.6.2.6 Link Maintenance Response Command and Status Register (LMRESPCSR)

The link maintenance response command and status register (LMRESPCSR), shown in Figure 20-21, 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.

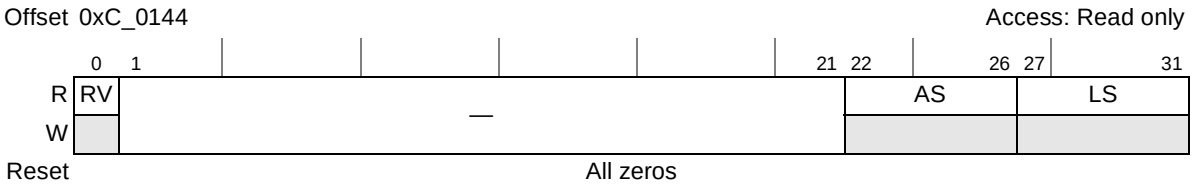


Figure 20-21. Link Maintenance Response Command and Status Register (LMRESPCSR)

Table 20-21 lists the fields of the LMRESPCSR.

Table 20-21. LMRESPCSR Field Descriptions

Bits	Name	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 (clears on read)
1–21	—	Reserved
22–26	AS	AckID_status field from LINK_RESPONSE
27–31	LS	Link_status field from LINK_RESPONSE

20.6.2.7 Local ackID Status Command and Status Register (LASCSR)

The local ackID status command and status register (LASCSR), shown in Figure 20-22, 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. Care should be taken to use this register only for hot swap and not software error management.

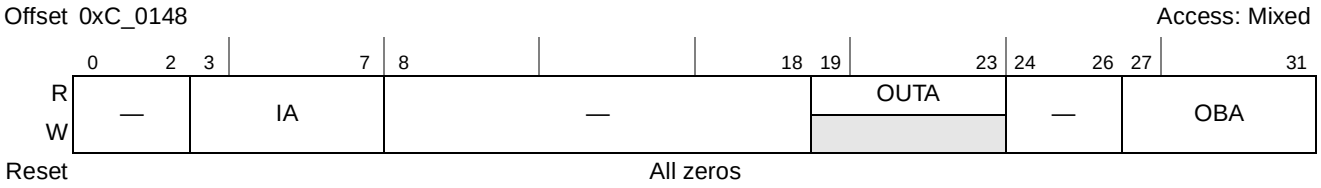


Figure 20-22. Local ackID Status Command and Status Register (LASCSR)

Table 20-22 lists LASCSR fields.

Table 20-22. LASCSR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	IA	Input port next expected ackID value.
8–18	—	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	—	Reserved
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, a newly-written value is ignored.

20.6.2.8 Error and Status Command and Status Register (ESCSR)

The error and status command and status register (ESCSR), shown in Figure 20-23, is accessed when the local processor or an external device wishes to examine the port error and status information.

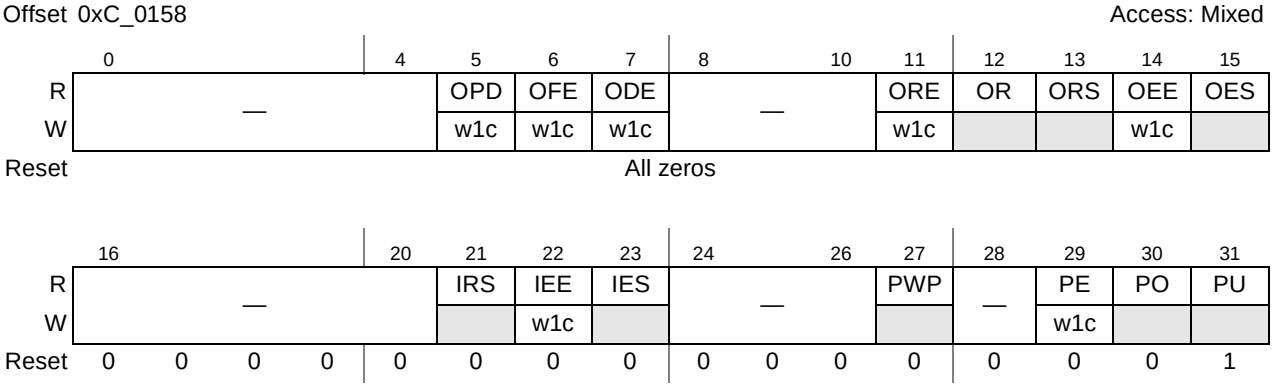


Figure 20-23. Error and Status Command and Status Register (ESCSR)

Table 20-23 lists the fields of the ESCSR.

Table 20-23. ESCSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: • It is received while OFE is set and CCSR[DPE] (drop packet enable) is set and CCSR[SPF] (stop on port failed) is set. • It is received while PCR[OBDEN] (output buffer drain enable) is set. • It is not-accepted by the link-partner while ERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and CCSR[DPE] is set and CCSR[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, will 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. Once cleared, will 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	—	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	—	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	—	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 will not be capable of issuing port-writes. This bit is hardwired to 0.
28	—	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 CCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.

Table 20-23. ESCSR Field Descriptions (continued)

Bits	Name	Description
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).

20.6.2.9 Control Command and Status Register (CCSR)

The control command and status register (CCSR), shown in Figure 20-24, contains control register bits for the RapidIO port.

Offset 0xC_015C Access: Read/Write

	0	1	2	4	5	7	8	9	10	11	12	13	27	28	29	30	31
R	PW		IPW		PWO		PD	OPE	IPE	ECD	MEP		—	SPF	DPE	PL	PT
W																	
Reset	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 20-24. Control Command and Status Register (CCSR)

Table 20-24 lists CCSR fields.

Table 20-24. CCSR Field Descriptions

Bits	Name	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. 000 No override 001 Reserved 010 Force single lane, lane 0 011 Force single lane, lane 2 100–111 Reserved, will cause undefined operation 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 CCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear CCSR[PD] (enabled).
8	PD	Port disable. 0 Input error state machine operates normally 1 Input error state machine is forced to normal state

Table 20-24. CCSR Field Descriptions (continued)

Bits	Name	Description
9	OPE	Output port transmit enable. 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. 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. Initial value read from configuration pins.
10	IPE	Input port receive enable. 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. This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly. Initial value read from configuration pins.
11	ECD	Error checking disable. This bit is hardwired to 0. This bit disables all RapidIO transmission error checking 0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.
12	MEP	Multicast-event participant. This bit is hard-wired to 0.
13–27	—	Reserved
28	SPF	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.
29	DPE	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.
30	PL	Port lockout. 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.
31	PT	Port type (read-only). 0 Reserved 1 Serial port.

20.6.3 RapidIO Extended Features Space—Error Reporting Logical Registers

20.6.3.1 Error Reporting Block Header Register (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.

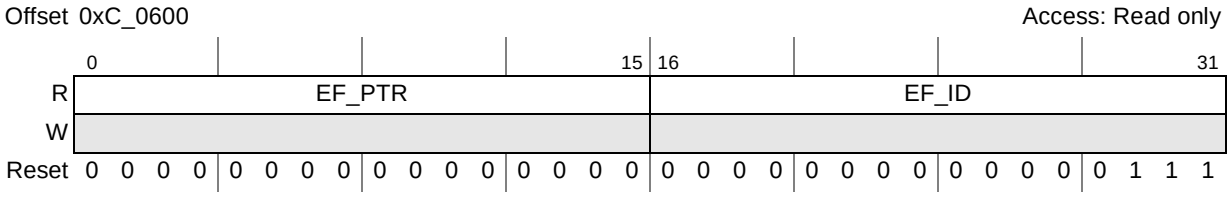


Figure 20-25. Error Reporting Block Header (ERBH)

Table 20-25 lists the fields of ERBH.

Table 20-25. ERBH Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

20.6.3.2 Logical/Transport Layer Error Detect Command and Status Register (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 the detected error and unlock the capture registers.

Error information that corresponds to two or more different error events will not be 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 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 this register. This can be used to emulate a hardware error during software development. Undefined results will occur if fields in this register are set while an actual Logical/Transport Layer error is being detected. Note that this register can be written with an invalid combination of bits set and care should be taken to avoid this

LTLEDCSR is shown in Figure 20-26.

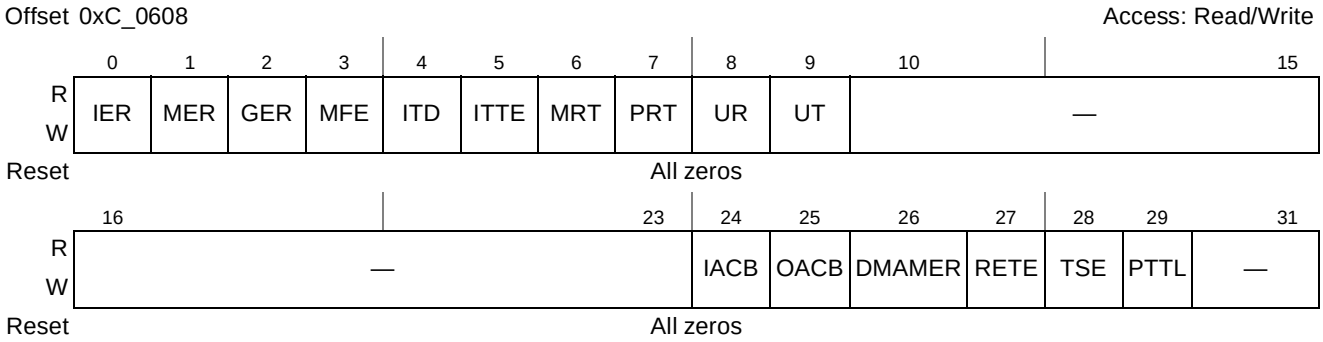


Figure 20-26. Logical/Transport Layer Error Detect Command and Status Register (LTLEDCSR)

Table 20-26 lists the LTLEDCSR fields.

Table 20-26. LTLEDCSR Field Descriptions

Bits	Name	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 when pass-through and accept-all are not enabled. 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.
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	—	Reserved
24	IACB	Inbound ATMU crossed boundary. A transaction is received that crosses an inbound ATMU boundary.

Table 20-26. LTLEDCSR Field Descriptions (continued)

Bits	Name	Description
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	—	Reserved

20.6.3.3 Logical/Transport Layer Error Enable Command and Status Register (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, shown in Figure 20-27, is stored in all ports and the message unit.

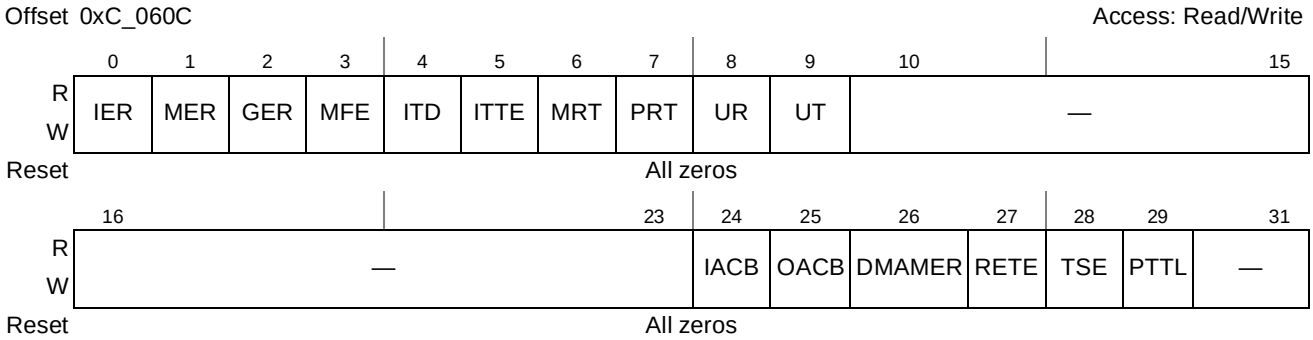


Figure 20-27. Logical/Transport Layer Error Enable Command and Status Register (LTLEECSR)

Table 20-27 lists the LTLEEDCSR fields.

Table 20-27. LTLEECSR Field Descriptions

Bits	Name	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).

Table 20-27. LTLEECSR Field Descriptions (continued)

Bits	Name	Description
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)
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	—	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	—	Reserved

20.6.3.4 Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

This register contains 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 the message unit, although the values in this register can differ between each port and message unit. The message unit LTLACCSR cannot lock if any port has locked; no port LTLACCSR can lock if the message unit or any other port has locked.

Undefined results will occur if this register is written while actual logical/transport layer errors are being detected by the port.

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. Undefined results will occur if fields in this register are set while an actual Logical/Transport Layer error is being detected.

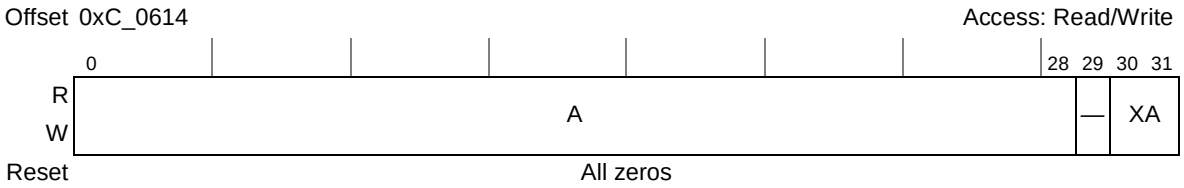


Figure 20-28. Logical/Transport Layer Address Capture Command and Status Register (LTLACCSR)

Table 20-28. LTLACCSR Field Descriptions

Bits	Name	Description
0–28	A	Normally the least significant 29 bits of the address associated with the error (for requests, for responses if available). Please see Section 20.8.12.3, “Logical Layer Errors and Error Handling,” for details.
29	—	Reserved
30–31	XA	xamsbs. Normally the extended address bits of the address associated with the error (for requests, responses, if available). Please see Section 20.8.12.3, “Logical Layer Errors and Error Handling,” for details.

20.6.3.5 Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

This register 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 the message unit, although the values in this register can differ between each port and message unit. The message unit LTLDIDCCSR cannot lock if any port has locked; no port LTLDIDCCSR can lock if the message unit or any other port has locked. Undefined results will occur if this register is written while actual logical/transport layer errors are being detected by the port.

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. Undefined results will occur if fields in this register are set while an actual Logical/Transport Layer error is being detected.

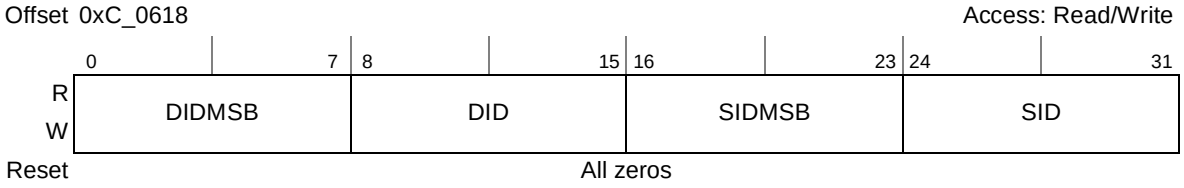


Figure 20-29. Logical/Transport Layer Device ID Capture Command and Status Register (LTLDIDCCSR)

Table 20-29. LTLDIDCCSR Field Descriptions

Bits	Name	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). Please see Section 20.8.12.3, “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. Please see Section 20.8.12.3, “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). Please see Section 20.8.12.3, “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. Please see Section 20.8.12.3, “Logical Layer Errors and Error Handling,” for details.

20.6.3.6 Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

This register contains error information. LTLCCCSR is stored in each port and the message unit, although the values in this register can differ between each port and message unit. The message unit LTLCCCSR cannot lock if any port has locked; no port LTLCCCSR can lock if the message unit or any other port has locked. Undefined results will occur if this register is written while actual logical/transport layer errors are being detected by the port.

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. Undefined results will occur if fields in this register are set while an actual Logical/Transport Layer error is being detected.

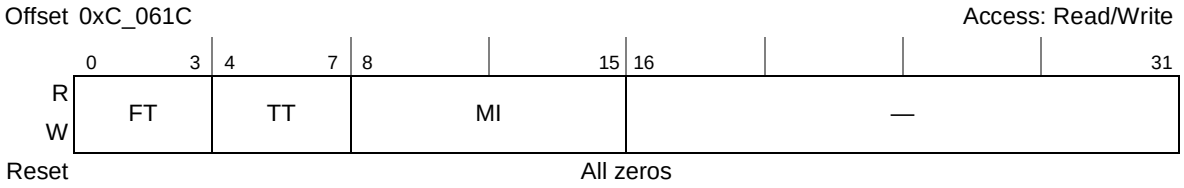


Figure 20-30. Logical/Transport Layer Control Capture Command and Status Register (LTLCCCSR)

Table 20-30. LTLCCCSR Field Descriptions

Bits	Name	Description
0–3	FT	Normally the format type associated with the error. Please see Section 20.8.12.3, “Logical Layer Errors and Error Handling,” for details.
4–7	TT	Normally the transaction type associated with the error. Please see Section 20.8.12.3, “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). Please see Section 20.8.12.3, “Logical Layer Errors and Error Handling,” for details.
16–31	—	Reserved

20.6.4 RapidIO Extended Features Space—Error Reporting Physical Registers

20.6.4.1 Error Detect Command and Status Register (EDCSR)

The error detect command and status register (EDCSR), shown in Figure 20-31, 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 will occur if this register is written while actual physical layer errors are being detected by the port.

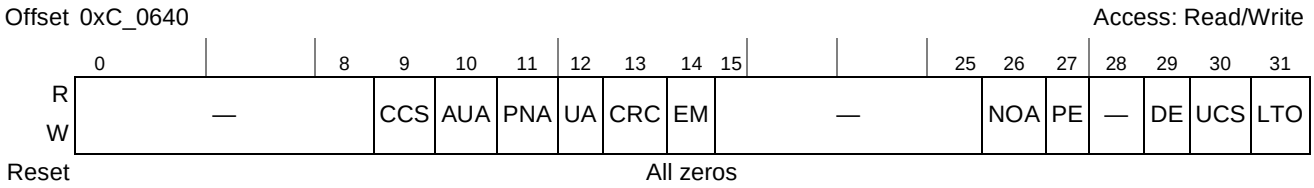


Figure 20-31. Error Detect Command and Status Register (EDCSR)

Table 20-31. EDCSR Field Descriptions

Bits	Name	Description
0–8	—	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	—	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.

20.6.4.2 Error Rate Enable Command and Status Register (ERECSR)

This register contains the bits that control when an error condition is allowed to increment the error rate counter in the error rate threshold register and lock the error capture registers.

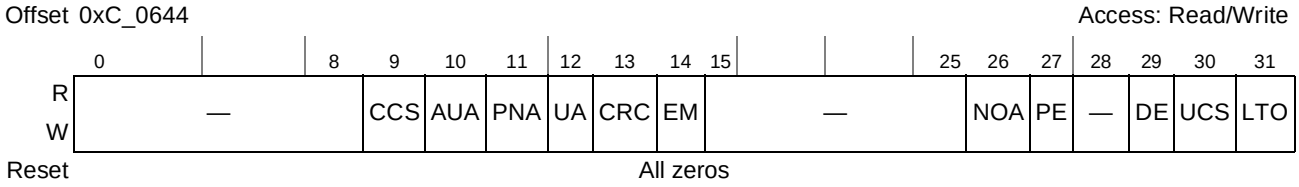


Figure 20-32. Error Rate Enable Command and Status Register (ERECSR)

Table 20-32. ERECSR Field Descriptions

Bits	Name	Description
0–8	—	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	—	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.
30	UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31	LTO	Enable error rate counting of link time-out errors.

20.6.4.3 Error Capture Attributes Command and Status Register (ECACSR)

The error capture attribute register indicates the type of information contained in the 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 will occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

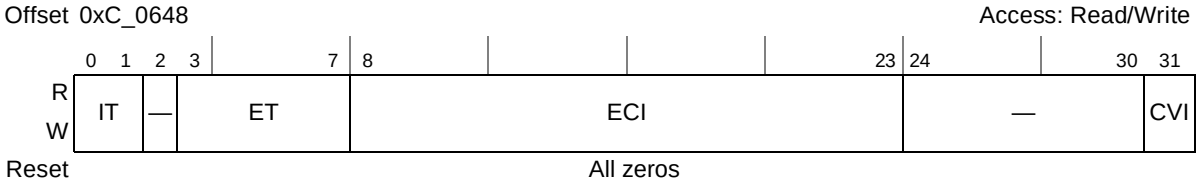


Figure 20-33. Error Capture Attributes Command and Status Register (ECACSR)

Table 20-33. ECACSR Field Descriptions

Bits	Name	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.)
2	—	Reserved
3–7	ET	The encoded value of the bit in the error detect CSR that describes the error captured in the 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. Each ECI bit reflects the validity of captured data. If a bit is set, then the designated byte of captured data is valid. If a bit is cleared, then the designated byte of the specified register does not contain valid data and should be disregarded until the bit is set. ECI[0] reflects validity of PCSECCSR0[0:7] ECI[1] reflects validity of PCSECCSR0[8:15] ECI[2] reflects validity of PCSECCSR0[16:23] ECI[3] reflects validity of PCSECCSR0[24:31] ECI[4] reflects validity of PECCSR1[0:7] ECI[5] reflects validity of PECCSR1[8:15] ... ECI[14] reflects validity of PECCSR3[16:23] ECI[15] reflects validity of PECCSR3[24:31]
24–30	—	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 will contain meaningful information.

20.6.4.4 Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)

This register contains the first 4 bytes of captured packet symbol information or a control character and control symbol. Undefined results will occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

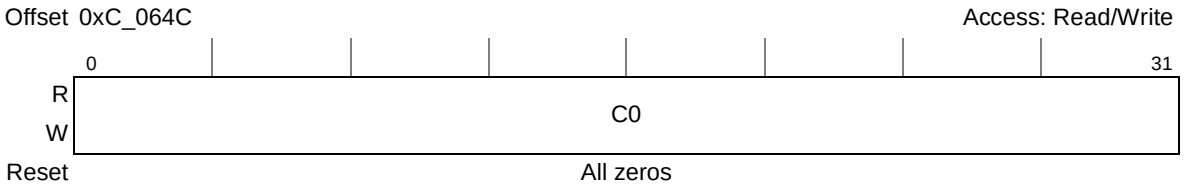


Figure 20-34. Packet/Control Symbol Error Capture Command and Status Register 0 (PCSECCSR0)

Table 20-34. PCSECCSR0 Field Descriptions

Bits	Name	Description
0–31	C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

20.6.4.5 Packet Error Capture Command and Status Register 1 (PECCSR1)

Error capture register 1 contains bytes 4 through 7 of the packet header. Undefined results will occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

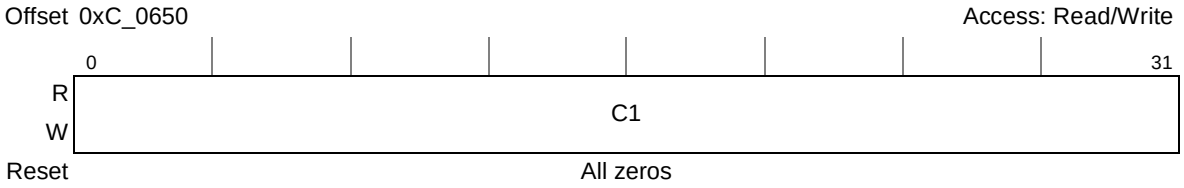


Figure 20-35. Packet Error Capture Command and Status Register 1 (PECCSR1)

Table 20-35. PECCSR1 Field Descriptions

Bits	Name	Description
0–31	C1	Capture 1. Bytes 4 to 7 of the packet header

20.6.4.6 Packet Error Capture Command and Status Register 2 (PECCSR2)

Error capture register 2 contains bytes 8 through 11 of the packet header. Undefined results will occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

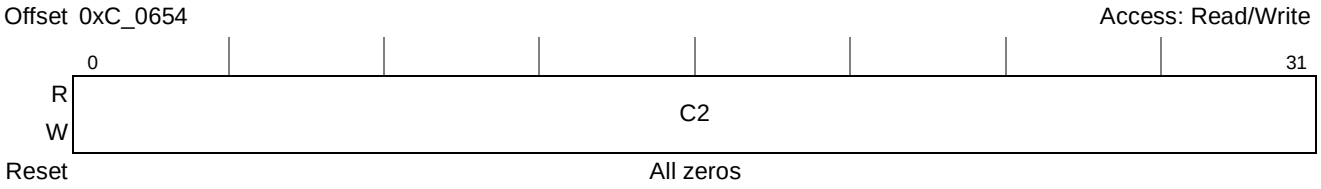


Figure 20-36. Packet Error Capture Command and Status Register 2 (PECCSR2)

Table 20-36. PECCSR2 Field Descriptions

Bits	Name	Description
0–31	C2	Capture 2. Bytes 8 to 11 of the packet header

20.6.4.7 Packet Error Capture Command and Status Register 3 (PECCSR3)

Error capture register 3 contains bytes 12 through 15 of the packet header. Undefined results will occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

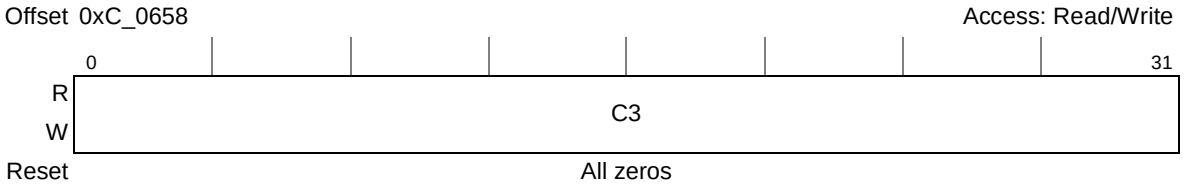


Figure 20-37. Packet Error Capture Command and Status Register 3 (PECCSR3)

Table 20-37. PECCSR3 Field Descriptions

Bits	Name	Description
0–31	C3	Capture 3. Bytes 12 to 15 of the packet header

20.6.4.8 Error Rate Command and Status Register (ERCSR)

The error rate register is a 32-bit register used with the error rate threshold register to monitor and control the reporting of transmission errors.

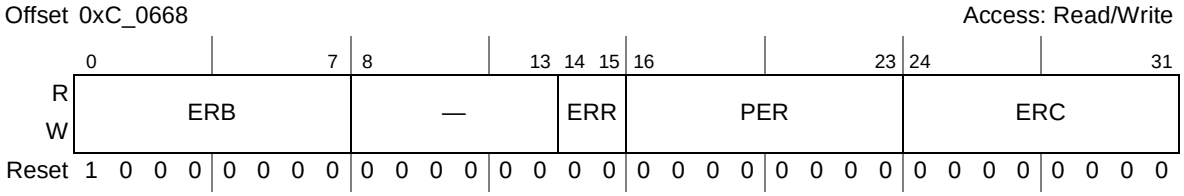


Figure 20-38. Error Rate Command and Status Register (ERCSR)

Table 20-38. ERCSR Field Descriptions

Bits	Name	Description
0–7	ERB	These bits provide the error rate bias value. 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms (±34%) 0x02 Decrement every 10 ms (±34%) 0x04 Decrement every 100 ms (±34%) 0x08 Decrement every 1 s (±34%) 0x10 Decrement every 10 s (±34%) 0x20 Decrement every 100 s (±34%) 0x40 Decrement every 1000 s (±34%) 0x80 Decrement every 10000 s (±34%) Other values are reserved and will cause undefined operation.
8–13	—	Reserved

Table 20-38. ERCSR Field Descriptions (continued)

Bits	Name	Description
14–15	ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. 0b00 Only count 2 errors above 0b01 Only count 4 errors above 0b10 Only count 16 error above 0b11 Do not limit incrementing the error rate count Note that the Error Rate Counter will never increment above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger might imply that it would.
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).

20.6.4.9 Error Rate Threshold Command and Status Register (ERTCSR)

The error rate threshold register is a 32-bit register used to control the reporting of the link status to the system host.

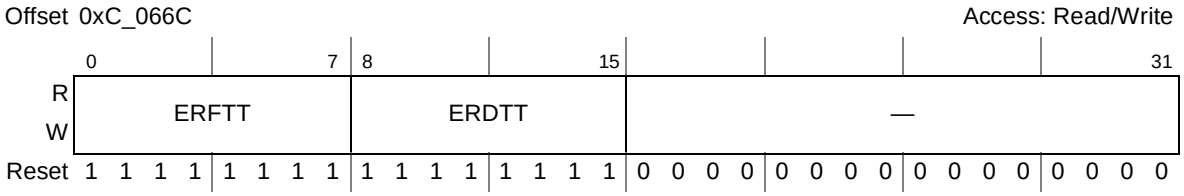


Figure 20-39. Error Rate Threshold Command and Status Register (ERTCSR)

Table 20-39. ERTCSR Field Descriptions

Bits	Name	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. 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. The ESCSR[OFE] bit will not be set if the ERFTT is written to a value lower than or equal to the ERCSR[ERC].
8–15	ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. 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. The ESCSR[ODE] bit will not be set if the ERDTT is written to a value lower than or equal to the ERCSR[ERC].
16–31	—	Reserved

20.6.5 RapidIO Implementation Space Registers

20.6.5.1 Logical Layer Configuration Register (LLCR)

The logical layer configuration register contains general port-common logical layer mode enables.

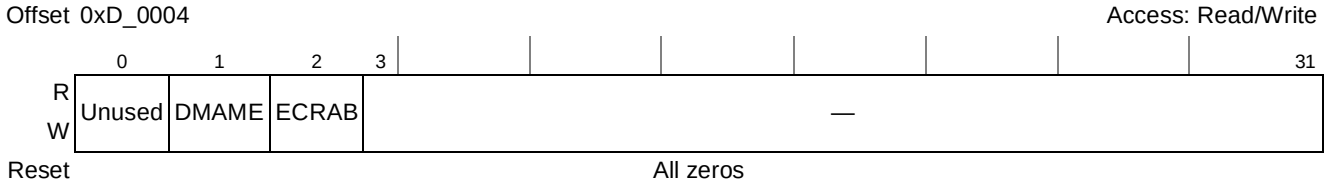


Figure 20-40. Logical Layer Configuration Register (LLCR)

Table 20-40. LLCR Field Descriptions

Bits	Name	Description
0	—	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

Table 20-40. LLCR Field Descriptions (continued)

Bits	Name	Description
2	ECRAB	External configuration register access block. When set, all maintenance requests and accesses that hit LCSBA1CSR is blocked; reads will return all 0's, and writes are ignored (both return done response). When clear, any external RapidIO device can access registers.
3–31	—	Reserved

20.6.5.2 Error/Port-Write Interrupt Status Register (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.

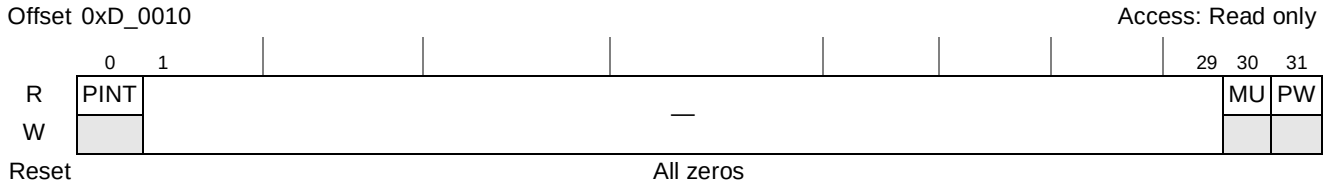


Figure 20-41. Error/Port-Write Interrupt Status Register (EPWISR)

Table 20-41. EPWISR Field Descriptions

Bits	Name	Description
0	PINT	A physical or logical/transport error interrupt was generated.
1–29	—	Reserved (can be used to indicate error on more ports if they exist).
30	MU	A logical/transport layer error interrupt was generated in the message unit.
31	PW	An inbound port-write was received.

20.6.5.3 Logical Retry Error Threshold Configuration Register (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 to 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.

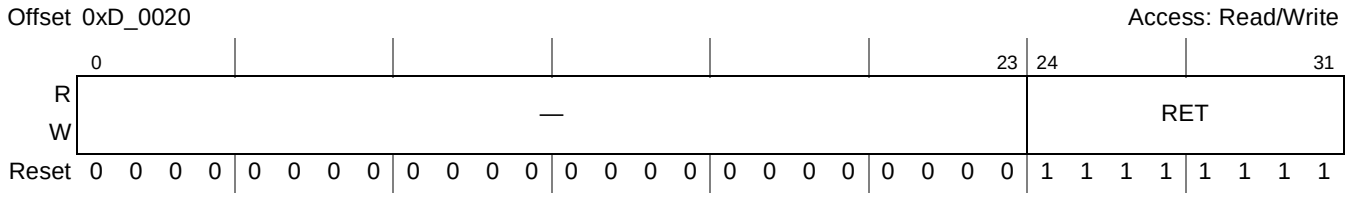


Figure 20-42. Logical Retry Error Threshold Configuration Register (LRETCR)

Table 20-42. LRETCR Field Descriptions

Bits	Name	Description
0–23	—	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 will cause 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

20.6.5.4 Physical Retry Error Threshold Configuration Register (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.

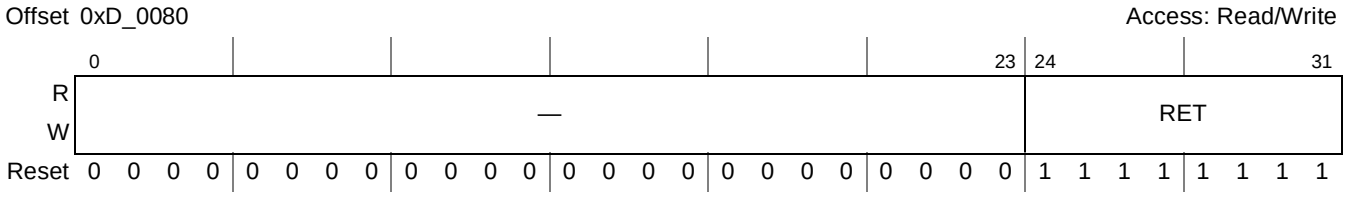


Figure 20-43. Physical Retry Error Threshold Configuration Register (PRETCR)

Table 20-43. PRETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. These bits provide the threshold value for the number of consecutive ACK retries received that will cause 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

20.6.5.5 Alternate Device ID Command and Status Register (ADIDCSR)

The alternate device id CSR contains an alternate deviceID. It is intended that this register should be enabled before the master enabled bit of the 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 will only accept packets sent with the deviceID contained in ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint will accept packets using the same common transport system). In addition, the outbound RapidIO endpoint will only generate requests using the deviceID contained in ADIDCSR; it will generate responses with the deviceID given in the original request packet (either from ADIDCSR or BDIDCSR).

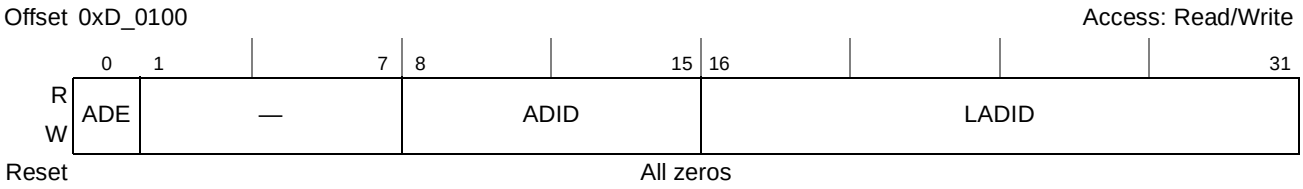


Figure 20-44. Alternate Device ID Command and Status Register (ADIDCSR)

Table 20-44. ADIDCSR Field Descriptions

Bits	Name	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	—	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

20.6.5.6 Accept-All Configuration Register (AACR)

The accept-all configuration register contains information on accept-all mode.

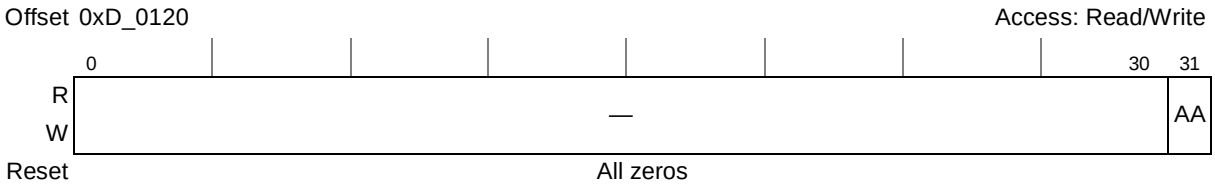


Figure 20-45. Accept-All Configuration Register (AACR)

Table 20-45. AACR Field Descriptions

Bits	Name	Description
0–30	—	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. 0 Normal RapidIO acceptance based on target ID.

20.6.5.7 Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)

The logical outbound packet time-to-live configuration register, shown in Figure 20-46, 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). The reset value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

When the packet time-to-live counter expires, PCR[OBDEN] is automatically set. PCR[OBDEN] must be cleared by software.

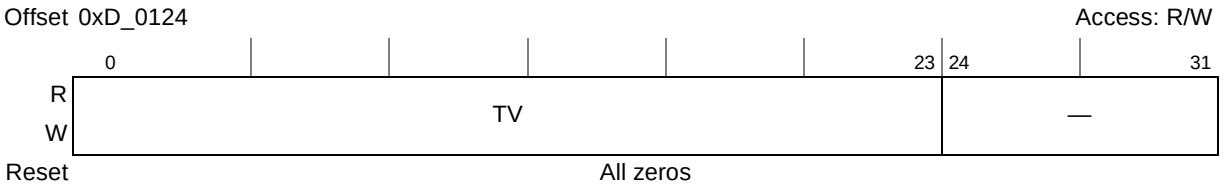


Figure 20-46. Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR)

Table 20-46 lists LOPTTLCR fields.

Table 20-46. LOPTTLCR Field Descriptions

Bits	Name	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	—	Reserved

20.6.5.8 Implementation Error Command and Status Register (IECSR)

The IECSR register contains status bits that are asserted whenever an implementation-defined error occurs.

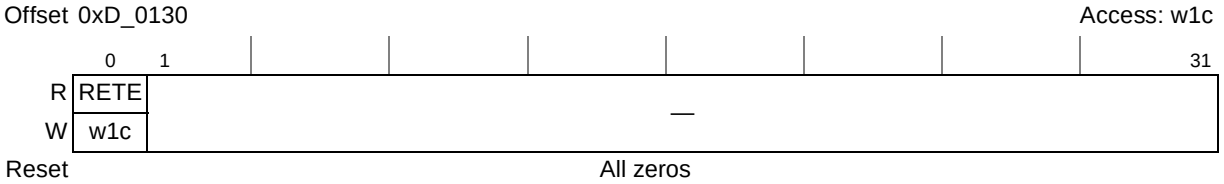


Figure 20-47. Implementation Error Command and Status Register (IECSR)

Table 20-47. IECSR Field Descriptions

Bits	Name	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 will set again if another retry is received and the number of consecutive retries continues to exceed the Retry Error Threshold.
1–31	—	Reserved

20.6.5.9 Physical Configuration Register (PCR)

The PCR contains general physical layer protocol and link mode enables.

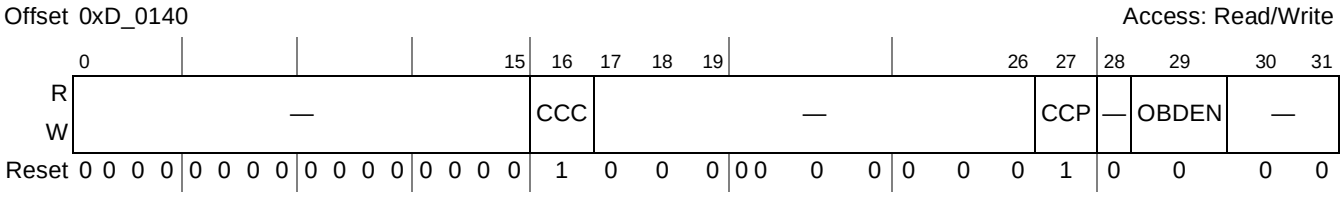


Figure 20-48. Physical Configuration Register (PCR)

Table 20-48. PCR Field Descriptions

Bits	Name	Description
0–15	—	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.
17-18	—	Reserved
19–26	—	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	—	Reserved
29	OBDEN	Output buffer drain enable. When set, the output drains packets from the outbound buffer and does not send them out. This will intentionally cause 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 will cause this bit to be set. (See Section 20.6.5.7, “Logical Outbound Packet Time-to-Live Configuration Register (LOPTTLCR).”)
30–31	—	Reserved

20.6.5.10 Serial Link Command and Status Register (SLCSR)

The SLCSR contains status of the of the serial physical link.

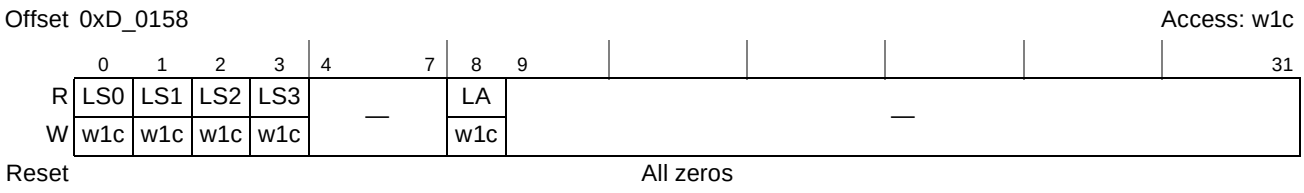


Figure 20-49. Serial Link Command and Status Register (SLCSR)

Table 20-49. SLCSR Field Descriptions

Bits	Name	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	—	Reserved
8	LA	Lane alignment achieved. Write with 1 to clear.
9–31	—	Reserved

20.6.5.11 Serial Link Error Injection Configuration Register (SLEICR)

The SLEICR is used to control the injection of bit errors into the transmit bit stream.

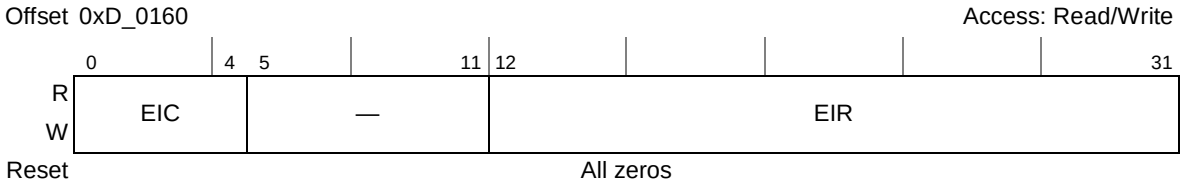


Figure 20-50. Serial Link Error Injection Configuration Register (SLEICR)

Table 20-50. SLEICR Field Descriptions

Bits	Name	Description
0–4	EIC	Error injection control. Enables and controls serial link error injection as follows: 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 All other values are reserved.
5–11	—	Reserved
12–31	EIR	Error injection range. The value of EIR × 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.

The 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 will occur 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, will determine the maximum number of character times between injected errors.

20.6.6 Revision Control Registers

20.6.6.1 IP Block Revision Register 1 (IPBRR1)

IP block revision register 1 is used to track changes and revisions of the RapidIO endpoint.

IP block revision register 1 is a read-only register.

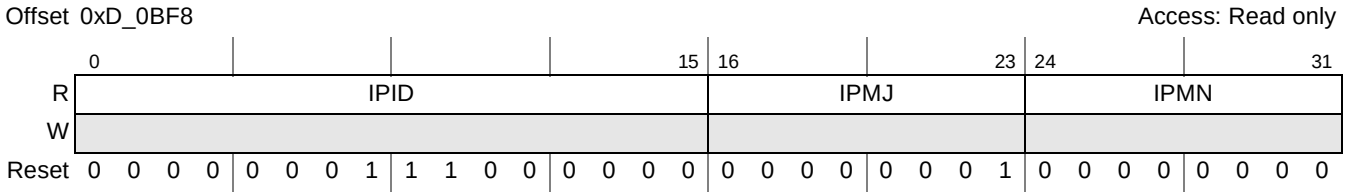


Figure 20-51. IP Block Revision Register 1 (IPBRR1)

Table 20-51. IPBRR1 Field Descriptions

Bits	Name	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 = 0x0

20.6.6.2 IP Block Revision Register 2 (IPBRR2)

IP block revision register 2 is used to track changes and revisions of the RapidIO endpoint.

IP block revision register 2 is a read-only register.

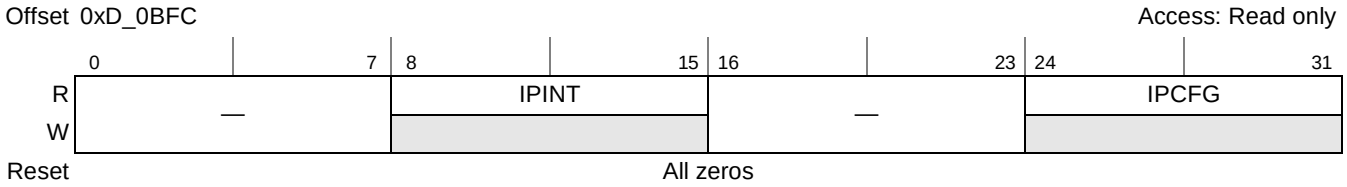


Figure 20-52. IP Block Revision Register 2 (IPBRR2)

Table 20-52. IPBRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IPBID	IP block Integration options = 0x0
16–23	—	Reserved
24–31	IPCFG	IP block Configuration options = 0x0

20.6.7 RapidIO Implementation Space—ATMU Registers

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 will result 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 will cause 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.

The RapidIO endpoint implementation allows up to a 34-bit (0:33) RapidIO address and a 36-bit (0:35) OCN address. In a device confined to 32-bit OCN addresses, the top 4 bits (0:3) of the Inbound translation address and the Outbound base address should be set to all 0’s; setting them otherwise will result in undefined behavior.

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.

20.6.7.1 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. Figure 20-53 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.

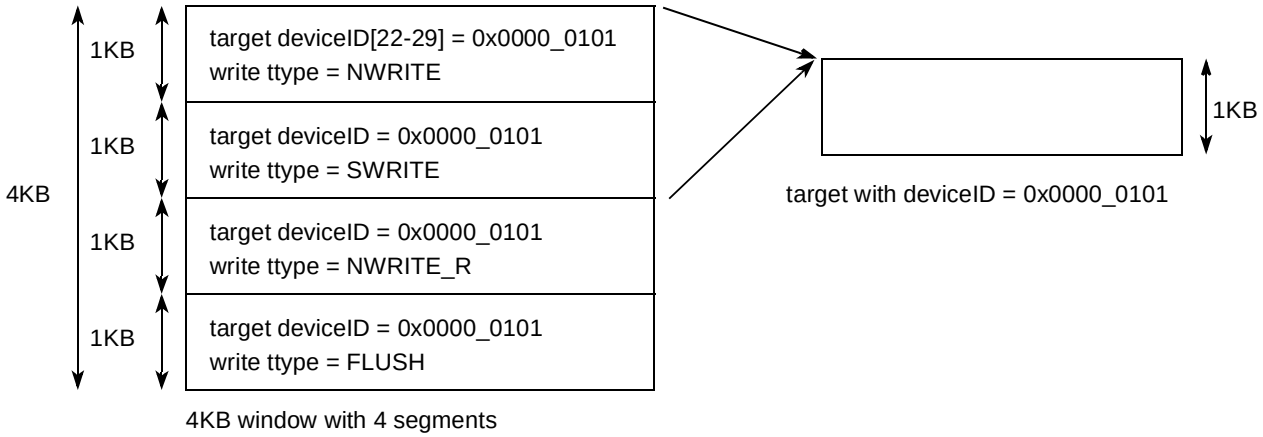


Figure 20-53. Example of Attribute Aliasing

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. Figure 20-54 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 ttype 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.

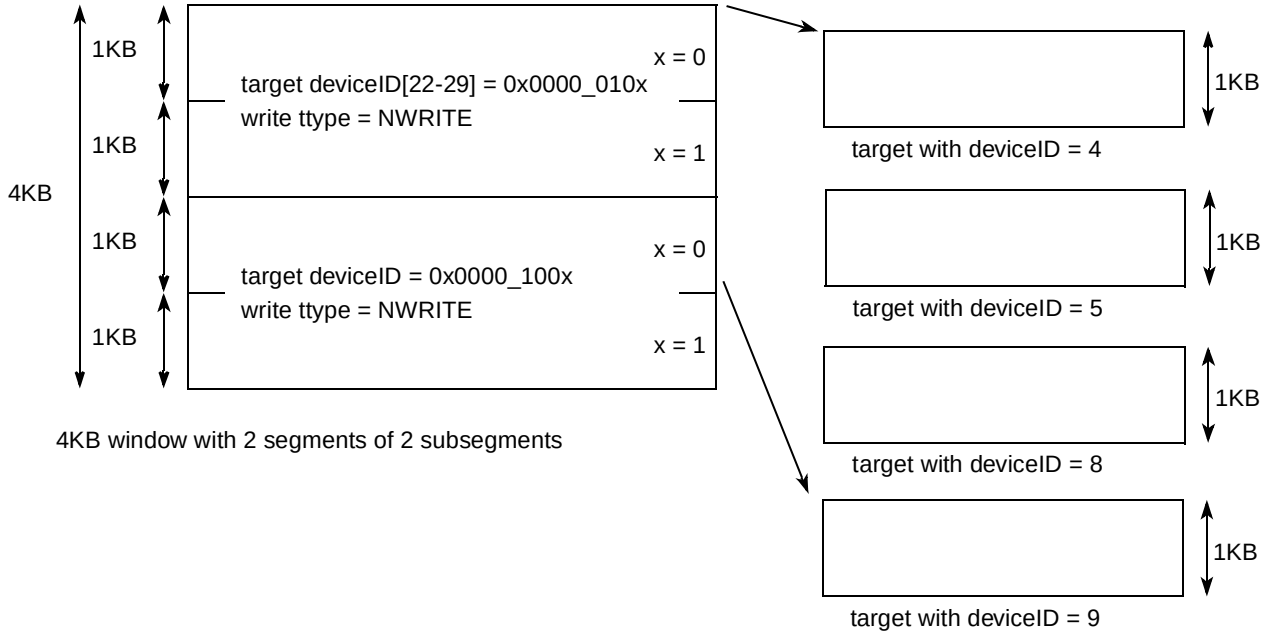


Figure 20-54. Example of Multi-Targeting

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.

20.6.7.2 RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)

The 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.

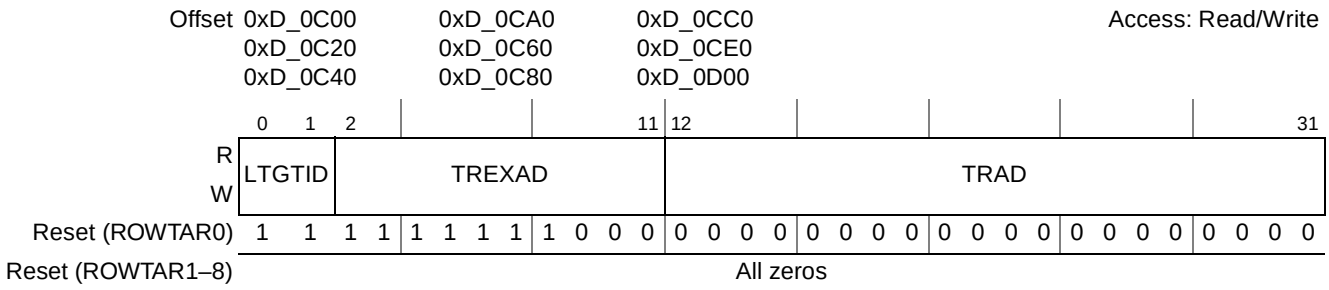


Figure 20-55. RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)

Table 20-53. ROWTAR*n* Field Descriptions

Bits	Name	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 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.

20.6.7.3 RapidIO Outbound Window Translation Extended Address Registers 0–8 (ROWTEAR*n*)

The RapidIO outbound window translation extended address registers contain bits 0–5 of the target ID for a common transport large system.

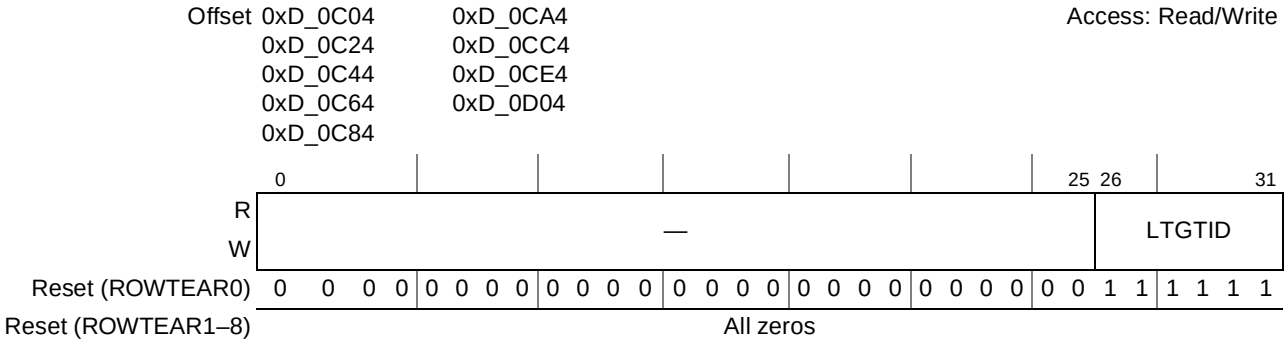


Figure 20-56. RapidIO Outbound Window Translation Extended Address Registers 0–8

Table 20-54. ROWTEAR*n* Field Descriptions

Bits	Name	Description
0–25	—	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 RapidIO outbound window translation address register.

20.6.7.4 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

The 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, please see Section 20.8.5.2, “Window Boundary Crossing Errors.”

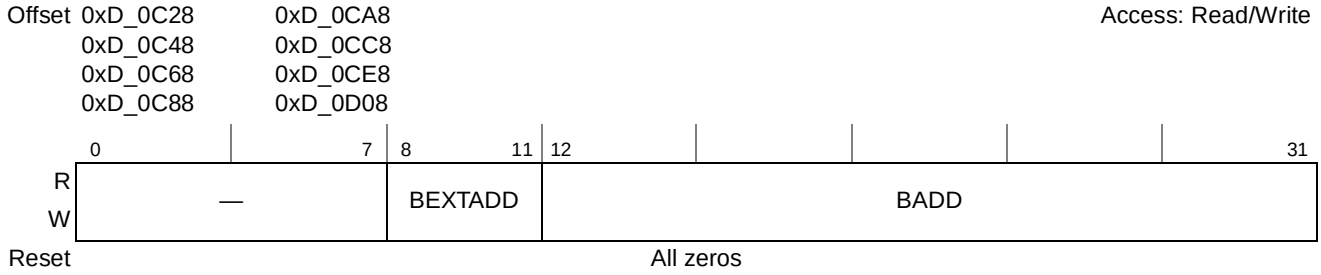


Figure 20-57. RapidIO Outbound Window Base Address Registers 1–8

Table 20-55. ROWBAR*n* Descriptions

Bits	Name	Description
0–7	—	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.

20.6.7.5 RapidIO Outbound Window Attributes Registers 0–8 (ROWAR*n*)

The RapidIO outbound window attributes registers, shown in Figure 20-58, 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.

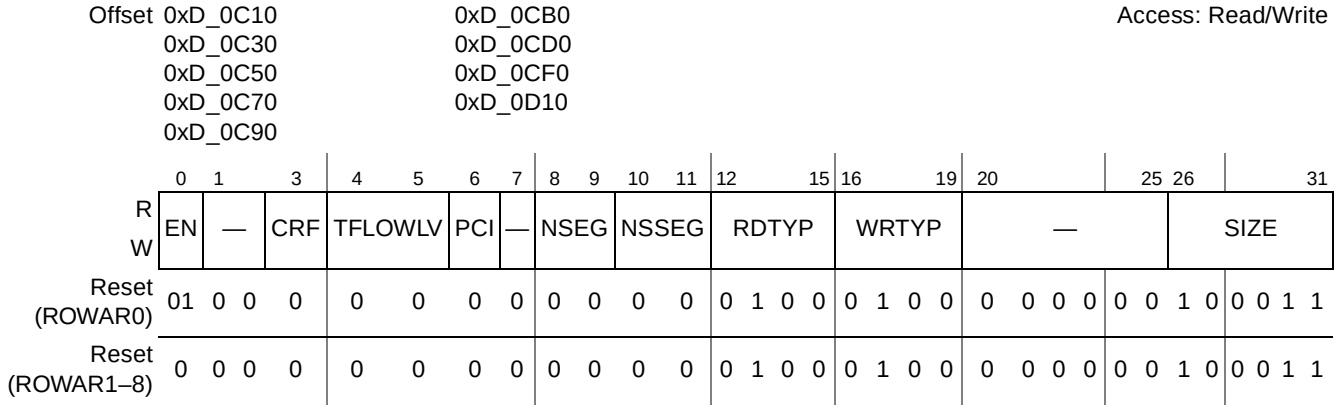


Figure 20-58. RapidIO Outbound Window Attributes Registers 0–8

Table 20-56. ROWAR*n* Field Descriptions

Bits	Name	Description
0	EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–2	—	Reserved
3	CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5	TFLOWLV	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved 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.
6	PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification 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.
7	—	Reserved
8–9	NSEG	Number of segments for this window. 00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved This field is reserved for default window 0.
10–11	NSSEG	Number of subsegments for this segment. 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 This field is reserved for default window 0. Note that this field is valid only when ROWARn[NSEG] contains a non-zero value (1 or 2).
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 ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear

Table 20-56. ROWAR*n* Field Descriptions (continued)

Bits	Name	Description
16–19	WRTYP	Transaction type to run on RapidIO interface if access was a write. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1111 Reserved 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 will generate a NWRITE_R instead.
20–25	—	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. 000000 Reserved ... 001011 4K window size 001100 8K window size ... 011111 4G window size 100000 8G window size 100001 16G window size 100010 32G window size 100011 64G window size 100100 Reserved ... 111111 Reserved This field is read only for default window 0.

20.6.7.6 RapidIO Outbound Window Segment 1–3 Registers 1–8 (ROWS*n*R*n*)

The 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).

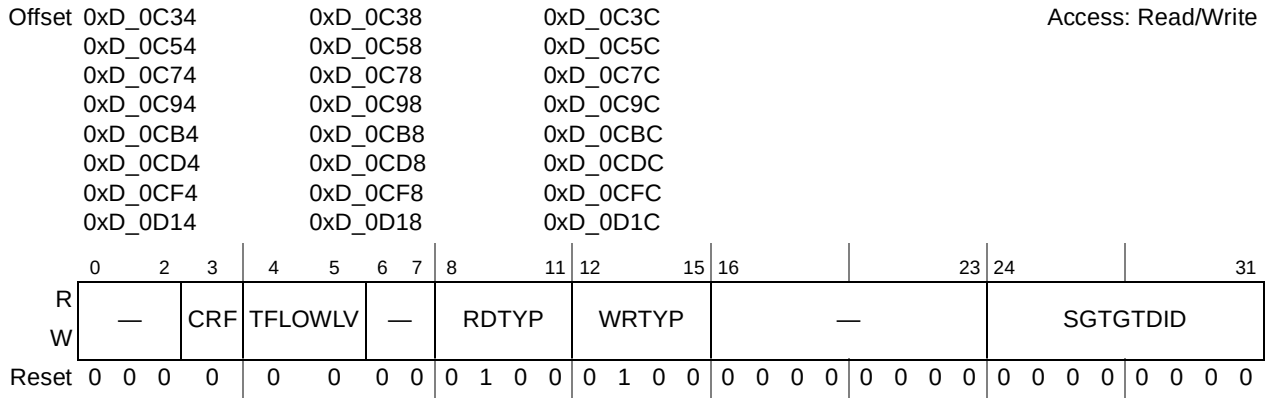


Figure 20-59. RapidIO Outbound Window Segment 1–3 Registers 1–8 (ROWS*n*R*n*)

Table 20-57. ROWS*n*R*n* Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5	TFLOWLV	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved This field must be set to 00 if the PCI_Window bit is set
6–7	—	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 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved ... 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15	WRTYP	Transaction type to run on RapidIO interface if access was a write. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved ... 1110 Reserved 1111 Reserved 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 will generate a NWRITE_R instead.
16–23	—	Reserved
24–28	SGTGTDID	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29	SGTGTDID	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30	SGTGTDID	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	SGTGTDID	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.

20.6.7.7 RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

The 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.

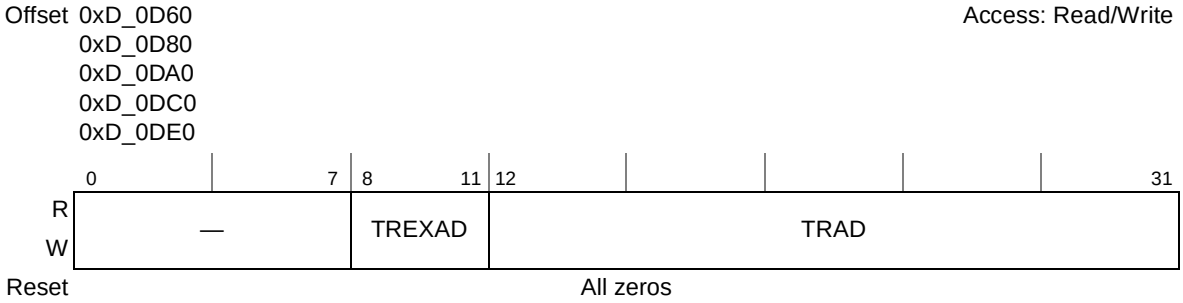


Figure 20-60. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

Table 20-58. RIWTAR*n* Field Descriptions

Bits	Name	Description
0–7	—	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.

20.6.7.8 RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

The 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, please see Section 20.8.6.2, “Window Boundary Crossing Errors.”

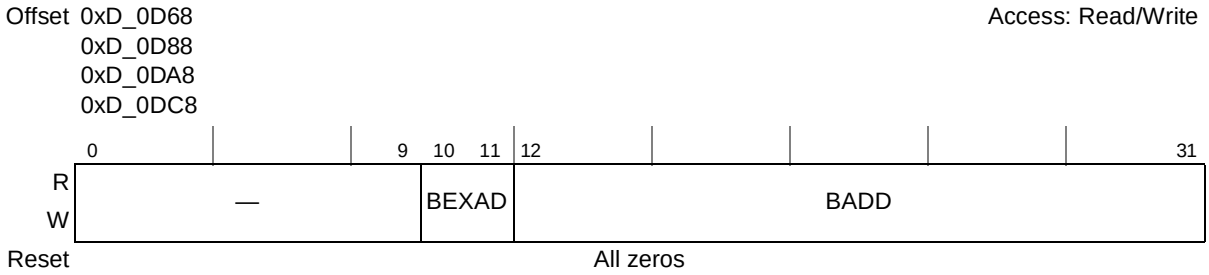


Figure 20-61. RapidIO Inbound Window Base Address Registers1–4

Table 20-59. RIWBAR*n* Field Descriptions

Bits	Name	Description
0–9	—	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.

20.6.7.9 RapidIO Inbound Window Attributes Registers 0–4 (RIWAR*n*)

The 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, 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.

Offset 0xD_0D70 0xD_0D90 0xD_0DB0 0xD_0DD0 Access: Read/Write

	0	1	2	7	8	11	12	15	16	19	20	25	26	31
R/W	EN	PW	—		TGINT		RDTYP		WRTYP		—		SIZE	
Reset	0	0	0	0 0 0 0 0 0 0	0	0 0 0	0	1 0 0	0	1 0 0	0 0 0 0 0 0		1 0	0 0 0 1

Table 20-60. RapidIO Inbound Window Attributes Register 1–4

Offset 0xD_0DF0 Access: Mixed

	0	1	2	7	8	11	12	15	16	19	20	25	26	31
R	EN	PW	—		TGINT		RDTYP		WRTYP		—		SIZE	
W														
Reset	1	0	0	0 0 0 0 0 0 0	0	0 0 0	0	1 0 0	0	1 0 0	0 0 0 0 0 0		1 0	0 0 0 1

Table 20-61. RapidIO Inbound Window Attributes Register 0

Table 20-62. RIWAR*n* Field Descriptions

Bits	Name	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 will generate an error response. Writes not requiring response are silently discarded.
2–7	—	Reserved

Table 20-62. RIWAR*n* Field Descriptions (continued)

Bits	Name	Description
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. This is the field definition for the MPC8572E: 0000 PCI Express 3 0001 PCI Express 2 0010 PCI Express 1 0011–1110 Reserved 1111 Local memory space
12–15	RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ... 0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, unlock L2 cache line 1000 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, don't snoop local processor 0101 Write, snoop local processor 0110 Write, allocate cache line 0111 Write, allocate and lock cache line 1000 Reserved ... 1111 Reserved
20–25	—	Reserved

Table 20-62. RIWAR*n* Field Descriptions (continued)

Bits	Name	Description
26–31	SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4K bytes. 000000 Reserved ... 001011 4K window size 001100 8K window size ... 011111 4G window size 100000 8G window size 100001 16G window size 100010 Reserved ... 111111 Reserved This field is read only for default window 0.

20.7 RapidIO Message Unit Registers

20.7.1 RapidIO Outbound Message 0 Registers

The registers in this section control RapidIO outbound messages. The following provide partial descriptions of these registers.

20.7.1.1 Outbound Message *n* Mode Registers (OM*n*MR)

The outbound message mode register allows software to start a message operation and to control various message operation characteristics.

Offset Msg unit 0: 0xD_3000
Msg unit 1: 0xD_3100

Access: Read/Write

	0–3	4–6	7–10	11	12–15	16–19	20–21	22	23	24	25	26	27	28–29	30	31
R W	—	SCTL	—	DES_SEN	—	CIRQ_SIZ	—	QOIE	QFIE	—	QEIE	EIE	—	MUTM	MUI	MUS
Reset	All zeros															

Figure 20-62. Outbound Message *n* Mode Registers (OM*n*MR)

Table 20-63. OM*n*MR Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–6	SCTL	Service control. Determines the number of descriptors to process before servicing the next queue. 0b000 Fixed priority based on outbound message unit number 0b001 1 descriptors 0b010 2 descriptors 0b011 4 descriptors 0b100 8 descriptors 0b101 16 descriptors 0b110 32 descriptors 0b111 64 descriptors 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.
7–10	—	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
12–14	—	Reserved
15	—	Reserved
16–19	CIRQ_SIZ	Circular descriptor queue size. Determines the number of descriptors that can be placed on the circular queue without overflow. 0b0000 2 — 0b1000 512 0b0001 4 — 0b1001 1024 0b0010 8 — 0b1010 2048 0b0011 16 — 0b1011 Reserved 0b0100 32 — 0b1100 Reserved 0b0101 64 — 0b1101 Reserved 0b0110 128 — 0b1110 Reserved 0b0111 256 — 0b1111 Reserved For proper operation, this field should only be modified when the outbound message controller is not enabled
20–21	—	Reserved
22	QOIE	Queue overflow interrupt enable. When set will generate 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	—	Reserved

Table 20-63. OM*n*MR Field Descriptions (continued)

Bits	Name	Description
25	QEIE	Queue empty interrupt enable. When set will generate 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 will generate 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	—	Reserved
29	MUTM	Message unit transfer mode. Setting this bit will put 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 will configure the message unit in chaining mode.
30	MUI	Message unit increment. Software will set this bit after writing a descriptor to memory. Hardware will then increment the OM*n*DQEPAR 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) will start the message unit. A 1 to 0 transition will have no effect. Chaining mode—If this bit is set the message unit will start whenever the enqueue and dequeue pointers are not equal.

20.7.1.2 Outbound Message *n* Status Registers (OM*n*SR)

The outbound message status register reports various message unit conditions during and after a message operation. Writing a 1 to the corresponding set bit will clear the bit.

Offset Msg unit 0: 0xD_3004 Access: Mixed
Msg unit 1: 0xD_3104

	0									10	11	12							18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—										QF	—								MER	RETE	PRT	—		TE	—	QOI	QFI	—	MUB	EOMI	QEI
W																				w1c	w1c	w1c			w1c		w1c	w1c			w1c	w1c
Reset	All zeros																														

Figure 20-63. Outbound Message *n* Status Registers (OM*n*SR)

Table 20-64. OM*n*SR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full - If the queue becomes full, then this bit is set. (Read-only)
12–18	—	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.

Table 20-64. OM*n*SR Field Descriptions (continued)

Bits	Name	Description
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	—	Reserved
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	—	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	—	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.

20.7.1.3 Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR) and Outbound Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)

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 number of queue entries × 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 Section 20.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

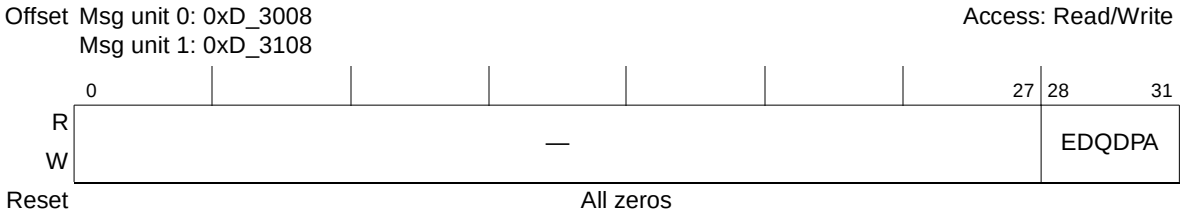


Figure 20-64. Extended Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (EOM*n*DQDPAR)

Table 20-65. EOM*n*DQDPAR Field Descriptions

Bits	Name	Description
0–27	—	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.

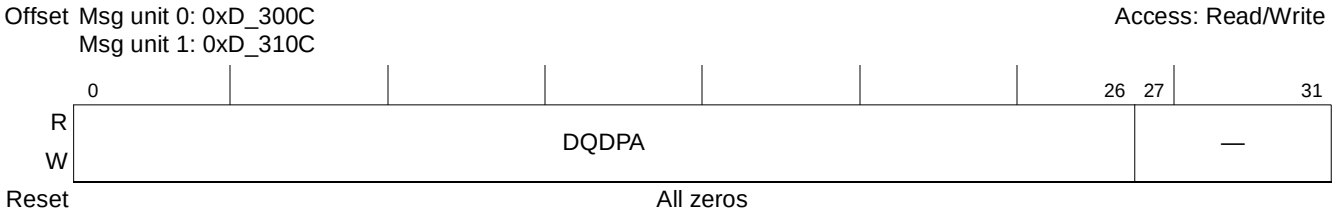


Figure 20-65. Outbound Message *n* Descriptor Queue Dequeue Pointer Address Registers (OM*n*DQDPAR)

Table 20-66. OM*n*DQDPAR Field Descriptions

Bits	Name	Description
0–26	DQDPA	Descriptor queue dequeue pointer address. Contains the address of the first descriptor in memory to process. The descriptor must be aligned to a 32-byte boundary. For proper operation, this field should only be modified when the outbound message controller is not enabled.
27–31	—	Reserved

20.7.1.4 Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (EOM*n*DQEPAR) and Outbound Message *n* Descriptor Queue Enqueue Pointer Address Registers (OM*n*DQEPAR)

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*DQEPAR and EOM*n*DQEPAR), and then either writes the OM*n*DQEPAR 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 number of queue entries × 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 Section 20.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).

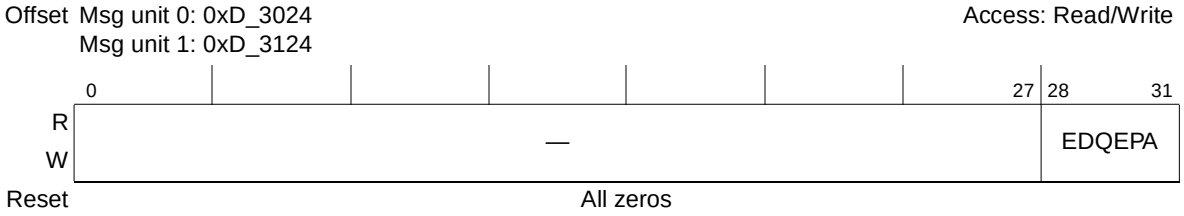


Figure 20-66. Extended Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (EOM*n*DQEPAR)

Table 20-67. EOM*n*DQEPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQEPA	Extended descriptor queue enqueue pointer address. These are the highest-order address bits.

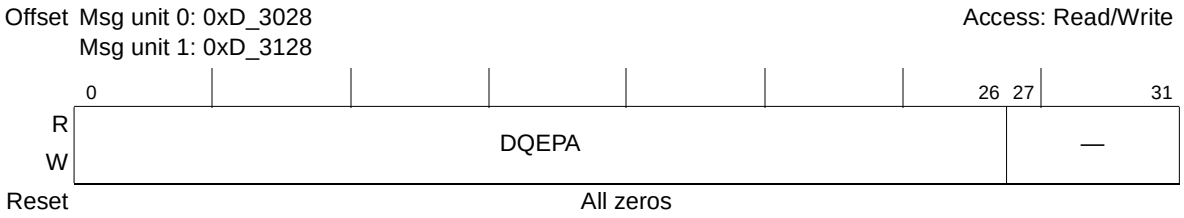


Figure 20-67. Outbound Message *n* Descriptor Queue Enqueue Pointer Registers (OM*n*DQEPAR)

Table 20-68. OM*n*DQEPAR Field Descriptions

Bits	Name	Description
0–26	DQEPA	Descriptor queue enqueue pointer address. Contains the address of the last descriptor in memory to process. The descriptor must be aligned to a 32 byte boundary and a descriptor queue boundary.
27–31	—	Reserved

20.7.1.5 Extended Outbound Message *n* Source Address Registers (EOM*n*SAR) and Outbound Message *n* Source Address Registers (OM*n*SAR)

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.

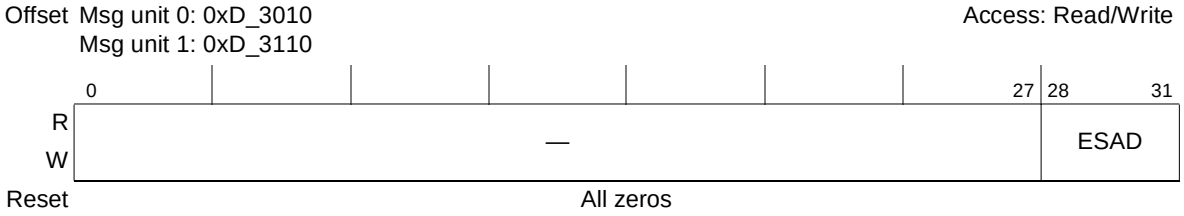


Figure 20-68. Extended Outbound Message *n* Source Address Registers (EOM*n*SAR)

Table 20-69. EOM*n*SAR Field Descriptions

Bits	Name	Description
0–27	—	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

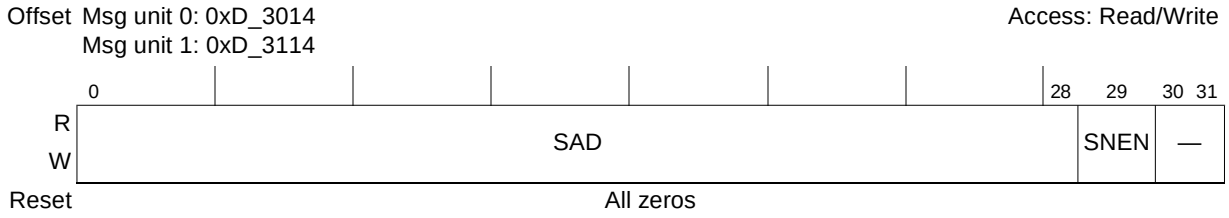


Figure 20-69. Outbound Message *n* Source Address Registers (OM*n*SAR)

Table 20-70. OM*n*SAR Field Descriptions

Bits	Name	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	—	Reserved

20.7.1.6 Outbound Message *n* Destination Port Registers (OM*n*DPR)

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.

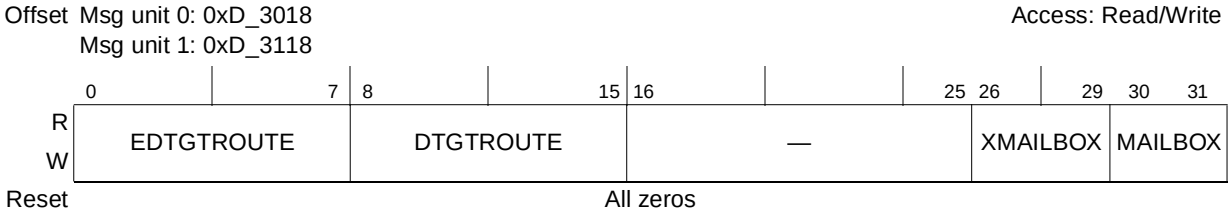


Figure 20-70. Outbound Message *n* Destination Port Registers (OM*n*DPR)

Table 20-71. OM*n*DPR Field Descriptions

Bits	Name	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	—	Reserved
26–29	XMAILBOX	Value for ‘xmbox’ 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

20.7.1.7 Outbound Message *n* Destination Attributes Registers (OM*n*DATR)

The outbound message destination attributes register contains the transaction attributes to be used for the message operation.

Offset Msg unit 0: 0xD_301C
Msg unit 1: 0xD_311C

Access: Read/Write

	0	1	2	3	4	5	6	7	8	11	12	31
R W	MM	—	EOMIE	CRF	DTFLOWLVL		—		DTGTINT		—	
Reset	All zeros											

Figure 20-71. Outbound Message *n* Destination Attributes Registers (OM*n*DATR)

Table 20-72. OM*n*DATR Field Descriptions

Bits	Name	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	—	Reserved
2	EOMIE	End-of-Message interrupt enable. When set will generate 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	CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5	DTFLOWLVL	Transaction flow level. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority transaction request flow 11 Reserved For proper operation, this field should only be modified when the outbound message controller is not enabled
6–7	—	Reserved
8–11	DTGTINT	Target interface. The value of this field should always be set to RapidIO (0x0).
12–31	—	Reserved

20.7.1.8 Outbound Message *n* Double-word Count Registers (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.

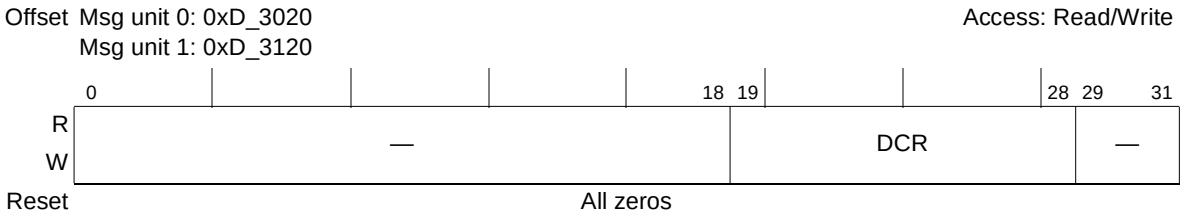


Figure 20-72. Outbound Message *n* Double Word Count Registers (OM*n*DCR)

Table 20-73. OM*n*DCR Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–28	DCR	Transfer count register. Contains the number of bytes for the message operation. 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 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) All other values yield undefined behavior. For proper operation, this field should only be modified when the outbound message controller is not enabled.
29–31	—	Reserved

20.7.1.9 Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)

The outbound message retry error threshold configuration register controls the number of times that the message unit will attempt 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.

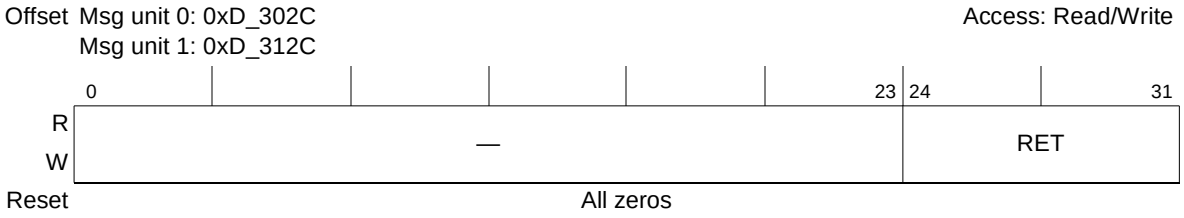


Figure 20-73. Outbound Message *n* Retry Error Threshold Configuration Registers (OM*n*RETCR)

Table 20-74. OM*n*RETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. This value is the number of times that the message unit will attempt to transmit a message segment to a particular target. 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 For proper operation, this field should only be modified when the outbound message controller is not enabled

20.7.1.10 Outbound Message *n* Multicast Group Registers (OM*n*MGR)

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), etc.

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.

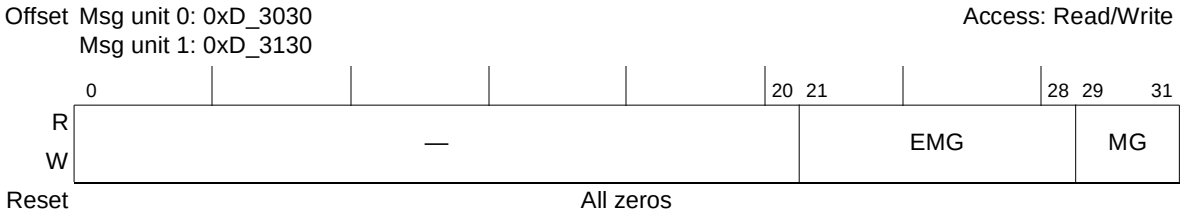


Figure 20-74. Outbound Message *n* Multicast Group Registers (OM*n*MGR)

Table 20-75. OM*n*MGR Field Descriptions

Bits	Name	Description
0–20	—	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

20.7.1.11 Outbound Message *n* Multicast List Registers (OM*n*MLR)

See the multicast group register description for more information.

Figure 20-75. Outbound Message *n* Multicast List Registers (OM*n*MLR)

Table 20-76. OM*n*MLR Field Descriptions

Bits	Name	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 deviceID 0, 32, 64, 96, etc., bit 1 corresponds to deviceID 1, 33, 65, 97, etc. 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.

20.7.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

20.7.2.1 Inbound Message *n* Mode Registers (IM*n*MR)

The inbound message mode register allows software to enable the mailbox controller and to control various message operation characteristics.

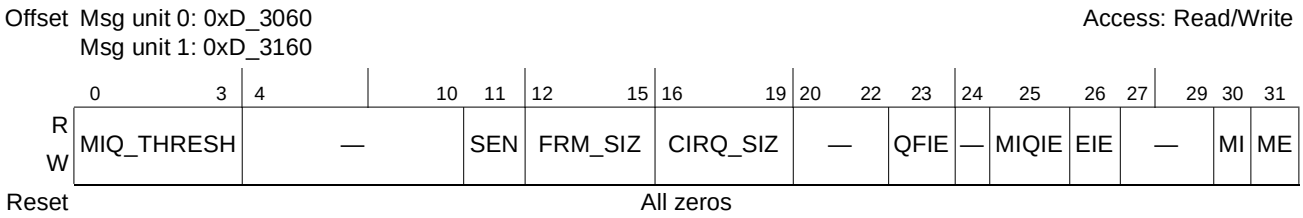


Figure 20-76. Inbound Message *n* Mode Registers (IM*n*MR)

Table 20-77. IM*n*MR Field Descriptions

Bits	Name	Description
0–3	MIQ_THRESH	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]). 0000 1 0111 128 0001 2 1000 256 0010 4 1001 512 0011 8 1010 1024 0100 16 1011 Reserved 0101 32 ... 0110 64 1111 Reserved For proper operation, this field should only be modified when the inbound message controller is not enabled.
4–10	—	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	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. 0000 Reserved 1000 512 bytes ... 1001 1024 bytes 0010 8 bytes 1010 2048 bytes 0011 16 bytes 1011 4096 bytes 0100 32 bytes 1100 Reserved 0101 64 bytes ... 0110 128 bytes 1111 Reserved 0111 256 bytes For proper operation, this field should only be modified when the inbound message controller is not enabled.
16–19	CIRQ_SIZ	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. 0000 2 0111 256 0001 4 1000 512 0010 8 1001 1024 0011 16 1010 2048 0100 32 1011 Reserved 0101 64 ... 0110 128 1111 Reserved For proper operation, this field should only be modified when the inbound message controller is not enabled.
20–22	—	Reserved
23	QFIE	Queue full interrupt enable. When set will generate 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 if this bit is cleared. If this bit is set and IM*n*SR[QF] is a 1, IM*n*SR[QFI] will assert.
24	—	Reserved

Table 20-77. IM*n*MR Field Descriptions (continued)

Bits	Name	Description
25	MIQIE	Message in queue interrupt enable. When set will generate 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] will assert. 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 will generate 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	—	Reserved
30	MI	Mailbox increment. Software will set this bit after processing an inbound message. Hardware will increment the IM*n*FQDPAR and clear this bit. Always reads as 0.
31	ME	Mailbox enable. If this bit is set the mailbox has been 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 will result (IM*n*SR[MRT]) and the busy bit (IM*n*SR[MB]) will clear if the port response timer value (PRTOCCSR[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.

20.7.2.2 Inbound Message *n* Status Registers (IM*n*SR)

The inbound message status register reports various mailbox conditions during and after a message operation. Writing a 1 to the corresponding set bit will clear the bit.

Offset Msg unit 0: 0xD_3064
Msg unit 1: 0xD_3164

Access: Mixed

	0										10	11	12		14	15	16				20	21	22	23	24		25	26	27	28	29	30	31
R	—											QF	—			MIQ	—					MRT	—		TE		—		QFI	—		MB	QE	MIQI
W																						w1c			w1c				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	1	0

Figure 20-77. Inbound Message *n* Status Registers (IM*n*SR)

Table 20-78. IM*n*SR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full. If the queue becomes full, then this bit is set. This bit will also be cleared if the message controller is disabled. (Read-only)
12–14	—	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 is set. This bit will also be cleared if the message controller is disabled. (Read-only)
16–20	—	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	—	Reserved

Table 20-78. IM*n*SR Field Descriptions (continued)

Bits	Name	Description
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	—	Reserved
27	QFI	Queue full interrupt. If the queue is 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	—	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)
30	QE	Queue empty. If the queue is empty, then this bit is set. This bit will also be set if 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], 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.

20.7.2.3 Extended Inbound Message Frame Queue Dequeue Pointer Address Registers (EIM*n*FQDPAR) and Inbound Message Frame Queue Dequeue Pointer Address Registers (IM*n*FQDPAR)

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 number of queue entries × 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 Section 20.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

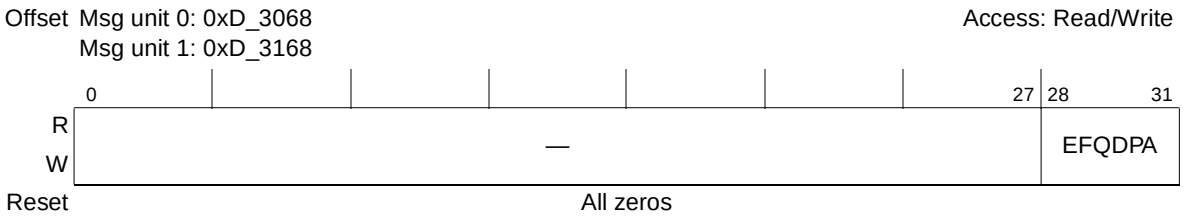


Figure 20-78. Extended Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (EIM*n*FQDPAR)

Table 20-79. EIM*n*FQDPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EFQDPA	Extended frame queue dequeue pointer address bits. These are the highest order address bits.

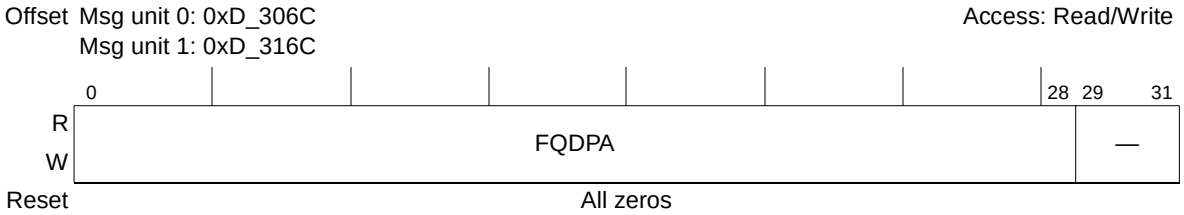


Figure 20-79. Inbound Message *n* Frame Queue Dequeue Pointer Address Registers (IM*n*FQDPAR)

Table 20-80. IM*n*FQDPAR Field Descriptions

Bits	Name	Description
0–28	FQDPA	Frame queue dequeue pointer address. Contains the address of the first message in memory to process.
29–31	—	Reserved

20.7.2.4 Extended Inbound Message Frame Queue Enqueue Pointer Address Registers (EIM*n*FQEPAR) and Inbound Message Frame Queue Enqueue Pointer Address Registers (IM*n*FQEPAR)

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*FQEPAR and EIM*n*FQEPAR) and then increments IM*n*FQEPAR and EIM*n*FQEPAR 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 number of queue entries × 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 Section 20.7.2.1, “Inbound Message n Mode Registers (IMnMR)”).

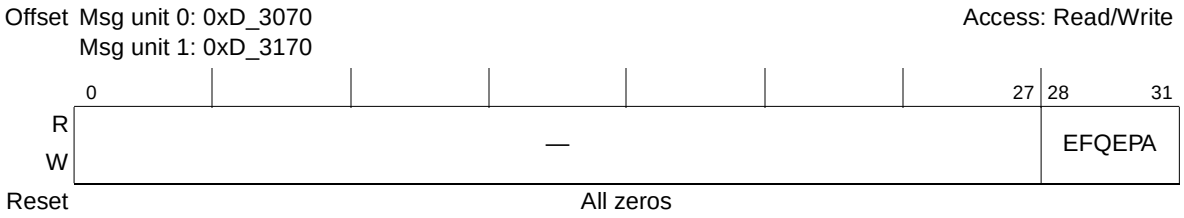


Figure 20-80. Extended Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (EIM*n*FQEPAR)

Table 20-81. EIM*n*FQEPAR Field Descriptions

Bits	Name	Description
0–27	—	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.

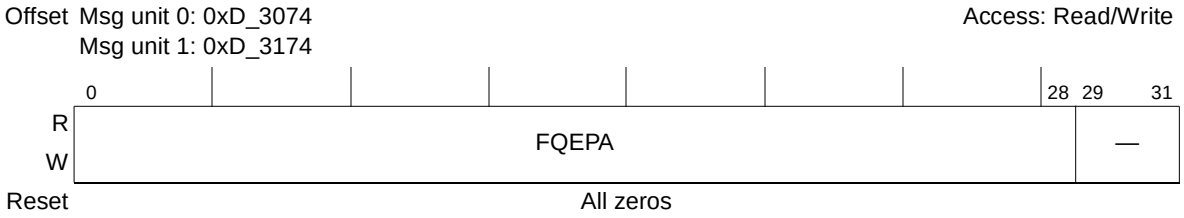


Figure 20-81. Inbound Message *n* Frame Queue Enqueue Pointer Address Registers (IM*n*FQEPAR)

Table 20-82. IM*n*FQEPAR Field Descriptions

Bits	Name	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	—	Reserved

20.7.2.5 Inbound Message *n* Maximum Interrupt Report Interval Registers (IM*n*MIRIR)

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 value is the maximum time-out interval. The timer decrements at one-half the platform clock rate; thus at a platform clock rate of 400 MHz, for example, the maximum time-out value is approximately 83.9 ms.

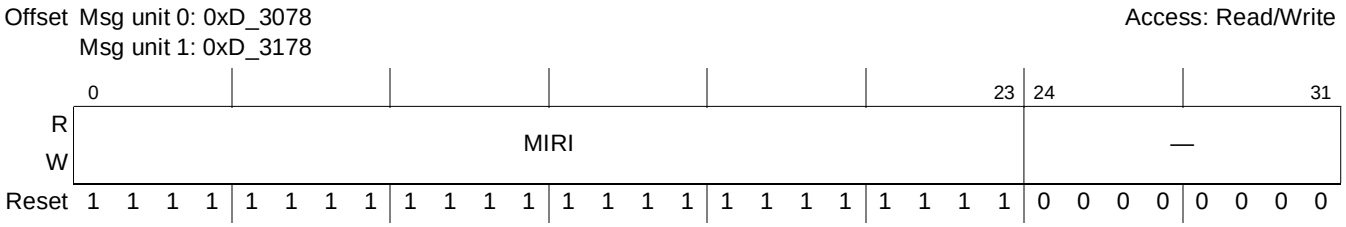


Figure 20-82. Inbound Message *n* Maximum Interrupt Report Interval Registers (IM*n*MIRIR)

Table 20-83. IM*n*MIRIR Field Descriptions

Bits	Name	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	—	Reserved

20.7.3 Outbound RapidIO Doorbell Controller Registers

These registers control the outbound RapidIO doorbell controller. The following sections provide descriptions of these registers.

20.7.3.1 Outbound Doorbell Mode Register (ODMR)

The outbound mode register allows software to start a doorbell operation and to control various doorbell operation characteristics.

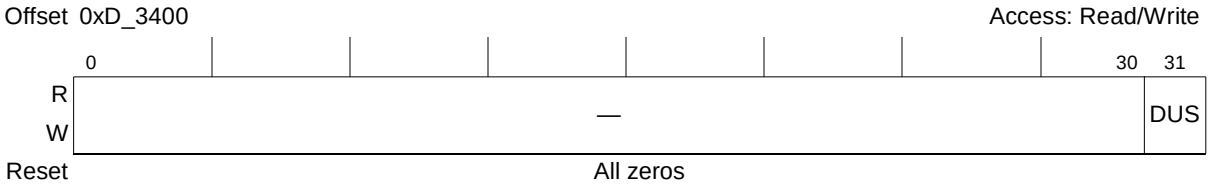


Figure 20-83. Outbound Mode Register (ODMR)

Table 20-84. ODMR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	DUS	Doorbell unit start. Direct mode - A 0 to 1 transition when the doorbell unit is not busy (DUB bit is 0) will start the doorbell unit. A 1 to 0 transition will have no effect.

20.7.3.2 Outbound Doorbell Status Register (ODSR)

The outbound status register reports various doorbell unit conditions during and after a doorbell operation. For some bits, writing a 1 to the bit will clear it.

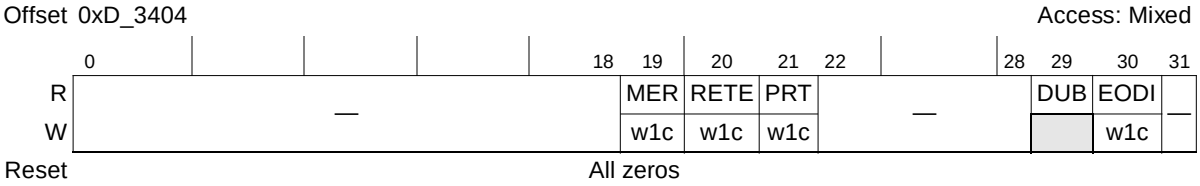


Figure 20-84. Outbound Doorbell Status Register (ODSR)

Table 20-85. ODSR Field Descriptions

Bits	Name	Description
0–18	—	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	—	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)
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 is 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	—	Reserved

20.7.3.3 Outbound Doorbell Destination Port Register (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.

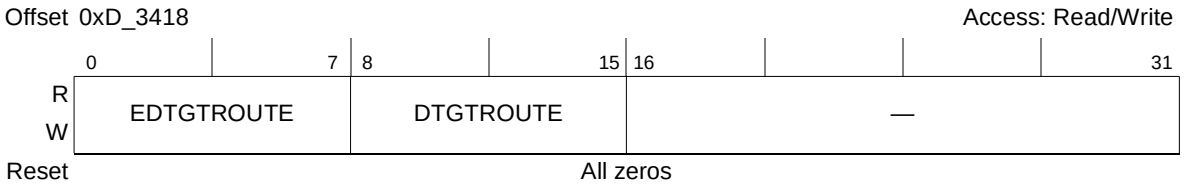


Figure 20-85. Outbound Doorbell Destination Port Registers (ODDPR)

Table 20-86. ODDPR Field Descriptions

Bits	Name	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 a doorbell operation is not in progress.
8–15	DTGTROUTE	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	—	Reserved

20.7.3.4 Outbound Doorbell Destination Attributes Register (ODDATR)

The outbound destination attributes register contains the transaction attributes to be used for the doorbell operation.

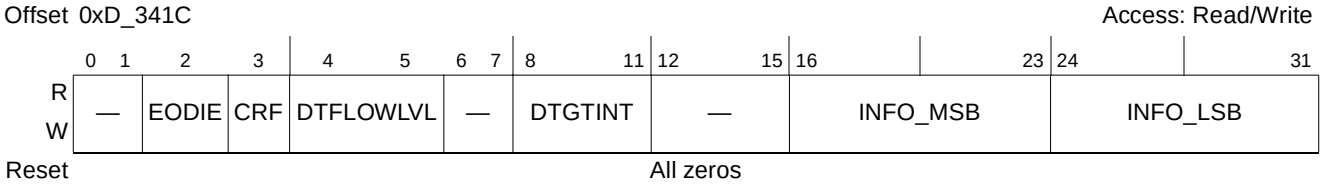


Figure 20-86. Outbound Doorbell Destination Attributes Register (ODDATR)

Table 20-87. ODDATR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	—	Reserved, hard wired to 0
2	EODIE	End-of-Doorbell interrupt enable. When set will generate 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	CRF	Critical request flow. 0 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a switch, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5	DTFLOWLVL	Transaction flow level. 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved For proper operation, this field should only be modified when a doorbell operation is not in progress.
6–7	—	Reserved
8–11	DTGTINT	Target interface. The value of this field should always be set to RapidIO (0x0).

Table 20-87. ODDATR Field Descriptions (continued)

Bits	Name	Description
12–15	—	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.

20.7.3.5 Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

The retry error threshold configuration register controls the number of times that the doorbell unit will attempt 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.

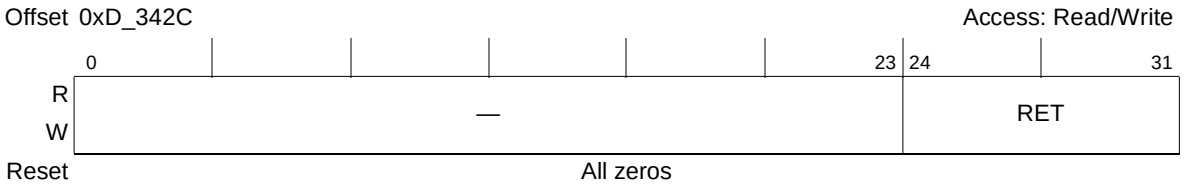


Figure 20-87. Outbound Doorbell Retry Error Threshold Configuration Register (ODRETCR)

Table 20-88. ODRETCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	RET	Retry error threshold. This value is the number of times that the doorbell unit will attempt to transmit a doorbell. 0x00 Disabled 0x01 Doorbell transmitted only 1 time 0x02 Doorbell transmitted up to 2 times ... 0xFF Doorbell transmitted up to 255 times For proper operation, this field should only be modified when a doorbell operation is not in progress.

20.7.4 Inbound RapidIO Doorbell Controller

These registers control the inbound RapidIO doorbell controller. The following sections provide descriptions of these registers.

20.7.4.1 Inbound Doorbell Mode Register (IDMR)

The doorbell mode register allows software to enable the doorbell controller to control various doorbell operation characteristics. The IDMR is shown in Figure 20-88.

Offset 0xD_3460 Access: Read/Write

	0–3	4–10	11	12–15	16–19	20–22	23	24	25	26	27–29	30	31
R W	DIQ_THRESH	—	SEN	—	CIRQ_SIZ	—	QFIE	—	DIQIE	EIE	—	DI	DE
Reset	All zeros												

Figure 20-88. Inbound Doorbell Mode Register (IDMR)

Table 20-89. IDMR Field Descriptions

Bits	Name	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. 0000 1 0111 128 0001 2 1000 256 0010 4 1001 512 0011 8 1010 1024 0100 16 1011 reserved 0101 32 ... 0110 64 1111 reserved For proper operation, this field should only be modified when the doorbell controller is not enabled.
4–10	—	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	—	Reserved
16–19	CIRQ_SIZ	Circular doorbell queue size. Determines the number of doorbell entries that can be place on the circular queue. CIRQ_SIZ × 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. 0000 2 0111 256 0001 4 1000 512 (4-kbyte page boundary) 0010 8 1001 1024 0011 16 1010 2048 0100 32 1011 reserved 0101 64 ... 0110 128 1111 reserved For proper operation, this field should only be modified when the doorbell controller is not enabled.
20–22	—	Reserved
23	QFIE	Queue full interrupt enable. 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). If this bit is set and IDSR[QF] is a 1, IDSR[QFI] will assert.
24	—	Reserved

Table 20-89. IDMR Field Descriptions (continued)

Bits	Name	Description
25	DIQIE	Doorbell in queue interrupt enable. When set will allow 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. 0 No DIQ interrupt is generated 1 Generates an interrupt when the DIQ bit is set (IDSR[DIQ]) If this bit is set and IDSR[DIQ] is a 1, IDSR[DIQI] will assert. If this bit is set and IDMR[DI] is also set simultaneously, IDSR[DIQI] indicates reflects the value of DIQ after the increment.
26	EIE	Error interrupt enable. When set will generate 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	—	Reserved
30	DI	Doorbell increment. Software sets this bit after processing an inbound doorbell. Hardware then increments the ODQEPAR 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.

20.7.4.2 Inbound Doorbell Status Register (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. The IDSR is shown in Figure 20-89.

Offset 0xD_3464 Access: Mixed

	0										10	11	12			14	15	16							23	24	25	26	27	28	29	30	31
R	—											QF	—				DIQ	—								TE	—		QFI	—	DB	QE	DIQI
W																										w1c			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	1	0

Figure 20-89. Inbound Doorbell Status Register (IDSR)

Table 20-90. IDSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full. If the queue is full, then this bit is set. This bit will also be cleared if the doorbell controller is disabled. (Read-only)
12–14	—	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 will also be cleared if the doorbell controller is disabled. (Read-only)
16–23	—	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.

Table 20-90. IDSR Field Descriptions (continued)

Bits	Name	Description
25–26	—	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	—	Reserved
29	DB	Doorbell busy. When set indicates that a doorbell has been received and the doorbell queue is currently being written. This bit is cleared 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 will also be 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.

20.7.4.3 Extended Inbound Doorbell Queue Dequeue Pointer Address Register (EIDQDPAR) and Inbound Doorbell Queue Dequeue Pointer Address Register (IDQDPAR)

The inbound doorbell queue dequeue pointer address registers (EIDQDPAR and IDQDPAR) 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 IDMR[DI]. The doorbell hardware then increments the doorbell queue dequeue pointer address register to point to the next doorbell entry in memory and clears IDMR[DI]. When processing multiple doorbells, instead of setting IDMR[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.

The EIDQDPAR is shown in Figure 20-90 and the IDQDPAR is shown in Figure 20-91.

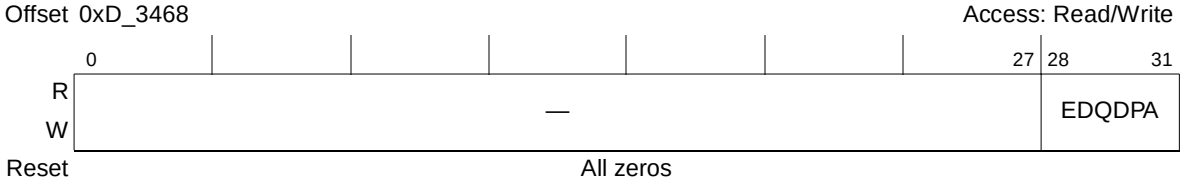


Figure 20-90. Extended Inbound Doorbell Queue Dequeue Pointer Address Registers (EIDQDPAR)

Table 20-91. EIDQDPAR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EDQDPA	Extended doorbell queue dequeue pointer address bits. These are the highest order address bits.

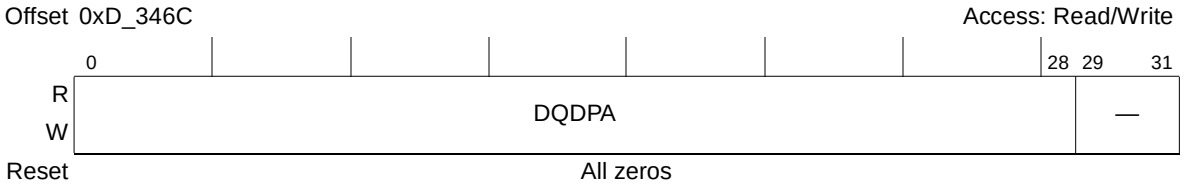


Figure 20-91. Inbound Doorbell Queue Dequeue Pointer Address Registers (IDQDPAR)

Table 20-92. IDQDPAR Field Descriptions

Bits	Name	Description
0–28	DQDPA	Doorbell queue dequeue pointer address. Contains the double-word address of the first doorbell in memory to process.
29–31	—	Reserved

20.7.4.4 Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR) and Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)

The doorbell queue enqueue pointer address registers (EIDQEPAR and IDQEPAR) contain the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize IDQEPAR and EIDQEPAR 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 IDQEPAR and EIDQEPAR) and then increments IDQEPAR and EIDQEPAR 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 will not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If the IDMR[QFIE] bit is set, then the IDSR[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 IDSR[DIQ] bit is set. If the DRM[DIQIE] bit is set, then the IDSR[DIQI] bit is also set and the inbound doorbell interrupt is generated.

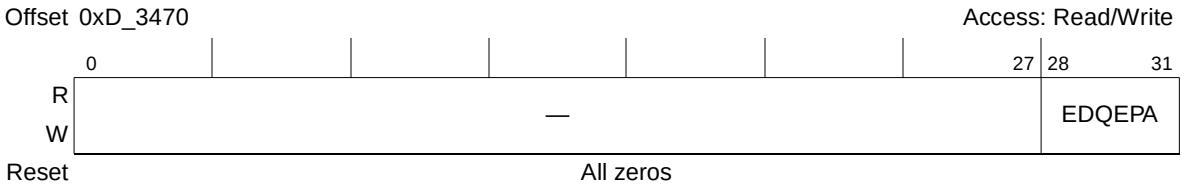


Figure 20-92. Extended Inbound Doorbell Queue Enqueue Pointer Address Register (EIDQEPAR)

Table 20-93. EIDQEPAR Field Descriptions

Bits	Name	Description
0–27	—	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 will result.

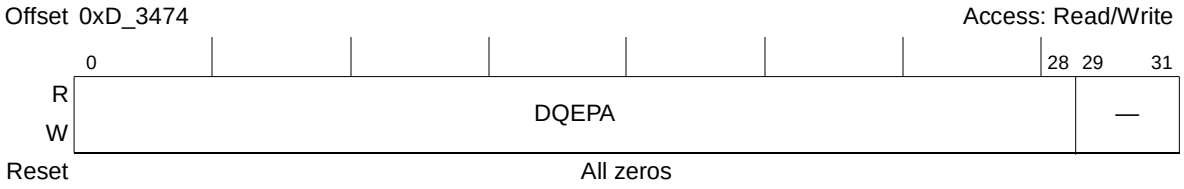


Figure 20-93. Inbound Doorbell Queue Enqueue Pointer Address Register (IDQEPAR)

Table 20-94. IDQEPAR Field Descriptions

Bits	Name	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 will result.
29–31	—	Reserved

20.7.4.5 Inbound Doorbell Maximum Interrupt Report Interval Register (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 value is the maximum time-out interval, and represents between 3 and 6 seconds.

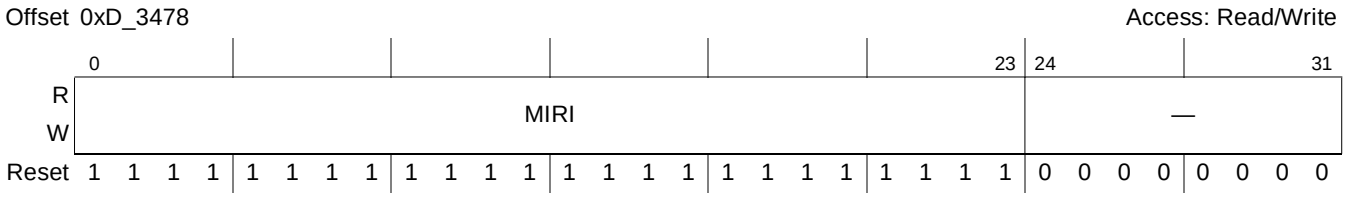


Figure 20-94. Inbound Doorbell Maximum Interrupt Report Interval Register (IDMIRIR)

Table 20-95. IDMIRIR Field Descriptions

Bits	Name	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 will result.
24–31	—	Reserved

20.7.5 RapidIO Port-Write Registers

These registers control the RapidIO port-write unit. The following sections provide partial descriptions of these registers.

20.7.5.1 Inbound Port-Write Mode Register (IPWMR)

The port-write mode register allows software to enable the port-write controller and to control various port-write operation characteristics. The IPWMR is shown in Figure 20-95.

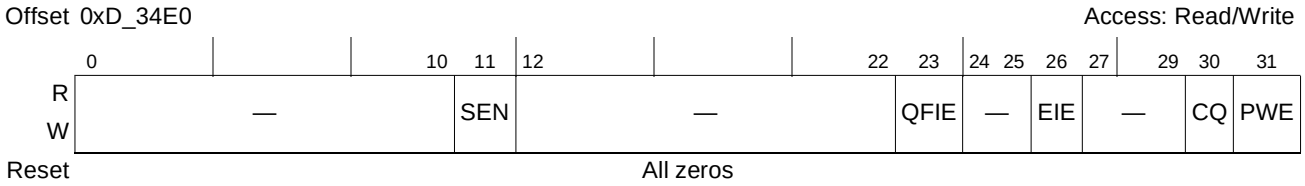


Figure 20-95. Inbound Port-Write Mode Register (IPWMR)

Table 20-96. IPWMR Field Descriptions

Bits	Name	Description
0–10	—	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	—	Reserved
23	QFIE	Queue full interrupt enable. When set will generate the error/port-write interrupt when 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 if this bit is cleared. For proper operation, this field should only be modified when the port-write controller is not enabled.
24–25	—	Reserved
26	EIE	Error Interrupt enable. When set will generate 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	—	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.

20.7.5.2 Inbound Port-Write Status Register (IPWSR)

The port-write status register reports various port-write conditions. The IPWSR is shown in Figure 20-96

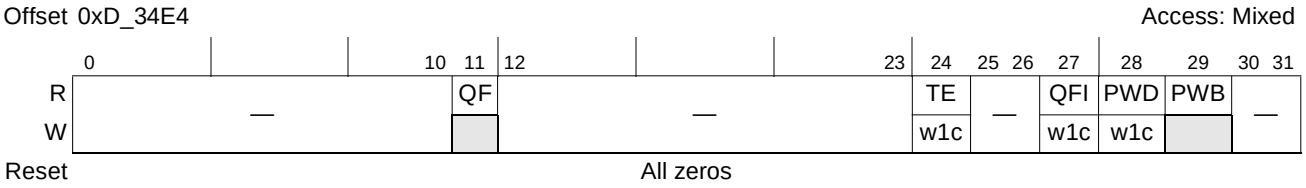


Figure 20-96. Inbound Port-Write Status Register (IPWSR)

Table 20-97. IPWSR Field Descriptions

Bits	Name	Description
0–10	—	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[CQ]) and the queue is not full. This bit will also be cleared if the port-write controller is disabled. (Read-only)
12–23	—	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	—	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). 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	—	Reserved

20.7.5.3 Extended Inbound Port-Write Queue Base Address Register (EIPWQBAR) and Inbound Port-Write Queue Base Address Register (IPWQBAR)

The port-write queue base address registers (EIPWQBAR and IPWQBAR) contain the 64-byte cache line address for the port-write data payload. Software must initialize these registers to the desired location in memory. The EIPWQBAR and IPWQBAR are shown in Figure 20-97 and Figure 20-98.

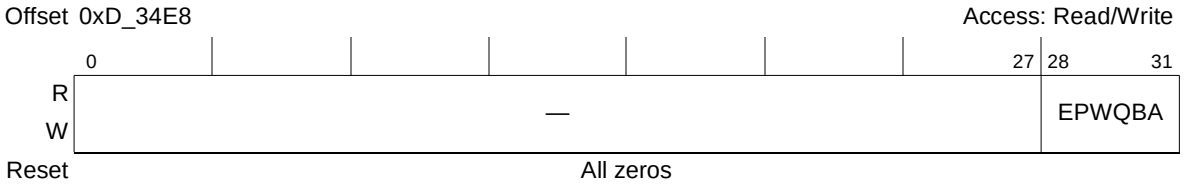


Figure 20-97. Extended Port-Write Queue Base Address Register (EIPWQBAR)

Table 20-98. EIPWQBAR Field Descriptions

Bits	Name	Description
0–27	—	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 will result.

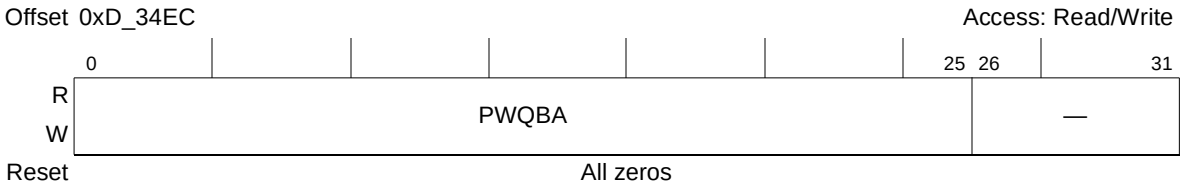


Figure 20-98. Inbound Port-Write Queue Base Address Register (IPWQBAR)

Table 20-99. IPWQBAR Field Descriptions

Bits	Name	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 will result.
26–31	—	Reserved

20.8 Functional Description

20.8.1 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 20-100 summarizes the RapidIO I/O transactions that are supported by this RapidIO endpoint.

Table 20-100. 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
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 20-101 summarizes the RapidIO message passing transactions that are supported by this RapidIO endpoint.

Table 20-101. 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 20-102 summarizes the RapidIO GSM transactions that are supported by this RapidIO endpoint.

Table 20-102. RapidIO GSM Transactions

GSM Transaction	ftype	ttype	status	Description
IO_READ_HOME	0010	0010	NA	I/O Read Home[1]
FLUSH w/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
RESPONSE with data		1000	0000	GSM done response
			0001	GSM data only response

1). Limited to RapidIO packet generation only

20.8.2 RapidIO Packet Format Summary

Table 20-103 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 20-103. RapidIO Small Transport Field Packet Format

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD, ATOMIC (inc/dec/inc/dec), IO_READ_HOME	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wd ptr	xam bs	NA
NWRITE_R, FLUSH w/data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wd ptr	xam bs	dword 0 -> dword n
NWRITE	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wd ptr	xam bs	dword 0 -> dword n
SWRITE	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1)=0, xambs(2)					dword 0 -> dword n			
MAINT read	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rd/wrs ize	src TID	hop cnt	cfg offset		wd ptr	rsv= 0	NA
MAINT write	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rd/wrs ize	src TID	hop cnt	cfg offset		wd ptr	rsv= 0	dword 0 (32-bit)
MAINT port-write	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	rd/wrs ize	rsv=0	hop cnt	rsv=0		wd ptr	rsv= 0	dword 0 -> dword n
MAINT response without data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				NA

Table 20-103. RapidIO Small Transport Field Packet Format (continued)

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
MAINT response with data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				dword 0 (32-bit)
RESPONSE without data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	NA					
RESPONSE without data for message	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	dword 0 -> dword n					
DOORBELL	ack ID	rsv =0	crf	prio	tt	ftype	des tID	src ID	rsv=0		src TID	Info-msb	Info-lsb	NA			
MESSAGE	ack ID	rsv =0	crf	prio	tt	ftype	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.

20.8.3 RapidIO Control Symbol Summary

Table 20-104 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, Chapter 4, 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 20-104. 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 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

Table 20-104. 1x/4x LP-Serial Control Symbol Format (continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
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
-			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)

20.8.4 Accessing Configuration Registers via RapidIO Packets

20.8.4.1 Inbound Maintenance Accesses

There are two suggested methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

The first method is based on RapidIO NREAD and NWRITE_R requests hitting a RapidIO address window defined by the local configuration space base address command and status register (LCSBACSR). See Section 20.6.1.11, “Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR),” for details.

If external configuration accesses are disabled (LLCR[ECRAB] = 1), any configuration access through the LCSBACSR window is denied. 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 (lower 64 Kbytes contain RapidIO architecture registers; upper 64 Kbytes contain 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.

20.8.4.1.1 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.

20.8.4.2 Outbound Maintenance Accesses

Outbound OCN NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the OCN 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.

20.8.5 RapidIO Outbound ATMU

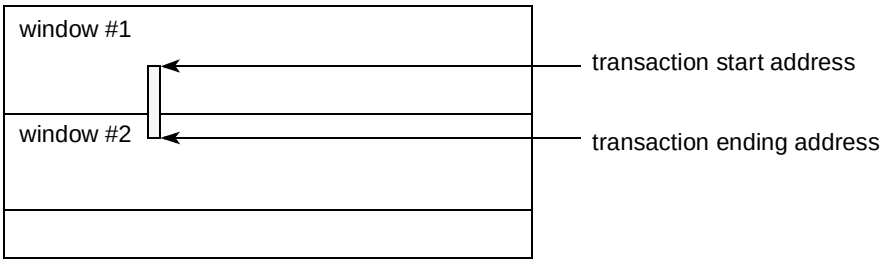
20.8.5.1 Valid Hits to Multiple ATMU Windows

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.

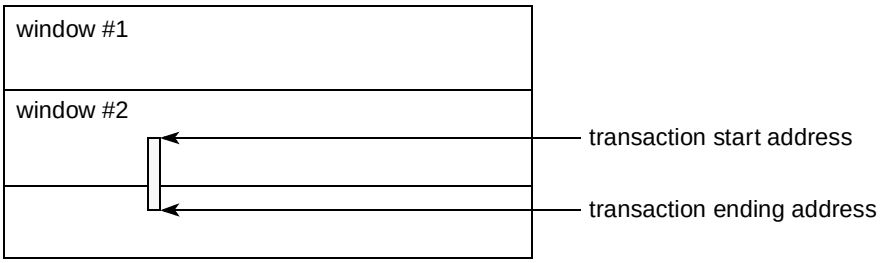
If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

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.



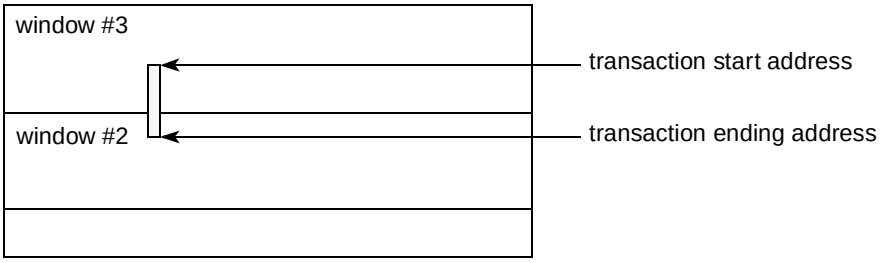
2. If a request hits (base address match) multiple ATMU windows (1–8) and the transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.



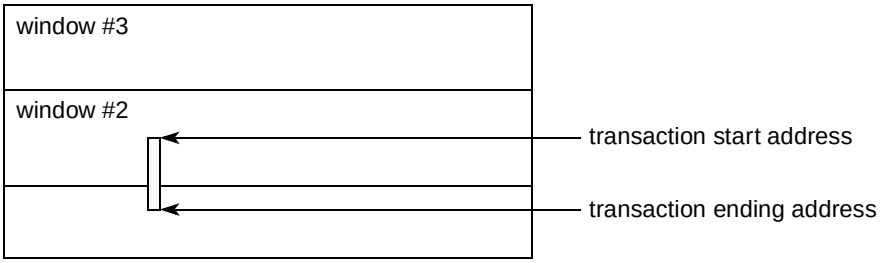
20.8.5.2 Window Boundary Crossing Errors

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.

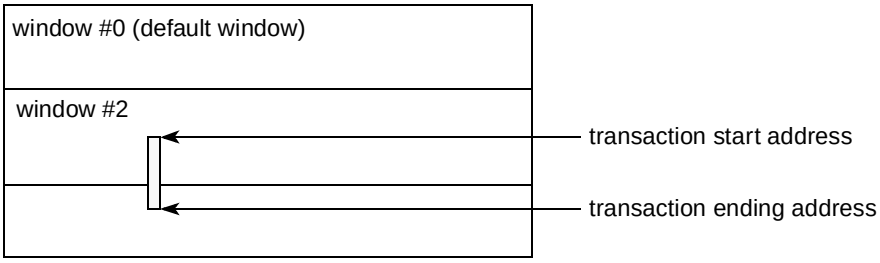


2. If a request hits multiple ATMU windows (1-8, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



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.

20.8.6 RapidIO Inbound ATMU

20.8.6.1 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.

20.8.6.2 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.

20.8.7 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).

20.8.8 Outbound Drain Mode

- The RapidIO port is placed into Drain mode when one of the following occurs:
  - PCR[OBDEN] is set
  - the Failed Threshold has been encountered and the CCSR[SPF] and CCSR[DPE] are both set
  - the packet time-to-live counter expires causing PCR[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 will show 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 Section 20.8.10, "Software Assisted Error Recovery Register Support."
- PCR[OBDEN] will also cause any queued up packet acknowledgements to be discarded if the port is uninitialized (the RapidIO port will let 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 will occur if the packet response timer is enabled (PRTOCCSR is non 0).

20.8.9 Input Port Disable Mode

- The RapidIO port is placed into Input Port Disable mode when CCSR[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 will show 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.

20.8.10 Software Assisted Error Recovery Register Support

- LMREQCSR 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:
     i. software polls for Port OK bit to be set
     ii. 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.)

     NOTE: If software can guarantee that the link-partner will 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 will actually cause the ackIDs to not match, and the link will not be recovered.
  e. 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.
  f. 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 LMREQCSR, software should make sure to successfully read LMRESPCSR 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 LASCSR, 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, CCSR[PL] could be set to prevent the inbound ackID from changing before LASCSR is written.

20.8.11 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.

20.8.11.1 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.

20.8.11.1.1 Extraction

- The host determines that the FRU has failed by getting a port response time-out when polling the port OK bit (PO bit in 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 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 PCR). Also note that setting OBDEN in PCR 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 PCR for normal operation
  - set the outbound and inbound ackID to 0x00 in the 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.

20.8.11.1.2 Insertion

- The FRU is inserted into the system
- Link initialization occurs and initialization complete status is indicated (port OK bit in the error 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 error and status command and status register)
- The host clears the output and input port enable bits and clears the port lockout bit in the 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 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 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

20.8.11.2 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.

20.8.11.2.1 Extraction

- 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
  - 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
- 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 control CSR
- The FRU is removed from the system

20.8.11.2.2 Insertion

- The FRU is inserted
- Link initialization occurs, initialization complete status is indicated in both RapidIO ports (PO bit = 1 in the error and status command and status register)
- After initialization is complete both devices set the port OK bit = 1 in the control CSR
- The FRU sets its port lockout bit in the control CSR
- The FRU generates a link-request/input-status to its link partner using the link maintenance request register. The FRU determines the link partner's inbound ackID by reading the link maintenance response register. The FRU then sets its outbound ackID in the local ackID status CSR.
- The FRU only enables maintenance transactions by clearing its output and input port enable bits in the control CSR and by clearing its port lockout bit in the control CSR
- The FRU generates a maintenance write to its link partner's 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 control CSR
- System operation resumes

20.8.12 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.

20.8.12.1 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 Error Detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by error enable CSR) is captured in the error 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.

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 error 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 implementation error CSR; the RapidIO hardware will continue to operate. In both cases, an interrupt is generated, and while the port will continue 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.

20.8.12.2 Physical Layer RapidIO Errors Detected

Table 20-105 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 ERECSR allows the error to increment the error rate counter and lock the error capture registers, and likewise which bit of the EDCSR is set when the error has been detected.

Table 20-106 lists the RapidIO endpoint behavior after exceeding certain preset limits (degraded threshold, failed threshold, retry threshold).

Table 20-105. 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	PCR[CCC] 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 w/o 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		
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.	PCR[CCP] 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

Table 20-105. Physical RapidIO Errors Detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable/ Detect
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.	—		
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.	PLTOCCSR[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	PLTOCCSR[TV] > 0 enables detect.	(re-) Enter Output Error Stopped.	—		

Table 20-106. Physical RapidIO Threshold Response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt clear[1]
Notification Errors					
Error Rate Counter has exceeded the Degraded Threshold.	ERTCSR[ERDTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Continue to operate normally.	Degraded Threshold	ESCSR[ODE]	Write 1 to ESCSR[ODE]
Fatal Errors					
Consecutive Retry Counter has exceeded the Retry Counter Threshold Trigger	PRETCR[RET] > 0 enables detect and interrupt generation	Generate Interrupt. Port is in priority order.	Consecutive Retry Threshold	IECSR[RETE]	Write 1 to IECSR[RETE]
Error Rate Counter has exceeded the Failed Threshold.	ERTCSR[ERFTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on CCSR[SPF] and CCSR[DPE] -- port can continue transmitting packets or can stop sending output packets, keeping or dropping them.	Failed Threshold	ESCSR[OFE]	Write 1 to ESCSR[OFE].

1 Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt.

20.8.12.3 Logical Layer Errors and Error Handling

This section describes how the logical layer will detect and react 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).

20.8.12.3.1 Logical Layer RapidIO Errors Detected

Table 20-107 through Table 20-121 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 with pass-through mode not enabled 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 20-107. 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 3	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, LTLCCCSR[FT] gets packet bits 12 - 15, LTLCCCSR[TT] gets packet bits 32 - 35, LTLCCCSR[MI] gets 0's For Large Transport type packetsr.LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92,LTLTLTLDIDCCSR[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.
Critical Request Flow Not Checked for error.	—	—	—	—	—

Table 20-107. Hardware Errors for NRead Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR[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 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 (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 Not Checked for error.	—	—	—	—	—
SrcTID Not Checked for error.	—	—	—	—	—
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to 20.8.6.2/20-97	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 20-108. 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 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,LTLTLTLDIDCCSR[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
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.	—	—	—	—	—

Table 20-108. 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
TransactionType Reserved Transaction Type for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	RapidIO packet is dropped
RdSize Read/Write request size is not for 4 bytes	Yes if LTLEECSR[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 LTLEECSR[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 LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	OCN error response is generated to self

Table 20-109. 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 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, 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,LTLTLTLDIDCCSR[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
Critical Request Flow Not Checked for error.	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR[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

Table 20-109. 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
SourceID Not Checked for error.	—	—	—	—	—
TransactionType Non-Atomic ttype 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 LTLEECSR[UT] is se	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 20.8.5.2/20-96	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 20.8.5.2/20-96	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 20-110. 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 Not unsupported	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, 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,LTLTLTLDIDCCSR[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.
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

Table 20-110. 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
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
SourceID Not applicable	—	—	—	—	—
TransactionType Received RapidIO packet for Atomic test-and-swap transaction	Yes if LTLEECSR[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 LTLEECSR[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 LTLEECSR[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 LTLEECSR[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 LTLEECSR[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 20-110. 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
Address:WdPtr:Xambs Write request hits overlapping ATMU windows Refer to 20.8.5.2/20-96	Yes if LTLEECSR[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
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 LTLEECSR[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 LTLEECSR[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:wrsize} field, payload is not double word aligned or does not have any payload	Yes if LTLEECSR[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 20-111. 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 LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
Critical Request Flow Not Checked for error.	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 62 - 63, LTLACCSR[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, 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 78-79, LTLACCSR[A] gets packet bits 48-76,LTLTLTLDIDCCSR[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
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

Table 20-111. Hardware Errors For SWrite Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Not Checked for error.	—	—	—	—	—
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to 20.8.5.2/20-96	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as third entry	RapidIO packet is dropped
Address:WdPtr:Xambs Request hits a protected ATMU window or the LCSBA1CSR	Yes if LTLEECSR[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 LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped

Table 20-112. 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 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, 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,LTLTLTLDIDCCSR[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
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 20-112. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	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
SourceID Does not match the request's DestID	Yes if LTLEECSR[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 LTLEECSR[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 LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
HopCount Not Checked for error.	—	—	—	—	—

Table 20-112. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Status Not UR Is not “Done” or “Error” Not “Done” status for “read_response” transaction type with payload “Error” status with payload.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR Error Response	Yes if LTLEECSR[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.
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[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

Table 20-112. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
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 20-113. 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, LTLACCSR[XA] gets packet bits 78 - 79 (if available), LTLACCSR[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, 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 (if available), LTLACCSR[A] gets packet bits 64-92 (if available),LTLTLTLDIDCCSR[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
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	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

Table 20-113. 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
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
SourceID Does not match the request's DestID	Yes if LTLEECSR [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 LTLEECSR [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 LTLEECSR [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 LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 20-113. 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 GSM Error response	Yes if LTLEECSR [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.
Status Not UR IO Error Response	Yes if LTLEECSR [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 LTLEECSR [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 LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 20-113. 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
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 LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
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 LTLEECSR [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.

Table 20-113. 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
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. 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 LTLEECSR [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 20-114. 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, 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,LTLTLTLDIDCCSR[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	RapidIO packet is dropped and ignored
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
TransportType Received reserved TT	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

Table 20-114. Hardware Errors For DMA Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN 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]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECSR [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	—	—	—	—	—
Status Not UR DMA Message Error response	Yes if LTLEECSR [DMAMER] is set	LTLEDCSR [DMAMER]	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 LTLEECSR [ITD] is set	Yes if LTLEDCSR [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 LTLEECSR [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) 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 LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table 20-114. Hardware Errors For DMA Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
Payload Size Not UR Payload size is not zero	Yes if LTLEECSR [ITD] is set	LTLEDCSR [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 LTLEECSR [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.	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 20-115. 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	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 packet bits 40-47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92,LTLTLTLDIDCCSR[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 packet bits 56-63.	RapidIO packet is dropped

Table 20-115. Hardware Errors For Message Request Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
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.	—	—	—	—	—
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.

Table 20-115. Hardware Errors For Message Request Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
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 se	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 20-116. 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 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,LTLTLTLDIDCCSR[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 packet bits 56-63.	—
Critical Request Flow Not Checked for error.	—	—	—	—	—
TransportType Received reserved TT	Yes if LTLEECSR[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 LTLEECSR[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 LTLEECSR[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

Table 20-116. Hardware Errors For Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	OCN Error Response Generated	Logical/Transport Layer Capture Register	Comments
SourceID Not Checked for error.	—	—	—	—	—
Status Not Checked for error.	—	—	—	—	—
Other Received Message response with SOCAR[M] disabled.	Yes if LTLEECSR[UR] is se	LTLEDCSR[UR]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored

Table 20-117. Hardware Errors for Doorbell Request 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, 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,LTLTLTLDIDCCSR[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 20-117. Hardware Errors for Doorbell Request Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
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.	—	—	—	—	—
SrcTID Not Checked for error.	—	—	—	—	—
Other Received Doorbell request with DOCAR[D] disabled.	Yes if LTLEECSR[UT] is se	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 20-118. 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 se	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,LTLTLTLDIDCCSR[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 and ignored
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR [ITD] is set	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
TransportType Received reserved TT	Yes if LTLEECSR [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 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 20-118. Hardware Errors For Doorbell Response Transactions (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 LTLEECSR [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 LTLEECSR [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 LTLEECSR [ITD] is se	LTLEDCSR [ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR [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 20-119. 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.	—	—	—	—	—

Table 20-119. Hardware Errors for PortWrite Transaction (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 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 zeros, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets zeros, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets zeros Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64–92,LTLTLTLDIDCCSR[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 zeros	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.	—	—	—	—	—
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

Table 20-119. Hardware Errors for PortWrite Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
SrcTID Not Checked for error.	—	—	—	—	—
HopCount Not Checked for error.	—	—	—	—	—
ConfigOffset Not Checked for error.	—	—	—	—	—
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 20-120. 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[IUT] 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 zeros, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets zeros, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] getszeros Large Transport type packets. LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92,LTLTLTLDIDCCSR[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 zeros	RapidIO packet is dropped
TransportType Received reserved TT	Yes if LTLEECSR[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

Table 20-120. 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
Address:WdPtr:Xambs Request hits overlapping ATMU windows. Refer to 20.8.5.2/20-96 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 20-121. 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 20.8.5.2/20-96	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, 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 zeros, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets zeros For Large Transport type packets, LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64–92,LTLTLTLDIDCCSR[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 zeros	—

Table 20-122. 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, LTLACCSR[XA] gets packet bits 78–79, LTLACCSR[A] gets packet bits 48–76, LTLDIDCCSR[DIDMSB] gets zeros, LTLDIDCCSR[DID] gets packet bits 16–23, LTLDIDCCSR[SIDMSB] gets zeros, LTLDIDCCSR[SID] gets packet bits 24–31, LTLCCCSR[FT] gets packet bits 12–15, LTLCCCSR[TT] gets packet bits 32–35, LTLCCCSR[MI] gets zeros For large transport mode, LTLACCSR[XA] gets packet bits 94–95, LTLACCSR[A] gets packet bits 64–92, LTLTLTLDIDCCSR[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 zeros	—

20.9 RapidIO Message Unit

This message unit supports multicast and non-multicast single segment messages and non-multicast multiple segment messages.

20.9.1 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.

20.9.2 Features

- Support for one or more 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 will 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 will not generate messages and the RapidIO endpoint is configured appropriately)
  - segment size up to 256 bytes
  - up to sixteen 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 one or more 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 sixteen 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

20.9.3 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.

20.9.4 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.

20.9.4.1 Direct Mode Operation

In direct mode (OM*n*MR[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*n*MR[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*n*MR[MUS] transitioning from 0 to 1 is ignored. Software is expected to program all the appropriate registers before setting OM*n*MR[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*n*SR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the status bits (OM*n*SR[MER], OM*n*SR[RETE], OM*n*SR[PRT], OM*n*SR[TE], OM*n*SR[QOI], OM*n*SR[QFI], OM*n*SR[EOMI], and OM*n*SR[QEI])

- Initialize the source address (EOM*n*SAR, OM*n*SAR), destination port (OM*n*DPR), destination attributes (OM*n*DATR), retry error threshold (OM*n*RETCR), and double-word count (OM*n*DCR) registers. If multicast mode is enabled (OM*n*DATR[MM]) initialize the multicast group and list (OM*n*MGR, OM*n*MLR).
- Initialize the outbound message mode register message unit transfer mode bit, OM*n*MR[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*n*MR[MUS], to start the message transfer.
- OM*n*SR[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*n*SAR, OM*n*SAR).
- 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*n*SR[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 (OM*n*DATR[EOMIE]).

20.9.4.1.1 Interrupts

The outbound message controller interrupt can be generated for one reason in direct mode.

- End-Of-Message - an interrupt is generated after the completion of a message if this interrupt event is enabled (OM*n*DATR[EOMIE] is a 1). The event that caused this interrupt is indicated by OM*n*SR[EOMI]. The interrupt is held until the OM*n*SR[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 (OM*n*MR[EIE])
- packet response time-out- an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (OM*n*MR[EIE])
- retry error threshold exceeded- an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (OM*n*MR[EIE])

- transaction error- an interrupt is generated after an OCN error response is received and this interrupt event is enabled (OM*n*MR[EIE])

20.9.4.1.2 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 (OM*n*SR[MER])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.1.3 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 (OM*n*SR[PRT])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.1.4 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*n*SR[RETE])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.1.5 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*n*SR[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*n*SR[MUB]) the message controller stops
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

20.9.4.1.6 Error Handling

When an error occurs and the SRIO error/write-port 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*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]

- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

20.9.4.1.7 Disabling and Enabling the Message Controller

Once the message controller is started, it cannot be stopped.

20.9.4.1.8 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 check are in addition to the error condition checks provided by the RapidIO port given in Section 20.8.12, “Errors and Error Handling.”

Table 20-123. 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/write-port if OM*n*MR[EIE] 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 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 segment local memory read. An internal error for a subsequent message segment local memory read for the same message may or may not occur.	2	No	None	Yes		Message controller stops after the current message operation completes.
Undefined Packet	Reserved ftype encoding[1]	3	SRIO error/write-port 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.

Table 20-123. 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	Reserved tt encoding[1]	3	SRIO error/write-port 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/write-port 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/write-port 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/write-port 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 received and no outbound mailboxes are supported[1]	3	SRIO error/write-port 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.

Table 20-123. 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	reserved response status (not done, retry, or error)	4a	SRIO error/write-port 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/write-port 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/write-port 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/write-port 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/write-port 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	RapidIO critical request flow (CRF) field is set when corresponding request's CRF field was clear	4c	SRIO error/write-port 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 20-123. 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	error response	5	SRIO error/write-port if LTLEECSR[MER] set. SRIO error/write-port if OM*n*MR[EIE].	Message error response in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MER]. OM*n*SR[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.
Message Response	number of retries exceeds limit	5	SRIO error/write-port if LTLEECSR[RETE] set. SRIO error/write-port if OM*n*MR[EIE].	Retry error threshold exceeded in the Logical/Transport Layer Error Detect CSR LTLEDCSR[RETE] OM*n*SR[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/write-port if LTLEECSR[PRT] set. SRIO error/write-port if OM*n*MR[EIE].	Packet response time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[PRT]. OM*n*SR[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 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) if the message request packet is captured or 0 if the message response packet is captured, LTLCCCSR[MI] gets the msg info (packet bits 56 - 63).

20.9.4.1.9 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 20-124. 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
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 will capture the transaction and generate an interrupt.
address for error enqueue address pointer is invalid	No	No	Local memory will capture the transaction and generate an interrupt.
register values changed during operation	No	No	Undefined operation results

20.9.4.2 Chaining Mode

In chaining mode, OM*n*MR[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*n*DQEPAR, EOM*n*DQEPAR) is maintained by software. The outbound message controller dequeues descriptors, processes them and increments the dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR) to point to the next descriptor in the queue.

Figure 20-99 below 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.

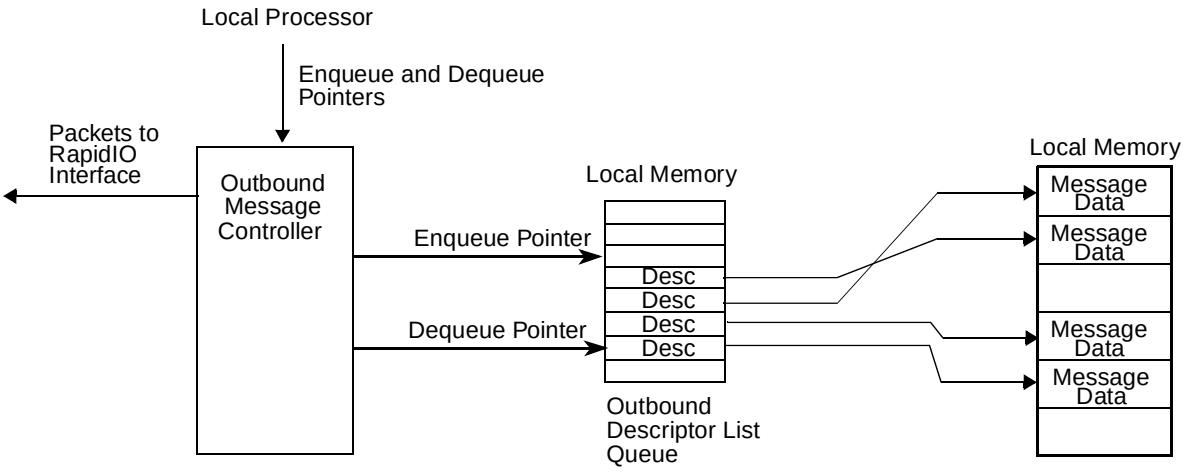


Figure 20-99. Outbound Frame Queue Structure

20.9.4.2.1 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*n*SR[MUB], to make sure the outbound message controller is not busy with a previously initiated message.
- Clear the message unit start bit (OM*n*MR[MUS]).
- Initialize the descriptor queue dequeue pointer address registers (OM*n*DQDPAR, EOM*n*DQDPAR) and the descriptor queue enqueue pointer address registers (OM*n*DQEPAR, EOM*n*DQEPAR). 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 × 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*n*MR[CIRQ_SIZ]; see Section 20.7.1.1, “Outbound Message n Mode Registers (OMnMR)”).
- Initialize the retry error threshold in the outbound message retry error threshold configuration register (OM*n*RETCR).
- Clear OM*n*MR[MUTM] for chaining mode.
- If using single segment multicast mode, set OM*n*DATR[MM].
- Configure the other control parameters in the mode register (OM*n*MR).
- Clear OM*n*SR[MER, PRT, RETE, TE, QOI, QFI, EOMI and QEI]. If OM*n*SR[MER, PRT, RETE, TE or QOI] are not cleared, the message controller will not start a new message operation. Incorrect status is indicated if the other status bits are not cleared.
- Set the message unit start (OM*n*MR[MUS]). This enables the outbound message controller and causes the descriptor dequeue pointer (OM*n*DQDPAR, EOM*n*DQDPAR) to be saved off as the base address of the descriptor queue.

20.9.4.2.2 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 (OM*n*DQEPAR, EOM*n*DQEPAR)
- Either increment the enqueue pointer address registers (OM*n*DQEPAR, EOM*n*DQEPAR) by setting OM*n*MR[MUI] for each descriptor entry added or directly change the enqueue pointer address register (OM*n*DQEPAR). If OM*n*MR[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 (OM*n*DQDPAR, EOM*n*DQDPAR) and sets the busy bit (OM*n*SR[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 (OM*n*DQDPAR, EOM*n*DQDPAR). The message controller will not prefetch more than one descriptor.
- OM*n*SR[MUB] is set by the message controller to indicate that the message transfer is in progress. OM*n*SR[MUB] will remain 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 (OM*n*SR[MUB]). Software can continue adding descriptors as long as the descriptor queue is not full. If software is adding descriptors using the OM*n*MR[MUI] bit, overflowing the queue can be prevented by polling the queue full bit (OM*n*SR[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 (OM*n*SR[MER]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If a packet response time-out occurs, the packet response time-out bit is set (OM*n*SR[PRT]). If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If the retry error threshold value is exceeded for a specific segment, the retry error threshold exceeded bit is set (OM*n*SR[RETE]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- If an internal error occurs while reading local memory, the transaction error bit is set (OM*n*SR[TE]) and outbound message controller operation stops after all message segments complete. If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- The above process continues until the descriptor queue is empty (dequeue pointer equals the enqueue pointer).
- The message unit clears OM*n*SR[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.

20.9.4.2.3 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 (OM*n*SR[MUB]). Software then disables the message controller by clearing OM*n*MR[MUS], changes the enqueue and dequeue descriptor pointers (EOM*n*DQEPAR, OM*n*DQEPAR and EOM*n*DQDPAR, OM*n*DQDPAR) and re-enables the message unit by setting OM*n*MR[MUS].

20.9.4.2.4 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*n*MR[MUI]) software can poll the queue full bit (OM*n*SR[QF]) before enqueueing another descriptor. When software sets the enqueue pointer directly, software is responsible for not overflowing the descriptor queue.

20.9.4.2.5 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*n*SR[MUB] being cleared. When switching from direct mode to chaining mode, if OM*n*MR[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*n*MR[MUS] must also be cleared and set.

20.9.4.2.6 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 × 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 20-125. 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.
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 20-100 depicts the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

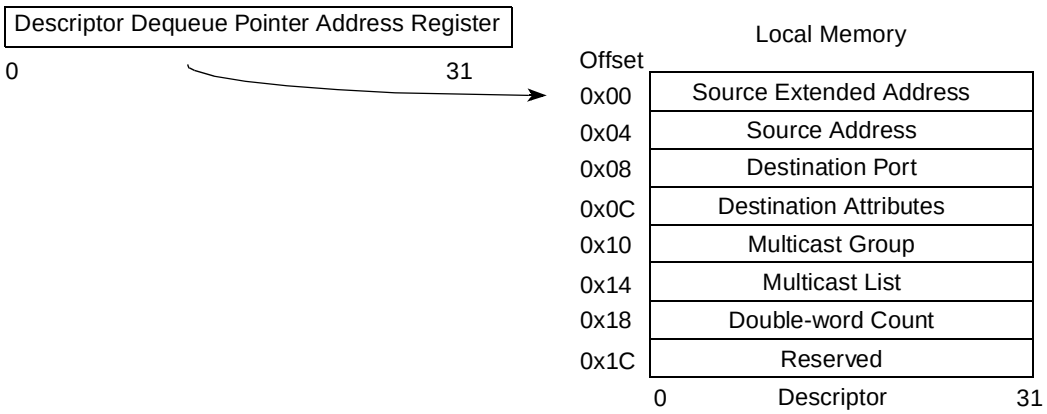


Figure 20-100. Descriptor Dequeue Pointer and Descriptor

20.9.4.2.7 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 (OM*n*MR[QEIE] is a 1). The event that caused the outbound message

interrupt is indicated by OM*n*SR[QEI]. The interrupt is held until the queue is not empty and the OM*n*SR[QEI] bit has been 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 (OM*n*MR[QFIE] is a 1). The event that caused the outbound message interrupt is indicated by OM*n*SR[QFI]. The interrupt is held until the queue is not full and the OM*n*SR[QFI] bit has been 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 (OM*n*MR[MUI]) and the interrupt event is enabled (OM*n*MR[QOIE] is a 1). The event that caused the outbound message interrupt is indicated by OM*n*SR[QOI]. The message unit must be re-initialized. The interrupt is held until the OM*n*SR[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 (OM*n*DATR[EOMIE] is a 1). The event that caused this interrupt is indicated by OM*n*SR[EOMI]. The interrupt is held until the OM*n*SR[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 (OM*n*MR[EIE])
- Packet response time-out—an interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (OM*n*MR[EIE])
- Retry error threshold exceeded—an interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (OM*n*MR[EIE])
- Transaction error—an interrupt is generated after a message error response is received and this interrupt event is enabled (OM*n*MR[EIE])

20.9.4.2.8 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 (OM*n*SR[MER])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.2.9 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 (OM*n*SR[PRT])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.2.10 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*n*SR[RETE])
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the message operation completes (indicated by OM*n*SR[MUB]) the message controller stops

20.9.4.2.11 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*n*SR[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*n*SR[MUB]) the message controller stops
- If OM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

20.9.4.2.12 Error Handling

When an error occurs and the SRIO error/write-port 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*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

When an error occurs and the SRIO error/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling OM*n*SR[MUB]
- Software disables the message controller by clearing OM*n*MR[MUS]
- Software clears the error by writing a 1 to the corresponding status bit (OM*n*SR[MER], OM*n*SR[PRT], OM*n*SR[RETE], or OM*n*SR[TE])

20.9.4.2.13 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 Section 20.8.12, “Errors and Error Handling.”

Table 20-126. 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/write-port if OM*n*MR[EIE] 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.

20.9.4.2.14 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 20-127. Outbound Message Chaining Mode Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Enqueued descriptor address is invalid	No	No	Local memory will capture the transaction and generate an interrupt.
Address for descriptor enqueue address pointer is invalid	No	No	Local memory will capture 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 (OM*n*MR[QOIE])	Queue overflow (OM*n*SR[QOI])	Message controller stops.
Queue misaligned	No	No	May result in duplicate messages being sent

20.9.4.3 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 unit. 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, etc.). The number of messages a message unit can send is from 1 to 64.

OM*n*MR[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, etc.) 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.

20.9.5 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 × 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 20-101 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 (IM*n*MR[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.

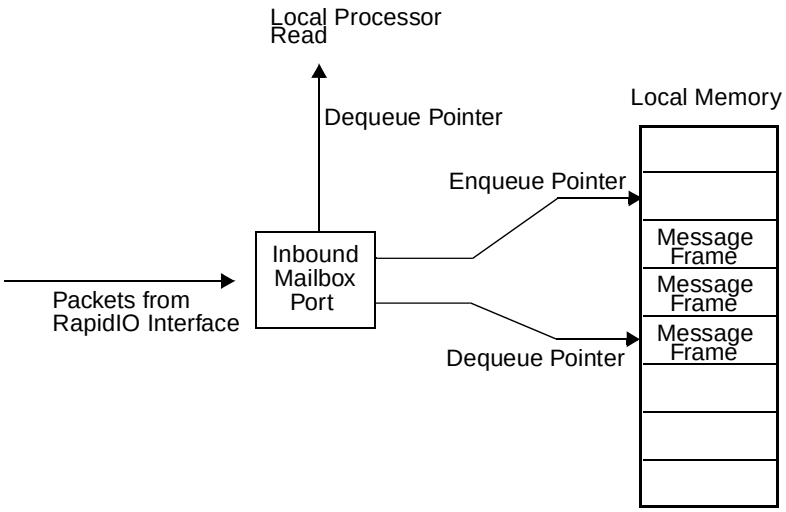


Figure 20-101. Inbound Message Structure

20.9.5.1 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*n*FQDPAR, EIM*n*FQDPAR) and the frame queue enqueue pointer address registers (EIM*n*FQEPAR, IM*n*FQEPAR). 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 × 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*n*SR).
- Set the mailbox enable bit (IM*n*MR[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*n*MR).

20.9.5.2 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 (IM*n*MR[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 IM*n*SR[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 (IM*n*MR[MIQ_THRESH) and this event is enabled to generate the interrupt (IM*n*MR[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 (IM*n*SR[MIQI]).
- Software processes the frame queue entry pointed to by the frame dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR).
- Software increments the dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) by setting the message increment bit (IM*n*MR[MI]). Software determines if there are any more messages to process by reading the queue empty bit (IM*n*SR[QE]). If the queue is not empty, the previous two steps are repeated.
- Optionally, software reads the enqueue pointer address registers (IM*n*FQEPAR, EIM*n*FQEPAR) and processes all the received messages. After the message processing is complete the dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) are written.
- Software clears the message-in-queue interrupt bit (IM*n*SR[MIQI]) by writing a 1 to the IM*n*SR[MIQI] bit.

20.9.5.3 Message Steering

Messages are forwarded to the inbound message controllers as follows:

- Messages directed to mailbox 0 are forwarded to message controller 0
- Messages directed to mailbox 1, 2 or 3 are forwarded to message controller 1

20.9.5.4 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 will not occur.

20.9.5.5 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 (IM*n*MR[MIQIE]). The event that caused this interrupt is indicated by IM*n*SR[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 IM*n*SR[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 (IM*n*SR[MIQIE]). The event that caused this interrupt is indicated by IM*n*SR[MIQI]. The interrupt is held until the IM*n*SR[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 (IM*n*SR[QFIE]). The event that caused this interrupt is indicated by IM*n*SR[QFI]. The interrupt is held until the queue is not full and the IM*n*SR[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 (IM*n*MR[EIE])
- Transaction error— an interrupt is generated after a OCN error response is received and this interrupt event is enabled (IM*n*MR[EIE])

20.9.5.5.1 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 (IM*n*SR[MRT])
- All message segments that have not yet been received are considered complete
- If IM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated
- Once all message segments complete (indicated by IM*n*SR[MB]), the message controller stops

20.9.5.5.2 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 (IM*n*SR[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 IM*n*SR[MB]) the message controller stops
- If IM*n*MR[EIE] is set, the interrupt SRIO error/write-port is generated.

20.9.5.5.3 Error Handling

When an error occurs and the SRIO error/write-port 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 IM*n*SR[MB]
- Software clears the error by writing a 1 to the corresponding status bit (IM*n*SR[MRT], and/or IM*n*SR[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/write-port interrupt is not enabled, the following occurs.

- Software determines that an error has occurred by polling the status bits (IM*n*SR[MRT] and/or IM*n*SR[TE])
- Software verifies the message controller has stopped operation by polling IM*n*SR[MB]
- Software disables the message controller by clearing IM*n*MR[ME]
- Software clears the error by writing a 1 to the corresponding status bit (IM*n*SR[MRT] and/or IM*n*SR[TE])

20.9.5.5.4 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 20-128.

Table 20-128. 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/write-port 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/write-port 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/write-port 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/write-port 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.
An incorrect message packet size (payload is not than the specified ssize except for the last segment. The last segment's payload can be less than or equal to the segment size but not 0. Note that payload sizes are dword multiples.)[1]	1	SRIO error/write-port 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.
Reserved ssize field[1]	1	SRIO error/write-port 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.

Table 20-128. 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 received and no mailboxes are supported as indicated by DOCAR[M][1]	1	SRIO error/write-port 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.
Message packet size larger than the configured frame size and message controller is enabled	2	SRIO error/write-port 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.
An inbound message packet with a RapidIO priority of 3	2	SRIO error/write-port 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.
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
message segment number is larger than the number of message segments in the message	2	SRIO error/write-port 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.

Table 20-128. 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
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/write-port 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.
msglen (number of segments in a message) is not consistent in all segments of a multi segment message	2	SRIO error/write-port 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.
ssize (segment size) is not consistent in all segments of a multi segment message	2	SRIO error/write-port 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.
Message received for an unsupported mailbox but at least one mailbox is supported	2	SRIO error/write-port if LTLEECSR[UT] set	Unsupported transaction in the Logical/Transport Layer Error Detect CSR LTLEDCSR[UT]	No	Error	—	Packet is ignored and discarded. This error will only apply if a mailbox is not supported. This error is not currently supported since all mailboxes are supported.
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.

Table 20-128. 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
Internal error occurred during the write of the frame queue entry to memory	3	SRIO error/write-port if IM*n*MR[EIE] set	Transaction error in the Message status register (IM*n*SR[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	—	—

Table 20-128. 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 request to request time-out for multi segment messages	unrelated	SRIO error/write-port if LTLEECSR[MRT] set. SRIO error/write-port if OM*n*MR[EIE].	Message request time-out in the Logical/Transport Layer Error Detect CSR LTLEDCSR[MRT]. IM*n*SR[MRT] bit set.	No	No	Updated with the previous message request packet 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]	All message segments received before the time-out update memory. The enqueue pointer is not incremented. The message operation completes.

Note:

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).

20.9.5.5.5 Programming Errors

The following is a partial list of programming errors that result in undefined or undesired hardware operation.

Table 20-129. Inbound Message Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the message in queue threshold (IM*n*MR[MIQ_THRESH]) or reserved value of the circular frame queue size (IM*n*MR[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

Table 20-129. Inbound Message Programming Errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Frame queue entry written to non-existent memory	No	No	Memory controller will cause the interrupt and update capture registers. IM*n*SR[TE] is set due to the internal error.
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

20.9.5.5.6 Disabling and Enabling the Inbound Message Controller

When the message controller is disabled by clearing IM*n*MR[ME] the following occurs.

- Queue full clears (IM*n*SR[QF])
- Message-in-queue clears (IM*n*SR[MIQ])
- Queue empty is set (IM*n*SR[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 (IM*n*SR[MB]).

Before the message controller is re-enabled the message busy bit must be clear (IM*n*SR[MB) and the (IM*n*MR[ME]) the frame queue dequeue pointer address registers (IM*n*FQDPAR, EIM*n*FQDPAR) and the frame queue enqueue pointer address registers (IM*n*FQEPAR, EIM*n*FQEPAR) must be initialized to the same value for proper message controller operation.

20.9.6 RapidIO 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 (IM*n*MR[ME]), the inbound message controller is not in the internal error state (IM*n*SR[TE] = 0) and the inbound message controller did not detect a message request time-out (IM*n*SR[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 IM*n*SR[MB]
- Failed (MCSR[FA])—This bit is set if any of the following bits are set:
  - The inbound message controller transaction error status bit IM*n*SR[TE]
  - The inbound message controller message request time-out status bit IM*n*SR[MRT]
  - The outbound message controller transaction error status bit OM*n*SR[TE] is set
  - The outbound message controller packet response time-out bit OM*n*SR[PRT] is set
  - The outbound message controller message error response received status bit OM*n*SR[MER] is set
  - The outbound message controller retry error threshold exceeded status bit OM*n*SR[RETE] is set
- Error (MCSR[ERR])—This bit is always 0

20.10 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.

20.10.1 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

20.10.2 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 20-102 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.

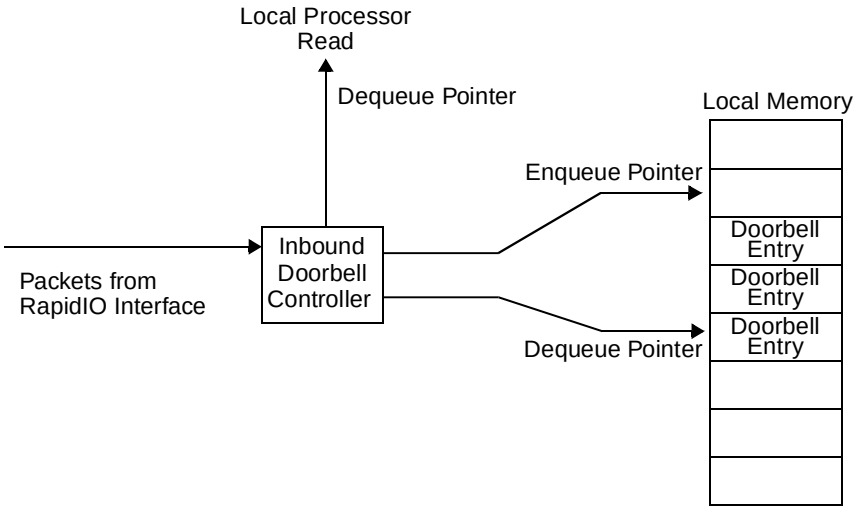


Figure 20-102. Inbound Doorbell Queue and Pointer Structure

20.10.2.1 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, ODMR[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 (ODDPR), destination attributes (ODDATR) and retry error threshold (ODRETCR) 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]).

20.10.2.1.1 Interrupts

The outbound doorbell controller interrupt can be generated for one reason.

- End-Of-Doorbell. An 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 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 (ODMR[EIE])
- Packet response time-out. An interrupt is generated after a packet response time-out occurs and this interrupt event is enabled (ODMR[EIE])
- Retry error threshold exceeded. An interrupt is generated after a retry threshold exceeded error occurs and this interrupt event is enabled (ODMR[EIE])

20.10.2.1.2 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 bit (ODSR[MER])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

20.10.2.1.3 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 bit (ODSR[PRT])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

20.10.2.1.4 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 bit (ODSR[RETE])
- If ODMR[EIE] is set, the interrupt SRIO error/write-port is generated.
- After the doorbell operation completes (indicated by ODSR[DUB]) the doorbell controller stops

20.10.2.1.5 Error Handling

When an error occurs and the SRIO error/write-port 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 ODSR[DUB]
- Software disables the doorbell controller by clearing ODMR[DUS]
- Software clears the error by writing a 1 to the corresponding status bit (ODSR[PRT], ODSR[PRT], and/or ODSR[RETE])

When an error occurs and the SRIO error/write-port 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 bit (ODSR[MER], ODSR[PRT], and/or ODSR[RETE])

20.10.2.1.6 Disabling and Enabling the Doorbell Controller

Once the doorbell controller is started, it cannot be stopped.

20.10.2.1.7 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 check are in addition to the error condition checks provided by the RapidIO port given in Section 20.8.12, “Errors and Error Handling.”

Table 20-130. 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/write-port 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/write-port 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/write-port 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/write-port 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/write-port 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/write-port 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/write-port 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 20-130. Outbound Doorbell Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Doorbell Response	RapidIO critical request flow (CRF) field is set when corresponding request's CRF field was clear[1]	2	SRIO error/write-port 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/write-port 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	reserved response status [1]	2	SRIO error/write-port 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/write-port 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/write-port if LTLEECSR[MER] set. SRIO error/write-port if OM*n*MR[EIE].	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

Table 20-130. Outbound Doorbell Hardware Errors (continued)

Transaction	Error	Error Checking Level	Interrupt Generated	Status Bit Set	Doorbell Sent	Logical/Transport Layer Capture Register	Comments
Doorbell Response	number of retries exceeds limit	3	SRIO error/write-port if LTLEECSR[RETE] set. SRIO error/write-port if OM*n*MR[EIE].	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/write-port if LTLEECSR[PRT] set. SRIO error/write-port if OM*n*MR[EIE].	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.

20.10.2.1.8 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 20-131. 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 will start using this new transaction flow level.

Table 20-131. Outbound Doorbell Programming Errors (continued)

Error	Interrupt Generated	Status Bit Set	Comments
Target interface set to an invalid RapidIO port	No	None	Undefined operation results
Register values changed during operation	No	No	Undefined operation results

20.10.2.2 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.

20.10.2.2.1 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 (DQEPAR, EDQEPAR). These need to be initialized to the same value for proper operation. These also must be queue size 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).

20.10.2.2.2 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 (DQEPAR, EDQEPAR).
- 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.

20.10.2.2.3 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 20-132. 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 20-133. 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 20-103 depicts the doorbell queue entry fields and their related offsets.

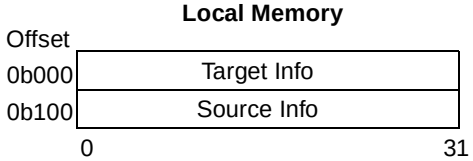


Figure 20-103. Doorbell Entry Format

20.10.2.2.4 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 will not occur.

20.10.2.2.5 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])

20.10.2.2.6 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/write-port is generated.

20.10.2.2.7 Error Handling

When an error occurs and the SRIO error/write-port 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.

20.10.2.2.8 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 Section 20.8.12, “Errors and Error Handling.”

Table 20-134. 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/write-port 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/write-port 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/write-port 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.

Table 20-134. 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
Illegal Destination ID[1]	1	SRIO error/write-port 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/write-port 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/write-port 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/write-port 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.

Table 20-134. 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
Internal error occurred during the write of the doorbell queue entry to memory	4	SRIO error/write-port if OMMR[EIE] set	Transaction error in the Doorbell status register (ISSR[TE]). Doorbell Failed in the Port-write and Doorbell CSR (PWDCSR[FA])	No	Error	—	Doorbell controller stops after the current 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), 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.

20.10.2.2.9 Programming Errors

The following is the list of programming errors that result in undefined or undesired hardware operation.

Table 20-135. Inbound Doorbell Programming Errors

Error	Interrupt Generated	Status Bit Set	Comments
Reserved value of the doorbell in queue threshold (ID*n*MR[DIQ_THTRES]) or reserved value of the circular doorbell queue size (ID*n*MR[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 will cause 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

20.10.2.2.10 Disabling and Enabling the Doorbell Controller

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 clear (IDSR[DB) and the doorbell queue dequeue pointer address registers (DQDPAR, EDQDPAR) and the doorbell queue enqueue pointer address registers (DQEPAR, EDQEPAR) must be initialized to the same value for proper doorbell controller operation.

20.10.2.3 RapidIO 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

20.10.3 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 20-104 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/write-port) for notification of incoming port-writes.

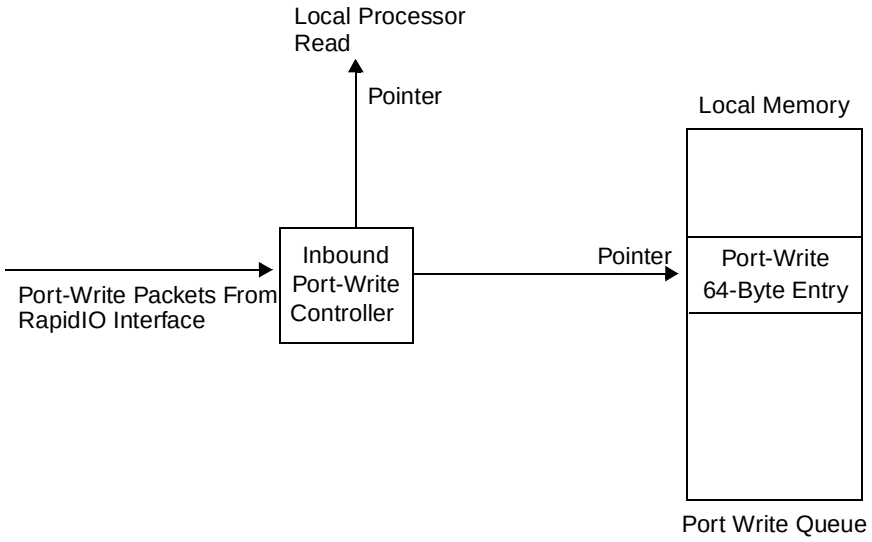


Figure 20-104. Inbound Port-Write Structure

20.10.3.1 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).

20.10.3.2 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/write-port 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].

20.10.3.3 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 not 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])

20.10.3.4 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.

20.10.3.5 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/write-port is generated.

20.10.3.5.1 Error Handling

When an error occurs and the SRIO error/write-port 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/write-port 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])

20.10.3.6 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 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 Section 20.8.12, “Errors and Error Handling.”

Table 20-136. 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/write-port 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/write-port 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/write-port 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/write-port if LTLEECSR[ITTE] set	Illegal transaction target in the Logical/Transport Layer Error Detect CSR LTLEDCSR[ITTE]	No	No Response	Updated with the packet[2]	Packet is ignored and discarded.

Table 20-136. 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 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/write-port 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/write-port 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/write-port 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 - An inbound port-write packet with a RapidIO priority of 3	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 20-136. 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
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/write-port 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), 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.

20.10.3.6.1 Programming Errors

The following is the list of programming errors.

Table 20-137. 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 will cause its own interrupt and update its own capture registers. An internal error response is 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.

20.10.3.7 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]).

20.10.3.8 RapidIO 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 21
PCI Express Interface Controller

The PCI Express interface complies with the *PCI Express™ Base Specification, Revision 1.0a* (available from 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.

21.1 Introduction

The PCI Express controller provides the mechanism to communicate with PCI Express devices. Figure 21-1 is a high-level block diagram of the PCI Express controller.

21.1.1 Overview

The PCI Express controller connects the internal platform to a 2.5-GHz serial interface. The MPC8572E offers three instantiations of this controller yielding up to three PCI Express links simultaneously.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.

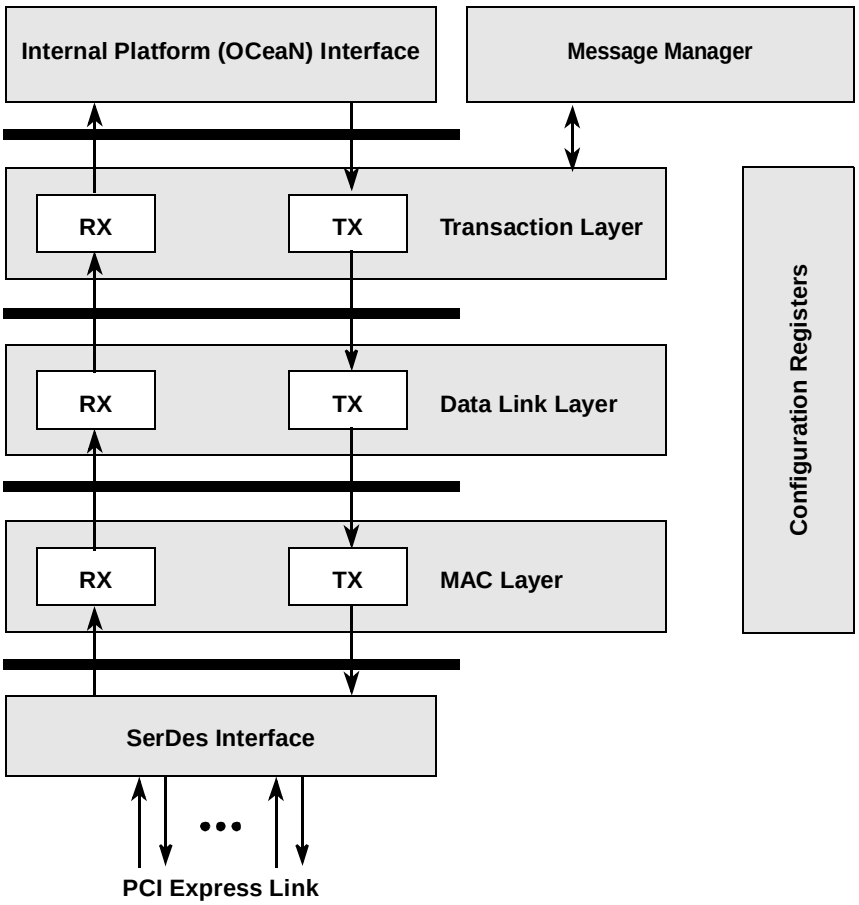


Figure 21-1. PCI Express Controller Block Diagram

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.

21.1.1.1 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 (OCeaN). 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 (OCeaN), 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.

21.1.1.2 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 (OCeaN).

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.

21.1.2 Features

The following is a list of features supported by the PCI Express controller:

- Complies with the *PCI Express™ Base Specification, Revision 1.0a*
- Supports root complex (RC) and endpoint (EP) configurations
- 32- and 64-bit 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
- 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 and one configuration window
- Supports four outbound translation windows and one default window
- Supports eight non-posted and four posted PCI Express transactions
- Supports up to six priority 0 internal platform reads and eight priority 0 to 2 internal platform writes. (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 (OCeaN)

21.1.3 Modes of Operation

Several parameters that affect the PCI Express controller modes of operation are determined at power-on reset (POR) by reset configuration signals as described in Chapter 4, “Reset, Clocking, and Initialization.”

Table 21-1. POR Parameters for PCI Express Controller

Parameter	Description	Section/Page
Host/Agent Configuration	Selects between root complex (RC) and endpoint (EP) modes.	4.4.3.5/4-16
I/O Port Selection	Selects the width of the PCI Express link	4.4.3.6/4-17

21.1.3.1 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 input signals cfg_host_agt[0:2] multiplexed on $\overline{\text{LWE}}$/$\overline{\text{LBS}}$[1:3] determine the RC/EP mode. (See Section 4.4.3.5, “Host/Agent Configuration.”)

21.1.3.2 Link Width

The I/O port selection configuration input signals cfg_IO_ports[0:3] multiplexed on TSEC1_TXD[3:1] and TSEC2_TX_ER determine the initial link width. (See Section 4.4.3.6, “I/O Port Selection.”)

See Section 4.4.4.3, “SGMII Clocks,” for proper selection of CCB clock frequency with regard to PCI Express link width selection.

Note that the link width also depends on the configuration of the serial RapidIO width.

21.2 External Signal Descriptions

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 (TX*n* and $\overline{\text{TX}n}$) and a differential pair for receiving (RX*n* and $\overline{\text{RX}n}$) with an embedded data clock.

Although the generic PCI Express controller described here accommodates up to a single x8 link, there are three PCI Express controllers instantiated on the MPC8572E. Please refer to Section 4.4.3.6, "I/O Port Selection," for specific pin muxing details.

Table 21-2 contains detailed descriptions of the external PCI Express interface signals.

Table 21-2. PCI Express Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SD1_RX[7:0]	I	Receive data. The receive data signals carry PCI Express packet information. Subject to Section 4.4.3.6, "I/O Port Selection, PCI Express controller 1 is available on SD1_RX[7:0], PCI Express controller 2 is available on SD1_RX[7:4], and PCI Express controller 3 is available on SD1_RX[7:6]. Note that lane reversal may affect the logical lane assignment. Refer to Section 21.4.1.3, "Lane Reversal," for more information.	
		State Meaning	Asserted/Negated—Represents data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.
$\overline{\text{SD1\_RX}}$[7:0]	I	Receive data, inverted. $\overline{\text{SD1\_RX}}$[7:0] are the inverted forms of the receive data signals (SD1_RX[7:0]).	
		State Meaning	Asserted/Negated—Represents the inverse of data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.
SD1_TX[7:0]	O	Transmit data. The transmit data signals carry PCI Express packet information. Subject to Section 4.4.3.6, "I/O Port Selection, PCI Express controller 1 is available on SD1_TX[7:0], PCI Express controller 2 is available on SD1_TX[7:4], and PCI Express controller 3 is available on SD1_TX[7:6]. Note that lane reversal may affect the logical lane assignment. Refer to Section 21.4.1.3, "Lane Reversal," for more information.	
		State Meaning	Asserted/Negated—Represents data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.
$\overline{\text{SD1\_TX}}$[7:0]	O	Transmit data, inverted. $\overline{\text{SD1\_TX}}$[7:0] are the inverted form of the transmit data signals (SD1_TX[7:0]).	
		State Meaning	Asserted/Negated—Represents the inverse of data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 1.0a*.

21.3 Memory Map/Register Definitions

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 Section 21.3.1, "PCI Express Memory Mapped Registers," and its subsections.

- 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 Section 21.3.7, "PCI Express Configuration Space Access," and its subsections.

21.3.1 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 but are located at a different block base address. Memory-mapped registers for PCI Express controller 1 begin at block base address 0x0_A000, controller 2 registers begin at 0x0_9000, and controller 3 registers begin at 0x0_8000.

Table 21-3 lists the memory-mapped registers. In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 21-3. PCI Express Memory-Mapped Register Map

PCI Express Controller 1 —Block Base Address 0x0_A000 PCI Express Controller 2—Block Base Address 0x0_9000 PCI Express Controller 3—Block Base Address 0x0_8000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
PCI Express Controller 1 Memory-Mapped Registers				
PCI Express Configuration Access Registers				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	0x0000_0000	21.3.2.1/21-10
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	0x0000_0000	21.3.2.2/21-11
0x008	Reserved	—	—	
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0010_FFFF	21.3.2.3/21-11
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	21.3.2.4/21-12
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x0000_0000	21.3.2.5/21-13

Table 21-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_A000 PCI Express Controller 2—Block Base Address 0x0_9000 PCI Express Controller 3—Block Base Address 0x0_8000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x018–0x01C	Reserved	—	—	
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	w1c	0x0000_0000	21.3.3.1/21-13
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	R/W	0x0000_0000	21.3.3.2/21-15
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	R/W	0x0000_0000	21.3.3.3/21-16
0x02C	PEX_PMCR—PCI Express power management command register	R/W	0x0000_0000	21.3.3.4/21-18
0x030–0xBF4	Reserved	—	—	—
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0100	21.3.4.1/21-19
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	0x0000_0000	21.3.4.2/21-19
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	R/W	0x0000_0000	21.3.5.1.1/21-20
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	R/W	0x0000_0000	21.3.5.1.2/21-21
0xC08–0xC0C	Reserved	—	—	—
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	Mixed	0x8004_4023	21.3.5.1.4/21-22
0xC14–0xC1C	Reserved	—	—	—
Outbound Window 1				
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W	0x0000_0000	21.3.5.1.1/21-20
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W	0x0000_0000	21.3.5.1.2/21-21
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W	0x0000_0000	21.3.5.1.3/21-22
0xC2C	Reserved	—	—	—
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W	0x0004_4023	21.3.5.1.4/21-22
0xC34–0xC3C	Reserved	—	—	—
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W	0x0000_0000	21.3.5.1.1/21-20

Table 21-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_A000 PCI Express Controller 2—Block Base Address 0x0_9000 PCI Express Controller 3—Block Base Address 0x0_8000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W	0x0000_0000	21.3.5.1.2/21-21
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W	0x0000_0000	21.3.5.1.3/21-22
0xC4C	Reserved	—	—	—
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W	0x0004_4023	21.3.5.1.4/21-22
0xC54–0xC5C	Reserved	—	—	—
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W	0x0000_0000	21.3.5.1.1/21-20
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W	0x0000_0000	21.3.5.1.2/21-21
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W	0x0000_0000	21.3.5.1.3/21-22
0xC6C	Reserved	—	—	—
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W	0x0004_4023	21.3.5.1.4/21-22
0xC74–0xC7C	Reserved	—	—	—
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W	0x0000_0000	21.3.5.1.1/21-20
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W	0x0000_0000	21.3.5.1.2/21-21
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W	0x0000_0000	21.3.5.1.3/21-22
0xC8C	Reserved	—	—	—
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W	0x0004_4023	21.3.5.1.4/21-22
0xC94–0xC9C	Reserved	—	—	—
0xD14–0xD9C	Reserved	—	—	—
Inbound Window 3				
0xDA0	PEXITAR3—PCI Express inbound translation address register 3	R/W	0x0000_0000	21.3.5.2.3/21-26
0xDA4	Reserved	—	—	—
0xDA8	PEXIWBAR3—PCI Express inbound window base address register 3	R/W	0x0000_0000	21.3.5.2.4/21-27
0xDAC	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W	0x0000_0000	21.3.5.2.5/21-27
0xDB0	PEXIWAR3—PCI Express inbound window attributes register 3	R/W	0x20F4_4023	21.3.5.2.6/21-28

Table 21-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_A000 PCI Express Controller 2—Block Base Address 0x0_9000 PCI Express Controller 3—Block Base Address 0x0_8000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xDB4–0xDBC	Reserved	—	—	
Inbound Window 2				
0xDC0	PEXITAR2—PCI Express inbound translation address register 2	R/W	0x0000_0000	21.3.5.2.3/21-26
0xDC4	Reserved	—	—	—
0xDC8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W	0x0000_0000	21.3.5.2.4/21-27
0xDCC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W	0x0000_0000	21.3.5.2.5/21-27
0xDD0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W	0x20F4_4023	21.3.5.2.6/21-28
0xDD4–0xDDC	Reserved	—	—	—
Inbound Window 1				
0xDE0	PEXITAR1—PCI Express inbound translation address register 1	R/W	0x0000_0000	21.3.5.2.3/21-26
0xDE4	Reserved	—	—	—
0xDE8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W	0x0000_0000	21.3.5.2.4/21-27
0xDEC	Reserved	—	—	—
0xDF0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W	0x20F4_4023	21.3.5.2.6/21-28
0xDF4–0xDFC	Reserved	—	—	—
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	0x0000_0000	21.3.6.1/21-30
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	0x0000_0000	21.3.6.2/21-32
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	0x0000_0000	21.3.6.3/21-34
0xE14–0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	0x0000_0000	21.3.6.4/21-36
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	0x0000_0000	21.3.6.5/21-37
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	0x0000_0000	21.3.6.6/21-38
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	0x0000_0000	21.3.6.7/21-40
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	0x0000_0000	21.3.6.8/21-42
0xE38–0xFFC	Reserved	—	—	—
PCI Express Controller 2 Memory-Mapped Registers				

Table 21-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller 1 —Block Base Address 0x0_A000 PCI Express Controller 2—Block Base Address 0x0_9000 PCI Express Controller 3—Block Base Address 0x0_8000				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x000–0xFFC	PCI Express Controller 2 registers **Note:** All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 2; the offsets of PCI Express Controller 2 registers are the same except they have a different block base address.			
PCI Express Controller 3 Memory-Mapped Registers				
0x000–0xFFC	PCI Express Controller 3 registers **Note:** All registers defined for PCI Express Controller 1 are also defined for PCI Express Controller 3; the offsets of PCI Express Controller 3 registers are the same except they have a different block base address.			

21.3.2 PCI Express Configuration Access Registers

21.3.2.1 PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The PCI Express configuration address register, shown in Figure 21-2, contains address information for accesses to PCI Express internal and external configuration registers.

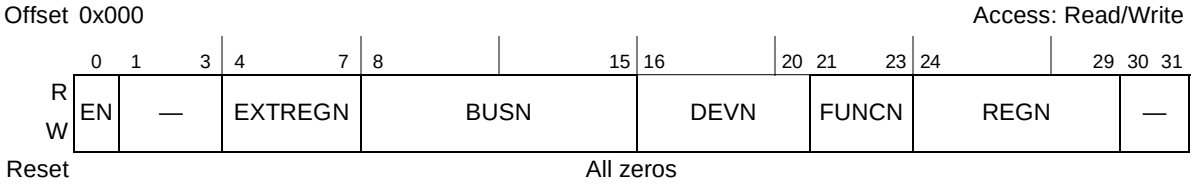


Figure 21-2. PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The fields of the PCI Express configuration address register are described in Table 21-4.

Table 21-4. PEX_CONFIG_ADDR Field Descriptions

Bits	Name	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	—	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 0xFFF).
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	—	Reserved

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 || 0b00.

21.3.2.2 PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The PCI Express configuration data register, show in Figure 21-3, 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 Section 21.4.1.2.1, "Byte Order for Configuration Transactions," for more information.

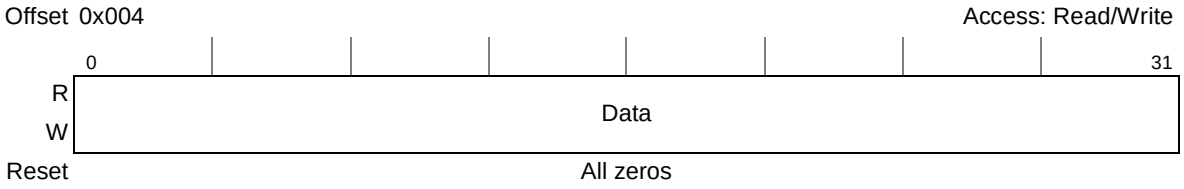


Figure 21-3. PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The fields of the PCI Express configuration data register are described in Table 21-5.

Table 21-5. PEX_CONFIG_DATA Field Descriptions

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI Express configuration cycle if the PEX_CONFIG_ADDR enable bit is set (PEX_CONFIG_ADDR[EN] = 1).

21.3.2.3 PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register, shown in Figure 21-4, 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.

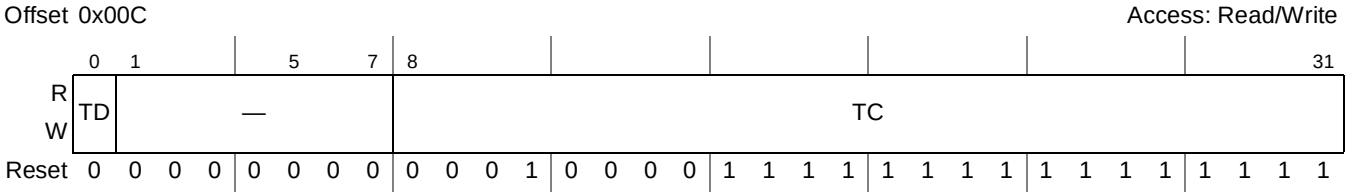


Figure 21-4. PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The fields of the PCI Express outbound completion timeout register are described in Table 21-6.

Table 21-6. PEX_OTB_CPL_TOR Field Descriptions

Bits	Name	Description
0	TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
1–7	—	Reserved
8–31	TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. One TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz, and 30 ns at 266.66 MHz. The following are examples of timeout periods based on different TC settings: 0x00_0000 Reserved 0x10_FFFF 22.28 ms at 400 MHz controller clock; 33.34 ms at 266.66 MHz controller clock 0xFF_FFFF335.54 ms at 400 MHz controller clock; 503.31 ms at 266.66 MHz controller clock

21.3.2.4 PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The PCI Express configuration retry timeout register, shown in Figure 21-5, contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.

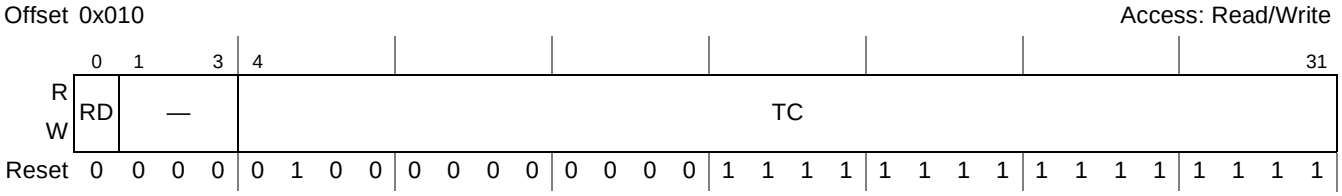


Figure 21-5. PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The fields of the PCI Express configuration retry timeout register are described in Table 21-7.

Table 21-7. PEX_CONF_RTY_TOR Field Descriptions

Bits	Name	Description
0	RD	Retry disable. This bit disables the retry of a configuration transaction that receives a CRS status response packet. 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. 1 Disable retry of a configuration transaction regardless of receiving a CRS status response.
1–3	—	Reserved
4–31	TC	Timeout counter. This is the value that is used to load the CRS response counter. One TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz and 30 ns at 266.66 MHz. Timeout period based on different TC settings: 0x000_0000 Reserved 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

21.3.2.5 PCI Express Configuration Register (PEX_CONFIG)

The PCI Express configuration register, shown in Figure 21-6, contains various control switches for the controller.

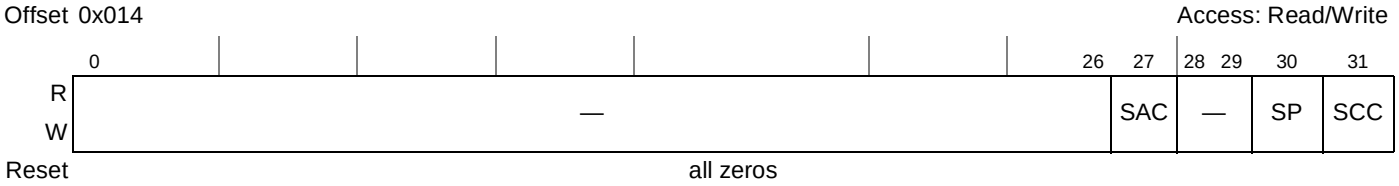


Figure 21-6. PCI Express Configuration Register (PEX_CONFIG)

The fields of the PCI Express configuration register are described in Table 21-8.

Table 21-8. PEX_CONFIG Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	SAC	Sense ASPM Control. This bit controls the default value of ASPM of PEX Link Control Register's bit 0. See Section 21.3.9.11, "PCI Express Link Control Register—0x5C," for more information.
28–29	—	Reserved
30	SP	Slot Present. This bit controls the default value of the PCI Express capabilities register [slot] bit. See Section 21.3.9.6, "PCI Express Capabilities Register—0x4E," for more information.
31	SCC	Slot Clock Configuration. This bit controls the default value of the PCI Express link status register [SCC] bit. See Section 21.3.9.12, "PCI Express Link Status Register—0x5E," for more information.

21.3.3 PCI Express Power Management Event and Message Registers

21.3.3.1 PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The PCI Express PME and message detect register, shown in Figure 21-7, logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.

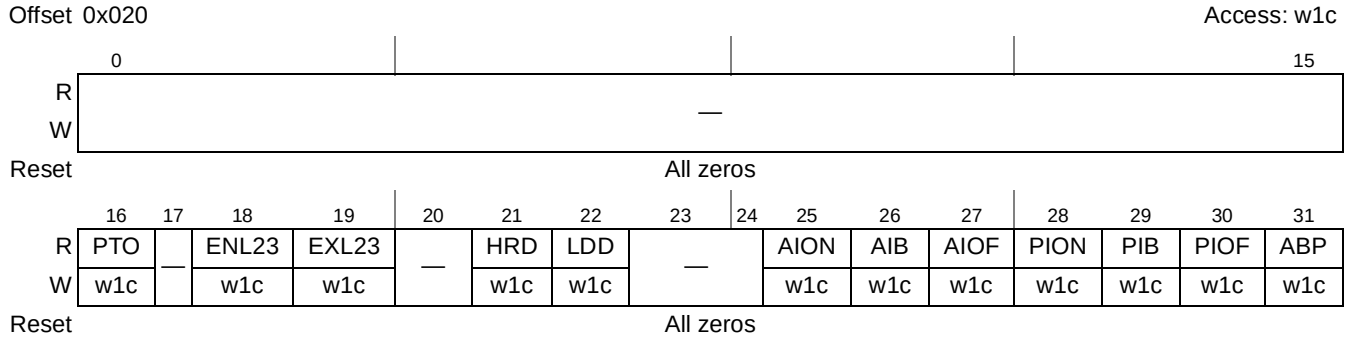


Figure 21-7. PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The fields of the PCI Express PME and message detect register are described in Table 21-9.

Table 21-9. PEX_PME_MES_DR Field Descriptions

Bits	Name	Description
0–15	—	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	—	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	—	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 will clean up all outstanding transactions. Link retraining will take 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 will clean up all outstanding transactions. Link retraining will take 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
23–24	—	Reserved
25	AION	Attention indicator on. This bit indicates the detection of an Attention_Indicator_On message. This bit is only valid in EP mode. 1 Attention indicator on message is detected 0 No attention indicator on message detected
26	AIB	Attention indicator blink. This bit indicates the detection of an Attention_Indicator_Blink message. This bit is only valid in EP mode. 1 Attention indicator blink message is detected 0 No attention indicator blink message detected
27	AIOF	Attention indicator off. This bit indicates the detection of an Attention_Indicator_Off message. This bit is only valid in EP mode. 1 Attention indicator off message is detected 0 No attention indicator off message detected
28	PION	Power indicator on. This bit indicates the detection of a Power_Indicator_On message. This bit is only valid in EP mode. 1 Power indicator on message is detected 0 No power indicator on message detected

Table 21-9. PEX_PME_MES_DR Field Descriptions (continued)

Bits	Name	Description
29	PIB	Power indicator blink. This bit indicates the detection of an Power_Indicator_Blink message. This bit is only valid in EP mode. 1 Power indicator blink message is detected 0 No power indicator blink message detected
30	PIOF	Power indicator off. This bit indicates the detection of an Power_Indicator_Off message. This bit is only valid in EP mode. 1 Power indicator off message is detected 0 No power indicator off message detected
31	ABP	Attention button pressed. This bit indicates the detection of an Attention_Button_Pressed message. This bit is only valid in EP mode. 1 Attention button press message is detected 0 No attention button press message detected

21.3.3.2 PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The PCI Express PME and message disable register, shown in Figure 21-8, when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.

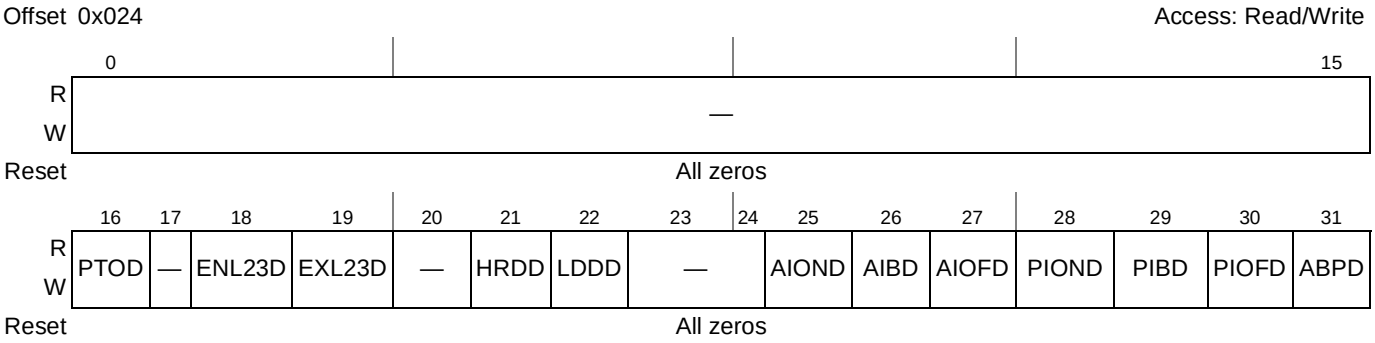


Figure 21-8. PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The fields of the PCI Express PME and message disable register are described in Table 21-10.

Table 21-10. PEX_PME_MES_DISR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOD	PME turn off disable. When set will disable the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
17	—	Reserved
18	ENL23D	Entered_L2/L3 ready disable. When set will disable 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
19	EXL23D	Exited_L2/L3 ready disable. When set will disable 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

Table 21-10. PEX_PME_MES_DISR Field Descriptions (continued)

Bits	Name	Description
20	—	Reserved
21	HRDD	Hot reset detected disable. When set will disable 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 will disable the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
23–24	—	Reserved
25	AIOND	Attention indicator on disable. When set will disable the setting of PEX_PME_MES_DR[AION] bit. 1 Disable attention indicator on message detection 0 Enable attention indicator on message detection
26	AIBD	Attention indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[AIB] bit. 1 Disable attention indicator blink message detection 0 Enable attention indicator blink message detection
27	AIOFD	Attention indicator off disable. When set will disable the setting of PEX_PME_MES_DR[AIOF] bit. 1 Disable attention indicator off message detection 0 Enable attention indicator off message detection
28	PIOND	Power indicator on disable. When set will disable the setting of PEX_PME_MES_DR[PION] bit. 1 Disable power indicator on message detection 0 Enable power indicator on message detection
29	PIBD	Power indicator blink disable. When set will disable the setting of PEX_PME_MES_DR[PIB] bit. 1 Disable power indicator blink message detection 0 Enable power indicator blink message detection
30	PIOFD	Power indicator off disable. When set will disable the setting of PEX_PME_MES_DR[PIOF] bit. 1 Disable power indicator off message detection 0 Enable power indicator off message detection
31	ABPD	Attention button pressed disable. When set will disable the setting of PEX_PME_MES_DR[ABP] bit. 1 Disable attention button press message detection 0 Enable attention button press message detection

21.3.3.3 PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

The PCI Express PME and message interrupt enable register, shown in Figure 21-9, 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.

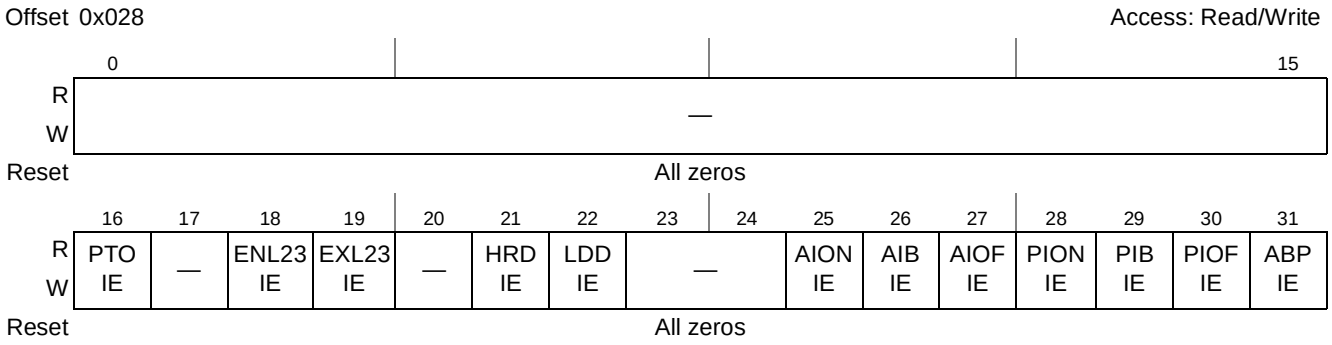


Figure 21-9. PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

Table 21-11 shows the fields of the PCI Express PME and message interrupt enable register.

Table 21-11. PEX_PME_MES_IER Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 will generate an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
17	—	Reserved
18	ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 will generate an interrupt. 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 will generate an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
20	—	Reserved
21	HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 will generate 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 will generate an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation
23–24	—	Reserved
25	AIONIE	Attention indicator on interrupt enable. When set and PEX_PME_MES_DR[AION]=1 will generate an interrupt. 1 Enable attention indicator on message interrupt generation 0 Disable attention indicator on message interrupt generation
26	AIBIE	Attention indicator blink interrupt enable. When set and PEX_PME_MES_DR[AIB]=1 will generate an interrupt. 1 Enable attention indicator blink message interrupt generation 0 Disable attention indicator blink message interrupt generation

Table 21-11. PEX_PME_MES_IER Field Descriptions (continued)

Bits	Name	Description
27	AIOFIE	Attention indicator off interrupt enable. When set and PEX_PME_MES_DR[AIOF]=1 will generate an interrupt. 1 Enable attention indicator off message interrupt generation 0 Disable attention indicator off message interrupt generation
28	PIONIE	Power indicator on interrupt enable. When set and PEX_PME_MES_DR[PION]=1 will generate an interrupt. 1 Enable power indicator on message interrupt generation 0 Disable power indicator on message interrupt generation
29	PIBIE	Power indicator blink interrupt enable. When set and PEX_PME_MES_DR[PIB]=1 will generate an interrupt. 1 Enable power indicator blink message interrupt generation 0 Disable power indicator blink message interrupt generation
30	PIOFIE	Power indicator off interrupt enable. When set and PEX_PME_MES_DR[PIOF]=1 will generate an interrupt. 1 Enable power indicator off message interrupt generation 0 Disable power indicator off message interrupt generation
31	ABPIE	Attention button pressed interrupt enable. When set and PEX_PME_MES_DR[ABP]=1 will generate an interrupt. 1 Enable attention button press message interrupt generation 0 Disable attention button press message interrupt generation

21.3.3.4 PCI Express Power Management Command Register (PEX_PMCR)

The PCI Express power management command register, shown in Figure 21-10, provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

Figure 21-10. PCI Express Power Management Command Register (PEX_PMCR)

The fields of the PCI Express power management command register are described in Table 21-12.

Table 21-12. PEX_PMCR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	SPMES	Set PME status. This will set the PME status bit and if PME is enabled (see Section 21.3.9.3, “PCI Express Power Management Status and Control Register—0x48,” on page 21-71 for more information) it will transmit a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.

Table 21-12. PEX_PMCR Field Descriptions (continued)

Bits	Name	Description
30	EXL2S	Exit L2 state. When set will exit 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 will broadcast a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing

21.3.4 PCI Express IP Block Revision Registers

21.3.4.1 IP Block Revision Register 1 (PEX_IP_BLK_REV1)

The IP block revision register 1 is shown in Figure 21-11.

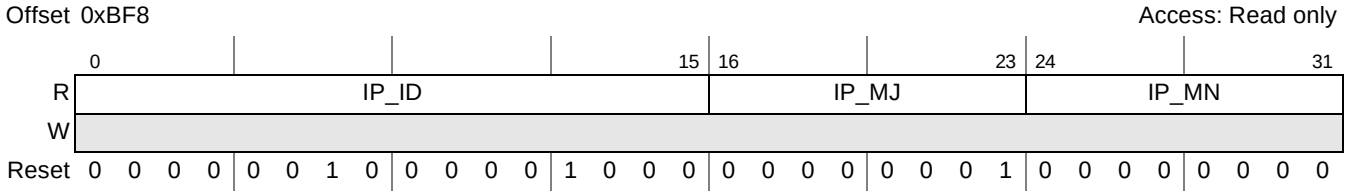


Figure 21-11. IP Block Revision Register 1

Table 21-13 contains descriptions of the fields of the IP block revision register 1.

Table 21-13. PCI Express IP Block Revision Register 1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	Block ID
16–23	IP_MJ	Block Major Revision
24–31	IP_MN	Block Minor Revision

21.3.4.2 IP Block Revision Register 2 (PEX_IP_BLK_REV2)

The IP block revision register 2 is shown in Figure 21-12.

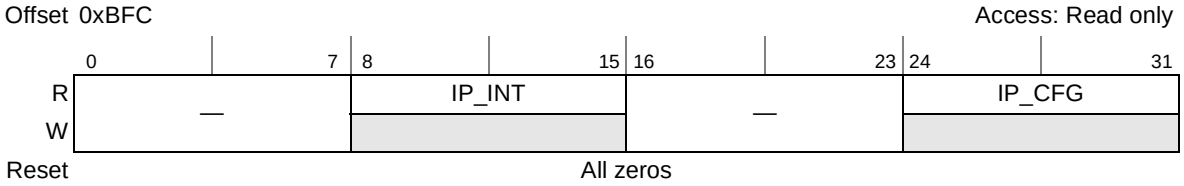


Figure 21-12. IP Block Revision Register 2

Table 21-14 contains descriptions of the fields of the IP block revision register 2.

Table 21-14. PCI Express IP Block Revision Register 2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	Block integration option
16–23	—	Reserved
24–31	IP_CFG	Block configuration option

21.3.5 PCI Express ATMU Registers

21.3.5.1 PCI Express Outbound ATMU Registers

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 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, there is no checking on whether the translated address actually hits into the memory base/limit range. It will just pass it through as is.

Figure 21-13 shows the outbound transaction flow.

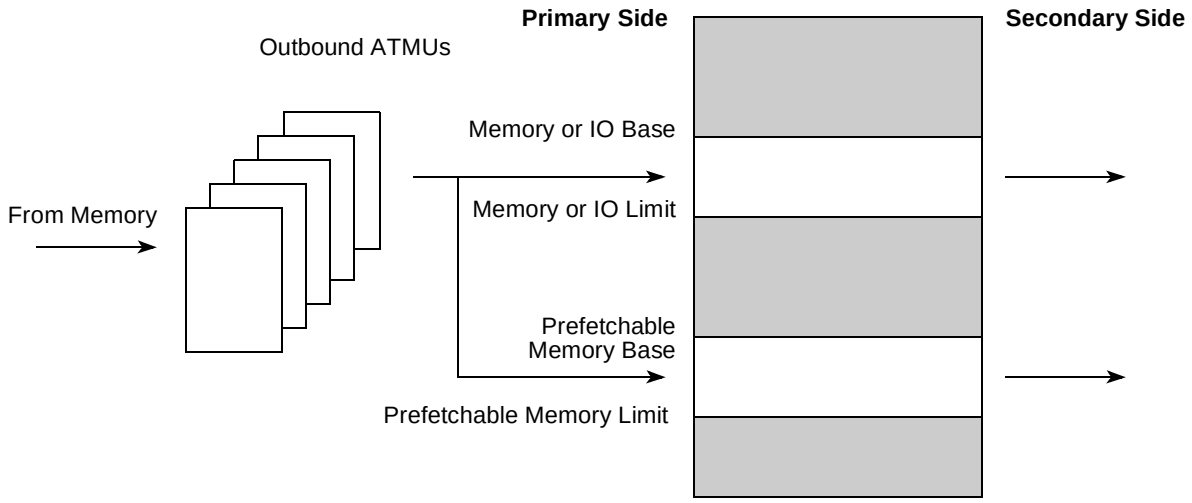


Figure 21-13. RC Outbound Transaction Flow

21.3.5.1.1 PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

The PCI Express outbound translation address registers, shown in Figure 21-14, 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.

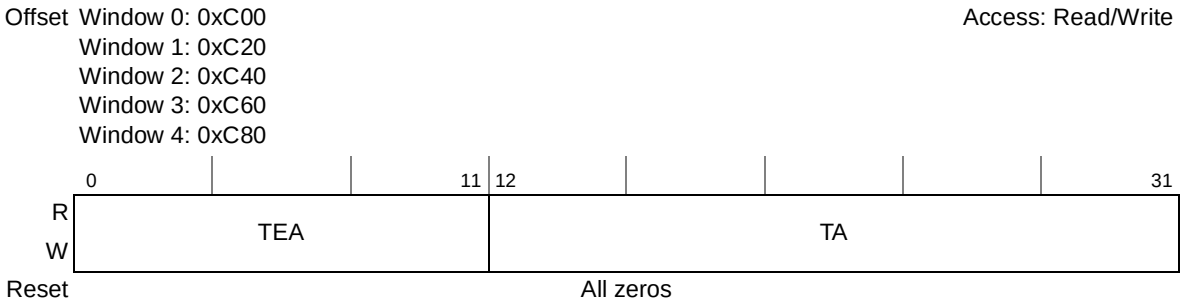


Figure 21-14. PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

Table 21-15 describes the fields of the PCI Express outbound translation address registers.

Table 21-15. PEXOTAR*n* Field Descriptions

Bits	Name	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].

21.3.5.1.2 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

The PCI Express outbound translation extended address registers, shown in Figure 21-15, contain the most-significant bits of a 64 bit translation address.

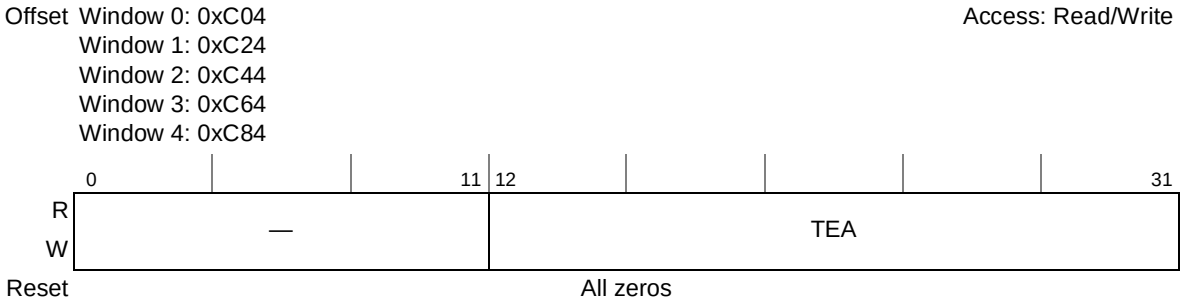


Figure 21-15. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

Table 21-16 describes the fields of the PCI Express outbound translation extended address registers.

Table 21-16. PCI Express Outbound Extended Address Translation Register *n* Field Descriptions

Bits	Name	Description
0–11	—	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].

21.3.5.1.3 PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

The PCI Express outbound window base address registers, shown in Figure 21-16, 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.

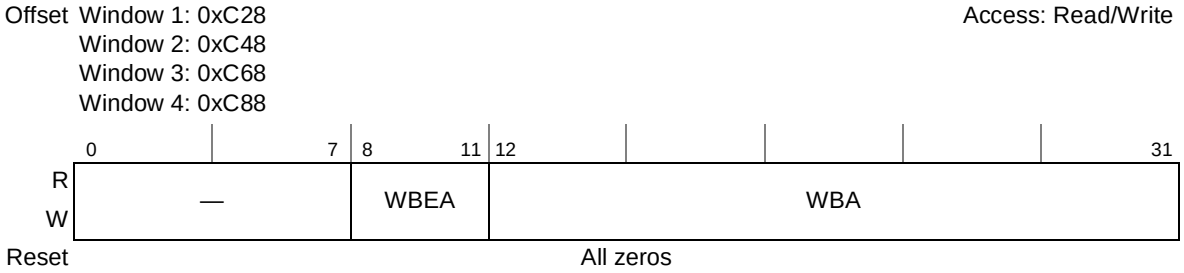


Figure 21-16. PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

Table 21-17 describes the fields of the PCI Express outbound window base address registers.

Table 21-17. PCI Express Outbound Window Base Address Register *n* Field Descriptions

Bits	Name	Description
0–7	—	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].

21.3.5.1.4 PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)

The PCI Express outbound window attributes registers, shown in Figure 21-17 and Figure 21-18, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed. Figure 21-17 shows the outbound window attributes register 0 (PEXOWAR0).

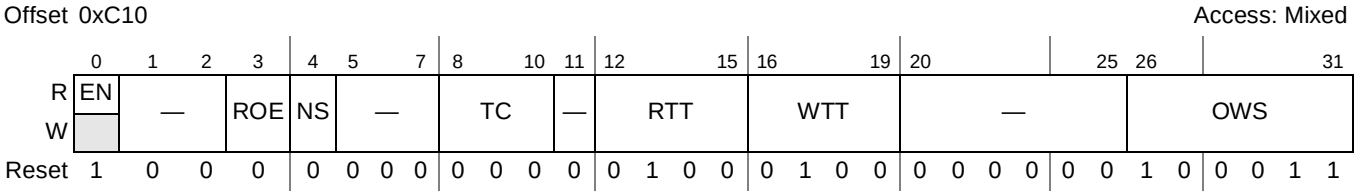


Figure 21-17. PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)

Figure 21-18 shows the PCI Express outbound window attributes registers 1–4 (PEXOWAR*n*).

Offset Window 1: 0xC30
Window 2: 0xC50
Window 3: 0xC70
Window 4: 0xC90

Access: Read/Write

	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	EN	—			ROE	NS	—		TC			—	RTT				WTT				—						OWS					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 21-18. PCI Express Outbound Window Attributes Registers 1–4 (PEXOWAR*n*)

Table 21-18 describes the fields of the PCI Express outbound window attributes registers.

Table 21-18. PEXOWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1–2	—	Reserved
3	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set will enable the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set will enable the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7	—	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. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7 **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.
11	—	Reserved

Table 21-18. PEXOWAR*n* Field Descriptions (continued)

Bits	Name	Description
12–15	RTT	Read transaction type. Read transaction type to run on the PCI Express link 0000 Reserved 0000 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 ... Reserved 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. ... Reserved 1111 Reserved
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0000 Reserved 0001 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. Note that inbound write transactions on one PCI express port must not translate to outbound configuration write transactions on another PCI Express port. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. ... Reserved 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. Note that inbound write transactions on one PCI express port must not translate to outbound I/O write transactions on another PCI Express port. ... Reserved 1111 Reserved
20–25	—	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 ... 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

21.3.5.2 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

21.3.5.2.1 EP Inbound ATMU Implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible via PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism. Note that host software must program these BAR using configuration type 0 cycles. There are 4 inbound BARs.

- Default inbound window BAR0 at configuration address 0x10 (32-bit). Also known as PEXCSRBAR. This is a fixed 1-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 for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR is used.

21.3.5.2.2 RC Inbound ATMU Implementation

In RC mode, the PEXIWBAR[1–3] registers reside outside of the type 1 header; PEXIWBAR0 is the only inbound BAR that resides in the Type 1 header (at offset 0x10).

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 21-19 shows the inbound transaction flow in RC mode.

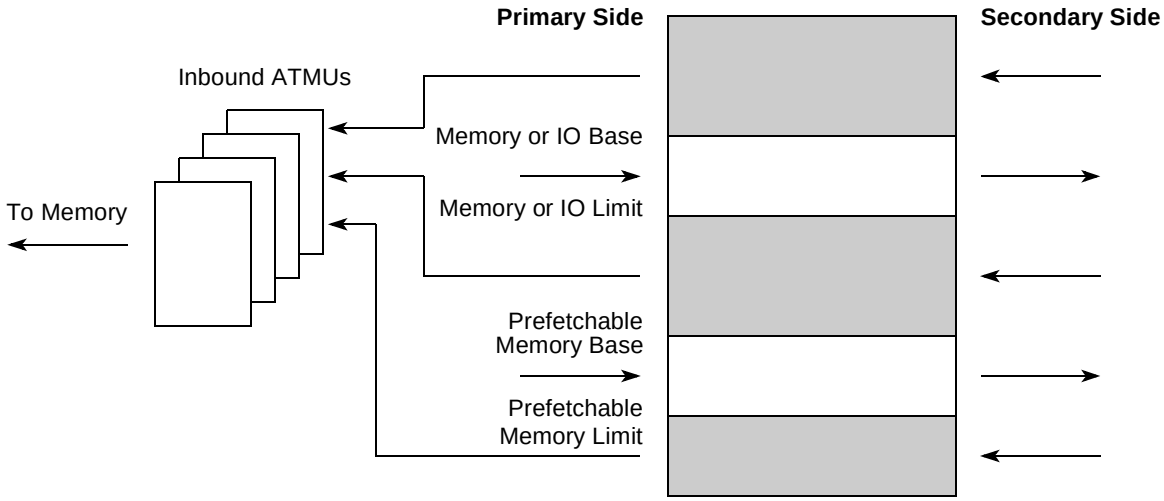


Figure 21-19. RC Inbound Transaction Flow

21.3.5.2.3 PCI Express Inbound Translation Address Registers (PEXITAR*n*)

The PCI Express inbound translation address registers, shown in Figure 21-20, contain the translated internal platform address to be used. Note that PEXITAR0 does not exist in the memory-mapped space; it is a fixed 1-Mbyte translation to the internal configuration (CCSRBAR) space.

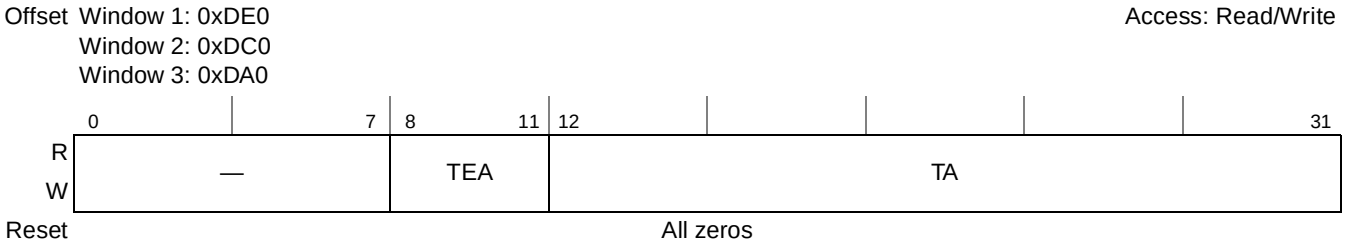


Figure 21-20. PCI Express Inbound Translation Address Registers (PEXITAR*n*)

Table 21-19 describes the fields of the PCI Express inbound translation address registers.

Table 21-19. PCI Express Inbound Translation Address Registers Field Descriptions

Bits	Name	Description
0–7	—	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].

21.3.5.2.4 PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

The PCI Express inbound window base address registers, shown in Figure 21-21, 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, PEXIWBAR0 is located in the PCI Express type 1 configuration header space and PEXIWBAR[1–3] registers 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 will return 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.

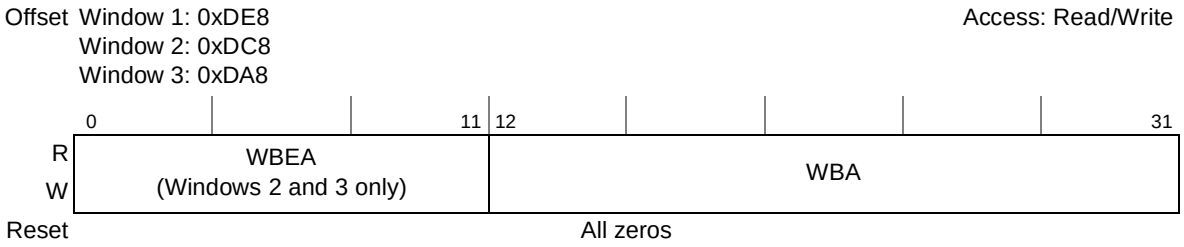


Figure 21-21. PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

Table 21-20 describes the fields of the PCI Express inbound window base address registers.

Table 21-20. PCI Express Inbound Window Base Address Register Field Descriptions

Bits	Name	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].

21.3.5.2.5 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

The PCI Express inbound window base extended address registers, shown in Figure 21-22, contain the most-significant bits of a 64 bit base address.

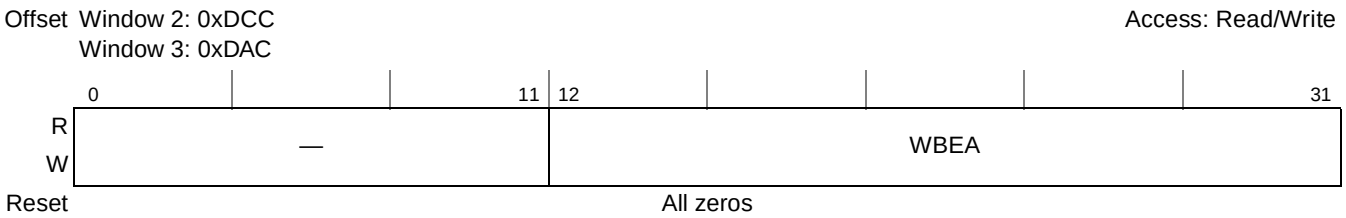


Figure 21-22. PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

Table 21-21 describes the fields of the PCI Express inbound window base extended address registers.

Table 21-21. PCI Express Inbound Window Base Extended Address Register Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

21.3.5.2.6 PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

The PCI Express inbound window attributes registers, shown in Figure 21-23, define the window sizes to translate and other attributes for the translations. 64 Gbytes is the largest window size allowed.

Offset Window 1: 0xDF0
Window 2: 0xDD0
Window 3: 0xDB0

Access: Read/Write

	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	EN	—	PF	—					TRGT				RTT				WTT				—						IWS					
Reset	0	0	1	0	0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1

Figure 21-23. PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

Table 21-22 describes the fields of the PCI Express inbound window attributes registers.

Table 21-22. PCI Express Inbound Window Attributes Registers Field Descriptions

Bits	Name	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	—	Reserved
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	—	Reserved
8–11	TRGT	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. 0000 PCI Express 3 — PCI Express controller 3 should not use this encoding 0001 PCI Express 2 — PCI Express controller 2 should not use this encoding 0010 PCI Express 1 — PCI Express controller 1 should not use this encoding 0011-1011 Reserved 1100 Serial RapidIO 1101-1110 Reserved 1111 Local memory space **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.

Table 21-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
12–15	RTT	Read transaction type. Read transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved
16–19	WTT	Write transaction type. Write transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved

Table 21-22. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
20–25	—	Reserved
26–31	IWS	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. 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

21.3.6 PCI Express Error Management Registers

21.3.6.1 PCI Express Error Detect Register (PEX_ERR_DR)

The PCI Express error detect register, shown in Figure 21-24, 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 will set the appropriate error bits in the error detection registers, but only the first error for a particular unit will 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.

Offset 0xE00 Access: w1c

	0	1	7	8	9	10	11	12	13	14	15
R	ME	—		PCT	—	PCAC	PNM	CDNSC	CRSNC	ICCA	IACA
W	w1c			w1c		w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros										

	16	17	18	19	20	21	22	23	24	31
R	CRST	MIS	IOIS	CIS	CIEP	IOIEP	OAC	IOIA	—	
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c		
Reset	All zeros									

Figure 21-24. PCI Express Error Detect Register (PEX_ERR_DR)

Table 21-23 describes the fields of the PCI Express error detect register.

Table 21-23. PCI Express Error Detect Register Field Descriptions

Bits	Name	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.
1–7	—	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. 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	—	Reserved
10	PCAC	PCI Express completer abort (CA) completion. A completion with CA status was received. 1 A completion with CA status was detected. 0 No completion with CA status detected.
11	PNM	PCI Express no map. Detect an inbound transaction that was not mapped to any inbound windows. In RC mode, a completion without data (Cpl) packet with a UR completion status is sent back 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 will not set this bit. 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 (such as UR, CA or CRS status). 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. 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. 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. 1 Invalid ATMU configuration access was detected 0 No invalid ATMU configuration access detected
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 Section 21.3.2.4, “PCI Express Configuration Retry Timeout Register (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

Table 21-23. PCI Express Error Detect Register Field Descriptions (continued)

Bits	Name	Description
17	MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See Section 21.4.1.9.1, "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
24–31	—	Reserved

21.3.6.2 PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

The PCI Express error interrupt enable register, shown in Figure 21-25, allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.

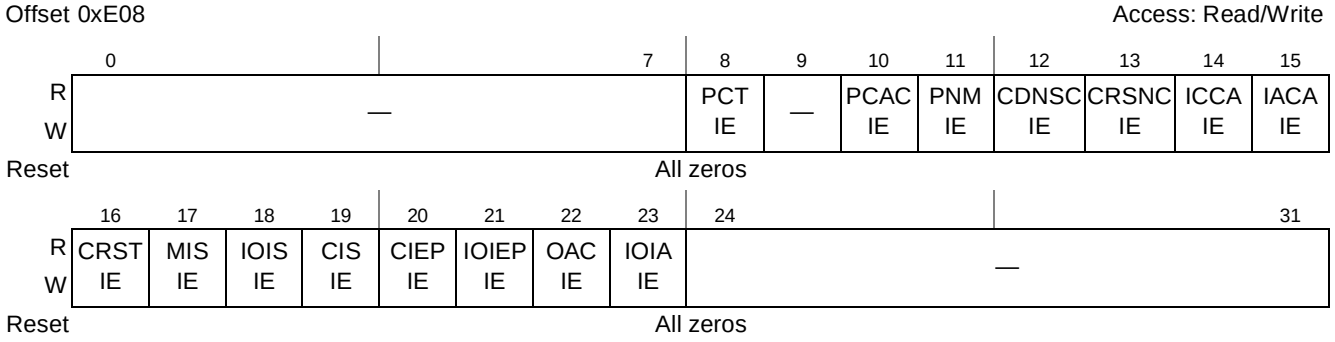


Figure 21-25. PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

Table 21-24 describes the fields of the PCI Express error interrupt enable register.

Table 21-24. PCI Express Error Interrupt Enable Register Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 will generate an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
9	—	Reserved
10	PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 will generate 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 will generate 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 will generate 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 will generate 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 will generate 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 will generate 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 will generate 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 will generate 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 will generate 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 will generate an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation

Table 21-24. PCI Express Error Interrupt Enable Register Field Descriptions (continued)

Bits	Name	Description
20	CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 will generate an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation
21	IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 will generate 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 will generate 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 will generate an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
24–31	—	Reserved

21.3.6.3 PCI Express Error Disable Register (PEX_ERR_DISR)

The PCI Express error disable register, shown in Figure 21-26, controls the setting of the PCI Express error detect register's bits.

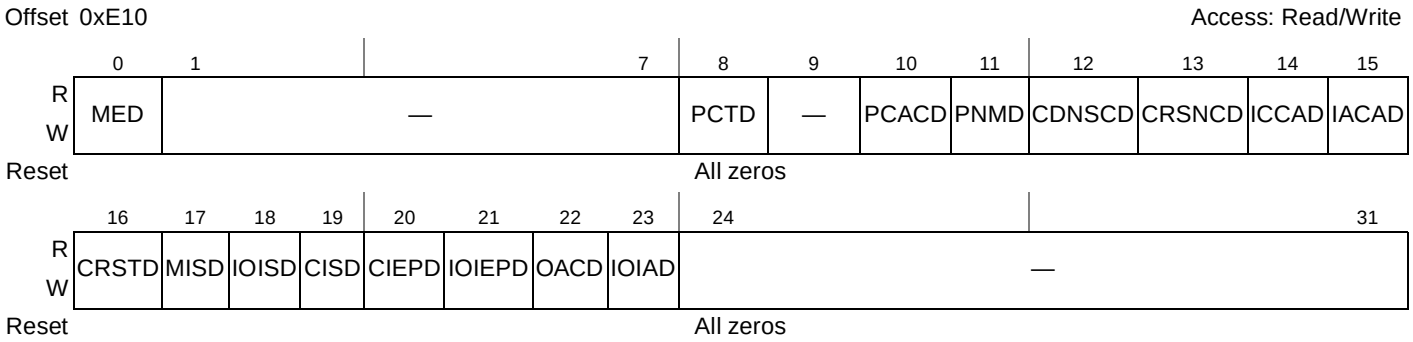


Figure 21-26. PCI Express Error Disable Register (PEX_ERR_DISR)

Table 21-25 describes the fields of the PCI Express error disable register.

Table 21-25. PCI Express Error Disable Register Field Descriptions

Bits	Name	Description
0	MED	Multiple errors disable. When set will disable the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
1–7	—	Reserved
8	PCTD	PCI Express completion time-out disable. When set will disable the setting of PEX_ERR_DR[PCT] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection

Table 21-25. PCI Express Error Disable Register Field Descriptions (continued)

Bits	Name	Description
9	—	Reserved
10	PCACD	PCI Express CA completion disable. When set will disable 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 will disable the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection
12	CDNSCD	Completion with data not successful disable. When set will disable 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 will disable 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 will disable the setting of PEX_ERR_DR[ICCA] bit. 1 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection 0 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection
15	IACAD	Invalid ATMU configuration access. When set will disable the setting of PEX_ERR_DR[IACA] bit. 1 Disable invalid ATMU configuration access detection 0 Enable invalid ATMU configuration access detection
16	CRSTD	CRS thresholded disable. When set will disable the setting of PEX_ERR_DR[CRST] bit. 1 Disable CRS threshold detection 0 Enable CRS threshold detection
17	MISD	Message invalid size disable. When set will disable the setting of PEX_ERR_DR[MIS] bit. 1 Disable invalid outbound message size detection 0 Enable invalid outbound message size detection
18	IOISD	I/O invalid size disable. When set will disable the setting of PEX_ERR_DR[IOIS] bit. 1 Disable invalid outbound I/O size detection 0 Enable invalid outbound I/O size detection
19	CISD	Configuration invalid size disable. When set will disable the setting of PEX_ERR_DR[CIS] bit. 1 Disable invalid outbound configuration size detection 0 Enable invalid outbound configuration size detection
20	CIEPD	Configuration invalid EP disable. When set will disable the setting of PEX_ERR_DR[CIEP] bit. 1 Disable outbound configuration transaction EP mode detection 0 Enable outbound configuration transaction EP mode detection
21	IOIEPD	I/O invalid EP disable. When set will disable the setting of PEX_ERR_DR[IOEP] bit. 1 Disable outbound I/O transaction EP mode detection 0 Enable outbound I/O transaction EP mode detection
22	OACD	Outbound ATMU crossing disable. When set will disable the setting of PEX_ERR_DR[OAC] bit. 1 Disable outbound crossing ATMU detection 0 Enable outbound crossing ATMU detection

Table 21-25. PCI Express Error Disable Register Field Descriptions (continued)

Bits	Name	Description
23	IOIAD	I/O invalid address disable. When set will disable the setting of PEX_ERR_DR[IOIA] bit. 1 Disable greater than 4G I/O address detection 0 Enable greater than 4G I/O address detection
24–31	—	Reserved

21.3.6.4 PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

The PCI Express error capture status register, shown in Figure 21-27, 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.

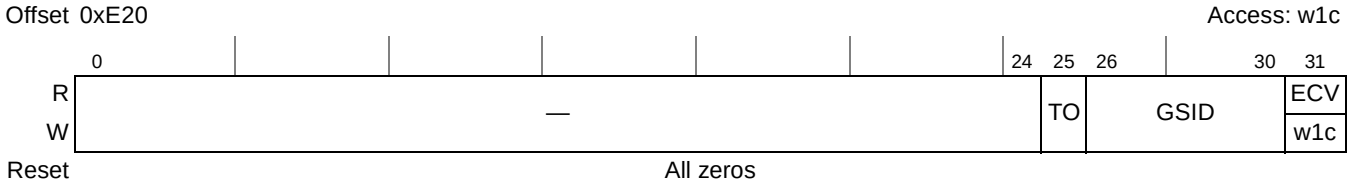


Figure 21-27. PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

Table 21-26 describes the fields of the PCI Express error capture status register.

Table 21-26. PCI Express Error Capture Status Register Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25	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.
26–30	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. 0 0000 PCI Express 3 0 0001 PCI Express 2 0 0010 PCI Express 1 0 1100 SRIO 0 1110 TLU 1, TLU 2 1 0000 Processor 0 instruction 1 0001 Processor 0 data 1 0010 Processor 1 instruction 1 0011 Processor 1 data 1 0101 DMA 1 1 0110 DMA 2 1 0111 SAP, PME 1 1000 eTSEC 1, eTSEC 2, Security 1 1010 eTSEC 3, eTSEC 4, Boot sequencer 1 1100 RapidIO Message Unit, Doorbell, Port-write All other settings reserved.
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.

21.3.6.5 PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R0 allows 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.

21.3.6.5.1 PEX_ERR_CAP_R0—Outbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 21-28.

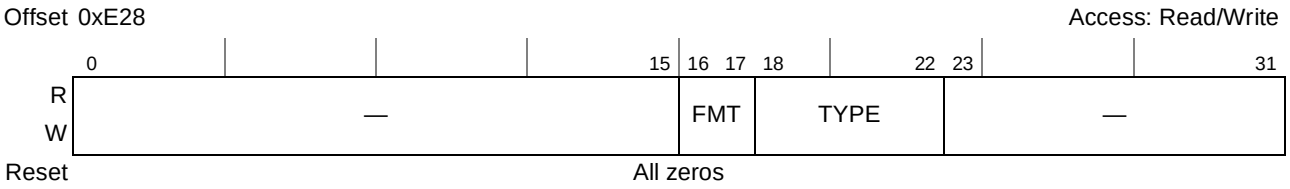


Figure 21-28. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction

Table 21-27 describes the fields of the PCI Express error capture register 0 for the case when the error is caused by an outbound transaction from an internal source.

Table 21-27. PCI Express Error Capture Register 0 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–15	—	Reserved
16-17	FMT	PCI Express format. This field indicates the PCI Express packet format. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
18–22	TYPE	PCI Express type. This field indicates the PCI express packet type. See PCI Express Spec 1.0a for more information on 3 or 4 DW (4-byte) header format.
23–31	—	Reserved

21.3.6.5.2 PEX_ERR_CAP_R0—Inbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 21-29.

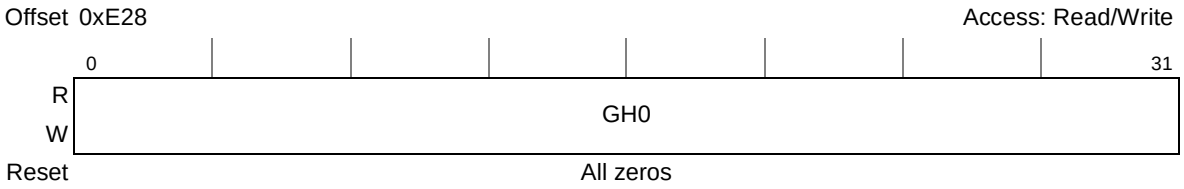


Figure 21-29. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0)
External Source, Inbound Transaction

Table 21-28 describes the fields of PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source.

Table 21-28. PCI Express Error Capture Register 0 Field Descriptions
External Source, Inbound Transaction

Bits	Name	Description
0–31	GH0	PCI Express first DW (4-byte) header. This field contains the first DW (4-byte) of the captured PCI Express packet header. 27–31 TYPE 25–26 FMT 20–24 Reserved 17–19 TC 16 Reserved 14–15 LENGTH[9:8] 12–13 Reserved 10–11 ATTR 9 EP 8 TD 0–7 LENGTH[7:0]

21.3.6.6 PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R1 allows 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.

21.3.6.6.1 PEX_ERR_CAP_R1—Outbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 21-30.

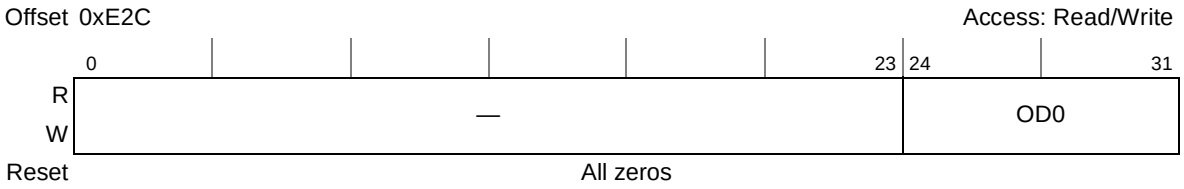


Figure 21-30. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) Internal Source, Outbound Transaction

Table 21-29 describes the fields of PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source.

Table 21-29. PCI Express Error Capture Register 1 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–23	—	Reserved
24–31	OD0	Internal platform transaction information. Reserved for factory debug.

21.3.6.6.2 PEX_ERR_CAP_R1—Inbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 21-31.

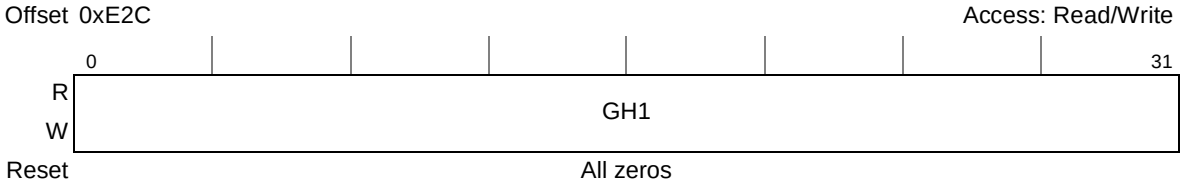


Figure 21-31. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) External Source, Inbound Transaction

Table 21-30 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 21-28) indicate the error was caused by an inbound completion transaction.

Table 21-30. PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound CompletionTransaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Comp ID[15:8] 16–23 Comp ID[7:0] 12–15 Byte Count[11:8] 11 BCM 8–10 Comp Status 0–7 Byte Count[7:0]

Table 21-31 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 21-28) indicate the error was caused by an inbound memory request transaction.

Table 21-31. PCI Express Error Capture Register 1 Field Descriptions
External Source, Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Requester ID[15:8] 16–23 Requester ID[7:0] 8–15 Tag[7:0] 4–7 First DW BE[3:0] 0–3 Last DW BE[3:0]

21.3.6.7 PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R2 allows 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.

21.3.6.7.1 PEX_ERR_CAP_R2—Outbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 21-32.

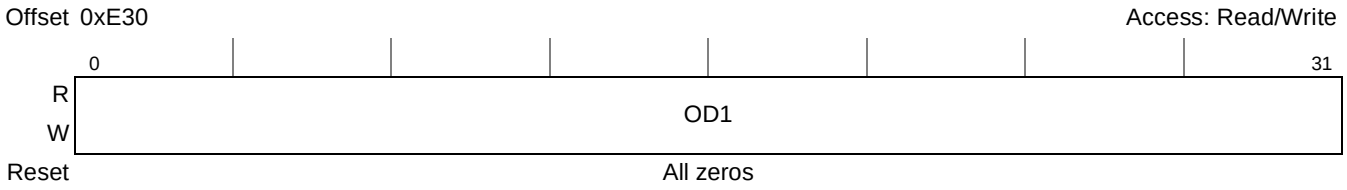


Figure 21-32. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)
Internal Source, Outbound Transaction

Table 21-32 describes the fields of PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source.

Table 21-32. PCI Express Error Capture Register 2 Field Descriptions
Internal Source, Outbound Transaction

Bit	Name	Description
0–31	OD1	Internal platform transaction information. Reserved for factory debug.

21.3.6.7.2 PEX_ERR_CAP_R2—Inbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 21-33.

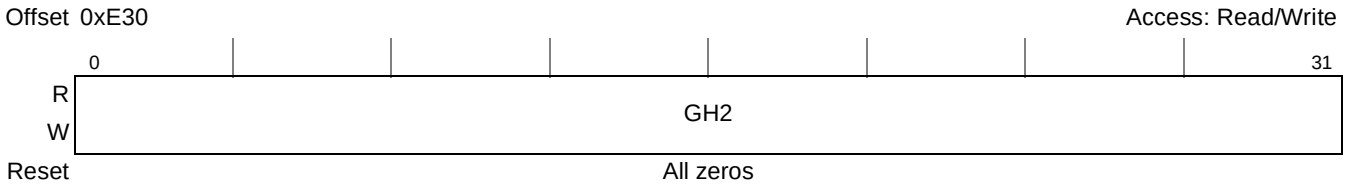


Figure 21-33. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)
External Source, Inbound Transaction

Table 21-33 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 21-28) indicate the error was caused by an inbound completion transaction.

Table 21-33. PCI Express Error Capture Register 2 Field Descriptions
External Source, Inbound Completion Transaction

Bits	Name	Description
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header. 24–31 Req ID[15:8] 16–23 Req ID[7:0] 8–15 Tag[7:0] 1–7 Lower Address[6:0] 0 Reserved

Table 21-34 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 21-28) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R2 captures the 32-bit address for a 3 DW memory request header or the upper half of the 64-bit address for a 4 DW memory request header; the lower half of the 64-bit address for a 4 DW memory request header is captured in PEX_ERR_CAP_R3.

Table 21-34. PCI Express Error Capture Register 2 Field Descriptions
External Source, Inbound Memory Request Transaction

Bits	Name	Description	
		3 DW Header	4 DW Header
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header.	
		24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]	24–31 Address[63:56] 16–23 Address[55:48] 8–15 Address[47:40] 0-7 Address[39:32]

21.3.6.8 PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R3 allows 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.

21.3.6.8.1 PEX_ERR_CAP_R3—Outbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source (that is, PEX_ERR_CAP_STAT[GSID] ≠ 0h02), is shown in Figure 21-34.

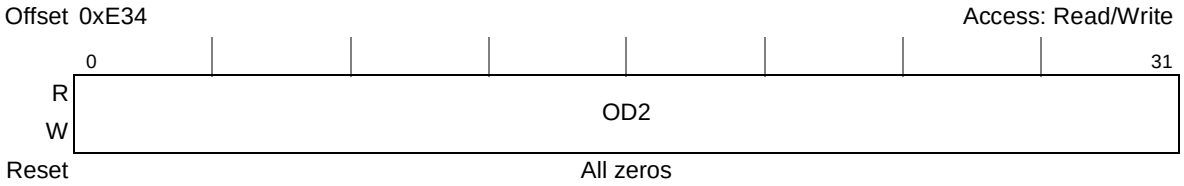


Figure 21-34. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) Internal Source, Outbound Transaction

Table 21-35 describes the fields of PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source.

Table 21-35. PCI Express Error Capture Register 3 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–31	OD2	Internal platform transaction information. Reserved for factory debug.

21.3.6.8.2 PEX_ERR_CAP_R3—Inbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an inbound transaction from an external source (that is, PEX_ERR_CAP_STAT[GSID] = 0h02 for controller 1), is shown in Figure 21-35.

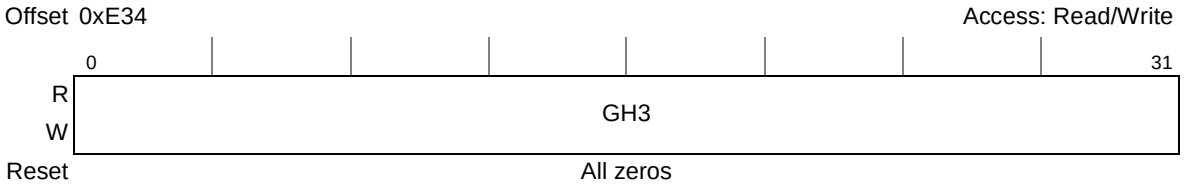


Figure 21-35. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) External Source, Inbound Transaction

Table 21-36 describes the fields of PEX_ERR_CAP_R3 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 21-28) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R3 captures the lower half of the 64-bit address for a 4 DW

memory request header; the upper half of the 64-bit address for a 4 DW memory request header or the 32-bit address for a 3 DW memory request header is captured in PEX_ERR_CAP_R2.

Table 21-36. PEX Error Capture Register 3 Field Descriptions
External Source, Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH3	PEX fourth DW (4-byte) header. This field contains the fourth DW (4-byte) of the captured PCI Express packet header. 24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]

21.3.7 PCI Express Configuration Space Access

There are two methods of accessing the PCI Express configuration header:

- PCI Express outbound ATMU window
- PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA)

21.3.7.1 RC Configuration Register Access

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 PCI Express bus, the device on that bus, the 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 that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 21.4.1.2.1, "Byte Order for Configuration Transactions," for more information.

Note that 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.

21.3.7.1.1 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 bus number, and device number equal to the PCI Express controller's bus number and device number, and the function number is zero, then an internal PCI Express configuration cycle access is performed.

- If the 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 device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the 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 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.

21.3.7.1.2 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 bus number, device number, function number, register, and extended register number sent are decoded from the outbound translated PCI Express address.

- bus number[7:0] = PCI Express address[27:20]
- device number[4:0] = PCI Express address[19:15]
- function number[2:0] = PCI Express address[14:12]

- extended register number[3:0] = PCI Express address[11:8]
- register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the bus number equals the secondary bus number (from the type 1 header) and device number is 0. A Type 1 configuration cycle is sent to the link if 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 will result in a response with error returned.

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.

21.3.7.2 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 up to 4096 bytes of the PCI Express configuration area. 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 bus number or 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 will get a response with an error.

21.3.8 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. The first 16 bytes of the predefined header are defined the same for all PCI Express devices. These common registers are shown in Figure 21-36.

Reserved

Address Offset (Hex)

Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C

Figure 21-36. PCI Express PCI-Compatible Configuration Header Common Registers

The remaining 48 bytes of 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.

21.3.8.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

21.3.8.1.1 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in Figure 21-37, is used to identify the manufacturer of the device.

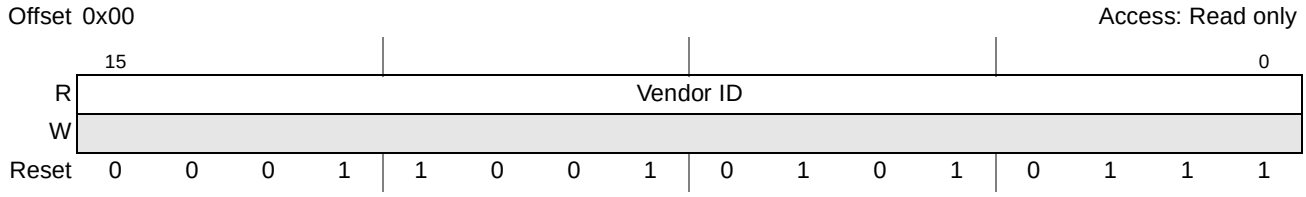


Figure 21-37. PCI Express Vendor ID Register

Table 21-37 describes the vendor ID register fields.

Table 21-37. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

21.3.8.1.2 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in Figure 21-38, is used to identify the device.

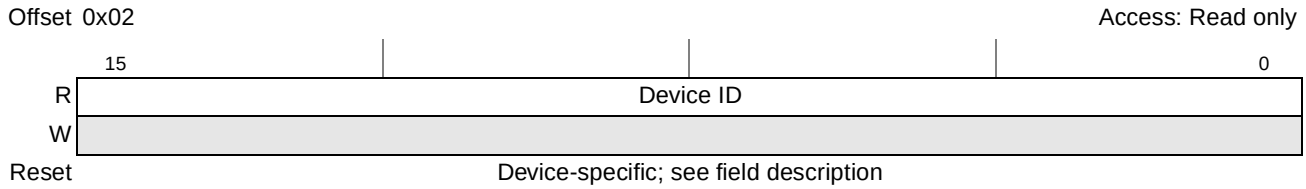


Figure 21-38. PCI Express Device ID Register

Table 21-38 describes the device ID register fields.

Table 21-38. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x0040 MPC8572E 0x0041 MPC8572

21.3.8.1.3 PCI Express Command Register—Offset 0x04

The command register, shown in Figure 21-39, provides control over the ability to generate and respond to PCI Express cycles.

Offset 0x04 Access: Mixed

	15	11	10	9	8	7	6	5	3	2	1	0
R	—		Interrupt Disable	—	SERR	—	Parity error response	—		Bus master	Memory space	I/O space
W												
Reset	All zeros											

Figure 21-39. PCI Express Command Register

Table 21-39 describes the bits of the command register.

Table 21-39. PCI Express Command Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	Interrupt Disable	Controls the ability to generate INTx interrupt messages. 0 Enables INTx interrupt messages 1 Disables INTx interrupt messages Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set.
9	—	Reserved
8	SERR	Controls the reporting of fatal and non-fatal errors detected by the device to the root complex. 0 Disables reporting 1 Enables reporting **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 Section 21.3.9.8, “PCI Express Device Control Register—0x54,” and the advance error reporting capability structure described in sections 21.3.10.1 through 21.3.10.12.
7	—	Reserved
6	Parity error response	Controls whether this PCI Express controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 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. **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 Section 21.3.9.8, “PCI Express Device Control Register—0x54,” and the advance error reporting capability structure described in sections 21.3.10.1 through 21.3.10.12.
5–3	—	Reserved
2	Bus master	Indicates whether this PCI Express device is configured as a master. 0 Disables the ability to generate PCI Express accesses 1 Enables this PCI Express controller to behave as a PCI Express bus master EP mode: Clearing this bit prevent 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. RC mode: Clearing this bit disables the ability of the device to forward memory transactions upstream. This causes any inbound memory transaction to be treated as an unsupported request.

Table 21-39. PCI Express Command Register Field Descriptions (continued)

Bits	Name	Description
1	Memory space	Controls whether this PCI Express device (as a target) responds to memory accesses. 0 This PCI Express device does not respond to PCI Express memory space accesses. 1 This PCI Express device responds to PCI Express memory space accesses. EP mode: Clearing this bit will prevent the device from accepting any memory transaction. RC mode: This bit is ignored. It does not affect outbound memory transaction
0	I/O space	I/O space. 0 This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1 This PCI Express device (as a target) does respond to PCI Express I/O space accesses. EP mode: Clearing this bit will prevent 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. RC mode: This bit is ignored. It does not affect outbound IO transaction.

21.3.8.1.4 PCI Express Status Register—Offset 0x06

The status register, shown in Figure 21-40, is used to record status information for PCI Express related events.

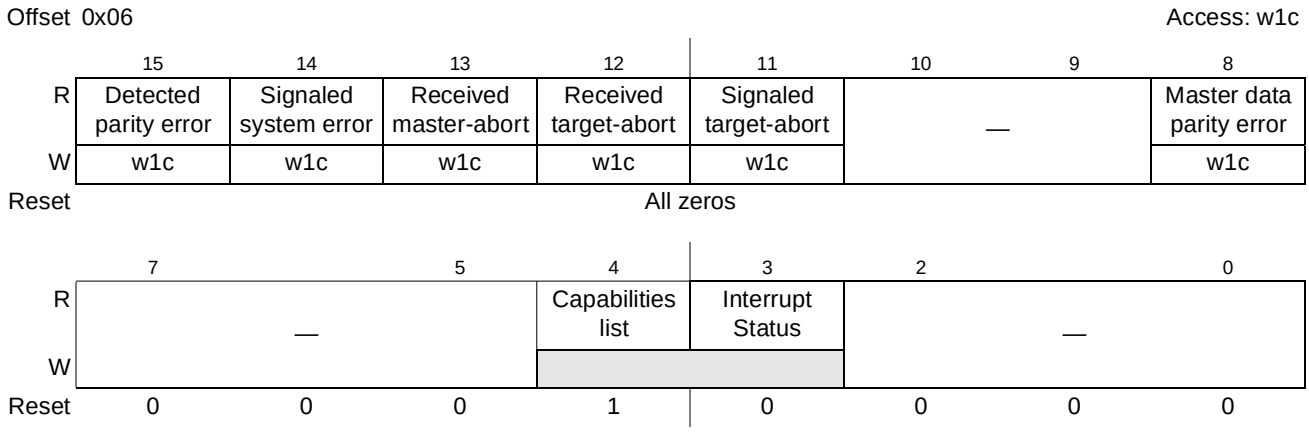


Figure 21-40. PCI Express Status Register

The definition of each bit is given in Table 21-40.

Table 21-40. PCI Express Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error[1]	Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register.
14	Signaled system error[1]	Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.
13	Received master-abort[1]	Set whenever a requestor receives a completion with unsupported request completion status.
12	Received target-abort[1]	Set whenever a device receives a completion with completer abort completion status.
11	Signaled target-abort[1]	Set whenever a device completes a request using completer abort completion status.

Table 21-40. PCI Express Status Register Field Descriptions (continued)

Bits	Name	Description
10–9	—	Reserved
8	Master data parity error detected[1]	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.
7–5	—	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.
2–0	—	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 Section 21.3.9.8, "PCI Express Device Control Register—0x54," and the advance error reporting capability structure described in sections 21.3.10.1 through 21.3.10.12.

21.3.8.1.5 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in Figure 21-41, is used to identify the revision of the device.

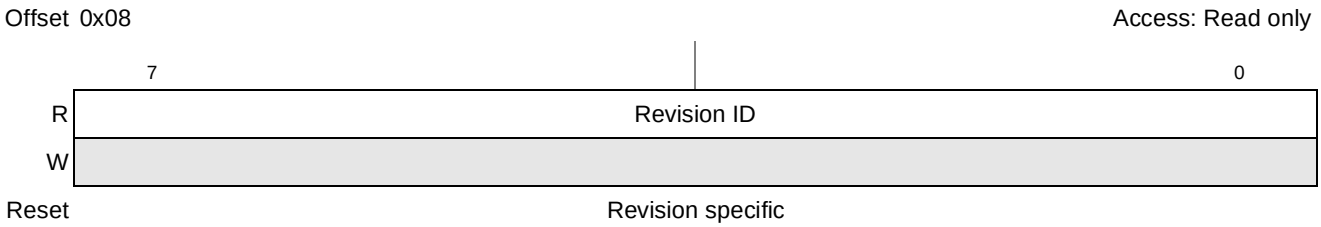


Figure 21-41. PCI Express Revision ID Register

Table 21-41 describes the revision ID register fields.

Table 21-41. PCI Express Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific.

21.3.8.1.6 PCI Express Class Code Register—Offset 0x09

The class code register, shown in Figure 21-42, 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.

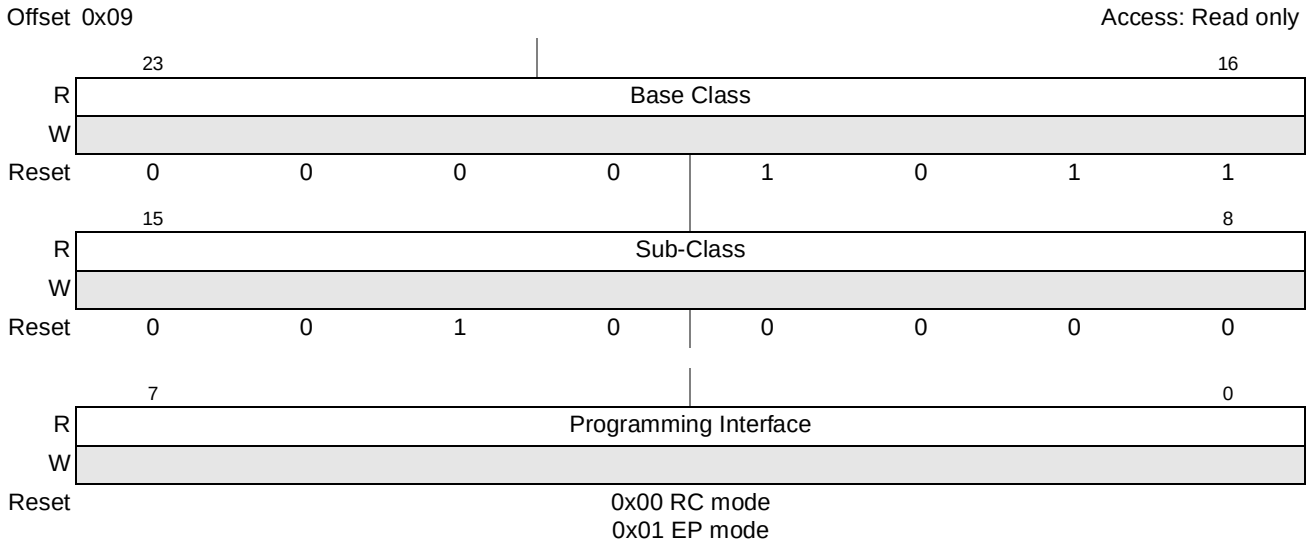


Figure 21-42. PCI Express Class Code Register

Table 21-42 describes the class code register fields.

Table 21-42. PCI Express Class Code Register Field Descriptions

Bits	Name	Description
23–16	Base Class	0x0B—Processor
15–8	Sub-Class	0x20—PowerPC
7–0	Programming Interface	0x00—RC mode 0x01—EP mode

21.3.8.1.7 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in Figure 21-43, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

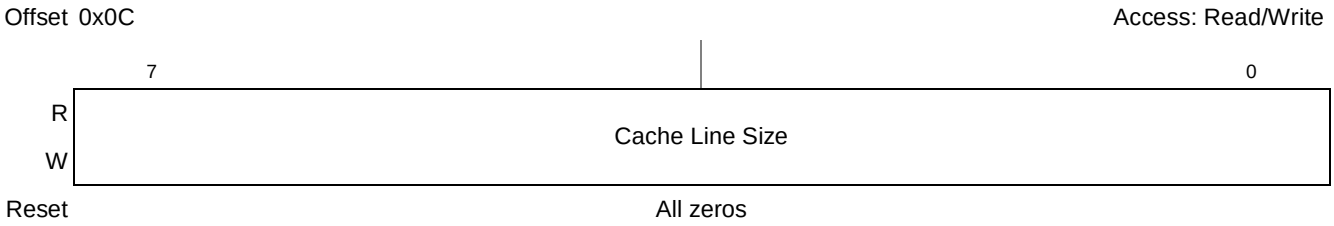


Figure 21-43. PCI Express Bus Cache Line Size Register

Table 21-43 describes the cache line size register.

Table 21-43. PCI Express Bus Cache Line Size Register Field Descriptions

Bits	Name	Description
7–0	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.

21.3.8.1.8 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in Figure 21-44, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

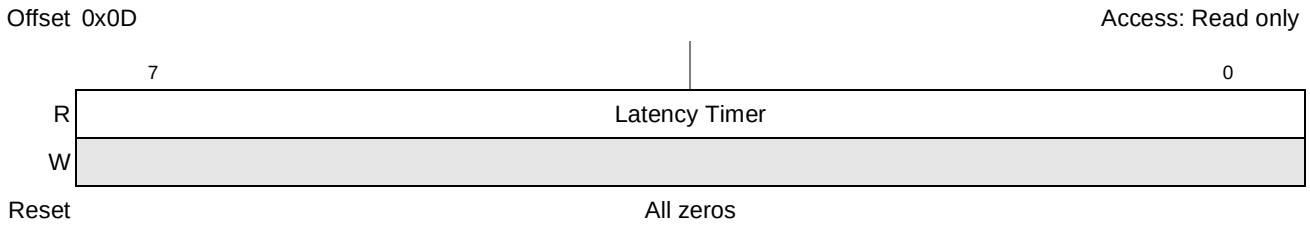


Figure 21-44. PCI Express Bus Latency Timer Register

Table 21-44 describes the PCI Express latency timer register (PLTR).

Table 21-44. PCI Express Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7–0	Latency Timer	Note that for PCI Express operation this register is ignored.

21.3.8.1.9 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in Figure 21-43, is used to identify the layout of the PCI compatible header.

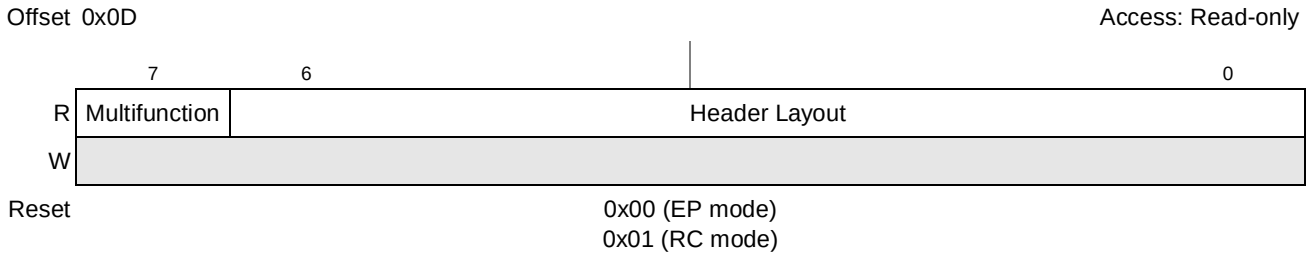


Figure 21-45. PCI Express Bus Latency Timer Register

Table 21-44 describes the PCI Express header type register.

Table 21-45. PCI Express Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7	Multifunction	Identifies whether a device supports multiple functions 0 Single function device 1 Multiple function device
6–0	Header Layout	0x00 Endpoint. See Figure 21-46 for type 0 layout. 0x01 Root Complex. See Figure 21-58 for type 1 layout. All other encodings reserved.

21.3.8.1.10 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

21.3.8.2 Type 0 Configuration Header

The type 0 header is shown in Figure 21-46.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Registers				10 14 18 1C 20 24
				28
Subsystem ID		Subsystem Vendor ID		2C
				30
			Capabilities Pointer	34
Expansion ROM Base Address				38
MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line	3C

Figure 21-46. PCI Express PCI-Compatible Configuration Header—Type 0

Section 21.3.8.1, "Common PCI Compatible Configuration Header Registers," describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 0 header beginning at offset 0x10.

21.3.8.2.1 PCI Express Base Address Registers—0x10–0x27

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 Section 21.3.5.2, "PCI Express Inbound ATMU Registers," for more information.

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 21-47.

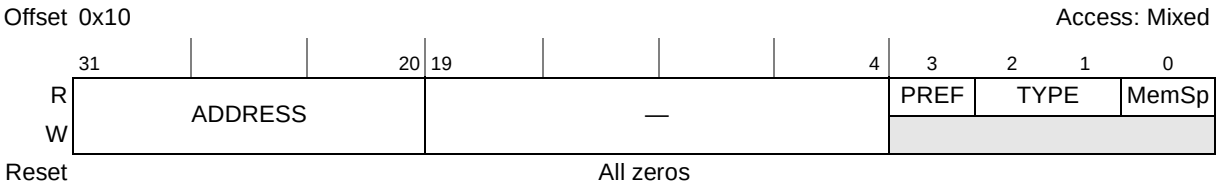


Figure 21-47. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 21-46 describes the PCI Express configuration and status register base address register.

Table 21-46. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space. The 32-bit memory BAR is shown in Figure 21-48.

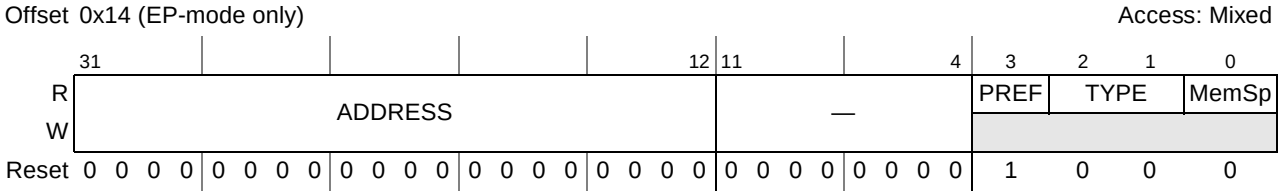


Figure 21-48. 32-Bit Memory Base Address Register (BAR1)

Table 21-47 describes the PCI Express 32-bit memory BAR fields.

Table 21-47. 32-Bit Memory Base Address Register (BAR1) Field Descriptions

Bits	Name	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	—	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.

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. The 64-bit low memory BARs are shown in Figure 21-49.

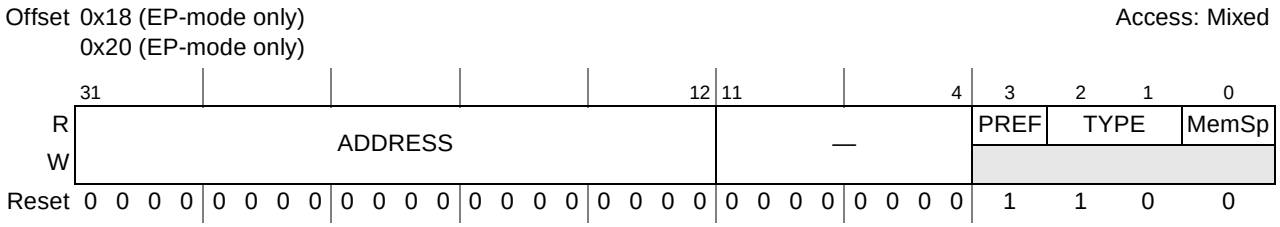


Figure 21-49. 64-Bit Low Memory Base Address Register

Table 21-48 describes the PCI Express 64-bit low memory BAR fields.

Table 21-48. 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	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	—	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

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. The 64-bit high memory BARs are shown in Figure 21-50.

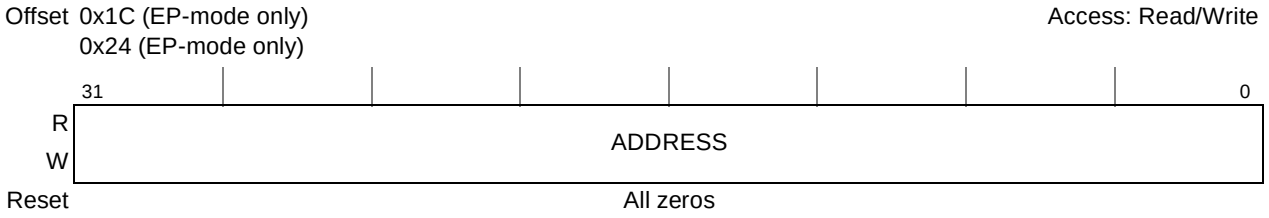


Figure 21-50. 64-Bit High Memory Base Address Register

Table 21-49 describes the PCI Express 64-bit low memory BAR fields.

Table 21-49. Bit Setting for 64-Bit High Memory Base Address Register

Bits	Name	Description
31–0	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.

21.3.8.2.2 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.

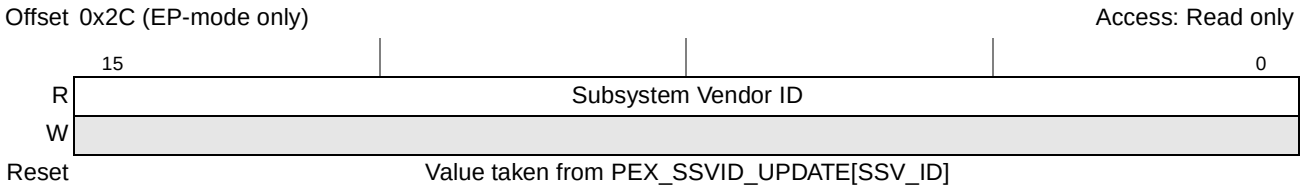


Figure 21-51. PCI Express Subsystem Vendor ID Register

Table 21-50. PCI Express Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	The value for subsystem vendor ID is determined by the PCI Express subsystem vendor ID update register. See Section 21.3.10.17, “PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,” for more information.

21.3.8.2.3 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.

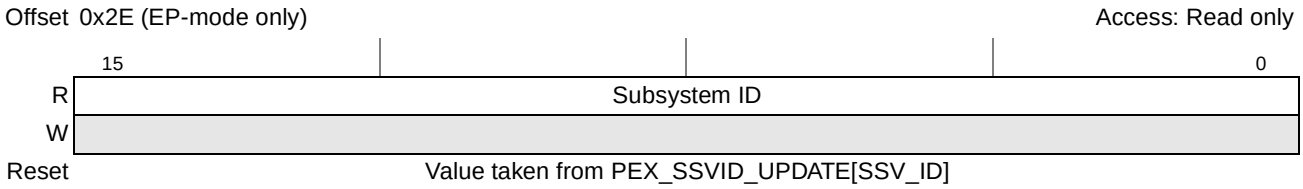


Figure 21-52. PCI Express Subsystem ID Register

Table 21-51. PCI Express Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	The value for subsystem ID is determined by the PCI Express subsystem vendor ID update register. See Section 21.3.10.17, “PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478,” for more information.

21.3.8.2.4 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

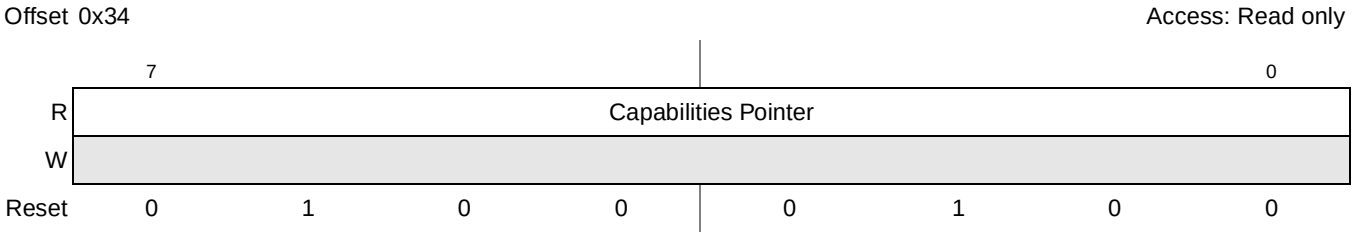


Figure 21-53. Capabilities Pointer Register

Table 21-52. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to Section 21.3.9, "PCI Compatible Device-Specific Configuration Space," for more information.

21.3.8.2.5 PCI Express Interrupt Line Register (EP-Mode Only)—0x3C

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.

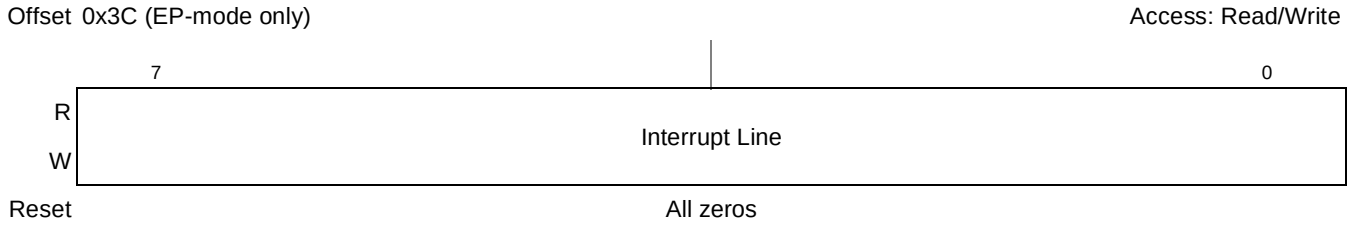


Figure 21-54. PCI Express Interrupt Line Register

Table 21-53. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

21.3.8.2.6 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

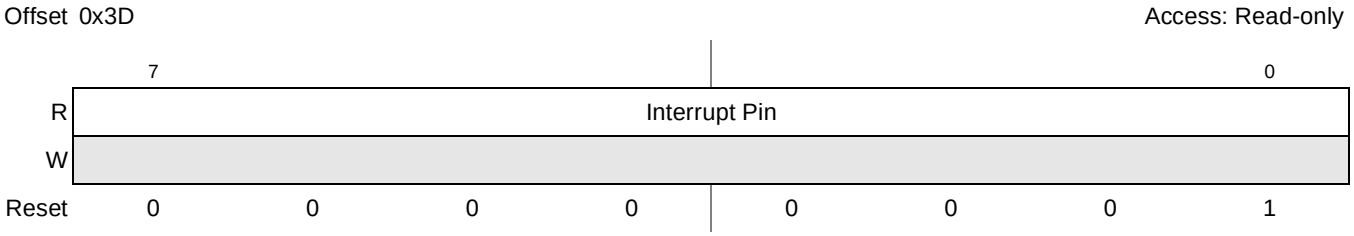


Figure 21-55. PCI Express Interrupt Pin Register

Table 21-54. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

21.3.8.2.7 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

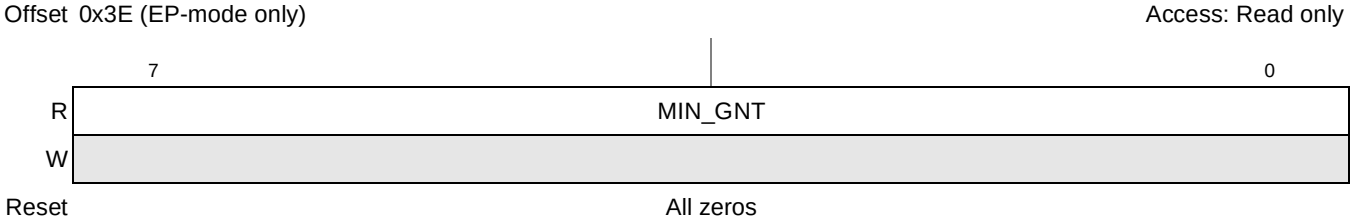


Figure 21-56. PCI Express Maximum Grant Register (MAX_GNT)

Table 21-55. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

21.3.8.2.8 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

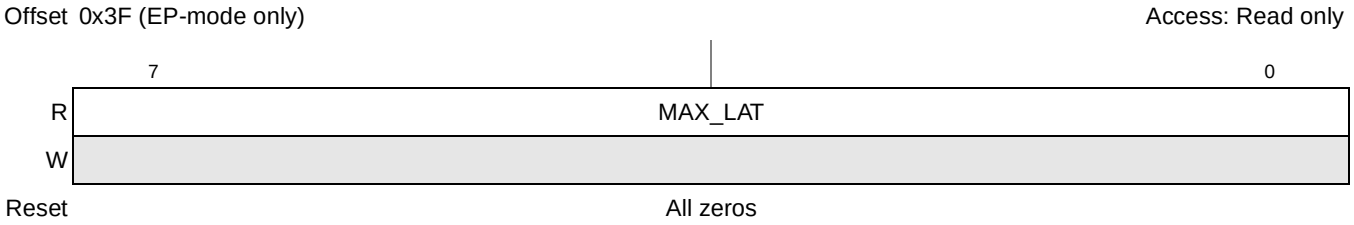


Figure 21-57. PCI Express Maximum Latency Register (MAX_LAT)

Table 21-56. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7–0	MAX_LAT	Does not apply for PCI Express.

21.3.8.3 Type 1 Configuration Header

The type 1 header is shown in Figure 21-58.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Register 0				10
				14
Secondary 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
			Capabilities Pointer	34
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 21-58. PCI Express PCI-Compatible Configuration Header—Type 1

Section 21.3.8.1, “Common PCI Compatible Configuration Header Registers,” describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 1 header beginning at offset 0x10.

21.3.8.3.1 PCI Express Base Address Register 0—0x10

Base address register 0 at offset 0x10 is a special fixed 1-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 21-47.

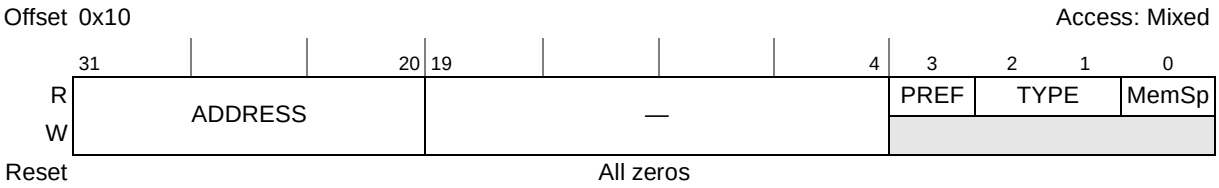


Figure 21-59. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 21-46 describes the PCI Express configuration and status register base address register.

Table 21-57. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

21.3.8.3.2 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in Figure 21-60.

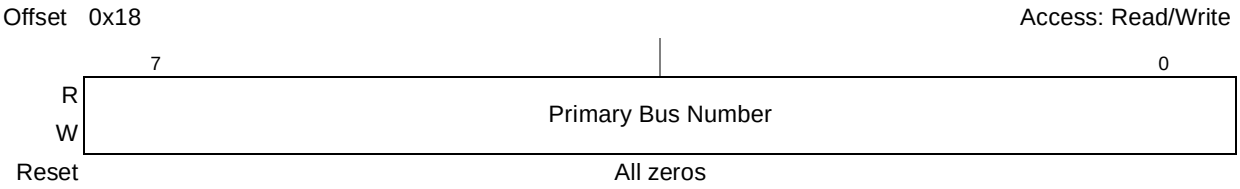


Figure 21-60. PCI Express Primary Bus Number Register

Table 21-58 describes the primary bus number register fields.

Table 21-58. PCI Express Primary Bus Number Register Field Description

Bits	Name	Description
7–0	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.

21.3.8.3.3 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in Figure 21-61.

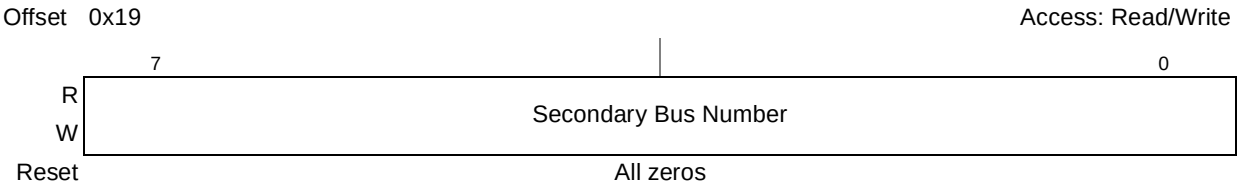


Figure 21-61. PCI Express Secondary Bus Number Register

Table 21-59 describes the secondary bus number register fields.

Table 21-59. PCI Express Secondary Bus Number Register Field Description

Bits	Name	Description
7–0	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.

21.3.8.3.4 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in Figure 21-62.

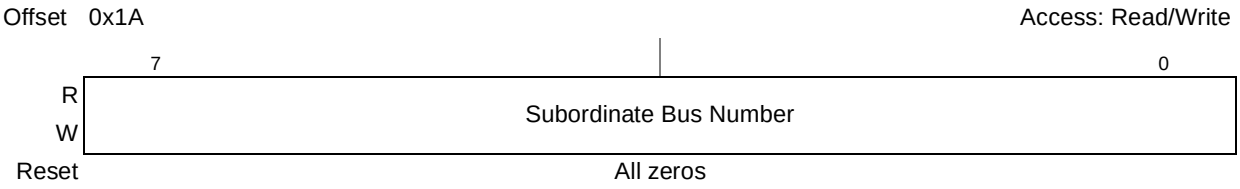


Figure 21-62. PCI Express Subordinate Bus Number Register

Table 21-60 describes the subordinate bus number register fields.

Table 21-60. PCI Express Subordinate Bus Number Register Field Description

Bits	Name	Description
7–0	Subordinate Bus Number	Highest bus number that is on the downstream interface.

21.3.8.3.5 PCI Express Secondary Latency Timer Register—0x1B

The secondary latency timer register does not apply to PCI Express. It must be read-only and return all zeros when read.

21.3.8.3.6 PCI Express I/O Base Register—0x1C

Note that this device does not support inbound I/O transactions. The I/O base register is shown in Figure 21-62.

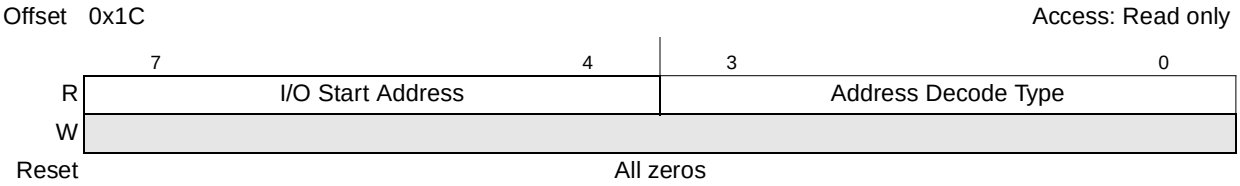


Figure 21-63. PCI Express I/O Base Register

Table 21-60 describes the I/O base register fields.

Table 21-61. PCI Express I/O Base Register Field Description

Bits	Name	Description
7–4	I/O Start Address	Specifies bits 15:12 of the I/O space start address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

21.3.8.3.7 PCI Express I/O Limit Register—0x1D

Note that this device does not support inbound I/O transactions. The I/O limit register is shown in Figure 21-62.

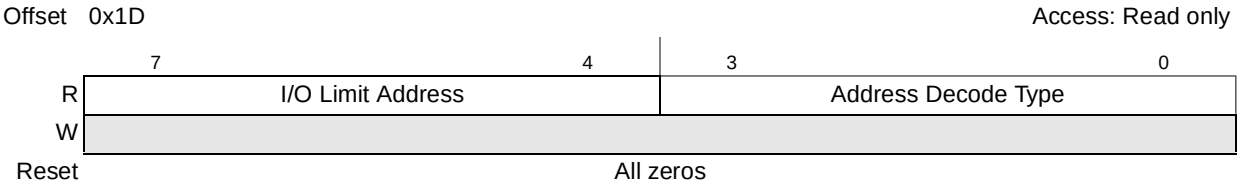


Figure 21-64. PCI Express I/O Limit Register

Table 21-60 describes the I/O limit register fields.

Table 21-62. PCI Express I/O Limit Register Field Description

Bits	Name	Description
7–4	I/O Limit Address	Specifies bits 15:12 of the I/O space ending address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

21.3.8.3.8 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in Figure 21-65. 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 Section 21.3.10.20, "Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0" for more information.

Offset 0x1E Access: w1c

	15	14	13	12	11	10	9	8	7	0
R	DPE	SSE	RMA	RTA	STA	—		MDPE	—	
W	w1c	w1c	w1c	w1c	w1c			w1c		
Reset	All zeros									

Figure 21-65. PCI Express Secondary Status Register

Table 21-63 describes the PCI Express secondary status register fields.

Table 21-63. PCI Express Secondary Status Register Field Description

Bits	Name	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.
11	STA	Signaled target abort. This bit is set when the secondary side issues a CA completion.
10–9	—	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.
7–0	—	Reserved

21.3.8.3.9 PCI Express Memory Base Register—0x20

The memory base register is shown in Figure 21-66.

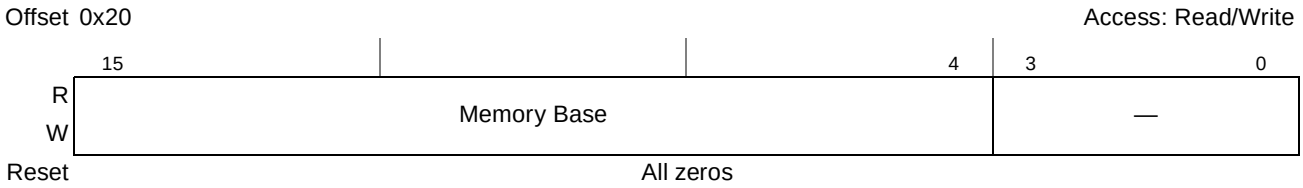


Figure 21-66. PCI Express Memory Base Register

Table 21-64 describes the memory base register fields.

Table 21-64. PCI Express Memory Base Register Field Description

Bits	Name	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.
3–0	—	Reserved

21.3.8.3.10 PCI Express Memory Limit Register—0x22

The memory limit register is shown in Figure 21-67.

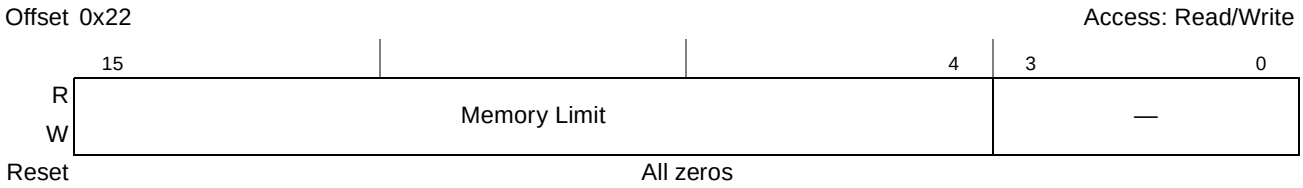


Figure 21-67. PCI Express Memory Limit Register

Table 21-65 describes the memory base register fields.

Table 21-65. PCI Express Memory Limit Register Field Description

Bits	Name	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. **Note:** Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range will result in unsupported request response.
3–0	—	Reserved

21.3.8.3.11 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in Figure 21-68.

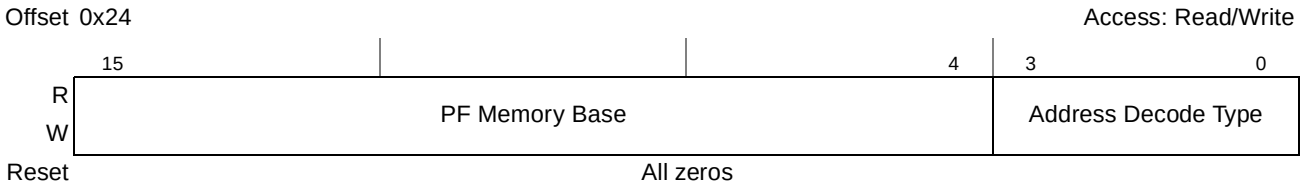


Figure 21-68. PCI Express Prefetchable Memory Base Register

Table 21-66 describes the prefetchable memory base register fields.

Table 21-66. PCI Express Prefetchable Memory Base Register Field Description

Bits	Name	Description
15–4	PF Memory Base	Specifies bits 31:20 of the prefetchable memory space start address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

21.3.8.3.12 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in Figure 21-69.

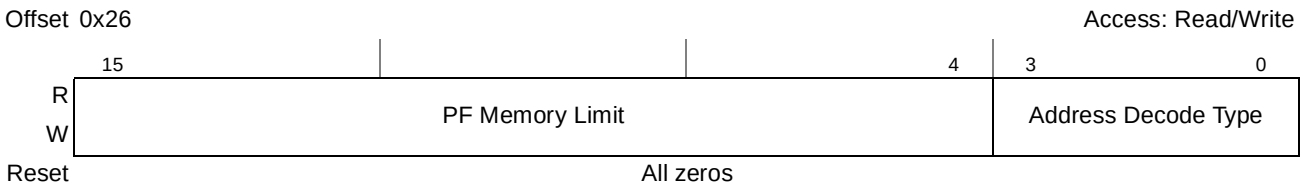


Figure 21-69. PCI Express Prefetchable Memory Limit Register

Table 21-67 describes the prefetchable memory limit register fields.

Table 21-67. PCI Express Prefetchable Memory Limit Register Field Description

Bits	Name	Description
15–4	PF Memory Limit	Specifies bits 31:20 of the prefetchable memory space ending address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

21.3.8.3.13 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 21-70.

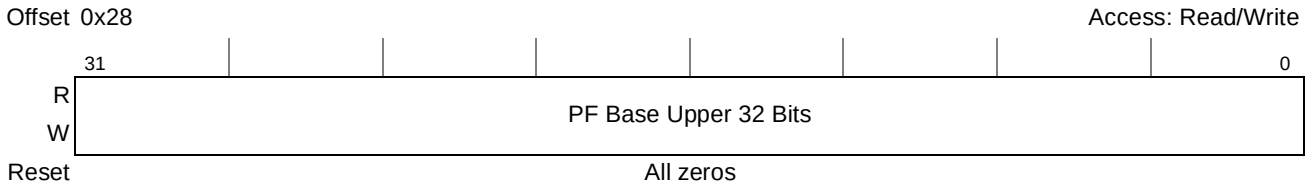


Figure 21-70. PCI Express Prefetchable Base Upper 32 Bits Register

Table 21-68 describes the PCI Express prefetchable memory base upper 32 bits register fields.

Table 21-68. PCI Express Prefetchable Base Upper 32 Bits Register

Bits	Name	Description
31–0	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.

21.3.8.3.14 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 21-71.

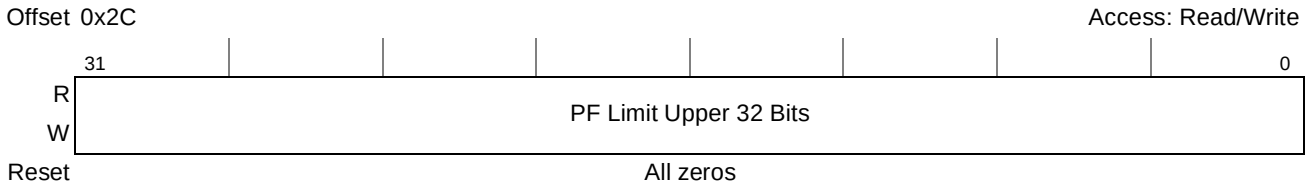


Figure 21-71. PCI Express Prefetchable Limit Upper 32 Bits Register

Table 21-69 describes the PCI Express prefetchable memory limit upper 32 bits register fields.

Table 21-69. PCI Express Prefetchable Limit Upper 32 Bits Register

Bits	Name	Description
31–0	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.

21.3.8.3.15 PCI Express I/O Base Upper 16 Bits Register—0x30

Note that this device does not support inbound I/O transactions. The I/O base upper 16 bits register is shown in Figure 21-72.

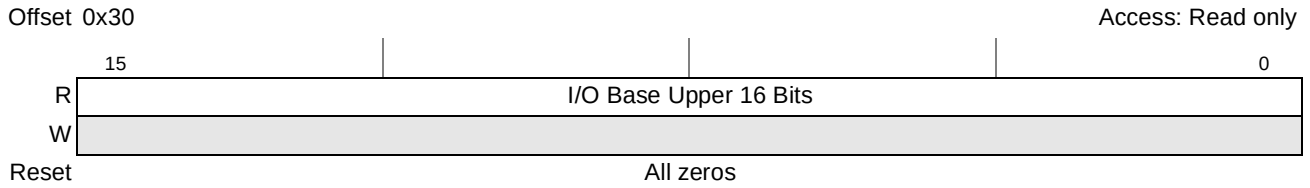


Figure 21-72. PCI Express I/O Base Upper 16 Bits Register

Table 21-70 describes the I/O base upper 16 bits register fields.

Table 21-70. PCI Express I/O Base Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O 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.

21.3.8.3.16 PCI Express I/O Limit Upper 16 Bits Register—0x32

Note that this device does not support inbound I/O transactions. The I/O limit upper 16 bits register is shown in Figure 21-73.

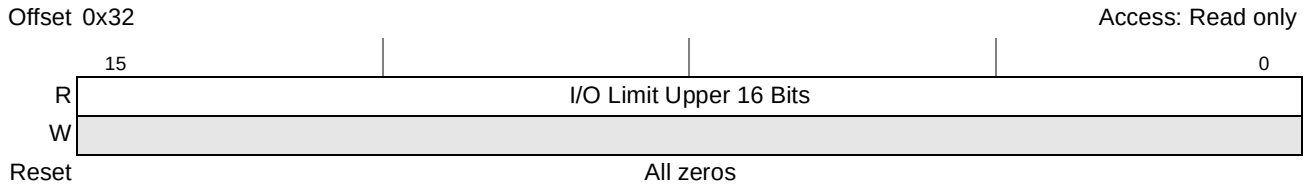


Figure 21-73. PCI Express I/O Limit Upper 16 Bits Register

Table 21-71 describes the I/O limit upper 16 bits register fields.

Table 21-71. PCI Express I/O Limit Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O 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.

21.3.8.3.17 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

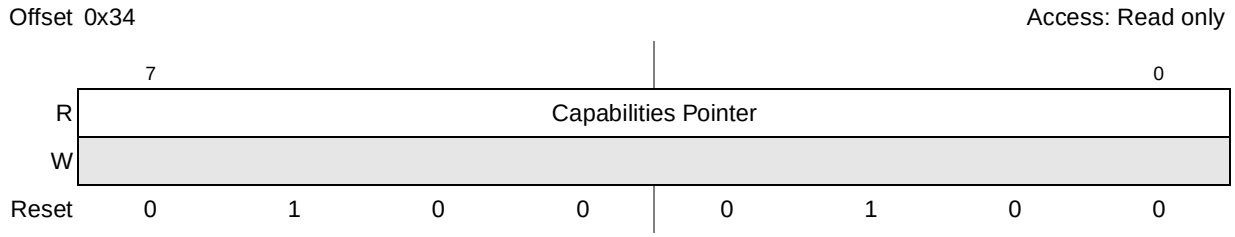


Figure 21-74. Capabilities Pointer Register

Table 21-72. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset (0x44) for additional PCI-compatible registers above the common 64-byte header. Refer to Section 21.3.9, “PCI Compatible Device-Specific Configuration Space,” for more information.

21.3.8.3.18 PCI Express Interrupt Line Register—0x3C

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.

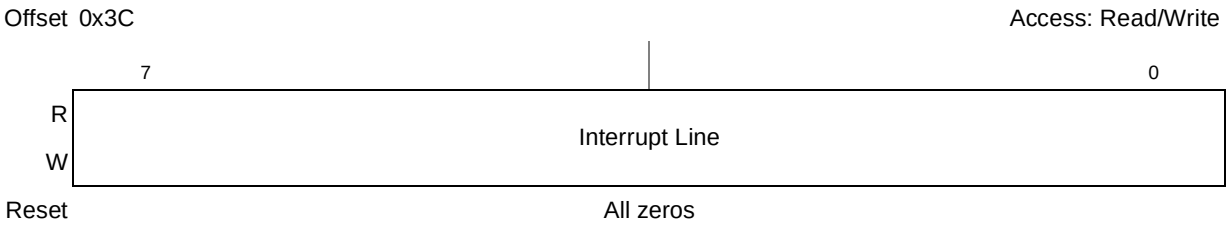


Figure 21-75. PCI Express Interrupt Line Register

Table 21-73. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

21.3.8.3.19 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

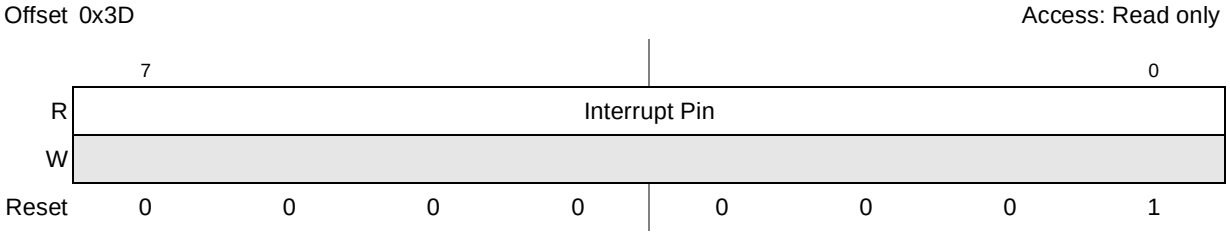


Figure 21-76. PCI Express Interrupt Pin Register

Table 21-74. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

21.3.8.3.20 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in Figure 21-77.

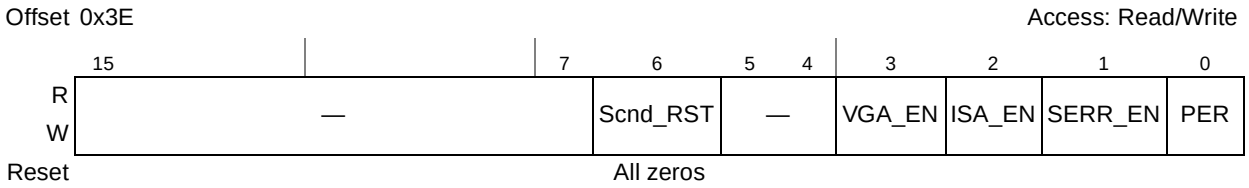


Figure 21-77. PCI Express Bridge Control Register

Table 21-75 describes the PCI Express bridge control register fields.

Table 21-75. PCI Express Bridge Control Register Field Description

Bits	Name	Description
15–7	—	Reserved
6	Scnd_RST	Secondary bus reset
5–4	—	Reserved
3	VGA_EN	VGA enable
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.

21.3.9 PCI Compatible Device-Specific Configuration Space

The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just above the 64-byte PCI-compatible configuration header).

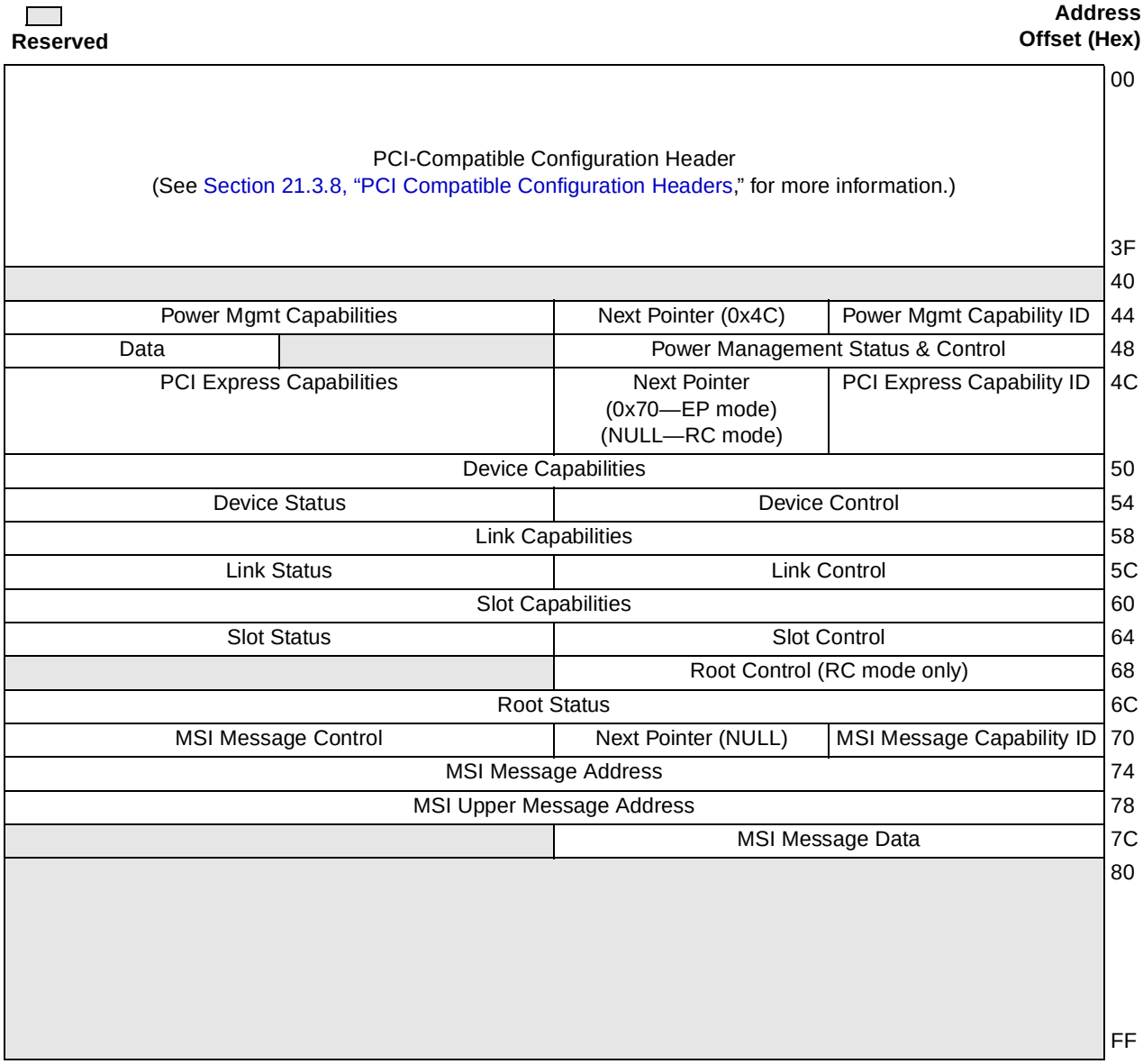


Figure 21-78. PCI Compatible Device-Specific Configuration Space

21.3.9.1 PCI Express Power Management Capability ID Register—0x44

The PCI Express power management capability ID register is shown in Figure 21-79.

Offset 0x44 Access: Read only

	7							0
R	Power Mgmt Capability ID							
W								
Reset	0	0	0	0	0	0	0	1

Figure 21-79. PCI Express Power Management Capability ID Register

Table 21-76. PCI Express Power Management Capability ID Register Field Description

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

21.3.9.2 PCI Express Power Management Capabilities Register—0x46

The PCI Express power management capabilities register is shown in Figure 21-80.

Offset 0x46 Access: Read only

	15				11	10	9	8		6	5	4	3	2		0
R	PME Support					D2	D1	AUX Curr			DSI	—	PME CLK	Version		
W																
Reset	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0

Figure 21-80. PCI Express Power Management Capabilities Register

Table 21-77. PCI Express Power Management Capabilities Register Field Description

Bits	Name	Description
15–11	PME Support	Indicates the power states that this device supports
10	D2	D2 Support
9	D1	D1 Support
8–6	AUX Curr	AUX Current
5	DSI	Device Specific Initialization
4	—	Reserved
3	PME CLK	Does not apply to PCI Express.
2–0	Version	Set to 0x2 for this version of the specification.

21.3.9.3 PCI Express Power Management Status and Control Register—0x48

The PCI Express power management status and control register is shown in Figure 21-81.

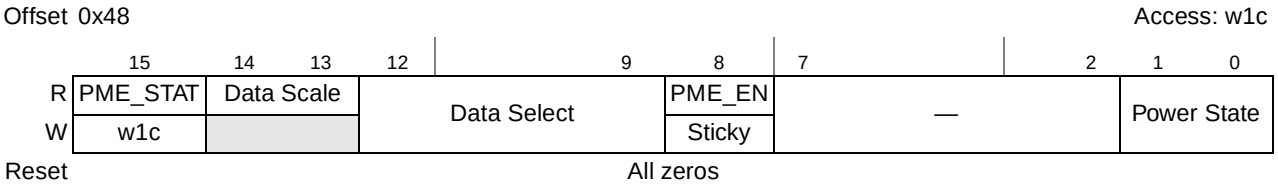


Figure 21-81. PCI Express Power Management Status and Control Register

Table 21-78. PCI Express Status and Control Register Field Description

Bits	Name	Description
15	PME_STAT	PME Status
14–13	Data Scale	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
12–9	Data Select	Obtained directly from *PCI Express™ Base Specification, Revision 1.0a*
8	PME_EN	PME Enable
7–2	—	Reserved
1–0	Power State	Power state. Indicates the current power state of the function. 0x00 D0 0x01 D1 0x02 D2 0x03 D3

21.3.9.4 PCI Express Power Management Data Register—0x4B

The PCI Express power management data register is shown in Figure 21-82.

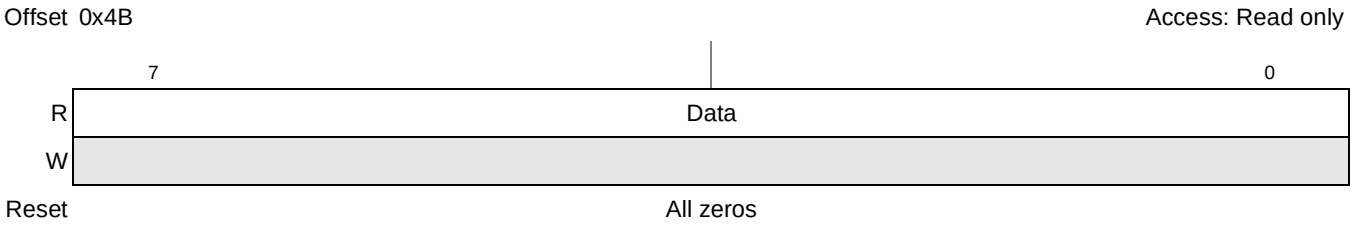


Figure 21-82. PCI Express Power Management Data Register

Table 21-79. PCI Express Power Management Data Register Field Description

Bits	Name	Description
7–0	Data	Obtained from *PCI Express™ Base Specification, Revision 1.0a*

21.3.9.5 PCI Express Capability ID Register—0x4C

The PCI Express capability ID register is shown in Figure 21-83.

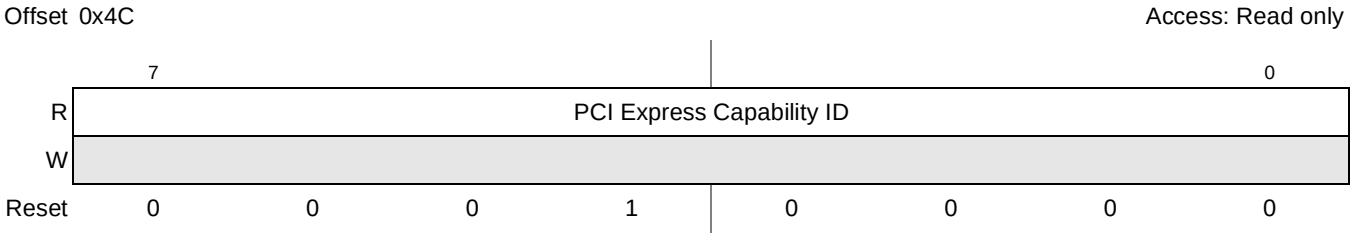


Figure 21-83. PCI Express Capability ID Register

Table 21-80. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

21.3.9.6 PCI Express Capabilities Register—0x4E

The PCI Express capabilities register is shown in Figure 21-84.

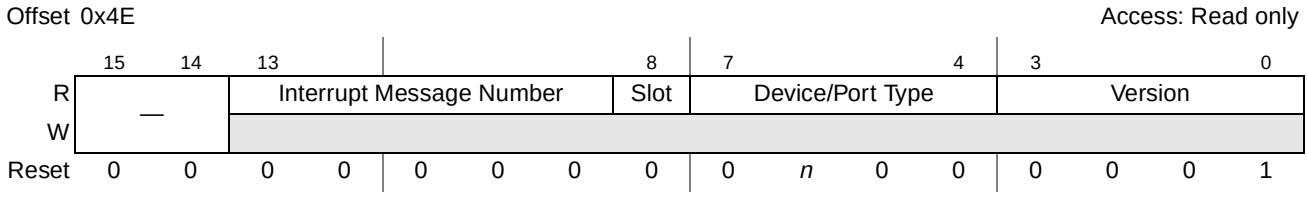


Figure 21-84. PCI Express Capabilities Register

Table 21-81. PCI Express Capabilities Register Field Description

Bits	Name	Description
15–14	—	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	0100 (RC mode) 0000 (EP mode)
3–0	Capability Version	Indicates the defined PCI Express capability structure version number. Must be 1h for 1.0, 1.0a, and 1.1 specification.

21.3.9.7 PCI Express Device Capabilities Register—0x50

The PCI Express device capabilities register is shown in Figure 21-85.

Offset 0x50 Access: Read-only

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—				CSPLS		CSPLV								—	
W																
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	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

Figure 21-85. PCI Express Device Capabilities Register

Table 21-82. PCI Express Device Capabilities Register Field Description

Bits	Name	Description
31–28	—	Reserved
27–26	CSPLS	Captured Slot Power Limit Scale
25–18	CSPLV	Captured Slot Power Limit Value
17–15	—	Reserved
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
2–0	MAX_PL_SIZE_SUP	Maximum payload size supported. 001 = 256-bytes

21.3.9.8 PCI Express Device Control Register—0x54

The PCI Express device control register is shown in Figure 21-86.

Offset 0x54 Access: Read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	MAX_READ_SIZE			NSE	APE	PFE	ETE	MAX_PAYLOAD_SIZE			RO	URR	FER	NFER	CER
W																
Reset	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0

Figure 21-86. PCI Express Device Control Register

Table 21-83. PCI Express Device Control Register Field Description

Bits	Name	Description
15	—	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
8	ETE	Extended tag field enable
7–5	MAX_PAYLOAD_SIZE	Maximum payload size
4	RO	Relaxed ordering
3	URR	Unsupported request reporting
2	FER	Fatal error reporting
1	NFER	Non-fatal error reporting
0	CER	Correctable error reporting

21.3.9.9 PCI Express Device Status Register—0x56

The PCI Express device status register is shown in Figure 21-87.

Offset 0x56 Access: w1c

	15–6	5	4	3	2	1	0
R	—	TP	APD	URD	FED	NFED	CED
W				w1c	w1c	w1c	w1c
Reset	All zeros						

Figure 21-87. PCI Express Device Status Register

Table 21-84. PCI Express Device Status Register Field Description

Bits	Name	Description
15–6	—	Reserved
5	TP	Transactions pending
4	APD	AUX power detected
3	URD	Unsupported request detected
2	FED	Fatal error detected
1	NFED	Non-fatal error detected
0	CED	Correctable error detected

21.3.9.10 PCI Express Link Capabilities Register—0x58

The PCI Express link capabilities register is shown in Figure 21-88.

Offset 0x58 Access: Read only

	31							24	23				18	17	16
R	Port Number								—					L1_EX_LAT	
W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

	15	14		12	11	10	9				4	3				0
R	L1_EX_LAT	L0s_EX_LAT			ASPM		MAX_LINK_W					MAX_LINK_SP				
W																
Reset	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1

Figure 21-88. PCI Express Link Capabilities Register

Table 21-85. PCI Express Link Capabilities Register Field Description

Bits	Name	Description
31–24	Port Number	
23–18	—	Reserved
17–15	L1_EX_LAT	L1 exit latency
14–12	L0s_EX_LAT	L0s exit latency
11–10	ASPM	Active state power management (ASPM) Support
9–4	MAX_LINK_W	Maximum link width
3–0	MAX_LINK_SP	Maximum link speed 0001 2.5 GT/s link

21.3.9.11 PCI Express Link Control Register—0x5C

The PCI Express link control register is shown in Figure 21-89.

Offset 0x5C Access: Mixed

	15							8	7	6	5	4	3	2	1	0
R	—								EXT_SYNC	CCC		LD	RCB	—	ASPM_CTL	
W											RL					
Reset	All zeros															

Figure 21-89. PCI Express Link Control Register

Table 21-86. PCI Express Link Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	EXT_SYNC	Extended synch

Table 21-86. PCI Express Link Control Register Field Description (continued)

Bits	Name	Description
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; reads of this bit always return 0.
4	LD	Link disable (Reserved for EP devices)
3	RCB	Read completion boundary
2	—	Reserved
1–0	ASPM_CTL	Active state power management (ASPM) control

21.3.9.12 PCI Express Link Status Register—0x5E

The PCI Express link status register is shown in Figure 21-90.

Offset 0x5E Access: Read only

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—			SCC	LT	—	NEG_LINK_W						LINK_SP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Figure 21-90. PCI Express Link Status Register

Table 21-87. PCI Express Link Status Register Field Description

Bits	Name	Description
15–13	—	Reserved
12	SCC	Slot clock configuration
11	LT	Link training
10	—	Reserved.
9–4	NEG_LINK_W	Negotiated link width
3–0	LINK_SP	Link speed.

21.3.9.13 PCI Express Slot Capabilities Register—0x60

The PCI Express slot capabilities register is shown in Figure 21-91.

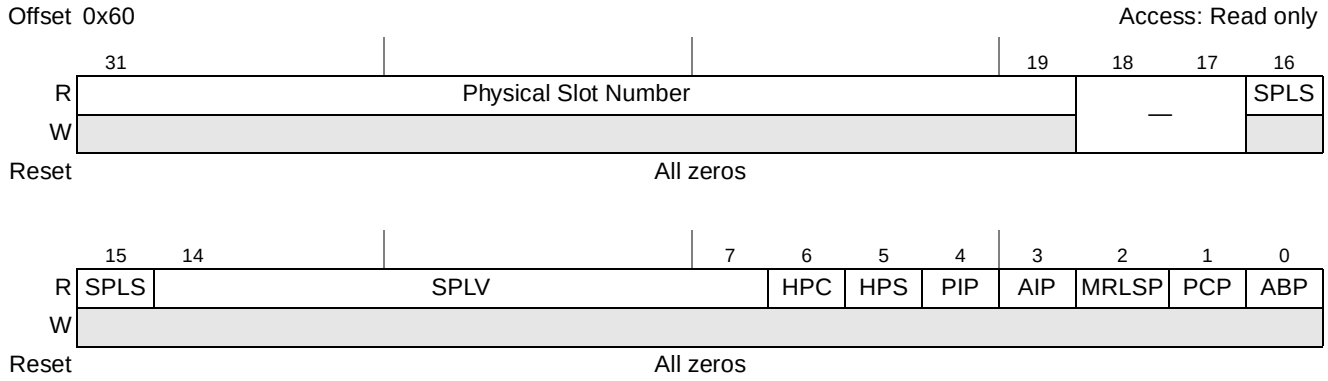


Figure 21-91. PCI Express Slot Capabilities Register

Table 21-88. PCI Express Slot Capabilities Register Field Description

Bits	Name	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	—	Reserved
16–15	SPLS	Slot power limit scale.
14–7	SPLV	Slot power limit value.
6	HPD	Hot plug capable.
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.

21.3.9.14 PCI Express Slot Control Register—0x64

The PCI Express slot control register is shown in Figure 21-92.

Offset 0x64 Access: Read/Write

	15–11	10	9–8	7–6	5	4	3	2	1	0
R W	—	PCC	PIC	AIC	HPIE	CCIE	PDCE	MRLSCE	PFDE	ABPE
Reset	All zeros									

Figure 21-92. PCI Express Slot Control Register

Table 21-89. PCI Express Slot Control Register Field Description

Bits	Name	Description
15–11	—	Reserved
10	PCC	Power controller control.
9–8	PIC	Power indicator control.
7–6	AIC	Attention indicator control.
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.

21.3.9.15 PCI Express Slot Status Register—0x66

The PCI Express slot status register is shown in Figure 21-93.

Offset 0x66 Access: w1c

	15–7	6	5	4	3	2	1	0
R	—	PDS	MRLSS	CC	PDC	MRLSC	PFD	ABP
W				w1c	w1c	w1c	w1c	w1c
Reset	0 0 0 0 0 0 0 0 0	1	0	0	0	0	0	0

Figure 21-93. PCI Express Slot Status Register

Table 21-90. PCI Express Slot Status Register Field Descriptions

Bits	Name	Description
15–7	—	Reserved
6	PDS	Presence detect state. This bit indicates the presence of a card in the slot. 0 Slot empty 1 Card is present
5	MRLSS	MRL sensor state. 0 MRL closed 1 MRL open

Table 21-90. PCI Express Slot Status Register Field Descriptions (continued)

Bits	Name	Description
4	CC	Command completed.
3	PDC	Presence detect changed.
2	MRLSC	MRL sensor changed.
1	PFD	Power fault detected.
0	ABP	Attention button pressed.

21.3.9.16 PCI Express Root Control Register (RC Mode Only)—0x68

The PCI Express root control register is shown in Figure 21-94.

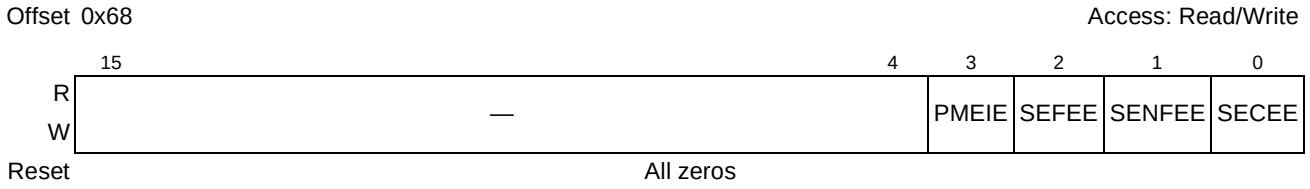


Figure 21-94. PCI Express Root Control Register

Table 21-91. PCI Express Root Control Register Field Description

Bits	Name	Description
15–4	—	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.

21.3.9.17 PCI Express Root Status Register (RC Mode Only)—0x6C

The PCI Express root status register is shown in Figure 21-95.

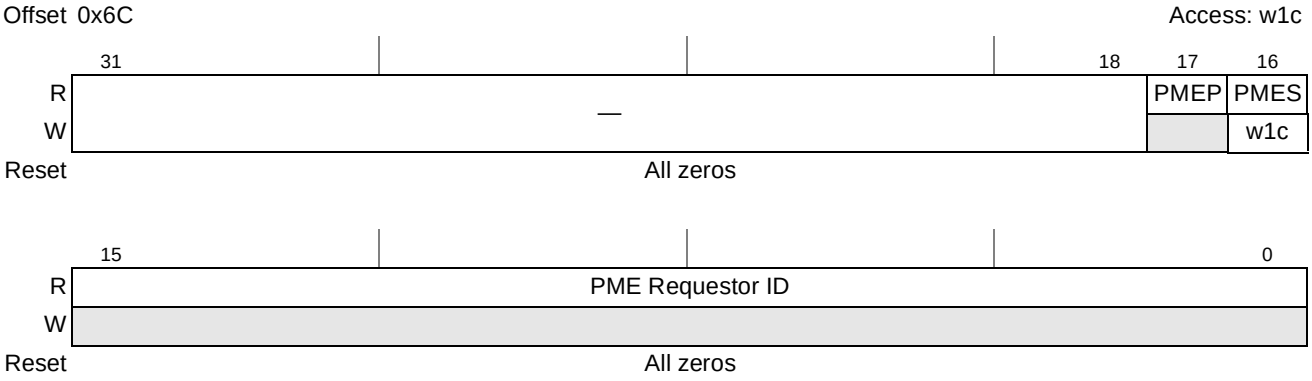


Figure 21-95. PCI Express Root Status Register

Table 21-92. PCI Express Root Status Register Field Description

Bits	Name	Description
31–18	—	Reserved
17	PMEP	PME pending.
16	PMES	PME status.
15–0	PME Requestor ID	PME requestor ID.

21.3.9.18 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x70

The PCI Express MSI message capability ID register is shown in Figure 21-96.

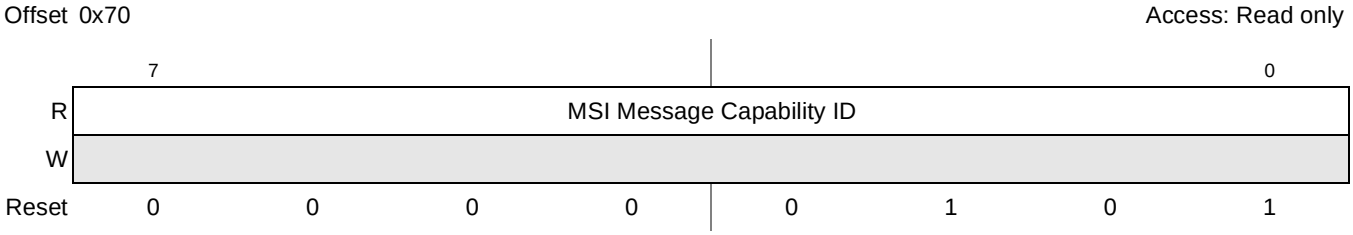


Figure 21-96. PCI Express Capability ID Register

Table 21-93. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

21.3.9.19 PCI Express MSI Message Control Register (EP Mode Only)—0x72

The PCI Express MSI message control register is shown in Figure 21-97.

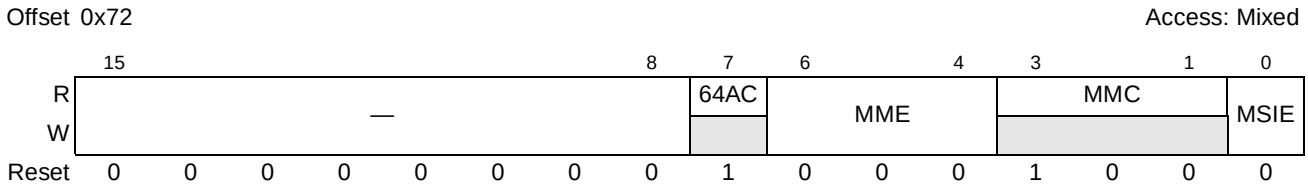


Figure 21-97. PCI Express MSI Message Control Register

Table 21-94. PCI Express MSI Message Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	64AC	64-bit address capable.
6–4	MME	Multiple message enable.

Table 21-94. PCI Express MSI Message Control Register Field Description (continued)

Bits	Name	Description
3–1	MMC	Multiple message capable.
0	MSIE	MSI enable.

21.3.9.20 PCI Express MSI Message Address Register (EP Mode Only)—0x74

The PCI Express MSI message address register is shown in Figure 21-98.

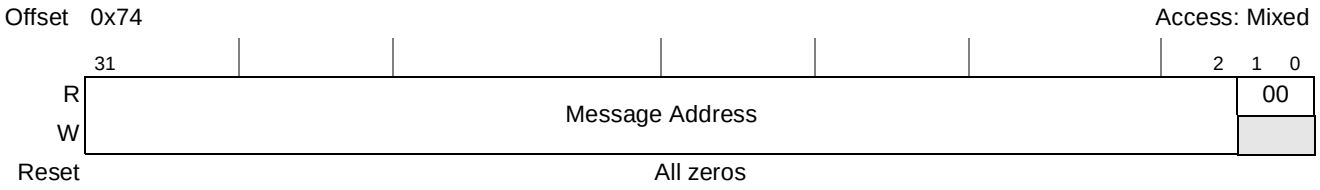


Figure 21-98. PCI Express MSI Message Address Register

Table 21-95. PCI Express MSI Message Address Register Field Description

Bits	Name	Description
31–2	Message Address	System-specified message address
1–0	00	Always returns 00 on reads; write operations have no effect.

21.3.9.21 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x78

The PCI Express MSI message upper address register is shown in Figure 21-99.

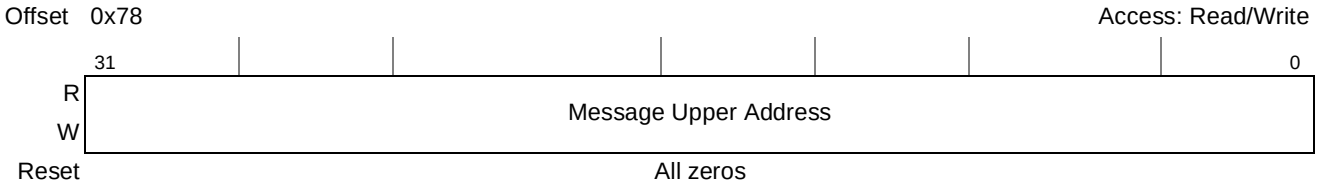


Figure 21-99. PCI Express MSI Message Upper Address Register

Table 21-96. PCI Express MSI Message Upper Address Register Field Description

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

21.3.9.22 PCI Express MSI Message Data Register (EP Mode Only)—0x7C

The PCI Express MSI message data register is shown in Figure 21-100.

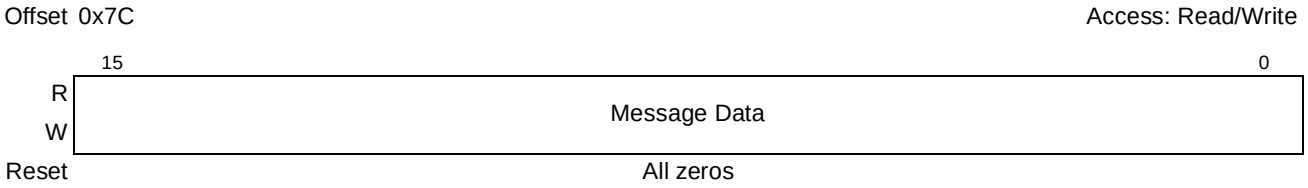


Figure 21-100. PCI Express MSI Message Data Register

Table 21-97. PCI Express MSI Message Data Register Field Description

Bits	Name	Description
15–0	Message Data	System-specified message.

21.3.10 PCI Express Extended Configuration Space

Reserved

		Address Offset (Hex)
PCI Compatible Configuration Header (See Section 21.3.8, “PCI Compatible Configuration Headers,” for more information.)		000 03F
PCI-Compatible Device-Specific Configuration Space (See Section 21.3.9, “PCI Compatible Device-Specific Configuration Space,” for more information.)		040 0FF
Next Capability Offset (NULL)/Capability Version	Advanced Error Reporting Capability ID	100
Uncorrectable Error Status		104
Uncorrectable Error Mask		108
Uncorrectable Error Severity		10C
Correctable Error Status		110
Correctable Error Mask		114
Advanced Error Capabilities and Control		118
Header Log		11C 120 124 128
Root Error Command		12C
Root Error Status		130
Error Source ID	Correctable Error Source ID	134
		138 3FF
PCI Express Controller Internal CSRs[1]		400 6FF
		700 FFF

Figure 21-101. PCI Express Extended Configuration Space

1 Note that the PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers will return all 0s.

21.3.10.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in Figure 21-102.

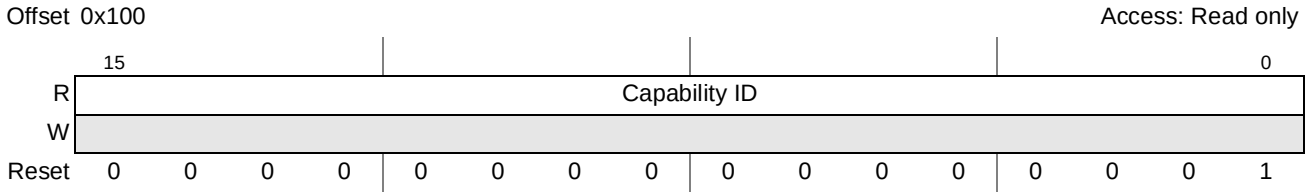


Figure 21-102. PCI Express Advanced Error Reporting Capability ID Register

Table 21-98. PCI Express Advanced Error Reporting Capability ID Register Field Description

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

21.3.10.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in Figure 21-103.

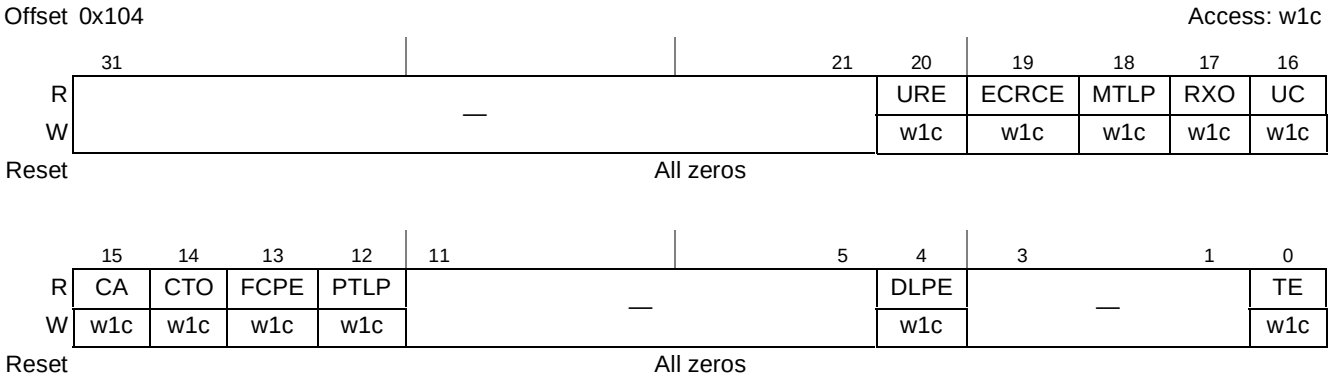


Figure 21-103. PCI Express Uncorrectable Error Status Register

Table 21-99. PCI Express Uncorrectable Error Status Register Field Description

Bits	Name	Description
31–21	—	Reserved
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.

Table 21-99. PCI Express Uncorrectable Error Status Register Field Description (continued)

Bits	Name	Description
13	FCPE	Flow control protocol error status.
12	PTLP	Poisoned TLP status.
11–5	—	Reserved
4	DLPE	Data link protocol error status.
3–1	—	Reserved
0	TE	Training error status.

21.3.10.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in Figure 21-104.

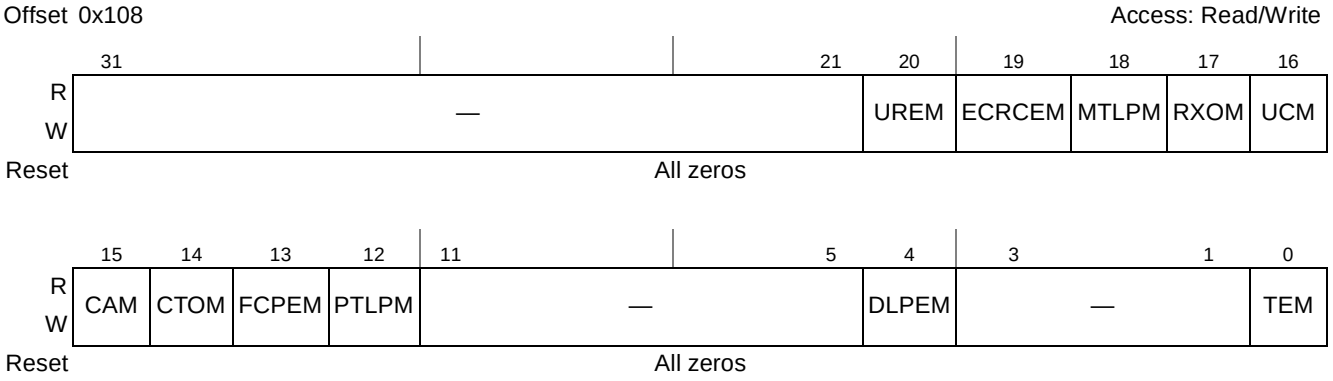


Figure 21-104. PCI Express Uncorrectable Error Mask Register

Table 21-100. PCI Express Uncorrectable Error Mask Register Field Description

Bits	Name	Description
31–21	—	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	—	Reserved
4	DLPEM	Data link protocol error mask.

Table 21-100. PCI Express Uncorrectable Error Mask Register Field Description (continued)

Bits	Name	Description
3–1	—	Reserved
0	TEM	Training error mask.

21.3.10.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in Figure 21-105.

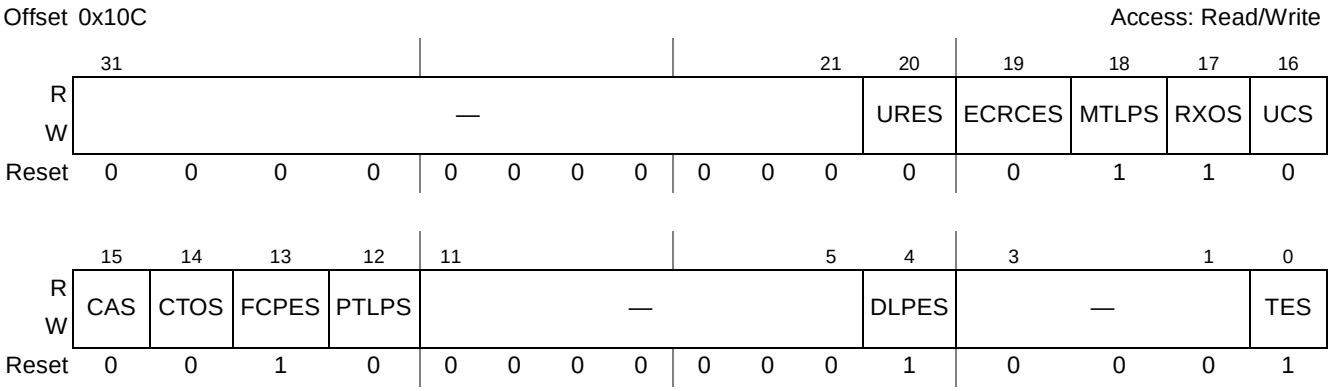


Figure 21-105. PCI Express Uncorrectable Error Severity Register

Table 21-101. PCI Express Uncorrectable Error Severity Register Field Description

Bits	Name	Description
31–21	—	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.
13	FCPES	Flow control protocol error severity.
12	PTLPS	Poisoned TLP severity.
11–5	—	Reserved
4	DLPES	Data link protocol error severity.
3–1	—	Reserved
0	TES	Training error severity.

21.3.10.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in Figure 21-106.

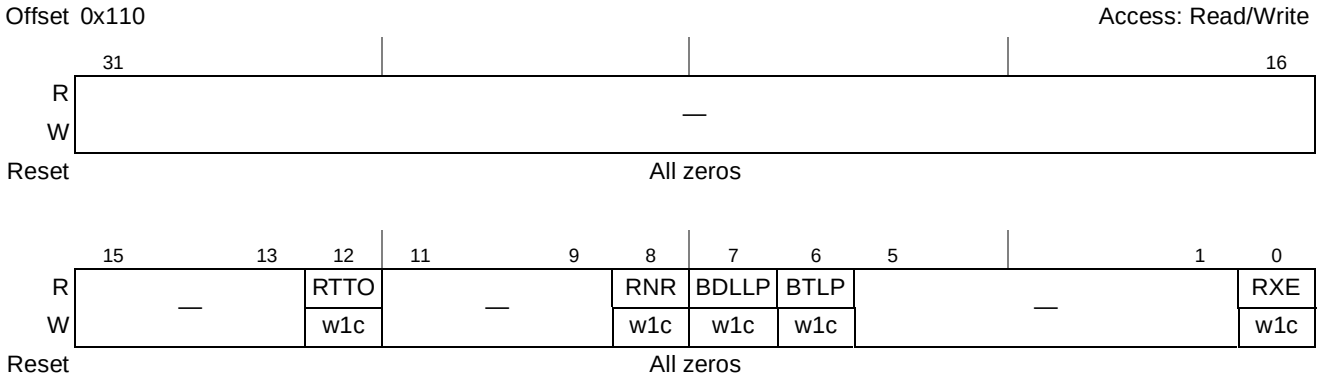


Figure 21-106. PCI Express Correctable Error Status Register

Table 21-102. PCI Express Correctable Error Status Register Field Description

Bits	Name	Description
31–13	—	Reserved
12	RTTO	Replay timer timeout status
11–9	—	Reserved
8	RNR	REPLAY_NUM Rollover status
7	BDLLP	Bad DLLP status
6	BTLP	Bad TLP status
5–1	—	Reserved
0	RXE	Receiver error status

21.3.10.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in Figure 21-107.

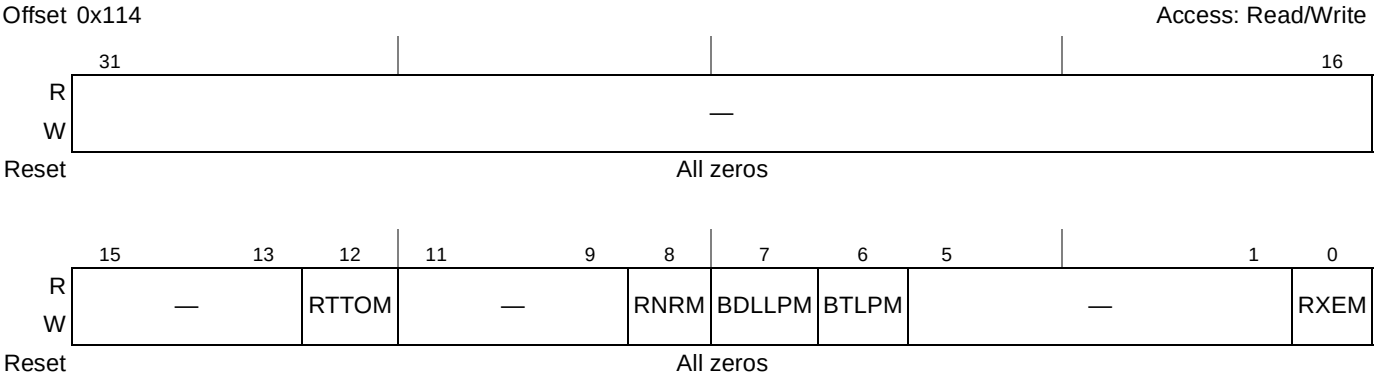


Figure 21-107. PCI Express Correctable Error Mask Register

Table 21-103. PCI Express Correctable Error Mask Register Field Description

Bits	Name	Description
31–13	—	Reserved
12	RTTOM	Replay timer timeout mask
11–9	—	Reserved
8	RNRM	REPLAY_NUM Rollover mask
7	BDLLPM	Bad DLLP mask
6	BTLPM	Bad TLP mask
5–1	—	Reserved
0	RXEM	Receiver error mask

21.3.10.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in Figure 21-108.

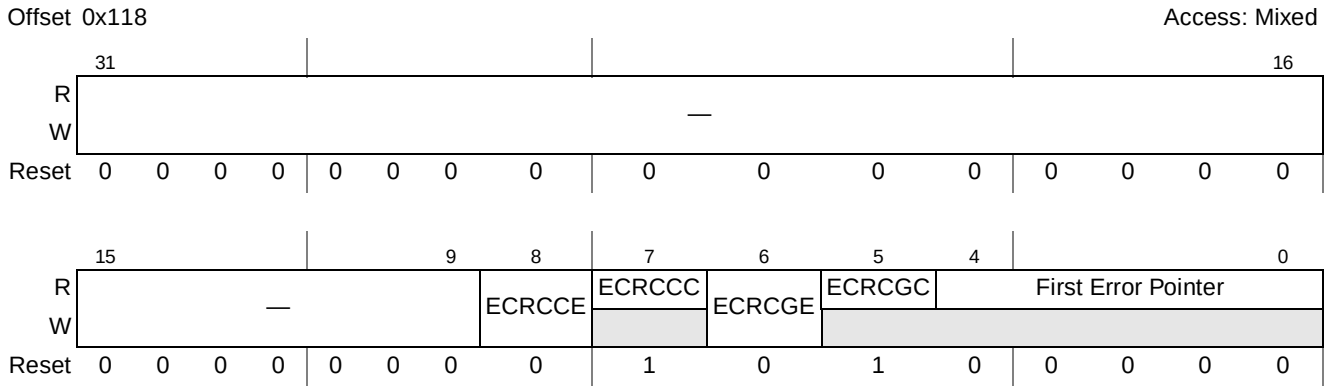


Figure 21-108. PCI Express Advanced Error Capabilities and Control Register

Table 21-104. PCI Express Advanced Error Capabilities and Control Register Field Description

Bits	Name	Description
31–9	—	Reserved.
8	ECRCCE	ECRC checking enable.
7	ECRCCC	ECRC checking capable.
6	ECRCGE	ECRC generation enable.
5	ECRCGC	ECRC generation capable.
4–0	First Error Pointer	The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

21.3.10.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in Figure 21-109.

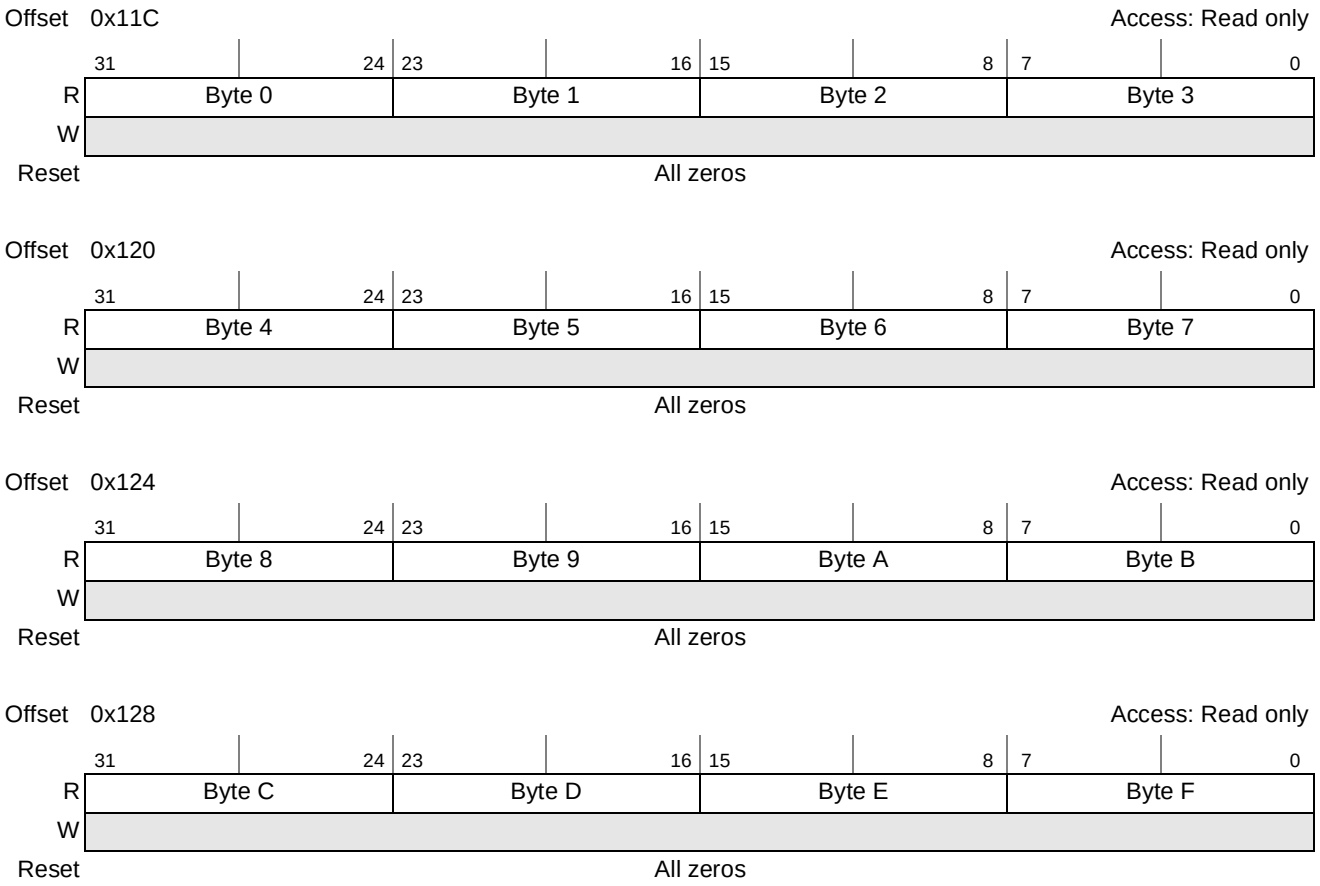


Figure 21-109. PCI Express Header Log Register

Table 21-105. PCI Express Header Log Register Field Description

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

21.3.10.9 PCI Express Root Error Command Register—0x12C

The PCI Express root error command register is shown in Figure 21-110.

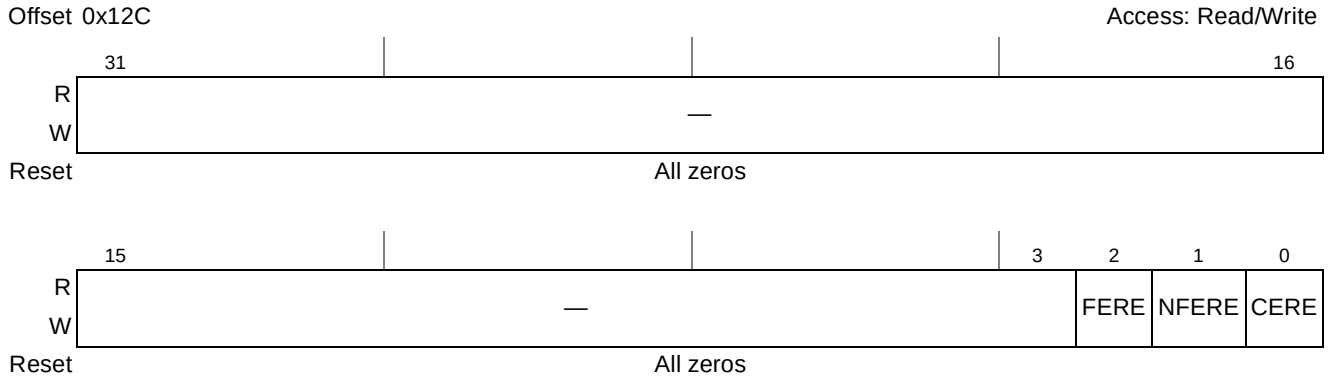


Figure 21-110. PCI Express Root Error Command Register

Table 21-106. PCI Express Root Error Command Register Field Description

Bits	Name	Description
31–3	—	Reserved
2	FERE	Fatal error reporting enable.
1	NFERE	Non-fatal error reporting enable
0	CERE	Correctable error reporting enable

21.3.10.10 PCI Express Root Error Status Register—0x130

The PCI Express root error status register is shown in Figure 21-111.

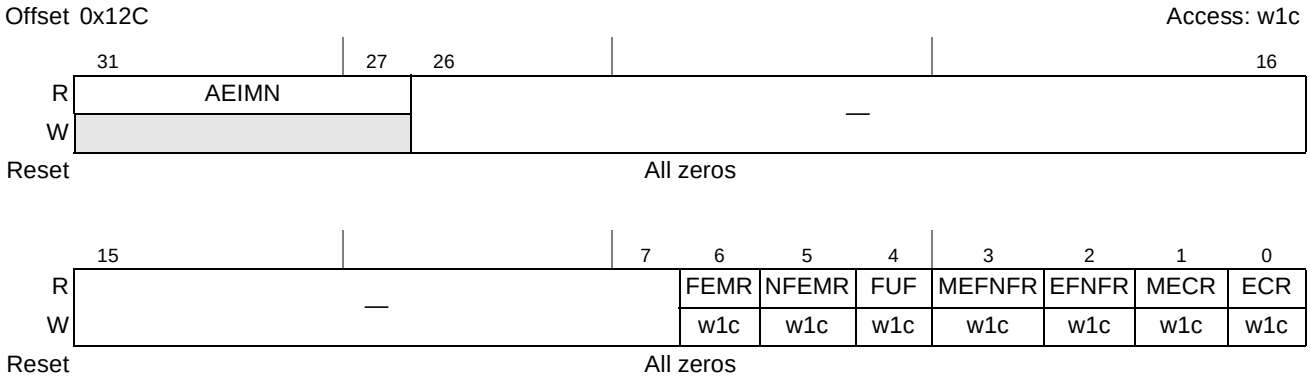


Figure 21-111. PCI Express Root Error Status Register

Table 21-107. PCI Express Root Error Command Status Field Description

Bits	Name	Description
31–27	AEIMN	Advanced error interrupt message number.
26–7	—	Reserved
6	FEMR	Fatal error messages received.

Table 21-107. PCI Express Root Error Command Status Field Description (continued)

Bits	Name	Description
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.
1	MECR	Multiple ERR_COR received.
0	ECR	ERR_COR received.

21.3.10.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in Figure 21-112.

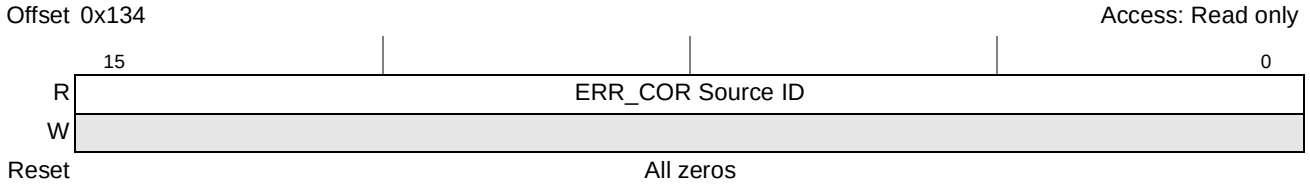


Figure 21-112. PCI Express Correctable Error Source ID Register

Table 21-108. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	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.

21.3.10.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in Figure 21-113.

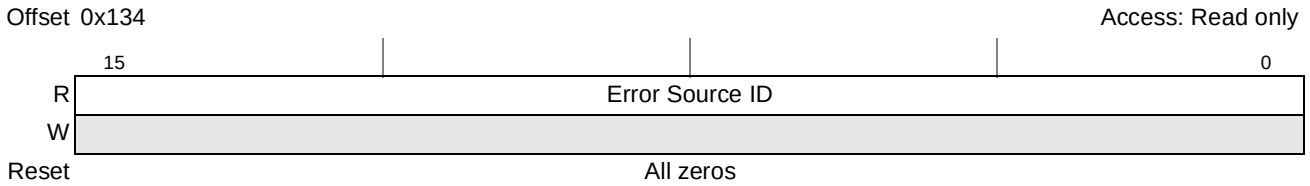


Figure 21-113. PCI Express Correctable Error Source ID Register

Table 21-109. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

21.3.10.13 LTSSM State Status Register—0x404

The PCI Express link training and status state machine (LTSSM) state status register, shown in Figure 21-114, provides detailed information about link training status. This register is useful for debugging link training failures.

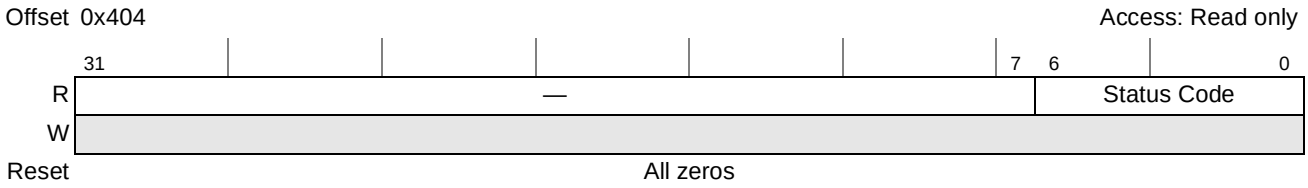


Figure 21-114. PCI Express LTSSM State Status Register (PEX_LTSSM_STAT)

The fields of the PCI Express LTSSM state status register are described in Table 21-110.

Table 21-110. PEX_LTSSM_STAT Field Descriptions

Bits	Name	Description
31–7	—	Reserved
6–0	Status code	Status code. See Table 21-111 for encodings.

Table 21-111 provides the encodings for the status code field of the PEX_LTSSM_STAT register.

Table 21-111. 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)

Table 21-111. PEX_LTSSM_STAT Status Codes (continued)

Status Code (Hex)	LTSSM State Description	Status Code (Hex)	LTSSM State Description
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
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

21.3.10.14 PCI Express Controller Core Clock Ratio Register—0x440

The PCI Express controller core clock ratio register, shown in Figure 21-115, 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.

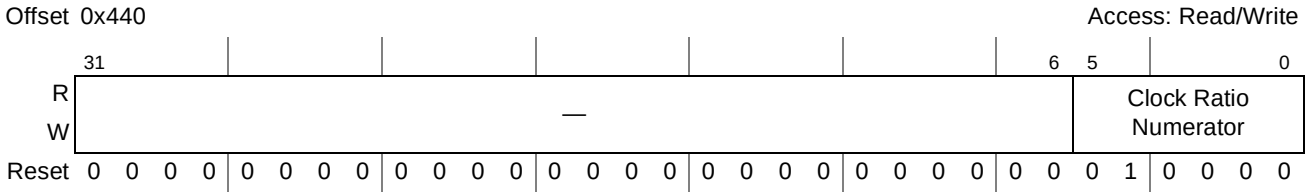


Figure 21-115. PCI Express IP Block Core Clock Ratio Register (PEX_GCLK_RATIO)

The fields of the PCI Express IP block core clock ratio register are described in Table 21-112.

Table 21-112. PEX_GCLK_RATIO Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5–0	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)

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.

21.3.10.15 PCI Express Power Management Timer Register—0x450

The PCI Express power management timer register, shown in Figure 21-116, is used to program the time-in values for entering L0s and L1 power management states.

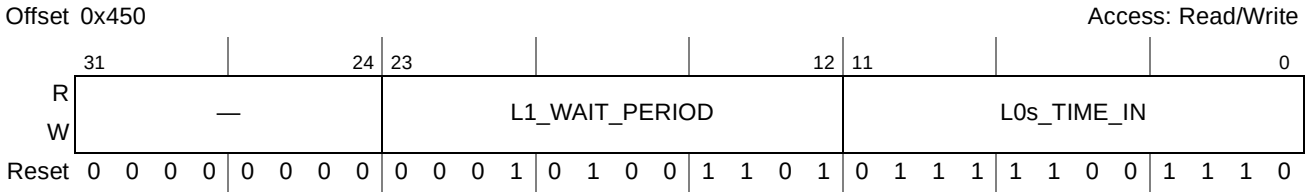


Figure 21-116. PCI Express Power Management Timer Register (PEX_PM_TIMER)

The fields of the PCI Express power management timer register are described in Table 21-113.

Table 21-113. PEX_PM_TIMER Field Descriptions

Bits	Name	Description
31–24	—	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 µsec) × PCI Express controller core clock frequency (in MHz) The time value must be less than 2 µsec; the default value (0x14D) is 1 µsec for the default clock frequency of 333 MHz.
11–0	L0s_TIME_IN	Time in value (in PCI Express controller core clock cycles) for entering L0s power state. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The maximum time value is 7 µsec; the default value (0x7CE) is 6 µsec for the default clock frequency of 333 MHz.

21.3.10.16 PCI Express PME Time-Out Register (EP-Mode Only)—0x454

The PCI Express PME time-out register, shown in Figure 21-117, 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. This register is supported only for EP mode.

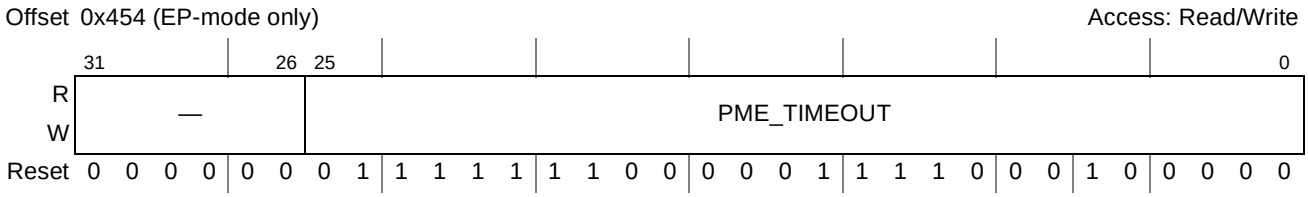


Figure 21-117. PCI Express PME Time-Out Register (PEX_PME_TIMEOUT)

The fields of the PCI Express PME time-out register are described in Table 21-113.

Table 21-114. PEX_PME_TIMEOUT Field Descriptions

Bits	Name	Description
31–26	—	Reserved
25–0	PME_TIMEOUT	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: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The minimum time value is 100 msec; the default value (0x1FC1E20) is 100 msec for the default clock frequency of 333 MHz.

21.3.10.17 PCI Express Subsystem Vendor ID Update Register (EP Mode Only)—0x478

The PCI Express subsystem vendor ID update register, shown in Figure 21-118, is used to set the values for the Subsystem ID and Subsystem Vendor ID registers in the Type 0 configuration header.

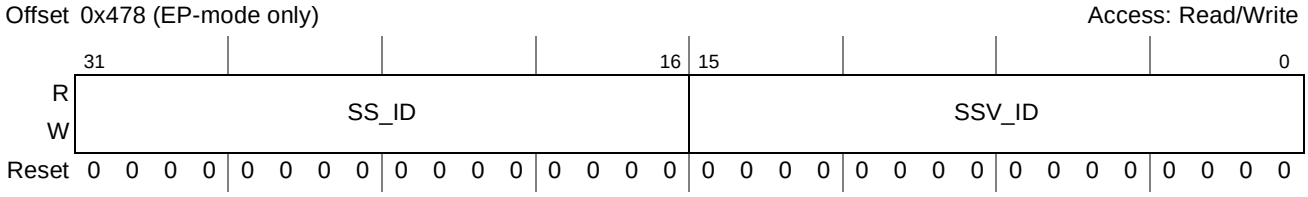


Figure 21-118. PCI Express Subsystem Vendor ID Update Register (PEX_SSVID_UPDATE)

The fields of the PCI Express subsystem vendor ID update register are described in Table 21-115.

Table 21-115. PEX_SSVID_UPDATE Field Descriptions

Bits	Name	Description
31–16	SS_ID	Subsystem ID [15–0] value
15–0	SSV_ID	Subsystem vendor ID [15–0] value

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 Section 21.3.10.18, "Configuration Ready Register—0x4B0"). That way, when the host begins system enumeration, the correct values are present in the Type 0 configuration header.

21.3.10.18 Configuration Ready Register—0x4B0

The PCI Express configuration ready register, shown in Figure 21-119, 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.

Note that the state of PEX_CFG_READY[CFG_READY] is dependent upon the POR configuration settings described in Section 4.4.3.5, "Host/Agent Configuration," and Section 4.4.3.7, "CPU Boot Configuration."

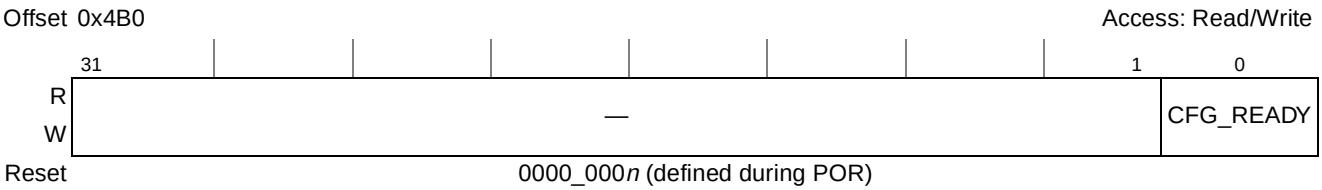


Figure 21-119. PCI Express Configuration Ready Register (PEX_CFG_READY)

The fields of the PCI Express configuration ready register are described in Table 21-116.

Table 21-116. PEX_CFG_READY Field Descriptions

Bits	Name	Description
31–1	—	Reserved
0	CFG_READY	Configuration ready 1 The transaction layer will accept inbound configuration requests. 0 The transaction layer will respond to all inbound configuration requests with retry (CRS) **Note:** The reset state of this bit is determined during POR. This bit defaults to 1 if the PCI Express controller is configured for EP mode and both cores are configured to be in boot holdoff.

21.3.10.19 PME_To_Ack Timeout Register (RC-Mode Only)—0x590

The PCI Express PME_To_Ack timeout register, shown in Figure 21-120, is used to program the timeout value for a PME_To_Ack message response in terms of PCI Express controller core clock cycles.

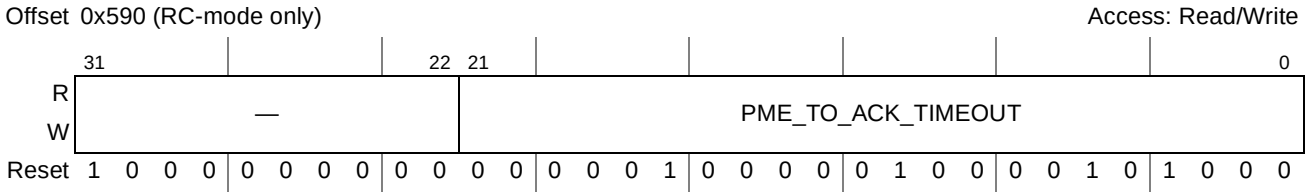


Figure 21-120. PCI Express PME_To_Ack Timeout Register (PEX_PME_TO_ACK_TOR)

The fields of the PCI Express PME_To_Ack timeout register are described in Table 21-117.

Table 21-117. PEX_PME_TO_ACK_TOR Field Descriptions

Bits	Name	Description
31–22	—	Reserved
21–0	PME_TO_ACK_TIMEOUT	After a PME_Turn_Off message is broadcast by the RC, the power management module waits for the duration of the PME_To_Ack timeout interval to receive a PME_To_Ack message from the downstream device. If the Ack message is not received within this interval, the power manager indicates that it is safe to switch off power, since timeout has occurred. The value is calculated as: Time (in µsec) × PCI Express controller core clock frequency (in MHz) The recommended timeout duration is 1 msec to 10 msec to make sure that the downstream devices get enough time to prepare for power-off condition.

21.3.10.20 Secondary Status Interrupt Mask Register (RC-Mode Only)—0x5A0

The PCI Express secondary status interrupt mask register, shown in Figure 21-121, can be used to disable sideband interrupt generation when error bits in the PCI Express secondary status register are set. See Section 21.3.8.3.8, “PCI Express Secondary Status Register—0x1E,” for more information. By default, interrupt generation due to secondary status errors is disabled.

Offset 0x5A0 (RC-mode only) Access: Read/Write

	31–6	5	4	3	2	1	0
R W	—	M_DPE	M_SSE	M_RMA	M_RTA	M_STA	M_MDPE
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	1

Figure 21-121. PCI Express PCI Interrupt Mask Register (PEX_SS_INTR_MASK)

The fields of the PCI Express secondary status interrupt mask register are described in Table 21-118.

Table 21-118. PEX_SS_INTR_MASK Field Descriptions

Bits	Name	Description
31–6	—	Reserved
5	M_DPE	Mask detected parity error
4	M_SSE	Mask signaled system error
3	M_RMA	Mask received master abort
2	M_RTA	Mask received target abort
1	M_STA	Mask signaled target abort
0	M_MDPE	Mask master data parity error

21.4 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.

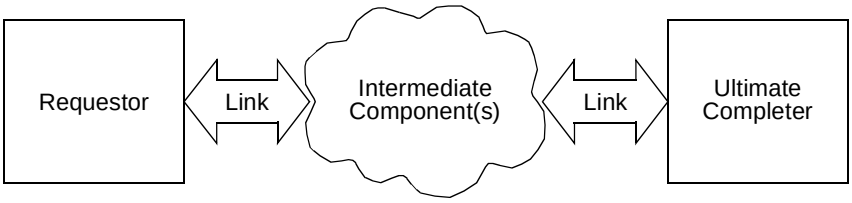


Figure 21-122. Requestor/Completer Relationship

Each PCI 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 21-123.

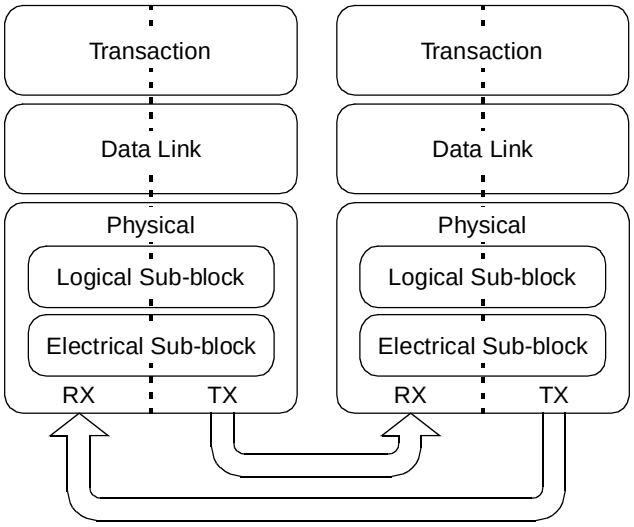


Figure 21-123. PCI Express High-Level Layering

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 21-124. 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.

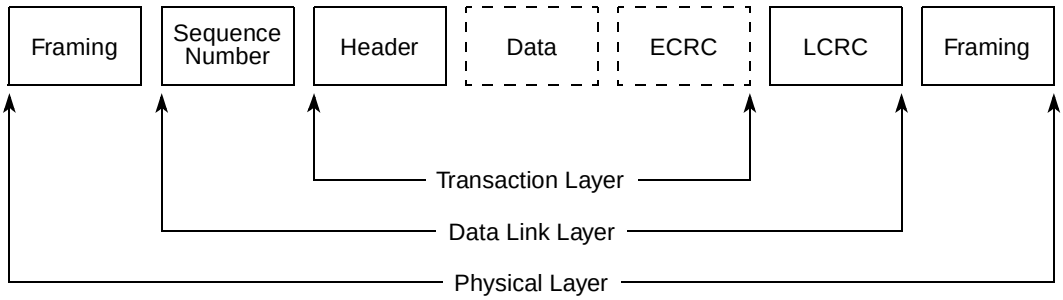


Figure 21-124. PCI Express Packet Flow

21.4.1 Architecture

This section describes implementation details of the PCI Express controller.

21.4.1.1 PCI Express Transactions

Table 21-119 contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 21-119. 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
CplD	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.
CplDLk	No	No	Completion for Locked Memory Read with Data

21.4.1.2 Byte Ordering

Whenever data must 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 byte ordering policy.

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.

Figure 21-125 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.

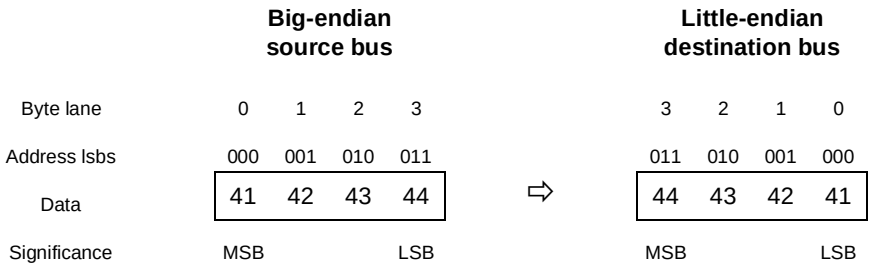


Figure 21-125. Address Invariant Byte Ordering—4 Bytes Outbound

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.

Figure 21-127 shows data flowing the other way, from a little-endian source to a big-endian destination.

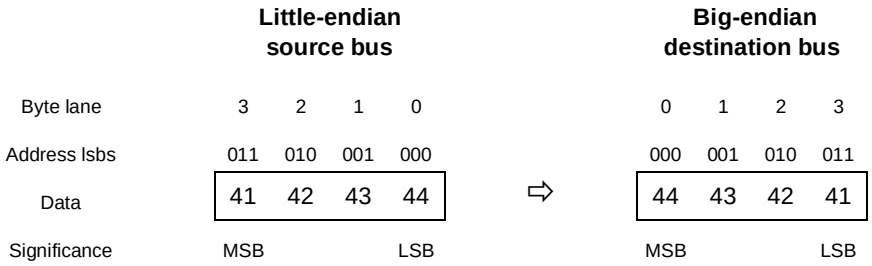


Figure 21-126. Address Invariant Byte Ordering—4 Bytes Inbound

Figure 21-127 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

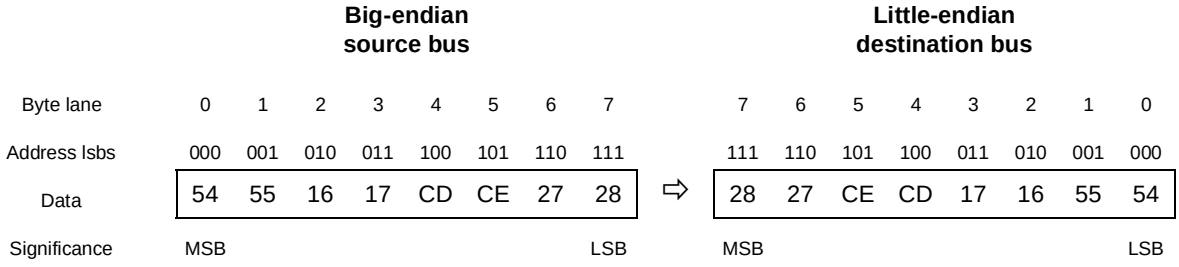


Figure 21-127. Address Invariant Byte Ordering—8 Bytes Outbound

Figure 21-128 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

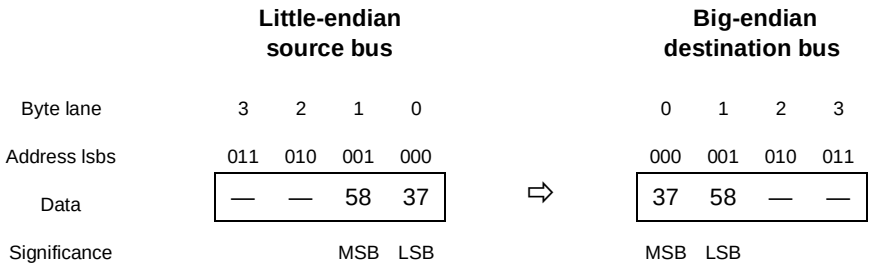


Figure 21-128. Address Invariant Byte Ordering—2 bytes Inbound

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.

21.4.1.2.1 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 the those targeting the internal PCI Express configuration registers, use the address invariance policy as shown in Figure 21-129. Therefore, software must access PEX_CONFIG_DATA with little-endian formatted data—either by using the **lwbrx**/**stwbrx** instructions or by manipulating the data before writing to and after reading from PEX_CONFIG_DATA.

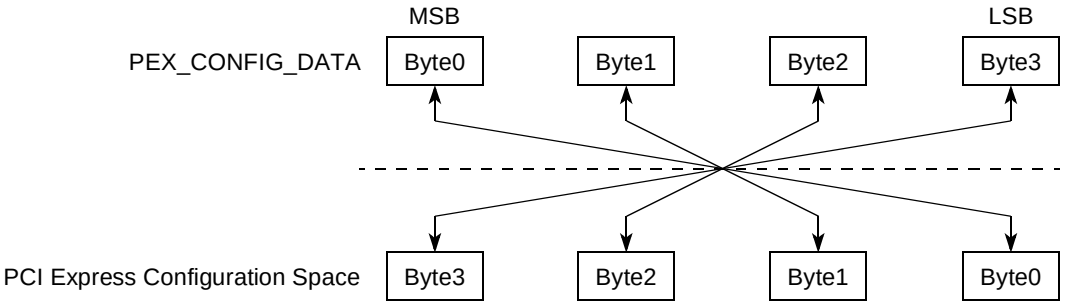


Figure 21-129. PEX_CONFIG_DATA Byte Ordering

21.4.1.3 Lane Reversal

The PCI Express link supports lane reversal. Table 21-120 describes the supported configurations.

Table 21-120. Lane Assignment With and Without Lane Reversal

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

Table 21-120. Lane Assignment With and Without Lane Reversal (continued)

Link Configuration	Lane 0	Lane 1	Lane 2	Lane 3	Lane 4	Lane 5	Lane 6	Lane 7
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 (0–7) 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 8 lanes. That is, if a x4 device is connected to lanes 0–3 and the link training fails without lane reversal, the lane reversal will cause the link to attempt connection on lanes 7–4 which would be impossible.

21.4.1.4 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. The following are the ordering rules for sending the next outstanding request:

- A posted request can and will bypass all other transactions except another posted request.
- A completion can and will only bypass non-posted. It can and will bypass posted requests only if the relaxed ordering (RO) bit is set.
- 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.

21.4.1.5 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.

21.4.1.6 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.

21.4.1.7 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 Section 21.3.7, "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. The PCI Express TLP header for a type 0 configuration read transaction has Type[4:0] = 00100 and Fmt[1] = 0; the PCI Express TLP header for a type 0 configuration write transaction has Type[4:0] = 00100 and Fmt[1] = 1. The PCI Express TLP header for a type 1 configuration read transaction has Type[4:0] = 00101 and Fmt[1] = 0; the PCI Express TLP header for a type 1 configuration write transaction has Type[4:0] = 00101 and Fmt[1] = 1. Note that all configuration transactions sent on PCI Express require a response regardless whether they are read or a write configuration transactions.

The controller does not generate configuration transactions in EP mode. Only inbound configuration transactions are supported in EP mode.

21.4.1.8 Serialization of Configuration and I/O Writes

Configuration and I/O writes originating from the PCI Express outbound ATMUs are serialized by the controller. The logic after issuing a configuration write or IO write will 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 will re-issue the configuration write transaction. It will keep retrying the request until either a status other than CRS is returned or the transaction times out. Note that configuration writes originating from the PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA) are not serialized.

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.

21.4.1.9 Messages

Software message generation is supported in both RC and EP modes.

21.4.1.9.1 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. Table 21-121 describes the message data format.

Table 21-121. Internal Platform (OCeaN) Message Data Format

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.

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 Section 21.3.3.4, "PCI Express Power Management Command Register (PEX_PMCR)," on page 21-18 for more information.

Table 21-122 provides a complete list of supported outbound messages depending on whether RC or EP is configured.

Table 21-122. 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

Table 21-122. PCI Express ATMU Outbound Messages (continued)

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Vendor_Defined Type 1	0111 1111		No	No	Not supported
Attention_Indicator_On	0100 0001	100	Yes	N/A	Hot-plug message
Attention_Indicator_Blink	0100 0011	100	Yes	N/A	Hot-plug message
Attention_Indicator_Off	0100 0000	100	Yes	N/A	Hot-plug message
Power_Indicator_On	0100 0101	100	Yes	N/A	Hot-plug message
Power_Indicator_Blink	0100 0111	100	Yes	N/A	Hot-plug message
Power_Indicator_Off	0100 0100	100	Yes	N/A	Hot-plug message
Attention_Button_Pressed	0100 1000	100	Yes	N/A	Hot-plug message

21.4.1.9.2 Inbound Messages

Table 21-123 provides a complete list of supported inbound messages in RC mode.

Table 21-123. 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 21-123. PCI Express RC Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Attention_Indicator_On	0100 0001	100	No action taken
Attention_Indicator_Blink	0100 0011	100	No action taken
Attention_Indicator_Off	0100 0000	100	No action taken
Power_Indicator_On	0100 0101	100	No action taken
Power_Indicator_Blink	0100 0111	100	No action taken
Power_Indicator_Off	0100 0100	100	No action taken
Attention_Button_Pressed	0100 1000	100	Set PEX_PME_MES_DR[ABP] bit and send interrupt if enabled.

Table 21-124 provides a complete list of supported inbound messages in EP mode.

Table 21-124. 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
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

Table 21-124. PCI Express EP Inbound Message Handling (continued)

Name	Code[7:0]	Routing[2:0]	Action
Attention_Indicator_On	0100 0001	100	Set PEX_PME_MES_DR[AION] bit. Send interrupt if enabled.
Attention_Indicator_Blink	0100 0011	100	Set PEX_PME_MES_DR[AIB] bit. Send interrupt if enabled.
Attention_Indicator_Off	0100 0000	100	Set PEX_PME_MES_DR[AIOF] bit. Send interrupt if enabled.
Power_Indicator_On	0100 0101	100	Set PEX_PME_MES_DR[PION] bit. Send interrupt if enabled.
Power_Indicator_Blink	0100 0111	100	Set PEX_PME_MES_DR[PIB] bit. Send interrupt if enabled.
Power_Indicator_Off	0100 0100	100	Set PEX_PME_MES_DR[PIOF] bit. Send interrupt if enabled.
Attention_Button_Pressed	0100 1000	100	No action taken

21.4.1.10 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 21-130 uncorrectable errors can further be classified as fatal or non-fatal.

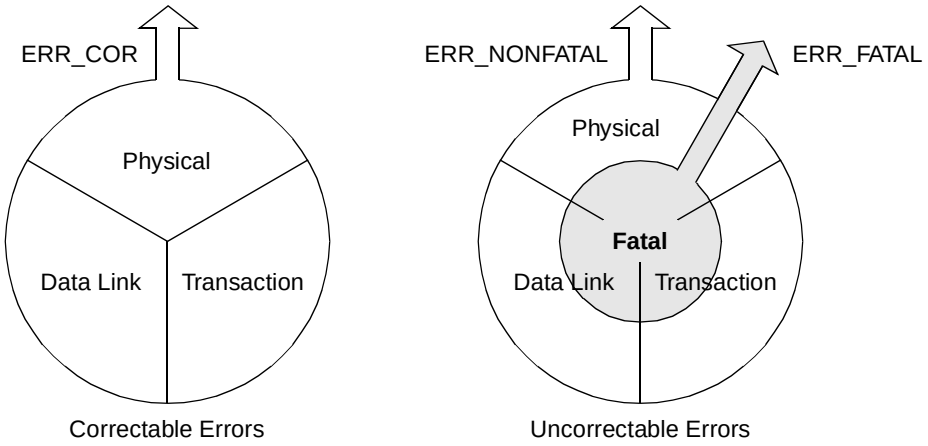


Figure 21-130. PCI Express Error Classification

21.4.1.10.1 PCI Express Error Logging and Signaling

Figure 21-131 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.

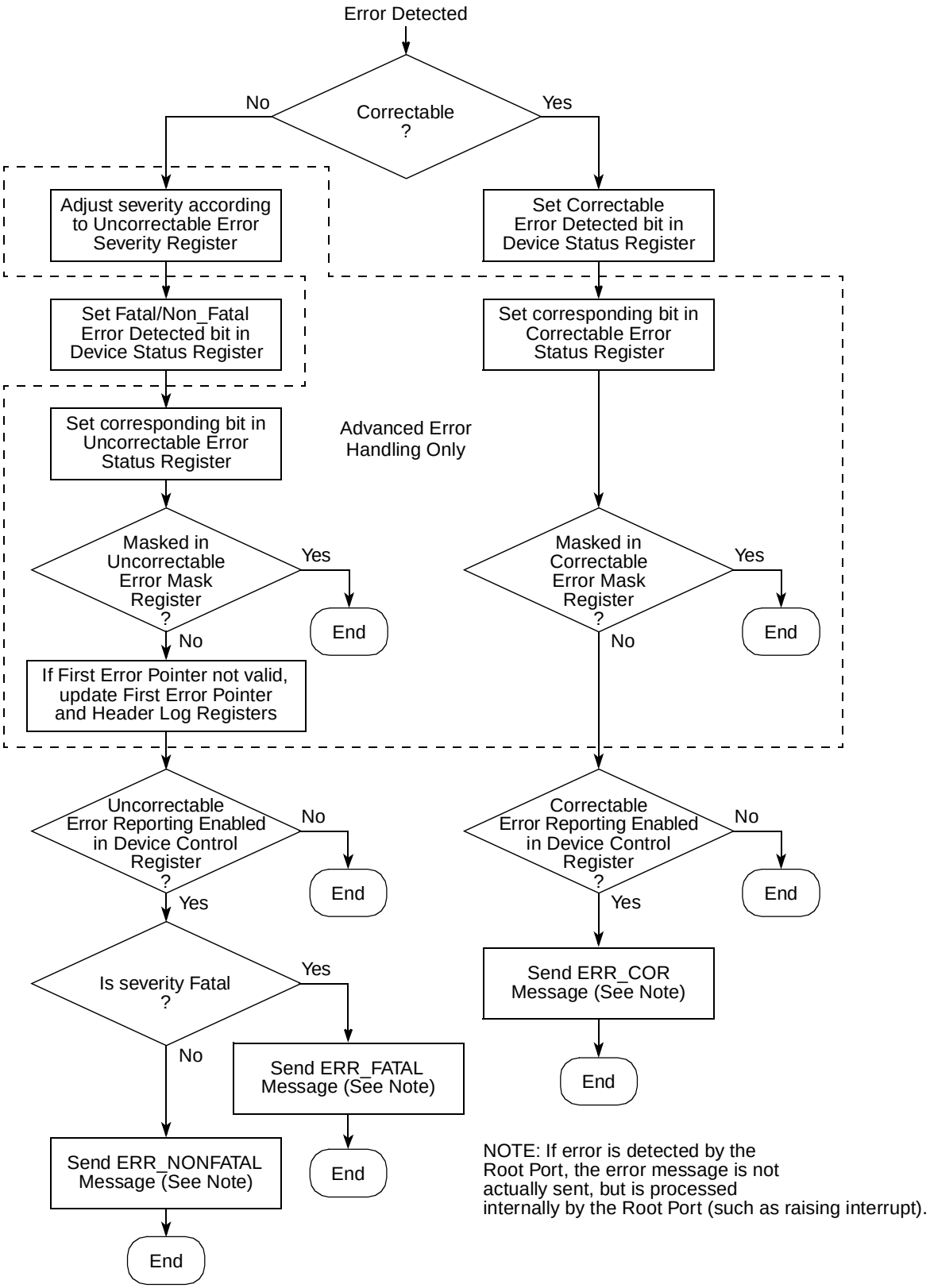


Figure 21-131. PCI Express Device Error Signaling Flowchart

21.4.1.10.2 PCI Express Controller Internal Interrupt Sources

Table 21-125 describes the sources of the PCI Express controller internal interrupt to the PIC and the preconditions for signaling the interrupt.

Table 21-125. 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.
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.

Table 21-125. PCI Express Internal Controller Interrupt Sources (continued)

Status Register Bit	Preconditions
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.

21.4.1.10.3 Error Conditions

Table 21-126 describes specific error types and the action taken for various transaction types.

Table 21-126. 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 will not be 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 will not be 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 will not be 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 will not be 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 (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.

Table 21-126. Error Conditions (continued)

Transaction Type	Error Type	Action
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 will not pass the response back to the core. Therefore, a completion timeout error will eventually occur.
Inbound request	Poisoned TLP (EP=1)	1. If it is a posted transaction, the controller will drop it. 2. If it is a non-posted transaction, the controller will return a completion with UR status. 3. Release the proper credits
Inbound request	ECRC error	1. If it is a posted transaction, the controller will drop it. 2. If it is a non-posted transaction, the controller will return 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 will not be sent out on the link.
Outbound request	Illegal message size	Log the error (PEX_ERR_DR[MIS]). The transaction will not be sent out on the link.
Outbound request	Illegal I/O size	Log the error (PEX_ERR_DR[IOIS]). The transaction will not be sent out on the link.
Outbound request	Illegal I/O address	Log the error (PEX_ERR_DR[IOIA]). The transaction will not be sent out on the link.
Outbound request	Illegal configuration size	Log the error (PEX_ERR_DR[CIS]). The transaction will not be sent out on the link.

Table 21-126. Error Conditions (continued)

Transaction Type	Error Type	Action
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.

21.4.2 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.

21.4.2.1 EP Interrupt Generation

21.4.2.1.1 Hardware INTx Message Generation

Hardware INTx message generation is not supported in EP mode.

21.4.2.1.2 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 *irq_out* (an internal version of the IRQ_OUT signal) to trigger the generation of the MSI. To trigger the MSI, interrupt events must be routed to the to *irq_out* by setting the EP (external pin) bit in the associated Interrupt Destination register in the PIC. Note that the IRQ_OUT signal should not be used for any other function if it is being used to trigger MSI transactions.

The remote root complex is expected 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 *irq_out* 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).

21.4.2.1.3 Software INTx Message Generation

Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in Section 21.4.1.9.1, “Outbound ATMU Message Generation.”

21.4.2.1.4 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 will generate the desired MSI transaction to the remote RC.

21.4.2.2 RC Handling of INTx Message and MSI Interrupts

21.4.2.2.1 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 PIC.

The internal INTx signals from the PCI Express controller are 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 PIC. Refer to Chapter 10, "Programmable Interrupt Controller (PIC)," for more information about handling of PCI Express INTx interrupts and the external interrupt request (IRQ*n*) signals.

If a PCI Express INTx interrupt is being used, then the PIC must be configured so that external interrupts are level-sensitive (EIVPRn[S] = 1).

21.4.2.2.2 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.

21.4.3 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 21-127. Initial credit advertisement

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	4
PD (Memory Write, Message Write)	(256/16)x4=64
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

21.4.4 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 21-128. Power Management State Supported

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffics are stalled. All inbound traffics are 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 via the PEX Power Management Command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffics are 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 via the PEX Power Management Command register.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffics are stalled. All inbound traffics are 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 via 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 will restart.
D3cold	L3	Completely off.

21.4.4.1 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 $\overline{\text{WAKE}}$ 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 $\overline{\text{WAKE}}$ signal, as shown in Figure 21-132.

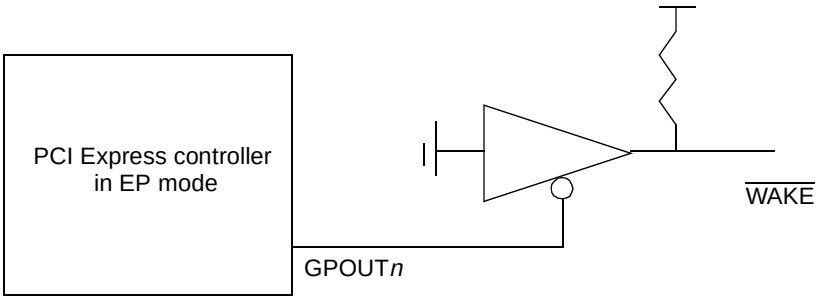


Figure 21-132. $\overline{\text{WAKE}}$ Generation Example

In RC mode, the $\overline{\text{WAKE}}$ signal from the EP device can be connected to one of the external interrupt inputs to service the $\overline{\text{WAKE}}$ request.

21.4.5 Hot Reset

When a hot reset condition occurs, the controller (in both RC and EP mode) will initiate 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 will take 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 detected a hot reset condition and initiates the clean-up procedure appropriately.

21.4.6 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 will return 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.

21.5 Initialization/Application Information

21.5.1 Boot Mode and Inbound Configuration Transactions

In normal boot mode (cfg_cpu_boot = 1), the core is allowed to boot and configure the device. During this time, the PCI Express interface will retry 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 Section 21.3.10.18, “Configuration Ready Register—0x4B0,” for more information about the CFG_READY bit.

In boot hold-off mode (cfg_cpu_boot = 0), 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 EEBPCR[CPU*n*_EN] bit after which the core is granted the internal bus.

Chapter 22
General Purpose I/O (GPIO)

22.1 Introduction

This chapter describes the general-purpose I/O module, including pin descriptions, register settings, and interrupt capabilities. Figure 22-1 shows the block diagram of the GPIO module.

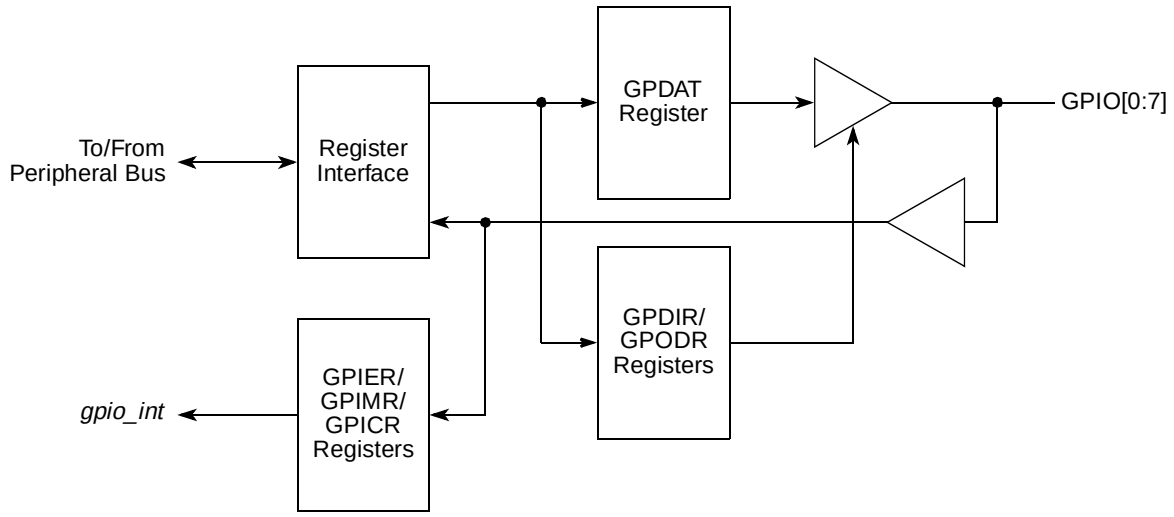


Figure 22-1. GPIO Module Block Diagram

22.1.1 Overview

The GPIO module supports 8 general-purpose I/O ports. Each port can be configured as an input or as an output. If a port is configured as an input, it can optionally generate an interrupt on 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.

22.1.2 Features

The GPIO unit implements the following features:

- 8 input/output ports
- All signals are configured as inputs when the device comes out of reset and also when $\overline{\text{HRESET}}$ is asserted.
- Open-drain capability on all ports
- All ports can optionally generate an interrupt upon changing their state.

22.2 External Signal Description

The following section provides information about GPIO signals.

22.2.1 Signals Overview

Table 22-1 provides detailed descriptions of the external GPIO signals.

Table 22-1. IPIC External Signals—Detailed Signal Descriptions

Signal	I/O	Description	
GPIO[0:7]	I/O	General purpose I/O. Each signal can be set individually to act as input or output, according to application needs.	
		State Meaning	Asserted/Negated—Defined per application.
		Timing	Assertion/Negation—Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock

22.3 Memory Map/Register Definition

The GPIO has programmable registers that occupy memory-mapped space beginning at offset 0x0_F000. Note that reading undefined portions of the memory map returns all zeros and writing has no effect.

All GPIO registers are 32 bits wide and are located on 32-bit address boundaries.

Table 22-2 shows the memory map of GPIO.

Table 22-2. GPIO Register Address Map

Offset	Register	Access	Reset Value	Section/Page
0xC00	GPIO direction register (GPDIR)	R/W	0x0000_0000	22.3.1/22-2
0xC04	GPIO open drain register (GPODR)	R/W	0x0000_0000	22.3.2/22-3
0xC08	GPIO data register (GPDAT)	R/W	0x0000_0000	22.3.3/22-3
0xC0C	GPIO interrupt event register (GPIER)	w1c	Undefined	22.3.4/22-4
0xC10	GPIO interrupt mask register (GPIMR)	R/W	0x0000_0000	22.3.5/22-4
0xC14	GPIO external interrupt control register (GPICR)	R/W	0x0000_0000	22.3.6/22-5

22.3.1 GPIO Direction Register (GPDIR)

The GPIO direction registers (GPDIR), shown in Figure 22-2, defines the direction of the individual ports.

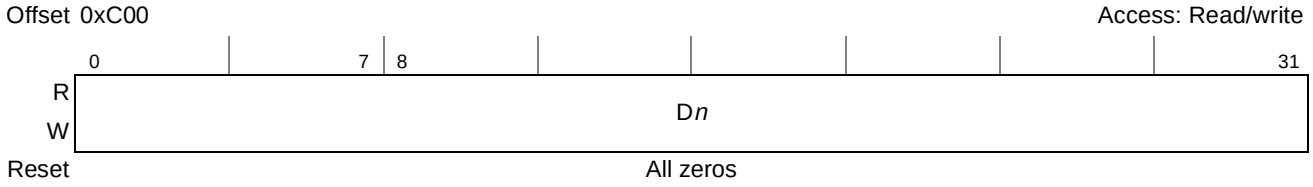


Figure 22-2. GPIO Direction Register (GPDIR)

Table 22-3 defines the bit fields of GPDIR.

Table 22-3. GPDIR Bit Settings

Bits	Name	Description
0–31	D*n*	Direction. Indicates whether a signal is used as an input or an output. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused. 0 The corresponding signal is an input. 1 The corresponding signal is an output.

22.3.2 GPIO Open Drain Register (GPODR)

The GPIO open drain register (GPODR), shown in Figure 22-3, defines the way individual ports drive their output.

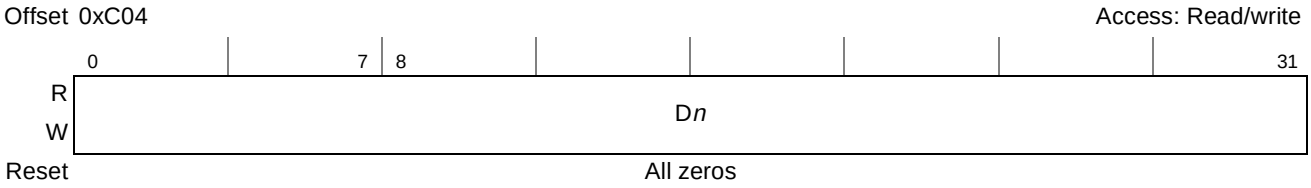


Figure 22-3. GPIO Open Drain Register (GPODR)

Table 22-4 defines the bit fields of GPODR.

Table 22-4. GPODR Bit Settings

Bits	Name	Description
0–31	D*n*	Open-drain configuration. Indicates whether a signal is actively driven as an output or an open-drain driver. This register has no effect on signals programmed as inputs in the corresponding GPDIR. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused. 0 The I/O signal is actively driven as an output. 1 The I/O signal is an open-drain driver. As an output, the signal is driven active-low, otherwise it is three-stated.

22.3.3 GPIO Data Register (GPDAT)

The GPIO data register (GPDAT), shown in Figure 22-4, carries the data in/out for the individual ports.

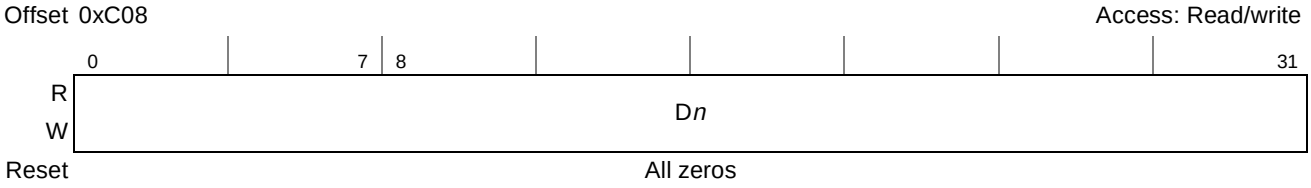


Figure 22-4. GPIO Data Register (GPDAT)

Table 22-5 defines the bit fields of GPDAT.

Table 22-5. GP*n*DAT Bit Settings

Bits	Name	Description
0–31	D*n*	Data. Write data is latched and presented on external signals if GPDIR has configured the port as an output. Read operation always returns the data at the signal. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused.

22.3.4 GPIO Interrupt Event Register (GPIER)

The GPIO interrupt event register (GPIER), shown in Figure 22-5, 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.

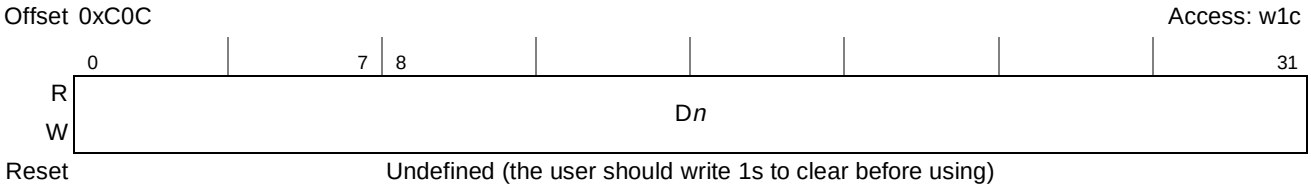


Figure 22-5. GPIO Interrupt Event Register (GPIER)

Table 22-6 defines the bit fields of GPIER.

Table 22-6. GPIER Bit Settings

Bits	Name	Description
0–31	D*n*	Interrupt events. Indicates whether an interrupt event occurred on the corresponding GPIO signal. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused. 0 No interrupt event occurred on the corresponding GPIO signal. 1 Interrupt event occurred on the corresponding GPIO signal.

22.3.5 GPIO Interrupt Mask Register (GPIMR)

The GPIO interrupt mask register (GPIMR), shown in Figure 22-6, 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 on chip interrupt controller.

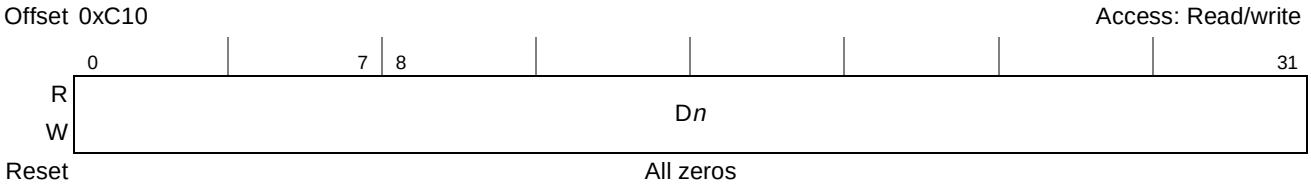


Figure 22-6. GPIO Interrupt Mask Register (GPIMR)

Table 22-7 defines the bit fields of GPIMR.

Table 22-7. GPIMR Bit Settings

Bits	Name	Description
0–31	D*n*	Interrupt mask. Indicates whether an interrupt event is masked or not masked. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused. 0 The input interrupt signal is masked (disabled). 1 The input interrupt signal is not masked (enabled).

22.3.6 GPIO Interrupt Control Register (GPICR)

The GPIO interrupt control register (GPICR), shown in Figure 22-7, determines whether the corresponding port line asserts an interrupt request on either a high-to-low change or any change on the state of the signal.

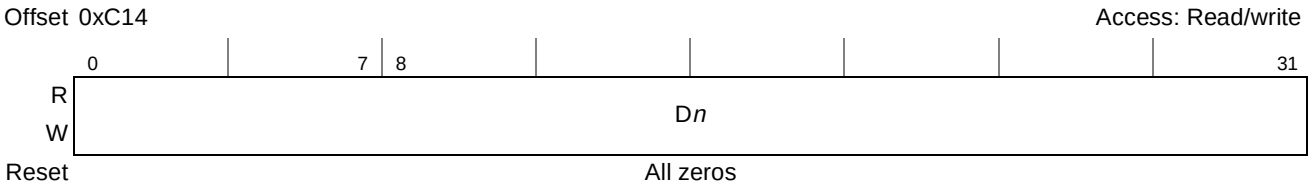


Figure 22-7. GPIO Interrupt Control Register (GPICR)

Table 22-8 defines the bit fields of GPICR.

Table 22-8. GPICR Bit Settings

Bits	Name	Description
0–31	D*n*	Edge detection mode. Bits D[0:7] correspond to signals GPIO[0:7]. Bits D[8:31] are unused. 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.

Part IV
Global Functions and Debug

Part IV defines other global blocks of the MPC8572E. The following chapters are included:

- Chapter 23, “Global Utilities,” defines the global utilities of the MPC8572E. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and signal multiplexing capabilities.
- Chapter 24, “Device Performance Monitor,” describes the performance monitor of the MPC8572E. Note that the MPC8572E performance monitor is similar to but separate from the performance monitor implemented on the e500v2 core.
- Chapter 25, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8572E.

Chapter 23
Global Utilities

This chapter describes the global utilities of the MPC8572E. It provides signal descriptions, register descriptions, and a functional description of these utilities.

23.1 Overview

The global utilities block controls power management, I/O device enabling, power-on-reset (POR) configuration monitoring, alternate function selection for multiplexed signals, and clock control.

23.2 Global Utilities Features

This section provides an overview of global utilities features.

23.2.1 Power Management and Block Disables

The following features affect the device's overall power consumption:

- Dynamic power management mode
- Software-controlled power management (doze, nap, sleep)
- Externally controlled power management (doze, sleep)
- Static power management (I/O block disables)

23.2.2 Accessing Current POR Configuration Settings

The POR configuration values of all device parameters sampled from pins at reset are available through memory-mapped registers in the global utilities block.

23.2.3 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

23.3 External Signal Description

The following subsections provide information about signals that serve as global utilities.

23.3.1 Signals Overview

Table 23-1 summarizes the external signals used by the global utilities block.

Table 23-1. External Signal Summary

Signal Name	I/O	Description	Reference (Section/page)
ASLEEP	O	Signals that the device has reached a sleep state.	23.5.1.5.3/23-33
$\overline{\text{CKSTP\_IN0}}$	I	Checkstop input 0	Table 23-2 on page 23-2
$\overline{\text{CKSTP\_IN1}}$	I	Checkstop input 1	Table 23-2 on page 23-2
$\overline{\text{CKSTP\_OUT0}}$	O	Checkstop output 0.	Table 23-2 on page 23-2
$\overline{\text{CKSTP\_OUT1}}$	O	Checkstop output 1.	Table 23-2 on page 23-2
CLK_OUT	O	Clock out. Selected by CLKOCR values.	23.4.1.20/23-26

23.3.2 Detailed Signal Descriptions

Table 23-2 describes signals in the global utilities block in detail.

Table 23-2. Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. See Section 23.5.1.5.3, “Sleep Mode.” After negation of $\overline{\text{HRESET}}$, 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 awakened 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.
$\overline{\text{CKSTP\_IN0}}$	I	Checkstop in 0	
		State Meaning	Asserted—Indicates that the e500 core 0 must enter a hard stop condition. All e500 clocks are turned off. $\overline{\text{CKSTP\_OUT0}}$ is asserted. The rest of MPC8572E device logic, including memory controllers, internal memories and registers, and I/O interfaces, remains functional. Negated—Indicates that normal operation should proceed.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the MPC8572E is reset with assertion of $\overline{\text{HRESET}}$.
$\overline{\text{CKSTP\_IN1}}$	I	Checkstop in 1	
		State Meaning	Asserted—Indicates that the e500 core 1 must enter a hard stop condition. All e500 core 1 clocks are turned off. $\overline{\text{CKSTP\_OUT1}}$ is asserted. The rest of MPC8572E device logic, including memory controllers, internal memories and registers, and I/O interfaces, remains functional. Negated—Indicates that normal operation should proceed.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the MPC8572E is reset with assertion of $\overline{\text{HRESET}}$.

Table 23-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{CKSTP\_OUT0}}$	O	Checkstop out 0	
		State Meaning	Asserted—Indicates that the e500 core 0 of the MPC8572E is in a checkstop state. The rest of the device logic remains functional. Negated—Indicates normal operation. After $\overline{\text{CKSTP\_OUT0}}$ has been asserted, it is negated after the next negation (low-to-high transition) of $\overline{\text{HRESET}}$.
		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 hard reset.
$\overline{\text{CKSTP\_OUT1}}$	O	Checkstop out 1	
		State Meaning	Asserted—Indicates that the e500 core 1 of the MPC8572E is in a checkstop state. The rest of the device logic remains functional. Negated—Indicates normal operation. After $\overline{\text{CKSTP\_OUT1}}$ has been asserted, it is negated after the next negation (low-to-high transition) of $\overline{\text{HRESET}}$.
		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 hard reset.
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR (see Section 23.4.1.20, “Clock Out Control Register (CLKOCR)”).	
		State Meaning	Asserted—If CLKOCR[ENB] = 1, clock signal selected by CLKOCR[CLK_SEL] is driven. High impedance—If CLKOCR[ENB] = 0.
		Timing	Assertion/Negation—Depends on the value of CLKOCR[CLK_SEL].

23.4 Memory Map/Register Definition

Table 23-3 summarizes the global utilities registers and their addresses.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 23-3. Global Utilities Block Register Summary

Offset	Register	Access	Reset	Section/page
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x*nnnn_nnnn*	23.4.1.1/23-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	23.4.1.2/23-6
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	Mixed	0x0000_007F	23.4.1.3/23-7
0xE_000C	PORDEVSR—POR device status register	R	see ref.	23.4.1.4/23-7

Table 23-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/page
0xE_0010	PORDBGMSR—POR debug mode status register	R	see ref.	23.4.1.5/23-11
0xE_0014	PORDEVSR2—POR device status register 2	R	0x0000_*n*00*n*	23.4.1.6/23-12
0xE_0020	GPPORCR—General-purpose POR configuration register	R	see ref.	23.4.1.7/23-13
Signal Multiplexing				
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	23.4.1.8/23-13
Device Disables				
0xE_0070	DEVDISR—Device disable control register	R/W	0xnn0*n*_00*nn*	23.4.1.9/23-14
Power Management Registers				
0xE_0080	POWMGTCSR—Power management control and status register	Mixed	0x0000_0000	23.4.1.10/23-16
Interrupt and Reset Status and Control				
0xE_0090	MCPSUMR—Machine check summary register	w1c	0x0000_0000	23.4.1.11/23-18
0xE_0094	RSTRSCR—Reset request status and control register	Mixed	0x0000_0000	23.4.1.12/23-20
0xE_0098	ECTRSTCR—Exception reset control register	Mixed	0x0000_0000	23.4.1.13/23-20
0xE_009C	RSTSR—Automatic reset status register	Mixed	0x0000_0000	23.4.1.14/23-21
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	23.4.1.15/23-22
0xE_00A4	SVR—System version register	R	MPC8572E system version	23.4.1.16/23-23
Status Registers				
0xE_00B0	RSTCR—Reset control register	R/W	0x0000_0000	23.4.1.17/23-23
0xE_00C0	LBCVSELCR—LBC voltage select control register	R/W	0x0000_0000	23.4.1.18/23-24
0xE_0B28	DDRCLKDR—DDR clock disable register	R/W	0x0000_0000	23.4.1.19/23-24
Debug Control Registers				
0xE_0E00	CLKOCR—Clock out control register	R/W	0x0000_0000	23.4.1.20/23-26
0xE_0F04	SRDS1CR—SerDes1 control register	R/W	0x1100_4418	23.4.1.21/23-27
0xE_0F10	SRDS2CR—SerDes2 control register	R/W	0x1100_4418	23.4.1.22/23-28

23.4.1 Register Descriptions

This section describes the global utilities registers in detail.

23.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in Figure 23-1, contains the settings for the PLL ratios as set by the cfg_sys_pll[0:3], cfg_ddr_pll[0:3], cfg_core0_pll[0:1], and cfg_core1_pll[0:1] POR configuration pins. See Section 4.4.3.1,

"System PLL Ratio," Section 4.4.3.2, "DDR PLL Ratio," and Section 4.4.3.3, "e500 Core PLL Ratios," for more information.

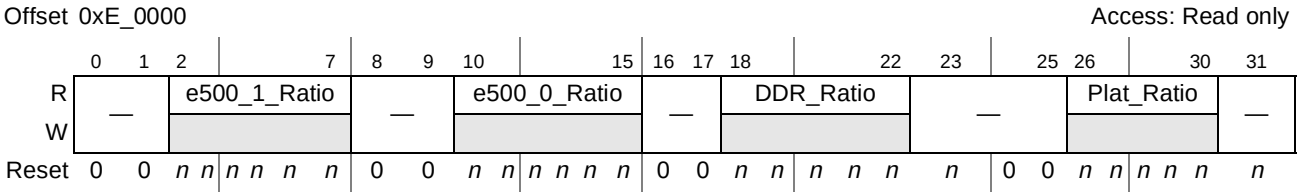


Figure 23-1. POR PLL Status Register (PORPLLSR)

Table 23-4 describes the bit settings of PORPLLSR.

Table 23-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–7	e500_1_Ratio	Clock ratio between the e500 core 1 and the CCB clock. The 3 lsbs of this field correspond to the values on cfg_core1_pll[0:2] at the negation of HRESET. (See Section 4.4.3.3, "e500 Core PLL Ratios.") Patterns not shown are reserved. 000011 3:2 000100 2:1 000101 5:2 000110 3:1 000111 7:2
8–9	—	Reserved
10–15	e500_0_Ratio	Clock ratio between the e500 core 0 and the CCB clock. The 3 lsbs of this field correspond to the values on cfg_core0_pll[0:2] at the negation of HRESET. (See Section 4.4.3.3, "e500 Core PLL Ratios.") Patterns not shown are reserved. 000011 3:2 000100 2:1 000101 5:2 000110 3:1 000111 7:2
16–17	—	Reserved
18–22	DDR_Ratio	Clock ratio between the DDR Complex clock and DDRCLK. Patterns not shown are reserved. 000011 3:1 000100 4:1 000110 6:1 001000 8:1 001010 10:1 001100 12:1 001110 14:1 000111 Synchronous Mode-DDR Complex Clocked by CCB clock
23–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the CCB (platform) clock and SYSCLK. Patterns not shown are reserved. 00100 4:1 00101 5:1 00110 6:1 01000 8:1 01010 10:1 01100 12:1
31	—	Reserved

23.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 23-2, reports setting of the POR configuration pins that control the boot mode settings (described in Section 4.4.3.4, “Boot ROM Location,” Section 4.4.3.7, “CPU Boot Configuration,” and Section 4.4.3.8, “Boot Sequencer Configuration”) and the default settings of PCI Express and Serial RapidIO host/agent mode (described in Section 4.4.3.5, “Host/Agent Configuration”).

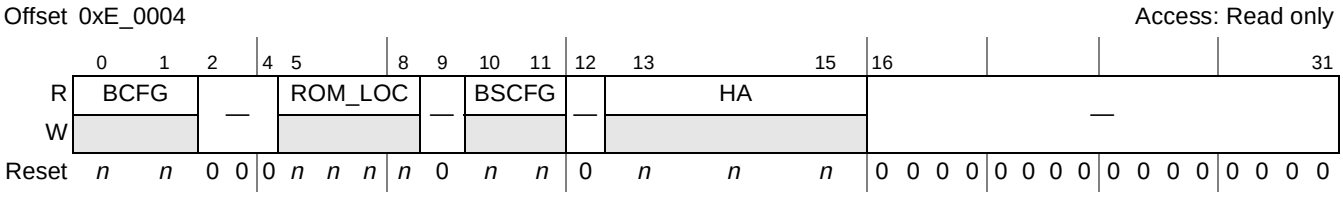


Figure 23-2. POR Boot Mode Status Register (PORBMSR)

Figure 23-5 describes the bit settings of the PORBMSR.

Table 23-5. PORBMSR Field Descriptions

Bits	Name	Description
0–1	BCFG	CPU boot configuration (See Section 4.4.3.7, “CPU Boot Configuration.”) 00 Both e500 cores are prevented from booting until configuration by an external master is complete. 01 The e500 core 1 is allowed to start fetching boot code, while e500 core 0 is prevented from booting until configuration by an external master or e500 core 0 is complete. 10 The e500 core 0 is allowed to start fetching boot code, while e500 core 1 is prevented from booting until configuration by an external master or e500 core 1 is complete. 11 Both e500 cores are allowed to start fetching boot code (default).
2–4	—	Reserved
5–8	ROM_LOC	Location of boot ROM. (See Section 4.4.3.4, “Boot ROM Location.”) 0000 PCI Express 1 0001 PCI Express 2 0010 Serial RapidIO 0011 Reserved 0100 DDR controller 1 0101 DDR controller 2 0110 DDR Interleaved 0111 PCI Express 3 1000 Local bus FCM: 8-bit NAND Flash small page 1001 Reserved 1010 Local bus FCM: 8-bit NAND Flash large page 1011 Reserved 1100 Reserved 1101 Local bus GPCM: 8-bit ROM 1110 Local bus GPCM: 16-bit ROM 1111 Local bus GPCM: 32-bit ROM (Default)
9	—	Reserved
10–11	BSCFG	Boot sequencer configuration (See Section 4.4.3.8, “Boot Sequencer Configuration.”) 00 Reserved 01 Boot sequencer enabled with normal I^2C addressing 10 Boot sequencer enabled with extended I^2C addressing 11 Boot sequencer disabled

Table 23-5. PORBMSR Field Descriptions (continued)

Bits	Name	Description
12	—	Reserved
13–15	HA	Host/agent mode configuration. When the MPC8572E is an agent on an interface, it is prevented from mastering transactions on that interface until the external host configures the interface appropriately. (See Section 4.4.3.5, “Host/Agent Configuration.”) 000 Agent on every interface 001 PCI Express 1 end point 010 PCI Express 2 end point / Serial RapidIO agent mode 011 PCI Express 3 end point 100 PCI Express 1 end point and PCI Express 2 end point / Serial RapidIO agent mode 101 PCI Express 1 end point and PCI Express 3 endpoint 110 PCI Express 2 end point / Serial RapidIO agent mode and PCI Express 3 end point 111 Root complex (Host mode)
16–31	—	Reserved

23.4.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

PORIMPSCR, shown in Figure 23-3,contains the current I/O driver impedances for local bus interfaces.

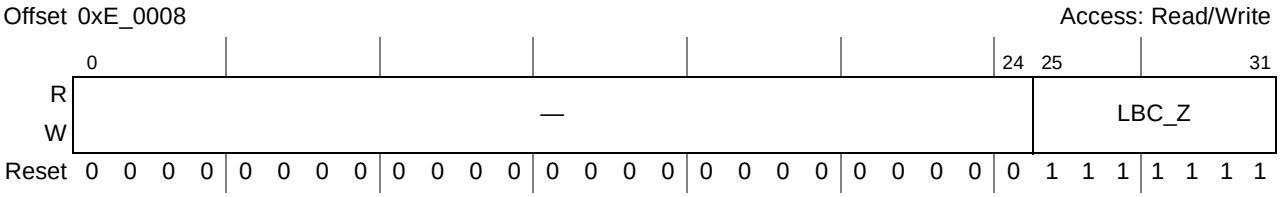


Figure 23-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

The I/O impedance of local bus signals (including the local bus clock) is controlled through this register. The *MPC8572E Integrated Processor Hardware Specification* provides exact I/O impedances.

Table 23-6 describes PORIMPSCR fields.

Table 23-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	LBC_Z	I/O impedance for these local bus signals: LAD[0:31], LDP[0:3], LA[27:31], $\overline{\text{LCS}}$[1:2], $\overline{\text{LWE}}$[0:3], LGPL[0:5], LCLK[0:2] Note: Other signals (for example, LALE, $\overline{\text{LCS}}$[0], $\overline{\text{LCS}}$[3:7]) use a fixed high I/O impedance 1111111 High impedance else Low impedance

23.4.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 23-4, PORDEVSR reports other POR settings for I/O devices as described in Section 4.4.3.12, “SGMII SerDes Reference Clock Configuration,” Section 4.4.3.14, “eTSEC3 and eTSEC4 Width,” and Section 4.4.3.16, “eTSEC2 Protocol.”

Offset 0xE_000C Access: Read Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ECW1	ECW2	SGMII1_DIS	SGMII2_DIS	SGMII3_DIS	SGMII4_DIS	ECP1		—	IO_SEL				—		
W																
Reset	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	0	*n*	*n*	*n*	*n*	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—		ECP2		ECP3		ECP4		FEC_DIS	RTYPE	—		RIO_CTLS	DEV_ID		
W																
Reset	0	0	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	0	0	*n*	*n*	*n*	*n*

Figure 23-4. POR Device Status Register (PORDEVSR)

Table 23-7 describes the bit settings of PORDEVSR.

Table 23-7. PORDEVSR Field Descriptions

Bits	Name	Description
0	ECW1	eTSEC1 and eTSEC2 controller width (See Section 4.4.3.13, “eTSEC1 and eTSEC2 Width.”) 0 Reduced interfaces for eTSEC1 and eTSEC2 (RMII, RGMII, RTBI, or 8-bit FIFO on eTSEC1) 1 Full interfaces for eTSEC1 and eTSEC2 (MII, GMII, TBI, or 16-bit FIFO on eTSEC1) note: FIFO mode on eTSEC2 is always 8-bits regardless of ECW1
1	ECW2	eTSEC3 and eTSEC4 controller width (See Section 4.4.3.14, “eTSEC3 and eTSEC4 Width.”) 0 Reduced interfaces for eTSEC3 and eTSEC4 (RMII, RGMII, RTBI) 1 Full interface for eTSEC3 (MII, GMII, TBI); eTSEC4 is available in SGMII mode if SGMII4_DIS = 0 (else eTSEC4 is disabled) note: FIFO mode on eTSEC3 is always 8-bits regardless of ECW2.
2	SGMII1_DIS	eTSEC1 in SGMII mode disabled (See Section 4.4.3.11, “eTSEC SGMII Mode.”) 0 eTSEC1 in SGMII mode 1 eTSEC1 in parallel mode
3	SGMII2_DIS	eTSEC2 in SGMII mode disabled (See Section 4.4.3.11, “eTSEC SGMII Mode.”) 0 eTSEC2 in SGMII mode 1 eTSEC2 in parallel mode
4	SGMII3_DIS	eTSEC3 SGMII mode disabled (See Section 4.4.3.11, “eTSEC SGMII Mode.”) 0 eTSEC3 in SGMII mode 1 eTSEC3 in parallel mode
5	SGMII4_DIS	eTSEC4 SGMII mode disabled (See Section 4.4.3.11, “eTSEC SGMII Mode.”) 0 eTSEC4 in SGMII mode 1 eTSEC4 in parallel mode
6–7	ECP1	eTSEC1 controller protocol (See Section 4.4.3.15, “eTSEC1 Protocol.”) 00 The eTSEC1 controller operates using the 16-bit FIFO protocol (or 8-bit FIFO if configured in reduced mode). 01 The eTSEC1 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode). note: 16-bit FIFO mode on eTSEC1, TSEC2 is available in SGMII mode if SGMII2_DIS = 0 (else TSEC2 is disabled).

Table 23-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
8	—	Reserved
9–12	IO_SEL *(1 of 2)*	I/O port selection mode (See Section 4.4.3.6, "I/O Port Selection.") 0000 Reserved 0001 Reserved 0010 PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] 0011 PCI Express 1 (x4) (2.5 Gbps), PCI Express 2 (x4) (2.5 Gbps) *100-MHz reference clock* PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] PCI Express 2: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] 0100 Reserved 0101 Reserved 0110 Serial RapidIO x4 (2.5 Gbps) *100 MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] 0111 PCI Express 1 (x4), PCI Express 2 (x2), PCI Express 3 (x2) *100-MHz reference clock* PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] PCI Express 2: RX lane[0:1] -> SD1_RX[4:5] TX lane[0:1] -> SD1_TX[4:5] PCI Express 3: RX lane[0:1] -> SD1_RX[6:7] TX lane[0:1] -> SD1_TX[6:7]

Table 23-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
9–12	IO_SEL *(2 of 2)*	1000 Reserved 1001 Reserved 1010 Reserved 1011 Serial RapidIO x4 (2.5 Gbps), PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* Serial RapidIO: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] 1100 Serial RapidIO x4 (1.25 Gbps), PCI Express 1 (x4) (2.5 Gbps) *100-MHz reference clock* Serial RapidIO: RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] PCI Express 1: RX lane[0:3] -> SD1_RX[0:3] TX lane[0:3] -> SD1_TX[0:3] 1101 Serial RapidIO x4 (3.125 Gbps) *125-MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] 1110 Serial RapidIO x4 (1.25 Gbps) *100-MHz reference clock* RX lane[0:3] -> SD1_RX[4:7] TX lane[0:3] -> SD1_TX[4:7] 1111 PCI Express 1 (x8) (2.5 Gbps) *100-MHz reference clock* RX lane[0:7] -> SD1_RX[0:7] TX lane[0:7] -> SD1_TX[0:7]
13–17	—	Reserved
18–19	ECP2	eTSEC2 controller protocol (See Section 4.4.3.16, “eTSEC2 Protocol.”) 00 The eTSEC2 controller operates using the 8-bit FIFO protocol. 01 The eTSEC2 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC2 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC2 controller operates using the TBI protocol (or RTBI if configured in reduced mode).
20–21	ECP3	eTSEC3 controller protocol (See Section 4.4.3.17, “eTSEC3 Protocol.”) 00 The eTSEC3 controller operates using the 8-bit FIFO protocol. 01 The eTSEC3 controller operates using the MII protocol (or RMII if configured in reduced mode). 10 The eTSEC3 controller operates using the GMII protocol (or RGMII if configured in reduced mode). 11 The eTSEC3 controller operates using the TBI protocol (or RTBI if configured in reduced mode). Note: Operation of eTSEC3 in MII, GMII, TBI, or FIFO modes disables eTSEC4 if SGMII4_DIS is set.

Table 23-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
22–23	ECP4	eTSEC4 controller protocol (See Section 4.4.3.18, “eTSEC4 Protocol.”) 00 Reserved 01 The eTSEC4 controller operates RMII 10 The eTSEC4 controller operates using the RGMII protocol 11 The eTSEC4 controller operates using the RTBI protocol Note: eTSEC4 is only available when SGMII4_DIS is cleared or when eTSEC3 is operating in a reduced interface mode—RMII, RGMII, or RTBI. eTSEC4 does not support FIFO mode.
24	FEC_DIS	Fast Ethernet controller disable (See Section 4.4.3.10, “FEC Configuration.”) 0 FEC enabled 1 FEC disabled Note: If the FEC is enabled, eTSEC3 and eTSEC4 are only available in SGMII mode.
25	RTYPE	DRAM Type for DDR Controllers (See Section 4.4.3.9, “DDR SDRAM Type.”) 0 DDR2 (1.8 V, CKE low at reset) 1 DDR3 (1.5 V, CKE low at reset)
26–27	—	Reserved
28	RIO_CTLS	RapidIO system size (See Section 4.4.3.20, “RapidIO System Size.”) 0 Device does not support common transport large systems (up to 256 devices) 1 Device supports common transport large systems (up to 65,536 devices)
29–31	DEV_ID	Serial RapidIO device ID (See Section 4.4.3.19, “RapidIO Device ID.”)

23.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 23-5, holds debug mode settings from the POR configuration pins as described in Section 4.4.3.21, “Memory Debug Configuration,”and Section 4.4.3.22, “DDR Debug Configuration.”

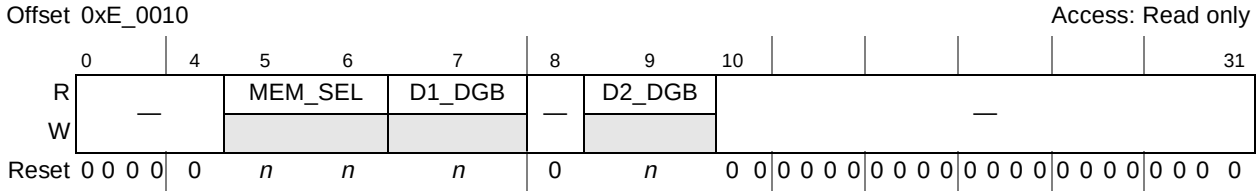


Figure 23-5. POR Debug Mode Status Register (PORDBGMSR)

Table 23-8 describes the bit settings of PORDBGMSR.

Table 23-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–6	MEM_SEL	Memory select. Indicates which controller is driving MSRCID[0:4] and MDVAL. (See Section 4.4.3.21, “Memory Debug Configuration.”) 00 Local bus controller is driving debug information 01 Reserved 10 DDR SDRAM controller 1 is driving debug information. 11 DDR SDRAM controller 2 is driving debug information.

Table 23-8. PORDBGMSR Field Descriptions (continued)

Bits	Name	Description
7	D1_DBG	DDR controller 1 debug configuration (See Section 4.4.3.22, “DDR Debug Configuration.”) 0 SourceID and data valid information is being driven on ECC pins of DDR SDRAM interface 1 1 Normal mode. ECC information is being driven on ECC pins of DDR SDRAM interface 1
8	—	Reserved
9	D2_DBG	DDR controller 2 debug configuration (See Section 4.4.3.22, “DDR Debug Configuration.”) 0 SourceID and data valid information is being driven on ECC pins of DDR SDRAM interface 2 1 Normal mode. ECC information is being driven on ECC pins of DDR SDRAM interface 2
10–31	—	Reserved

23.4.1.6 POR Device Status Register 2 (PORDEVSR2)

Shown in Figure 23-6,the PORDEVSR2 reports POR settings as described in Section 4.4.3.26, “SGMII SerDes Enable,” and Section 4.4.3.25, “SerDes1 Enable.”

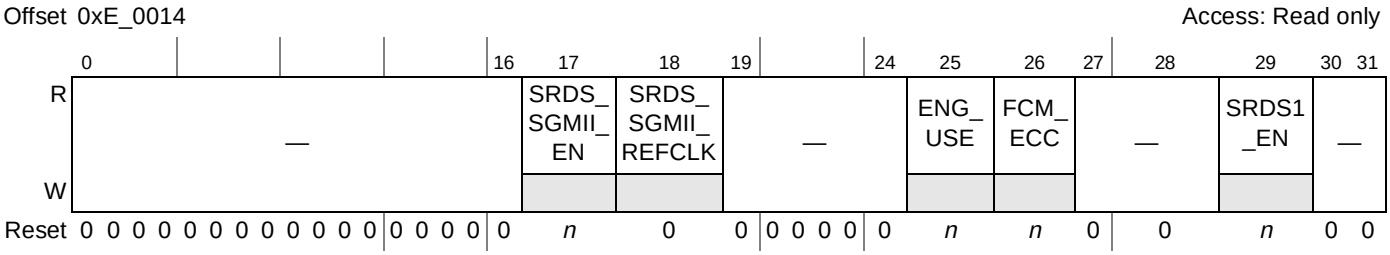


Figure 23-6. POR Device Status Register 2 (PORDEVSR2)

Table 23-9 describes the bit settings of PORDEVSR2.

Table 23-9. PORDEVSR2 Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	SRDS_SGMII_EN	Enable SGMII SerDes block. (See Section 4.4.3.26, “SGMII SerDes Enable.”) 0 SerDes interface is disabled. 1 SerDes interface is enabled.
18	SRDS_SGMII_REFCLK	SGMII SerDes reference clock configuration. (See Section 4.4.3.12, “SGMII SerDes Reference Clock Configuration.”) 0 125 MHz reference clock is expected for SGMII Serdes block. 1 100 MHz reference clock is expected for SGMII Serdes block.
19–24	—	Reserved
25	ENG_USE	Reserved for engineering use.
26	FCM_ECC	eLBC FCM ECC configuration. (Driven by POR reset signal: MSRCID[0]/ cfg_fcm_ecc) (See Section 4.4.3.24, “eLBC FCM ECC Configuration.”) 0 BRn[DECC} = 00. Data error checking is disabled; no ECC generation for FCM. 1 BRn[DECC} = 01. Data error checking is enabled, but ECC generation is disabled for FCM on full-page transfers.

Table 23-9. PORDEVSR2 Field Descriptions (continued)

Bits	Name	Description
27–28	—	Reserved
29	SRDS1_EN	Enable SerDes 1 block. (See Section 4.4.3.25, “SerDes1 Enable.”) 0 SerDes interface is disabled. 1 SerDes interface is enabled.
30–31	—	Reserved

23.4.1.7 General-Purpose POR Configuration Register (GPPORCR)

GPPORCR stores the value sampled from the local bus address/data signals, LAD[0:31], during POR, as described in Section 4.4.3.23, “General-Purpose POR Configuration.” 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.

GPPORCR is shown in Figure 23-7.

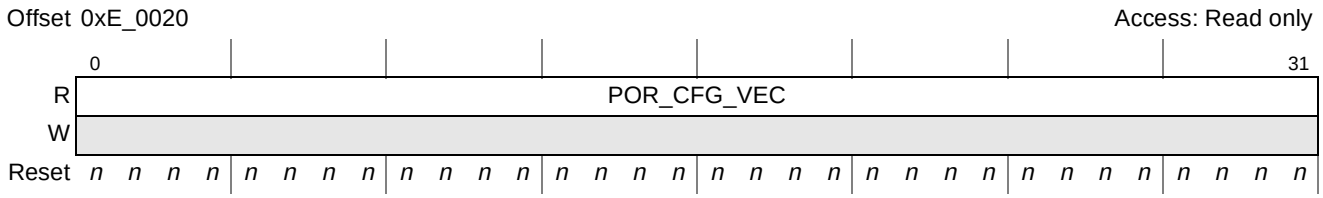


Figure 23-7. POR Configuration Register (GPPORCR)

Table 23-10 describes the bit settings of GPPORCR.

Table 23-10. GPPORCR Field Descriptions

Bits	Name	Description
0–31	POR_CFG_VEC	General-purpose POR configuration vector sampled from local bus address/data signals at the negation of $\overline{\text{HRESET}}$. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

23.4.1.8 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 23-8, PMUXCR contains bits that enable DMA2 channel 1 which exist as alternate functions on local bus chip select pins $\overline{\text{LCS}}$[5:7]. Specifically, DMA2 channel 1 request, acknowledge, and done signals comprise the secondary functions for the local bus chip select signals.

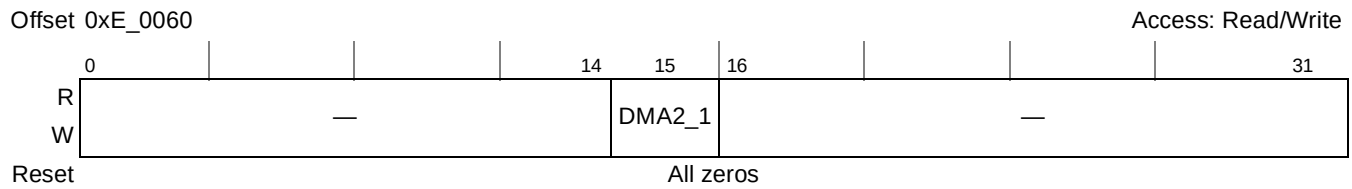


Figure 23-8. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 23-11 describes the bit settings of PMUXCR.

Table 23-11. PMUXCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	DMA2_1	Enables DMA2 channel 1 signals. 0 DMA2 channel 1 is not exposed to pins; the pins retain their primary function as local bus chip selects 1 DMA2 channel 1 is exposed to pins as follows: $\overline{\text{LCS5}}$ functions as $\overline{\text{DMA2\_DREQ1}}$ $\overline{\text{LCS6}}$ functions as $\overline{\text{DMA2\_DACK1}}$ $\overline{\text{LCS7}}$ functions as $\overline{\text{DMA2\_DDONE1}}$
16–31	—	Reserved

23.4.1.9 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 23-9, contains disable bits for various functional blocks.

Offset 0xE_0070 Access: Mixed

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—		PCIE1	—	LBC	PCIE2	PCIE3	SEC	PME	TLU1	TLU2	—	SRIO	—	D2	D1
Reset	0	0	n	0	0	n	n	0	0	0	0	0	n	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	E500_0	TB0	E500_1	TB1	—	DMA1	DMA2	—	TSEC1	TSEC2	TSEC3	TSEC4	FEC	I2C	DUART	—
Reset	0	0	0	0	0	0	0	0	0	0	n	n	n	0	0	0

Figure 23-9. Device Disable Register (DEVDISR)

Note that bits with a reset value of *n* depend on the state of POR configuration signals at reset.

All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. Section 23.5.1.4, "Shutting Down Unused Blocks," has more information on the use of DEVDISR. Table 23-12 describes DEVDISR fields.

Table 23-12. DEVDISR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	PCIE1	PCI Express 1 controller disable 0 PCI Express 1 controller enable 1 PCI Express 1 controller disable
3	—	Reserved
4	LBC	Local bus controller disable 0 Local bus controller enable 1 Local bus controller disable

Table 23-12. DEVDISR Field Descriptions (continued)

Bits	Name	Description
5	PCIE2	PCI Express 2 controller disable 0 PCI Express 2 controller enable 1 PCI Express 2 controller disable
6	PCIE3	PCI Express 3 controller disable 0 PCI Express 3 controller enable 1 PCI Express 3 controller disable
7	SEC	Security disable 0 Security enable 1 Security disable
8	PME	PME Deflate disable 0 PME Deflate enable 1 PME Deflate disable
9	TLU1	Table Look 1 disable 0 TLU 1 enable 1 TLU 1 disable
10	TLU2	Table Look 2 disable 0 TLU 2 enable 1 TLU 2 disable
11	—	Reserved
12	SRIO	Serial RapidIO controller disable 0 Serial RapidIO and RapidIO message units controllers enable 1 Serial RapidIO and RapidIO message units controllers disable
13	—	Reserved
14	D2	DDR controller 2 SDRAM controller disable 0 DDR 2 SDRAM controller enable 1 DDR 2 SDRAM controller disable
15	D1	DDR controller 1 SDRAM controller disable 0 DDR 1 SDRAM controller enable 1 DDR 1 SDRAM controller disable
16	E500_CORE0	e500 core 0 disable 0 e500 0 core enable 1 e500 0 core disable. Places the core in the core_stopped state in which it does not respond to interrupts. Equivalent to nap mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core including the timer facilities. For more information, see Section 23.5.1.4, “Shutting Down Unused Blocks.”
17	TB0	Time base (timer facilities) of the e500 core 0 disable 0 Timer facilities enabled 1 Timer facilities disabled.
18	E500_CORE1	e500 core 1 disable 0 e500 1 core enable 1 e500 1 core disable. Places the core in the core_stopped state in which it does not respond to interrupts. Equivalent to nap mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core including the timer facilities. For more information, see Section 23.5.1.4, “Shutting Down Unused Blocks.”

Table 23-12. DEVDISR Field Descriptions (continued)

Bits	Name	Description
19	TB1	Time base (timer facilities) of the e500 core 1 disable 0 Timer facilities enabled 1 Timer facilities disabled.
20	—	Reserved
21	DMA1	DMA 1 controller disabled 0 DMA 1 controller enabled 1 DMA 1 controller disabled
22	DMA2	DMA 2 controller disabled 0 DMA 2 controller enabled 1 DMA 2 controller disabled
23	—	Reserved
24	TSEC1	Three-speed Ethernet controller 1 disable 0 eTSEC1 enabled 1 eTSEC1 disabled
25	TSEC2	Three-speed Ethernet controller 2 disable 0 eTSEC2 enabled 1 eTSEC2 disabled.
26	TSEC3	Three-speed Ethernet controller 3 disable 0 eTSEC1 enabled 1 eTSEC1 disabled
27	TSEC4	Three-speed Ethernet controller 4 disable 0 eTSEC1 enabled 1 eTSEC1 disabled
28	FEC	Fast Ethernet controller disable 0 FEC enabled 1 FEC disabled
29	I2C	I^2C controllers disabled 0 I^2C controllers enabled 1 I^2C controllers disabled
30	DUART	Dual UART controller disabled 0 DUART enabled 1 DUART disabled
31	—	Reserved

23.4.1.10 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 23-10, POWMGTCSR contains bits for placing the MPC8572E into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 23.5.1.8.2, “Interrupts and Power Management Controlled by POWMGTCSR,” for more information.

Offset 0xE_0080 Access: Mixed

	0	1	2	3	4–11	12	13	14	15
R	CORE0_IRQ_MSK	CORE0_CI_MSK	CORE1_IRQ_MSK	CORE1_CI_MSK	—	CORE0_DOZ	CORE1_DOZ	SLP	—
W									
Reset	All zeros								

	16–25	26	27	28	29	30	31
R	—	CORE1_DOZING	CORE1_NAPPING	CORE0_DOZING	CORE0_NAPPING	SLPING	QSCED
W							
Reset	All zeros						

Figure 23-10. Power Management Control & Status Register (POWMGTCSR)

Table 23-13 describes the bit settings of POWMGTCSR.

Table 23-13. POWMGTCSR Field Descriptions

Bits	Name	Description
0	CORE0_IRQ_MSK	Core 0 interrupt input mask 0 Interrupts cause the device to wake up from a low-power state. 1 Interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of an interrupt request.
1	CORE0_CI_MSK	Core 0 critical interrupt input mask 0 Critical interrupts cause the device to wake up from a low power state. 1 Critical interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of a critical interrupt.
2	CORE1_IRQ_MSK	Core 1 interrupt input mask 0 Interrupts cause the device to wake up from a low-power state. 1 Interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of an interrupt request.
3	CORE1_CI_MSK	Core 1 critical interrupt input mask 0 Critical interrupts cause the device to wake up from a low power state. 1 Critical interrupts are masked as a wake-up condition. The device remains in a low-power state despite the presence of a critical interrupt.
4–11	—	Reserved
12	CORE0_DOZ	Core 0 doze mode. 0 No request to put device in doze mode. Note that this bit is automatically cleared on MCP, UDE, SRESET, *core_tbint* (from the core) and also $\overline{int}$ and $\overline{cint}$ if not masked. 1 Device is to be placed in doze mode. Instruction fetching is halted in the e500 core 0. Note that this bit is logically ORed with HID0[DOZE].
13	CORE1_DOZ	Core 1 doze mode. 0 No request to put device in doze mode. Note that this bit is automatically cleared on MCP, UDE, SRESET, *core_tbint* (from the core) and also $\overline{int}$ and $\overline{cint}$ if not masked. 1 Device is to be placed in doze mode. Instruction fetching is halted in the e500 core 1. Note that this bit is logically ORed with HID0[DOZE].

Table 23-13. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
14	SLP	Sleep mode 0 No request to put device in sleep mode. 1 Device is to be placed in sleep mode. Instruction fetching is halted, snooping of L1 caches is disabled, and most functional blocks are shut down in both the e500 cores and the system logic. Note that Sleep mode must not be requested while either core is in boot hold-off mode.
15–25	—	Reserved
26	CORE1 DOZING	Core 1 doze status 0 Device is not in doze mode. 1 The MPC8572E is in doze mode because POWMGTCSR[DOZ] is set or because HID0[DOZE] and MSR[WE] (in the e500 core 1) are set. The core has halted instruction fetching, but all other functional blocks in the core and device are running.
27	CORE1_ NAPPING	Core 1 nap status 0 Device is not in nap mode. 1 The MPC8572E is in nap mode because HID0[NAP] and MSR[WE] are set. The core has halted instruction fetching, snooping of the L1 caches is disabled, and all of the core's functional units except the timer facilities are shut down. All functional blocks in the device are running.
28	CORE0 DOZING	Core 0 doze status 0 Device is not in doze mode. 1 The MPC8572E is in doze mode because POWMGTCSR[DOZ] is set or because HID0[DOZE] and MSR[WE] (in the e500 core 0) are set. The core has halted instruction fetching, but all other functional blocks in the core and device are running.
29	CORE0_ NAPPING	Core 0 nap status 0 Device is not in nap mode. 1 The MPC8572E is in nap mode because HID0[NAP] and MSR[WE] are set. The core has halted instruction fetching, snooping of the L1 caches is disabled, and all of the core's functional units except the timer facilities are shut down. All functional blocks in the device are running.
30	SLPING	Sleep status 0 Device is not attempting to reach sleep mode. 1 The device is attempting to SLEEP because POWMGTCSR[SLP] is set or because HID0[SLEEP] and MSR[WE] (in the e500 core) are set. Most functional blocks in the core and device are shut down or are attempting to shut down.
31	—	Reserved. Should be cleared.

23.4.1.11 Machine Check Summary Register (MCPSUMR)

Shown in Figure 23-11, MCPSUMR contains bits summarizing some of the sources of a pending machine check interrupt. All MCPSUMR bits function as write-1-to-clear.

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

Note that other conditions can cause a machine check condition not summarized in MCPSUMR. For example, uncorrectable read errors cause the assertion of *core_fault_in*, which may directly cause a machine check (if HID1[RFXE] = 1). If RFXE = 0, the assertion of *core_fault_in* does not directly cause

a machine check interrupt, but must be handled by the block that generated the error. For more information about RFXE, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Offset 0xE_0090 Access: w1c

	0–23	24	25	26	27		29	30	31
R	—	MCP_RSP_CKSTP_P0	MPC_RSP_CKSTP_P1	WRS1	MCP1_IN	—	WRS0	SRESET	MCP0_IN
W		w1c	w1c	w1c	w1c		w1c	w1c	w1c
Reset	All zeros								

Figure 23-11. Machine Check Summary Register (MCPSUMR)

Table 23-14 describes the bit settings of MCPSUMR.

Table 23-14. MCPSUMR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	MCP_RSP_CKSTP_P0	Machine check to core 0 occurred in response to checkstop from core 1 0 Machine check exception was not caused by checkstop of core 1 1 Machine check exception was caused by checkstop of core 1
25	MCP_RSP_CKSTP_P1	Machine check to core 1 occurred in response to checkstop from core 0 0 Machine check exception was not caused by checkstop of core 0 1 Machine check exception was caused by checkstop of core 0
26	WRS1	Core 1 watchdog timer expiration caused machine check 0 Machine check exception was not caused by watchdog timer. 1 Machine check was caused by a soft reset condition from the e500_1 watchdog timer as configured in the core1’s TSR. Specifically, TSR[WRS] = 01 and a watchdog reset condition occurred. (See Section 6.6.1, “Timer Control Register (TCR).”)
27	MCP1_IN	$\overline{\text{MCP1}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{MCP1}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP1}}$ input signal.
28	—	Reserved
29	WRS0	Core 0 watchdog timer expiration caused machine check 0 Machine check exception was not caused by watchdog timer. 1 Machine check was caused by a soft reset condition from the e500_0 watchdog timer as configured in the core0’s TSR. Specifically, TSR[WRS] = 01 and a watchdog reset condition occurred. (See Section 6.6.1, “Timer Control Register (TCR).”)
30	SRESET	$\overline{\text{SRESET}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{SRESET}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{SRESET}}$ input signal.
31	MCP0_IN	$\overline{\text{MCP0}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{MCP0}}$ assertion. 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP0}}$ input signal.

23.4.1.12 Reset Request Status and Control Register (RSTRSCR)

Shown in Figure 23-14, the RSTRSCR contains status bits to record the reasons for $\overline{\text{HRESET\_REQ}}$ assertion as well as a mask bit to exclude RapidIO packets from generating such a reset request.

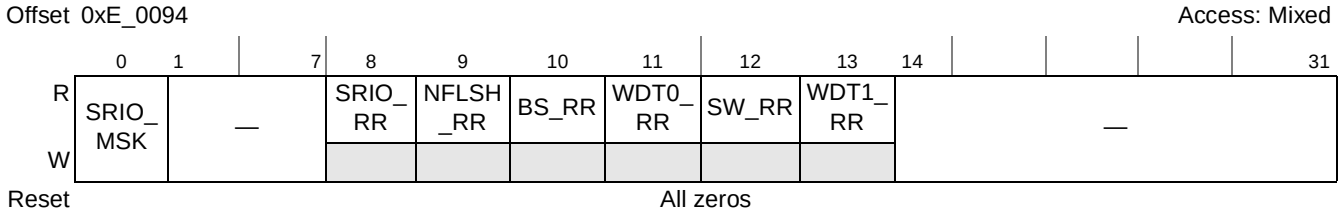


Figure 23-12. Reset Request Status and Control Register (RSTRSCR)

Table 23-17 describes the bit settings of RSTRSCR.

Table 23-15. RSTRSCR Field Descriptions

Bits	Name	Description
0	SRIO_MSK	Serial RapidIO (SRIO) reset request mask
1–7	—	Reserved
8	SRIO_RR	Serial RapidIO (SRIO) reset requested
9	NFLSH_RR	NAND Flash ECC error during boot reset requested
10	BS_RR	Boot sequence error during pre-boot initialization caused reset requested
11	WDT0_RR	Core0's watchdog timer expiration caused a device level reset request. Occurs when TSR[WRS] = 10 for core 0 and a watchdog expiration is reached. (See Section 6.6.1, “Timer Control Register (TCR).”)
12	SW_RR	Software settable reset requested (See Section 23.4.1.17, “Reset Control Register (RSTCR).”)
13	WDT1_RR	Core1's watchdog timer expiration caused a device level reset request. Occurs when TSR[WRS] = 10 for core 1and a watchdog expiration is reached. (See Section 6.6.1, “Timer Control Register (TCR).”)
14–31	—	Reserved

23.4.1.13 Exception Reset Control Register (ECTRSTCR)

Shown in Figure 23-14, the ECTRSTCR contains control bits for the exception reset of core 0 and core 1 in response to checkstop.

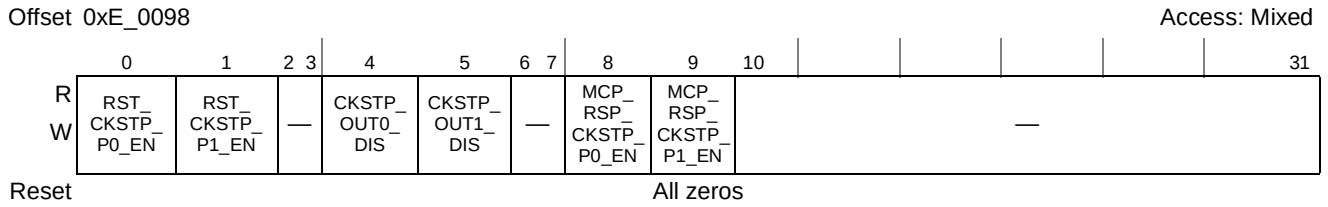


Figure 23-13. Checkstop Status and Control Register (ECTRSTCR)

Table 23-17 describes the bit settings of ECTRSTCR.

Table 23-16. ECTRSTCR Field Descriptions

Bits	Name	Description
0	RST_CKSTP_P0_EN	Enable automatic reset of core 0 in response to checkstop
1	RST_CKSTP_P1_EN	Enable automatic reset of core 1 in response to checkstop
2–3	—	Reserved
4	CKSTP_OUT0_DIS	Disable assertion of $\overline{\text{CKSTP\_OUT0}}$ pin
5	CKSTP_OUT1_DIS	Disable assertion of $\overline{\text{CKSTP\_OUT1}}$ pin
6–7	—	Reserved
8	MCP_RSP_CKSTP_P0_EN	Enable machine check to core 0 in response to check stop from core 1
9	MCP_RSP_CKSTP_P1_EN	Enable machine check to core 1 in response to check stop from core 0
10–31	—	Reserved

23.4.1.14 Automatic Reset Status Register (AUTORSTSR)

Shown in Figure 23-14, the AUTORSTSR contains the automatic reset status bits for core 0 and core 1.

Offset 0xE_009C Access: w1c

	0	1	2 3	4	5	6 7	8	9	10 11	12	13	14 15	16	17	18 31
R	RST_CKSTP_P0	RST_CKSTP_P1	—	RST_WRS_P0	RST_WRS_P1	—	RST_MPIC_P0	RST_MPIC_P1	—	RST_CORE_P0	RST_CORE_P1	—	READY_P0	READY_P1	—
W	w1c	w1c		w1c	w1c										
Reset	All zeros														

Figure 23-14. Checkstop Status and Control Register (AUTORSTSR)

Table 23-17 describes the bit settings of AUTORSTSR.

Table 23-17. AUTORSTSR Field Descriptions

Bits	Name	Description
0	RST_CKSTP_P0	Core 0 was reset in response to check stop 0 no reset 1 reset occurred.
1	RST_CKSTP_P1	Core 1 was reset in response to check stop 0 no reset 1 reset occurred.
2–3	—	Reserved
4	RST_WRS_P0	Core 0 was reset in response to its watchdog timer expiration. (See Section 6.6.1, “Timer Control Register (TCR).”) 0 no reset 1 reset occurred

Table 23-17. AUTORSTSR Field Descriptions (continued)

Bits	Name	Description
5	RST_WRS_P1	Core 1 was reset in response to its watchdog timer expiration. (See Section 6.6.1, "Timer Control Register (TCR).") 0 no reset 1 reset occurred
6–7	—	Reserved
8	RST_MPIC_P0	Core 0 was reset in response to MPIC reset request 0 no reset 1 reset occurred
9	RST_MPIC_P1	Core 1 was reset in response to MPIC reset request 0 no reset 1 reset occurred
10–11	—	Reserved
12	RST_CORE_P0	Core 0 was reset in response to internal core request to reset itself by setting bit DBCR0[RST] register 0 no reset 1 reset occurred
13	RST_CORE_P1	Core 1 was reset in response to internal core request to reset itself by setting bit DBCR0[RST] register 0 no reset 1 reset occurred
14–15	—	Reserved
16	READY_P0	Core 0 ready pin. This bit reflects what is driven on the READY_P0 external signal. 0 Core 0 not ready 1 Core 0 ready
17	READY_P1	Core 1 ready pin. This bit reflects what is driven on the READY_P1 external signal. 0 Core 1 not ready 1 Core 1 ready
18–31	—	Reserved

23.4.1.15 Processor Version Register (PVR)

Shown in Figure 23-15, the PVR contains the e500 processor version number. It is a memory-mapped copy of the PVR in the e500 core (and is therefore accessible to external devices). Section 5.2, "e500 Processor and System Version Numbers," lists the complete values for the MPC8572E.

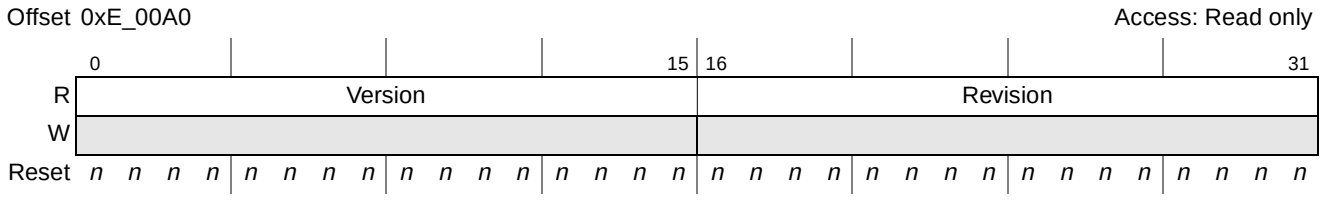


Figure 23-15. Processor Version Register (PVR)

Table 23-18 describes the fields of PVR.

Table 23-18. PVR Field Descriptions

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)
48–63	Revision	A 16-bit number that distinguishes between implementations of the version. Different revision numbers indicate minor differences between processors having the same version number, such as clock rate and engineering change level. (See Section 5.2, “e500 Processor and System Version Numbers,” for specific values.)

23.4.1.16 System Version Register (SVR)

Shown in Figure 23-16, the SVR contains the system version number for the MPC8572E implementation. This value can also be read though the SVR SPR of the e500 core. See Section 6.5.4, “System Version Register (SVR).” Section 5.2, “e500 Processor and System Version Numbers,” lists the complete values for the MPC8572E.

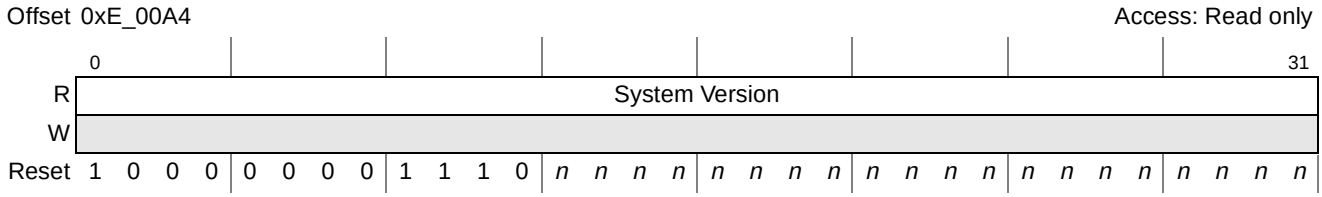


Figure 23-16. System Version Register (SVR)

Table 23-19 describes the fields of SVR.

Table 23-19. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version numbers for the MPC8572E system logic 0x80E8_0011 for MPC8572E with security 0x80E0_0011 for MPC8572 without security

23.4.1.17 Reset Control Register (RSTCR)

Shown in Figure 23-17, the RSTCR contains bits that allow for software to force assertion of $\overline{\text{HRESET\_REQ}}$. External hardware may then decide to issue $\overline{\text{HRESET}}$ to the device.

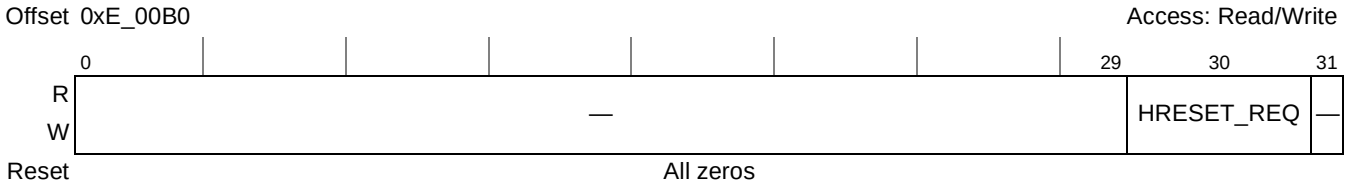


Figure 23-17. Reset Control Register (RSTCR)

Table 23-20 describes the bit settings of RSTCR.

Table 23-20. RSTCR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	HRESET_REQ	Hardware reset request 0 No reset request initiated. 1 Hardware reset request initiated by software.
31	—	Reserved

23.4.1.18 LBC Voltage Select Control Register (LBCVSELCR)

Shown in Figure 23-18, the LBCVSELR contains local bus voltage control bits.

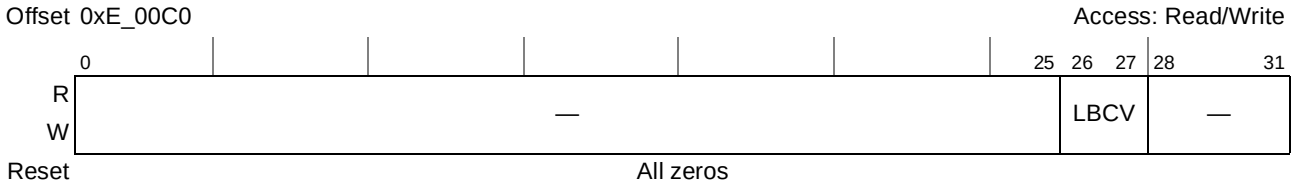


Figure 23-18. LBC Voltage Select Control Register (LBCVSELCR)

Table 23-21 describes the bit settings of LBCVSELCR.

Table 23-21. LBCVSELCR Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26–27	LBCV	Selects the I/O voltage for the local bus 00 (default) 3.3V 01 2.5V 10 1.8V 11 reserved
28–31	—	Reserved

23.4.1.19 DDR Clock Disable Register (DDRCLKDR)

Shown in Figure 23-19, the DDRCLKDR contains bits that allow disabling the clocks of the DDR SDRAM controller.

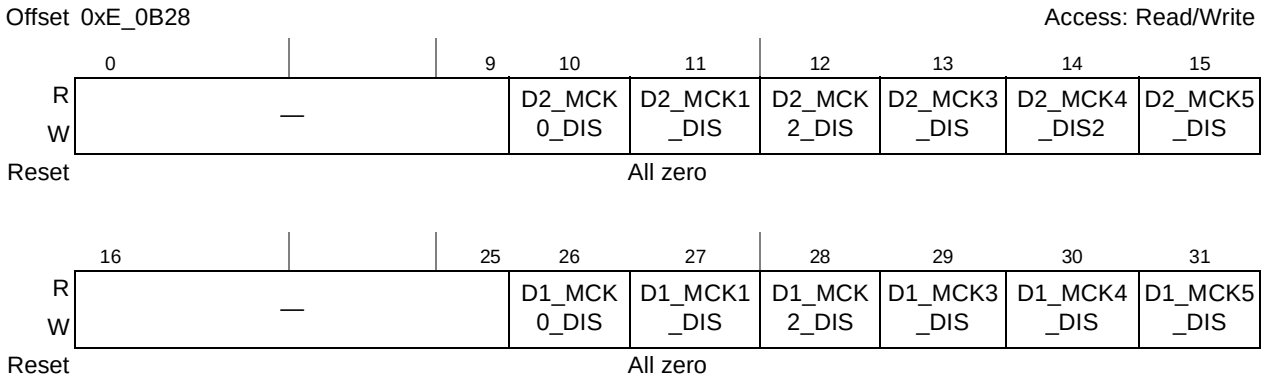


Figure 23-19. DDR Clock Disable Register (DDRCLKDR)

Table 23-22 describes the bit settings of DDRCLKDR.

Table 23-22. DDRCLKDR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10	D2_MCK0_DIS	DDR controller 2 clock 0 disable 0 MCK0 is enabled. 1 MCK0 is disabled.
11	D2_MCK1_DIS	DDR controller 2 clock 1 disable 0 MCK1 is enabled. 1 MCK1 is disabled.
12	D2_MCK2_DIS	DDR controller 2 clock 2 disable 0 MCK2 is enabled. 1 MCK2 is disabled.
13	D2_MCK3_DIS	DDR controller 2 clock 3 disable 0 MCK3 is enabled. 1 MCK3 is disabled.
14	D2_MCK4_DIS	DDR controller 2 clock 4 disable 0 MCK4 is enabled. 1 MCK4 is disabled.
15	D2_MCK5_DIS	DDR controller 2 clock 5 disable 0 MCK5 is enabled. 1 MCK5 is disabled.
16–25	—	Reserved
26	D1_MCK0_DIS	DDR controller 1 clock 0 disable 0 MCK0 is enabled. 1 MCK0 is disabled.
27	D1_MCK1_DIS	DDR controller 1 clock 1 disable 0 MCK1 is enabled. 1 MCK1 is disabled.
28	D1_MCK2_DIS	DDR controller 1 clock 2 disable 0 MCK2 is enabled. 1 MCK2 is disabled.

Table 23-22. DDRCLKDR Field Descriptions (continued)

Bits	Name	Description
29	D1_MCK3_DIS	DDR controller 1 clock 3 disable 0 MCK3 is enabled. 1 MCK3 is disabled.
30	D1_MCK4_DIS	DDR controller 1 clock 4 disable 0 MCK4 is enabled. 1 MCK4 is disabled.
31	D1_MCK5_DIS	DDR controller 1 clock 5 disable 0 MCK5 is enabled. 1 MCK5 is disabled.

23.4.1.20 Clock Out Control Register (CLKOCR)

Shown in Figure 23-20, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

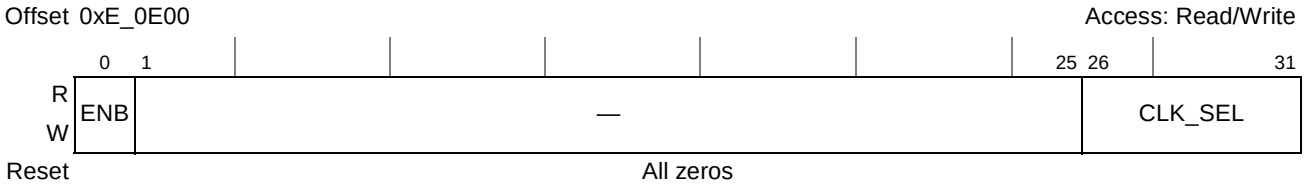


Figure 23-20. Clock Out Control Register (CLKOCR)

Table 23-23 describes the bit settings of CLKOCR.

Table 23-23. CLKOCR Field Descriptions

Bits	Name	Description
0	ENB	Clock out enable 0 CLK_OUT signal is three-stated 1 CLK_OUT signal is driven according to CLKOCR[CLK_SEL]
1–25	—	Reserved

Table 23-23. CLKOCR Field Descriptions (continued)

Bits	Name	Description
26–31	CLK_SEL	Clock out select 000000 CCB (platform) clock 000001 CCB (platform) clock divided by 2 000010 SYSCLK (echoes SYSCLK input) 000011 SYSCLK divided by 2 (demonstrates platform PLL lock) 000100 Reserved 000101 Reserved 000110 Reserved 000111 Reserved 001000 Reserved 001001 Reserved 001010 Reserved 001011 Reserved 001100 Reserved 001101 Reserved 001110 Reserved 001111 Reserved 01xx0x Reserved 01xx1x Reserved 10x000 Reserved 10x001 Reserved 10x010 Reserved 10x011 Reserved 10x100 Reserved 10x101 Reserved 10x110 Reserved 10x111 Logic 0 11x000 Reserved 11x001 Reserved 11x010 Reserved 11x011 Reserved 11x100 Reserved 11x101 Reserved 11x110 Reserved 11x111 Reserved

23.4.1.21 SerDes1 Control Register (SRDS1CR)

Shown in Figure 23-21, the SRDS1CR contains the control bits used for adjusting the transmit equalization of the SerDes 1 signals. Note that reserved fields with non-zero reset values are for internal use only and their values should be preserved.

Offset 0xE_0F04 Access: Read/Write

	0												15			
R W	—															
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

	16			19	20			23	24				31			
R W	XMITEQAD				XMITEQEH				—							
Reset	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0

Figure 23-21. SerDes1 Control Register (SRDS1CR)

Table 23-24 describes the bit settings of SRDS1CR.

Table 23-24. SRDS1CR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	XMITEQAD	Transmit equalization selection bus for lanes a-d (SD1_RX/TX[0:3]) If PCI Express is enabled: Default value = 4'b0100 MSB (bit 16) is amplitude select: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 17:19) are equalization amplitude: 000 No equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude 111 reserved
20–23	XMITEQEH	Transmit equalization selection bus for lanes e-h (SD1_RX/TX[4:7]) If SRIO is enabled: Default value = 4'b0011 If PCI Express is enabled: Default value = 4'b0100 MSB (bit 20) is amplitude select: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 21:23) are equalization amplitude: 000 No equalization 001 1.09x relative amplitude 010 1.2x relative amplitude 011 1.33x relative amplitude 100 1.5x relative amplitude 101 1.71x relative amplitude 110 2.0x relative amplitude 111 reserved
24–31	—	Reserved

23.4.1.22 SerDes2 Control Register (SRDS2CR)

Shown in Figure 23-22, the SRDS2CR contains the control bits used for adjusting the transmit equalization of the SerDes 2 signals. Note that reserved fields with non-zero reset values are for internal use only and their values should be preserved.

Offset 0xE_0F10 Access: Read/Write

	0	1	2	3	5	6	7	15
R W	—		LSTSAB		—	LSTSEF		—
Reset	0	0	0	1	0	0	0	1 0 0 0 0 0 0 0 0

	16	19	20	23	24	31
R W	XMITEQAB		XMITEQEF		—	
Reset	0	1 0 0	0	1 0 0	0	0 0 0 1 1 0 0 0

Figure 23-22. SerDes2 Control Register (SRDS2CR)

Table 23-25 describes the bit settings of SRDS2CR.

Table 23-25. SRDS2CR Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	LSTSAB	Loss of Signal Threshold Level Selection for lanes A & B of SerDes 2 in SGMII mode. This bit, when set, increases the SerDes 2 lanes A & B receiver's Vrx_diffp-p level on electrical idle detection. Refer to the VIos parameter (Loss of signal threshold) in the DC Receiver Electrical Characteristics of section "SGMII DC Electrical Characteristics" in *MPC8572E PowerQUICC III Integrated Processor Hardware Specifications*, for the details of how the LSTS bit affects the loss of signal threshold level.
3–5	—	Reserved
6	LSTSEF	Loss of Signal Threshold Level Selection for lanes E & F of SerDes 2 in SGMII mode. This bit, when set, increases the SerDes 2 lanes E & F receiver's Vrx_diffp-p level on electrical idle detection. Refer to the VIos parameter (Loss of signal threshold) in the DC Receiver Electrical Characteristics of section "SGMII DC Electrical Characteristics" in *MPC8572E PowerQUICC III Integrated Processor Hardware Specifications*, for the details of how the LSTS bit affects the loss of signal threshold level.
7–15	—	Reserved
16–19	XMITEQAB	Transmit equalization selection bus for SerDes 2 lanes A & B (SD2_RX/TX[0:1]) Default value = 4'b0100 (SGMII) MSB (bit 16) is the differential peak-to-peak amplitude selection: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 17:19) are the equalization amplitude selection: 000 = No equalization 001 = 1.09x relative amplitude 010 = 1.2x relative amplitude 011 = 1.33x relative amplitude 100 = 1.5x relative amplitude 101 = 1.71x relative amplitude 110 = 2.0x relatvie amplitude 111 = reserved

Table 23-25. SRDS2CR Field Descriptions (continued)

Bits	Name	Description
20–23	XMITEQEF	Transmit equalization selection bus for SerDes 2 lanes E & F (SD2_RX/TX[2:3]) Default value = 4'b0100 (SGMII) MSB (bit 20) is the differential peak-to-peak amplitude selection: 0 = Vdd-diff-pk=pk 1 = 5/6 Vdd-diff-pk=pk LSBs (bits 21:23) are the equalization amplitude selection: 000 = No equalization 001 = 1.09x rel.amplitude 010 = 1.2x rel.amplitude 011 = 1.33x rel.amplitude 100 = 1.5x rel.amplitude 101 = 1.71x rel.amplitude 110 = 2.0x rel.amplitude 111 = reserved
24–31	—	Reserved

23.5 Functional Description

This section describes the global utilities from a functional perspective.

23.5.1 Power Management

The MPC8572E has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to individual blocks when they are not needed through a memory-mapped register (DEVDISR). Additionally, software running on the e500 cores can access the cores's SPRs to put the device into doze or nap power down state. Finally, software can access a memory-mapped register (POWMGTCR) in the global utilities block to put the device in the doze or sleep states.

Note that the software that writes to either DEVDISR or POWMGTCR can be running either on the e500 cores or on an external master that can write to the MPC8572E memory-mapped registers through the PCI Express interfaces.

These features are described in further detail in this section.

23.5.1.1 Relationship Between Both Cores and Device Power Management States

The MPC8572E has three low-power states: doze, nap, and sleep. The mapping of both cores and device power management states is shown in Figure 23-23 showing state transitions from the perspective of the e500 cores.

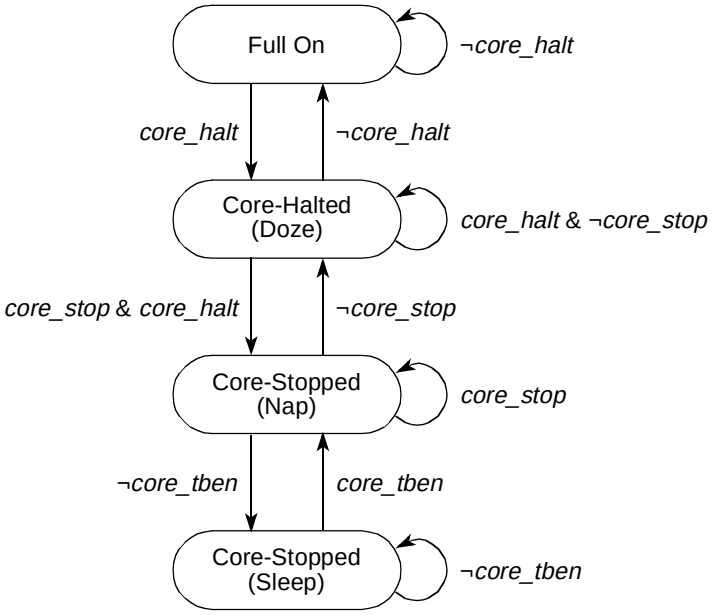


Figure 23-23. e500 Cores Power Management State Diagram

For each operating state represented in the diagram, the cores's state is listed first, with the corresponding state of the MPC8572E shown beneath it in parenthesis. Note that there are many other variables that control the state transitions between MPC8572E power management states. These additional variables are described in more detail in Section 23.5.1.7, "Power-Down Sequence Coordination."

Table 23-26 lists basic characteristics of the low-power modes and the full on mode.

Table 23-26. MPC8572E Power Management Modes—Basic Description

Mode	Description	Core Responds To		Signal States	
		Snoop	Interrupts	READY_P0 or READY_P1	ASLEEP
Full On	All units operating normally.	Yes	Yes	Asserted	Negated
Doze0/1	Core 0 or stops dispatching new instructions (core 0 or 1 is halted)	Yes	Yes	Negated	Negated
Nap0/1	Core 0 or 1 is stopped with clocks off except to time base Should flush data cache before entering	No	Yes	Negated	Negated
Sleep	Core is stopped with clocks off. Clocks powered down to all blocks (including core time base) except to the interrupt controller (PIC) unit	No	Yes	Negated	Asserted

23.5.1.2 $\overline{\text{CKSTP\_IN0/1}}$ is Not Power Management

$\overline{\text{CKSTP\_IN0/1}}$ are not described here because they are not considered power management signals, although asserting these do stop the cores and a stopped core is technically in a low-power mode. $\overline{\text{CKSTP\_IN0/1}}$ are described in Section 23.3.2, "Detailed Signal Descriptions."

23.5.1.3 Dynamic Power Management

Many blocks in the MPC8572E can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

23.5.1.4 Shutting Down Unused Blocks

As described in Section 23.4.1.9, "Device Disable Register (DEVDISR)," DEVDISR provides a way to shut down certain functional blocks within the MPC8572E when they are not needed in a particular system. DEVDISR can be written by the both e500 cores or by an external master. Powering down a block in this way turns off all clocks to that block.

DEVDISR was designed with the expectation that, once initialized by software, it would be modified only by a hard system reset ($\overline{\text{HRESET}}$). It is recommended that this register be written only during system initialization. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. (Setting DEVDISR[TB0/1] disables the cores's timer facilities, and setting DEVDISR[E500_0/1] places the cores in the core_stopped state in which they do not respond to interrupts.) The results of re-enabling previously disabled blocks (by clearing the corresponding DEVDISR field) without a hard reset are boundedly undefined.

NOTE

Functional blocks disabled using DEVDISR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

23.5.1.5 Software-Controlled Power-Down States

e500 software can place the device in doze or nap power-down states by writing to HID0 in the core. In addition, external masters can write to the memory-mapped POWMGTCR in the MPC8572E to cause the device to enter doze or sleep modes.

23.5.1.5.1 Doze Mode

In doze mode, the given e500 core suspends instruction execution, significantly reducing the power consumption of the cores. Snooping of the L1 data cache is still supported and thus the data in the data cache is kept coherent. Interrupts directed to the each core as described in Section 10.1.3, "Interrupts to the Processor Core," are monitored by the device and cause the MPC8572E to use the defined handshake mechanism to exit the cores from doze mode to allow the cores to recognize and process the interrupt; however, unless the interrupt subroutine turns off (or masks) the control bits that enabled doze mode (MSR[WE], and HID0[DOZE]), the device re-enters doze mode after the interrupt has been serviced. See Section 23.5.1.8, "Interrupts and Power Management," for more information.

The given e500 core's timer facilities are still enabled during doze mode, and core time base interrupts can be generated. All device logic external to the core remains fully operational in doze mode.

23.5.1.5.2 Nap Mode

In nap mode all clocks internal to the e500 core are turned off except for their timer facilities clock (the core time base). The L1 caches do not respond to snoops in nap mode, so if coherency with external I/O transactions is required, the L1 cache must be flushed before entering nap mode.

Similar to doze mode, interrupts occurring in nap mode cause the device to wake up the e500 core in order to service the interrupt. However, unless the interrupt service routine changes the control bits that caused the device to enter nap mode (MSR[WE], and HID0[NAP]), the core returns to nap mode after the interrupt is serviced. See Section 23.5.1.8, "Interrupts and Power Management," for more information.

All device logic external to the e500 core remains fully operational in nap mode.

23.5.1.5.3 Sleep Mode

In sleep mode, all clocks internal to both e500 cores are turned off, including the timer facilities clock. All I/O interfaces in the device logic are also shut down. Only the clocks to the MPC8572E PIC are still running so that an external interrupt can wake up the device.

After the core and I/O interfaces have shut down, ASLEEP is asserted and READY_P0 and READY_P1 are negated.

NOTE

Only external interrupts can wake the MPC8572E from sleep mode. Internal interrupt sources like the core interval timer or watchdog timer depend on an active clock for their operation and these are disabled in sleep mode.

23.5.1.6 Power Management Control Fields

The e500 cores provide the following fields to signal power management requests to the MPC8572E device logic.

- MSR[WE]—Used to qualify the values of HID0[DOZE0/1, NAP] in the generation of the internal *doze* and *nap* signals.
- HID0[DOZE*n*]—Signals the MPC8572E to initiate doze mode for the given core.
- HID0[NAP*n*]—Signals the MPC8572E to initiate nap mode for the given core.

These register fields and their functional relationship are shown in Figure 23-24. The *PowerPC e500 Core Reference Manual* has details on accessing these power management control bits.

An external master can also initiate power management requests by setting the applicable DOZ or SLP bits in the memory-mapped power management control and status register (POWMGTCSR). Because the core responds to snoops while dozing but not while napping, maintaining cache coherency requires significant preparation by the core before entering nap mode. For this reason only the core can initiate a nap during normal operation while other masters can initiate a doze.

23.5.1.7 Power-Down Sequence Coordination

To preserve cache coherency and otherwise avoid loss of system state, a core's transition to low-power modes is coordinated by a set of handshaking signals, shown in Figure 23-24, and protocols with all other MPC8572E functional blocks that respond to power-down requests. The mode-transition protocol is executed automatically under these conditions and is shown in Figure 23-23 and described in Table 23-27.

The column in Table 23-27 showing the global utilities block as initiating a low-power mode corresponds to the external masters that can write to the POWMGTCSR that resides in the global utilities block. For the MPC8572E, these are the PCI Express, and Serial RapidIO interfaces. However, note that the core can also write to POWMGTCSR and, in this case, can initiate power management through the global utilities block.

Table 23-27. Power Management Entry Protocol and Initiating Functional Units

Low-Power Mode	Entry Protocol	Initiating Functional Unit	
		Global Utilities	Core
Doze	1. Assert *core0/1_halt* input to core. 2. Wait for *core0/1_halted* handshake from core.	√	√
Nap	1. Follow doze protocol 2. Assert *core0/1_stop* input to core. 3. Wait for *core0/1_stopped* handshake from core.	—	√
Sleep	1. Follow doze protocol; send stop requests to rest of device. 2. Follow nap protocol. 3. Wait for all interfaces to acknowledge stop requests. 4. Assert ASLEEP, negate READY, power down all clocks except to PIC unit.	√	—

As shown in Figure 23-24, the e500 cores enter low-power modes only in response to the *core0/1_halt*, *core0/1_stop*, or *core0/1_tben* inputs from the MPC8572E's power management logic. These inputs may be prompted by the core (by setting the NAP0/1 or DOZE0/1 bits in the HID0 when enabled by setting MSR[WE]) or by an external master (by setting POWMGTCSR[DOZ0/1,SLP].

Figure 23-24 shows how all the clocking to the core timer facilities are disabled by clearing HID0[TB0/1EN]. When enabled, (HID0[TB0/1EN] = 1), the clock source is either the CCB clock divided by eight (the default) or a synchronized version of the RTC input. For more details, see Section 6.10.1, "Hardware Implementation-Dependent Register 0 (HID0)."

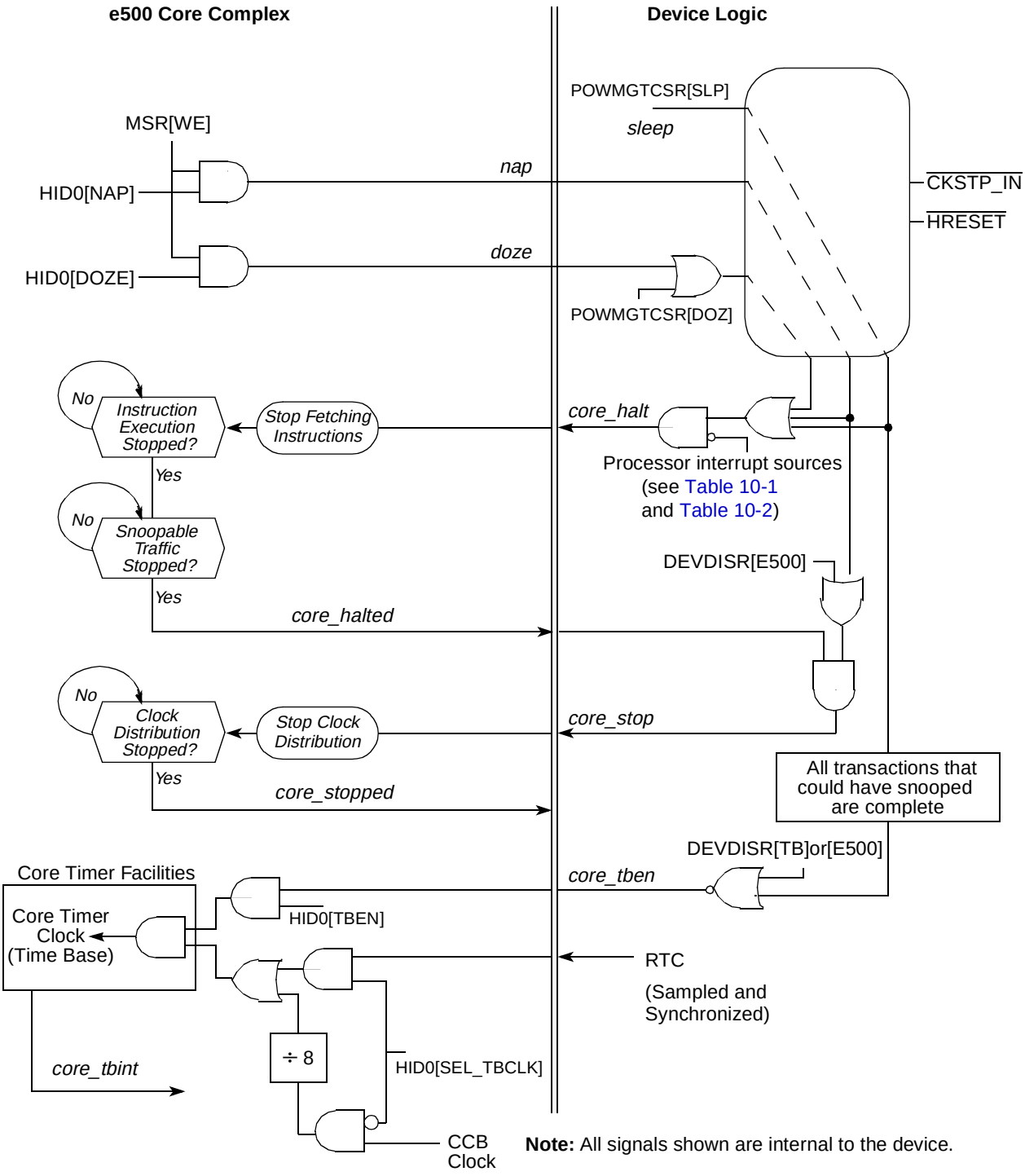


Figure 23-24. MPC8572E Power Management Handshaking Signals

23.5.1.8 Interrupts and Power Management

Whether low-power modes are automatically re-enabled after an interrupt is processed differs depending on whether the low power mode was entered due to a write to the core MSR[WE] bit or the low power mode was entered due to a write to POWMGTCSR.

23.5.1.8.1 Interrupts and Power Management Controlled by MSR[WE]

When an interrupt is asserted to the CPU, the core complex saves portions of the MSR to MCSRR1, CSRR1, or SRR1 (depending on the type of interrupt), and restores those values on return from the routine. MSR[WE], which gates the *doze* and *nap* power management outputs (internal device signals) from the core complex, is always among the bits saved and restored; hence these outputs negate to the MPC8572E power management logic when the interrupt begins processing in the core. They return to their previous state when the core executes an **rfi**, **rfci**, or **rfmci** instruction. Section 10.1.3, "Interrupts to the Processor Core," lists interrupts that cause the MPC8572E to wake up.

NOTE

Returning *doze* and *nap* signals to their original state when MSR[WE] is restored differs from how power management is implemented on earlier PowerPC devices where MSR[POW], which enables power-down requests, is cleared when the processor exits a low-power state and is not automatically restored, as it is in Book E implementations.

23.5.1.8.2 Interrupts and Power Management Controlled by POWMGTCSR

The IRQ_MSK and CI_MSK fields of the POWMGTCSR register prevent $\overline{int}$ interrupts or $\overline{cint}$ critical interrupts from waking the device from a low power state. This is true regardless of the method used to enter the low power state.

Any unmasked interrupt (not masked by the mask bits in the POWMGTCSR register) causes the POWMGTCSR[DOZ0/1,SLP] fields to be cleared when it occurs. When such an interrupt occurs, the device returns to the normal operating mode and does not automatically attempt to return to a low power state after the interrupt is handled.

Note that interrupts caused by the unconditional debug event ($\overline{UDE}$) and machine check ($\overline{MCP}$) signals are not masked by the IRQ_MSK and CI_MSK fields; therefore, when these signals assert, the POWMGTCSR[DOZ0/1,SLP] fields are cleared and the device will return to full power operation. See Section 23.4.1.10, "Power Management Control and Status Register (POWMGTCSR)," for detailed information about the bits of POWMGTCSR.

Note also that unmasked interrupts that occur while the device is in the process of going into the sleep state (before sleep is completely attained) can also cause the device to clear the POWMGTCSR[DOZ0/1,SLP] fields and return the device to full power operation.

23.5.1.9 Snooping in Power-Down Modes

When the MPC8572E is in doze mode, the e500 core is in the core-halted state and it snoops its L1 caches and full coherency is maintained. In deeper power-down modes, however, the e500 core does not respond to snoops.

The MPC8572E does not perform dynamic bus snooping as described in the *e500 Reference Manual*. That is, when the e500 core is in the core-stopped state the core is not awakened to perform snoops on global transactions. Therefore, before entering nap mode, the L1 caches should be flushed if coherency is required during this power-down mode.

23.5.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8572E logic (external to the core complex) to enter a power saving state. It is assumed that the desired power-saving state (doze or nap) was set up by setting the appropriate HID0 bit, typically at system start-up time. Setting WE has no direct effect on instruction execution, but is reflected on the internal *doze* and *nap* signals, depending on the HID0 settings. To ensure a clean transition into and out of a power-saving mode, the following program sequence is recommended:

```
            sync
            mtmsr (WE)
            isync
loop:       br loop
```

23.5.1.11 Requirements for Reaching and Recovering from Sleep State

In order to successfully reach the sleep state, I/O traffic to the device must be stopped. The logic that controls the power down sequence waits for all I/O interfaces to become idle. In some applications this may happen eventually without actively shutting down interfaces, but most likely, software will have to take steps to shut down the eTSEC, Serial RapidIO, and PCI Express interfaces among others before issuing the command (either the write to the core MSR[WE] as described above or writing to POWMGTCSR) to put the device into sleep state.

Chapter 24
Device Performance Monitor

This chapter describes the device performance monitor facility, which can be used to monitor and optimize performance. The e500 core implements a separate performance monitor for strictly core-related behavior, such as instruction timing and L1 cache operations. This is described in the *PowerPC™ e500 Core Reference Manual* (Freescale Document Order No. E500CORERM).

Section 24.4.7, "Performance Monitor Events," briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

24.1 Introduction

The device-level performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. Although the performance monitor described here is similar in many respects to the performance monitor facility implemented on the e500 core, it differs in that it is implemented using memory-mapped registers and it counts events outside the e500 core, for example, PCI, DDR, and L2 cache events.

Performance monitor counters (PMC0–PMC11) are used to count events selected by the performance monitor local control registers. PMC0 is a 64-bit counter specifically designated to count cycles. PMC1–PMC11 are 32-bit counters that can monitor 64 counter-specific events in addition to counting 64 reference events.

The benefits of the on-chip performance monitor are numerous, and include the following:

- Because some systems or software environments are not easily characterized by signal traces or benchmarks, the performance monitor can be used to understand the MPC8572E behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

24.1.1 Overview

Figure 24-1 is a high-level block diagram of the performance monitor, which consists of a global control register (PMGC0), one 64-bit counter (PMC0), eleven 32-bit counters, and two control registers per counter (24 total control registers). The global control register PMGC0 affects all counters and takes priority over local control registers. The local control registers are divided into two groups, as follows:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values, and the value of the threshold multiplier. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

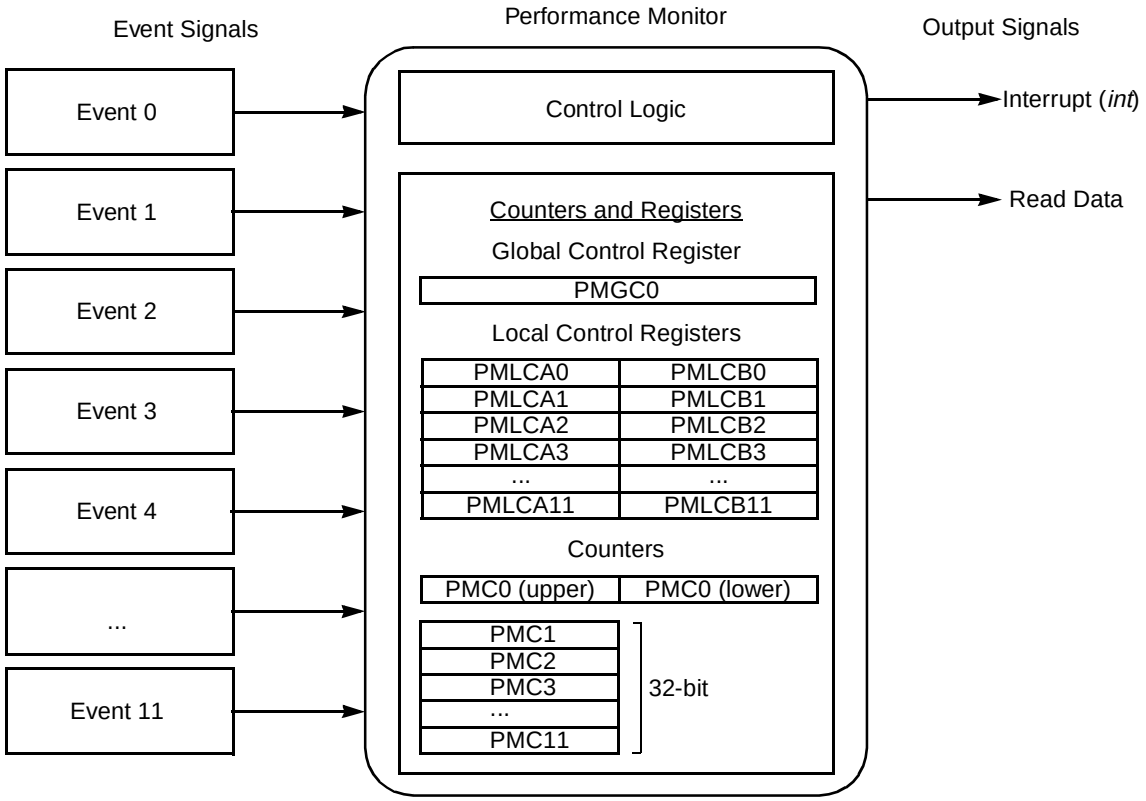


Figure 24-1. Performance Monitor Block Diagram

Performance monitor events are signalled by the functional blocks in the integrated device and are selectively recorded in the PMCs. Sixty-four of these events are referred to as reference events, which can be counted on any of the eleven 32-bit counters. Counter-specific events can be counted only on the counter where the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signalled. The PMCs can also be programmed to freeze when an interrupt is signalled.

24.1.2 Features

The MPC8572E performance monitor offers a rich set of features that permits a complete performance characterization of the implementation. These features include:

- One 64-bit counter exclusively dedicated to counting cycles
- Eleven 32-bit counters that count the occurrence of selected events
- One global control register (affects all counters) and two local control registers per counter
- Ability to count up to 64 reference events that may be counted on any of the eleven 32-bit counters
- Ability to count up to 704 counter-specific events
- Triggering and chaining capability
- Duration and quantity threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

24.2 Signal Descriptions

The performance monitor does not have any signals that are driven externally (off-chip) but it does assert the internal interrupt (*int*) signal on a performance monitor interrupt condition.

24.3 Memory Map and Register Definition

Performance monitor registers reside in the run-time register block starting at offset 0xE_1000. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved. This section describes the registers implemented to support the performance monitor facilities. Table 24-1 lists the performance monitor registers. These registers can be read or written only with 32-bit accesses.

24.3.1 Register Summary

The performance monitor uses twelve counter registers and a group of local control registers that are used to specify the method of counting. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 24-1. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	24.3.2.1/24-6
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1018	PMC0 (lower)—Performance monitor counter 0 upper	R/W	0x0000_0000	24.3.3.1/24-11
0xE_101C	PMC0 (upper)—Performance monitor counter 0 lower	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	24.3.3.1/24-11

Table 24-1. Control Register Memory Map (continued)

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	24.3.3.1/24-11
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	24.3.2.2/24-6
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	24.3.3.1/24-11
0xE_10A0	PMLCA9—Performance monitor local control register A9	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10A4	PMLCB9—Performance monitor local control register B9	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10A8	PMC9—Performance monitor counter 9	R/W	0x0000_0000	24.3.3.1/24-11
0xE_10B0	PMLCA10—Performance monitor local control register A10	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10B4	PMLCB10—Performance monitor local control register B10	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10B8	PMC10—Performance monitor counter 10	R/W	0x0000_0000	24.3.3.1/24-11
0xE_10C0	PMLCA11—Performance monitor local control register A11	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10C4	PMLCB11—Performance monitor local control register B11	R/W	0x0000_0000	24.3.2.2/24-6
0xE_10C8	PMC11—Performance monitor counter 11	R/W	0x0000_0000	24.3.3.1/24-11

In addition to these registers, the interrupt control provides four pairs of mask registers that can be used to monitor message, interprocessor, timer, and external interrupts. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs),” on page 10-34.

24.3.2 Control Registers

This section describes the performance monitor control registers in detail.

24.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 24-2, is a 32-bit register used to control all PMCs.

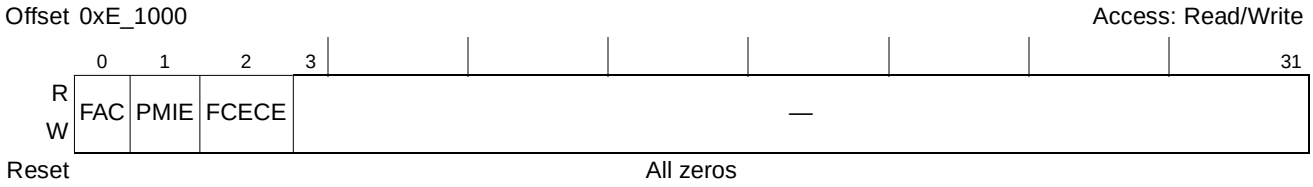


Figure 24-2. Performance Monitor Global Control Register (PMGC0)

Table 24-2 describes PMGC0 fields.

Table 24-2. PMGC0 Field Descriptions

Bits	Name	Description
0	FAC	Freeze all counters. 0 PMCs are incremented (if permitted by other PMGC0/PMLC bits). 1 PMCs are not incremented. Set by hardware when an interrupt is signalled and FCECE =1.
1	PMIE	Performance monitor interrupt enable. Interrupts are caused by PMC overflows. 0 Interrupts are disabled. 1 Interrupts are enabled and occur when an enabled condition or event occurs.
2	FCECE (DISCOUNT)	Freeze counters on enabled condition or event. An enabled condition or event is defined as: The msb = 1 in PMC*n* and PMLCA*n*[CE] = 1. The use of the trigger and freeze counter conditions depends on the enabled condition. 0 PMCs can be incremented (if permitted by other control bits). 1 PMCs can be incremented (if permitted by other control bits) only until an enabled condition or event occurs, at which time PMGC0[FAC] is set. It is up to software to clear FAC.
3–31	—	Reserved

24.3.2.2 Performance Monitor Local Control Registers (PMLCA*n*, PMLCB*n*)

The performance monitor local control registers (PMLCA*n* and PMLCB*n*) are used to control the operation of the PMCs. The performance monitor local control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter.

Figure 24-3 shows the performance monitor local control A0 register (PMLCA0).

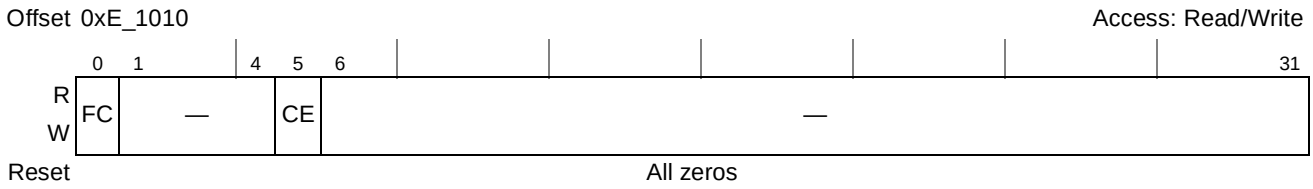


Figure 24-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 24-3 describes PMLCA0 fields.

Table 24-3. PMLCA0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter. Basic counter enable. 0 The PMCs are enabled and incremented (if permitted by other SPM control bits). 1 The PMCs are disabled–they do not increment.
1–4	—	Reserved
5	CE	Condition enable. Controls counter overflow condition. Should be cleared when PMC0 is used as a trigger or is selected for chaining. 0 Overflow conditions for PMC0 cannot occur (PMC0 cannot cause interrupts or freeze counters) 1 Overflow conditions occur when PMC0[msb] is set.
6–31	—	Reserved

Figure 24-4 shows the performance monitor local control registers A1–A11.

Figure 24-4. Performance Monitor Local Control A Registers (PMLCA1–PMLCA11)

Table 24-4 describes PMLCA*n* fields.

Table 24-4. PMLCA1–PMLCA11 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter 0 The PMCs are incremented (if permitted by other PMC control bits). 1 The PMCs are not incremented (if permitted by other PMC control bits).
1–4	—	Reserved
5	CE	Condition enable 0 Overflow conditions for PMC*n* cannot occur (PMC*n* cannot cause interrupts or freeze counters). Should be cleared when PMC*n* is used as a trigger or is selected for chaining. 1 Overflow conditions occur when PMC*n*[msb] is set.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable. Note that with counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64. See Table 24-10 for definition of events.
16–20	BSIZE	Burst size. Fewest event occurrences that constitute a burst, that is, a rapid sequence of events followed by a relatively long pause. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion. See Section 24.4.6, “Burstiness Counting,” for more information.
21–25	BGRAN	Burst granularity. The maximum number of clock cycles between events that are considered part of a single burst. See Section 24.4.6, “Burstiness Counting.”
26–31	BDIST	Burst distance (used with TBMULT). The number of clock cycles between bursts. Must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 24-5 shows the performance monitor local control B0 register (PMLCB0).

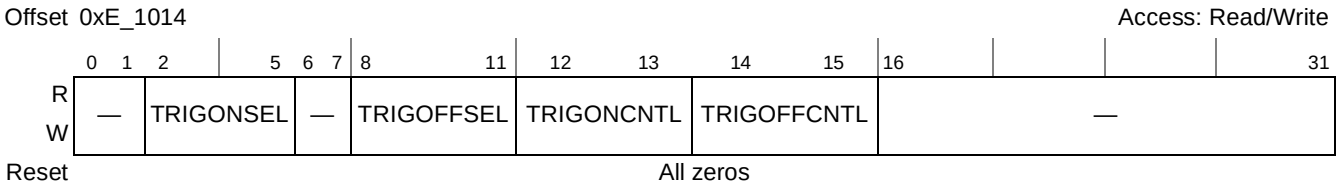


Figure 24-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 24-5 describes PMLCB0 fields.

Table 24-5. PMLCB0 Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. The number of the counter that starts event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved

Table 24-5. PMLCB0 Field Descriptions (continued)

Bits	Name	Description
8–11	TRIGOFFSEL	Trigger-off select. The number of the counter that stops event counting. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–31	—	Reserved

Figure 24-6 shows performance monitor local control registers 1–11.

Offset 0xE_1024 Access: Read/Write
0xE_1034
0xE_1044
0xE_1054
0xE_1064
0xE_1074
0xE_1084
0xE_1094
0xE_10A4
0xE_10B4
0xE_10C4

	0	1	2	5	6	7	8	11	12	13	14	15	16	20	21	23	24	25	26	31
R	—		TRIGONSEL		—		TRIGOFFSEL		TRIGONCNTL		TRIGOFFCNTL		—		TBMULT		—		THRESHOLD	
W																				
Reset	All zeros																			

Figure 24-6. Performance Monitor Local Control Register B (PMLCB1–PMLCB11)

Table 24-6 describes PMLCB*n* fields.

Table 24-6. PMLCB*n* Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. Set this field equal to the number of the counter that should trigger event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter.
6–7	—	Reserved

Table 24-6. PMLCB*n* Field Descriptions (continued)

Bits	Name	Description
8–11	TRIGOFFSEL	Trigger-off select. Set this field equal to the number of the counter that should trigger event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–20	—	Reserved
21–23	TBMULT	Threshold and burstiness multiplier. Threshold events are counted when the event duration exceeds a specified threshold value. The threshold is scaled based on the TBMULT settings.TBMULT is not used to scale the threshold value for quantity threshold events. The burst distance for burstiness counting is also scaled using the TBMULT settings. For all events that scale the threshold, the threshold field is multiplied by the factors shown below (ranging from 1 to 128). 000 1 001 2 010 4 011 8 100 16 101 32 110 64 111 128
24–25	—	Reserved
26–31	THRESHOLD	Threshold. Only events whose (number of) occurrences exceed this value are counted. By varying the threshold value, software can characterize the events subject to the threshold. For example, if PMC2 counts eTSEC BD read latencies for which the duration exceeds the threshold, software can obtain the distribution of eTSEC BD read latencies for a given program by monitoring the program using various threshold values.

24.3.3 Counter Registers

This section describes the PMCs in detail.

NOTE

Because accessing a PMC manually has priority over incrementing it due to event counting, writing a PMC while it is counting may affect the count. Likewise, writing a performance monitor control register while its target counter is counting may also affect the count.

24.3.3.1 Performance Monitor Counters (PMC0–PMC11)

PMC0–PMC11 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 24-7, is associated with two 32-bit registers that form a 64-bit counter designated to count clock cycles. PMC0 upper represents the upper 32 bits of counter 0, and PMC0 lower represents the lower 32 bits.

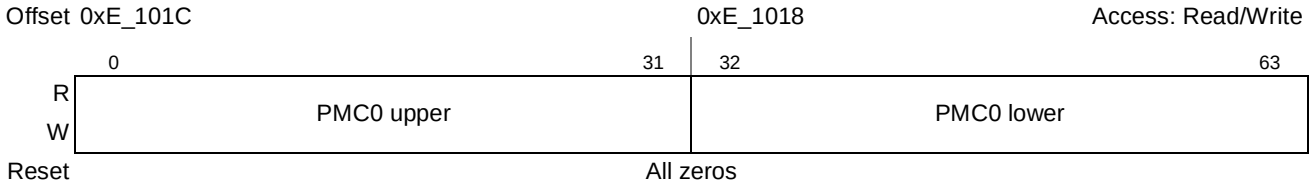


Figure 24-7. Performance Monitor Counter Register 0 (PMC0)

Table 24-7 describes PMC0 fields.

Table 24-7. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC11, shown in Figure 24-8, are 32-bit counters that can monitor 64 unique events in addition to the 64 reference events that can be counted on all of these registers.

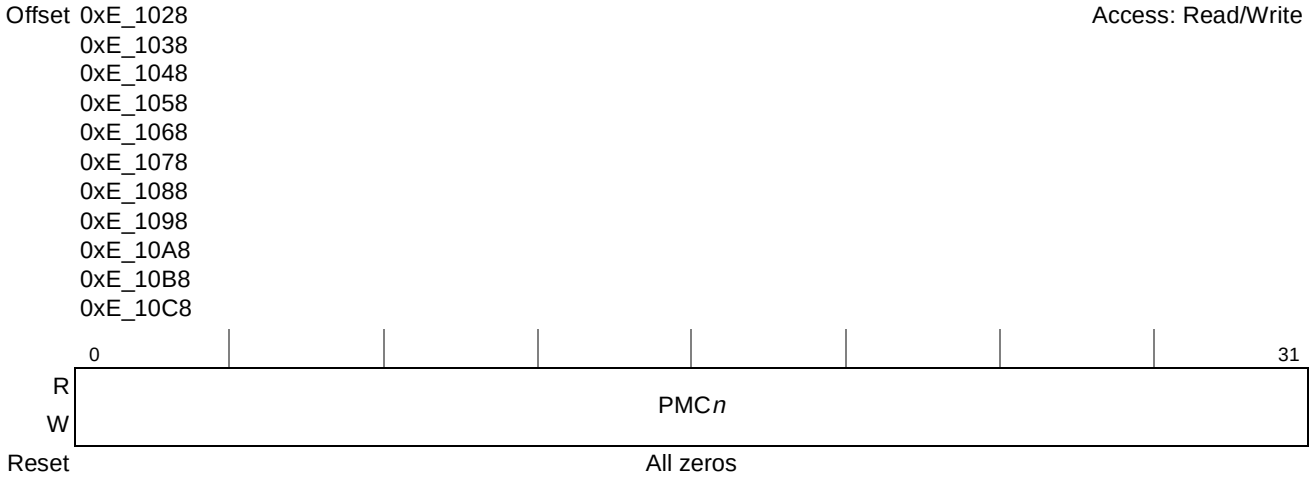


Figure 24-8. Performance Monitor Counter Register (PMC1–PMC11)

Table 24-8 describes PMC*n* fields.

Table 24-8. PMC[1–9] Field Descriptions

Bits	Name	Description
0–31	PMC*n*	Event count. An overflow is indicated when the msb = 1. Manually setting the msb can cause an immediate interrupt.

24.4 Functional Description

This section describes the use of some features of the performance monitor.

24.4.1 Performance Monitor Interrupt

PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit (PMLCA*n*[CE]) and performance monitor interrupt enable bit (PMGC0[PMIE]) must be set. When an interrupt is signalled and the freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set, PMGC0[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

24.4.2 Event Counting

Using the control registers described in Section 24.3.2, “Control Registers,” the twelve PMCs can count the occurrences of specific events. The 64-bit PMC0 is designated to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC11). Additionally, up to 64 unique events can be counted on each 32-bit counter.

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers.þCounters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.þ

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

24.4.3 Threshold Events

The threshold feature allows characterization of events that can take a variable number of clock cycles to occur. Threshold events are counted only if the latency is greater than the threshold value specified in PMLCB*n*[THRESHOLD]. There are two types of threshold events.

The first type of threshold events are duration threshold events. For duration threshold event sequences, the PMC increments only when the duration of the event is equal to or greater than the threshold value. The threshold value is scaled by a multiple specified in PMLCB*n*[TBMULT].

A duration threshold event requires two signals: The first indicates when a threshold event sequence begins, and the second indicates when it ends. An internal counter determines when the threshold count is exceeded and when the PMC can increment. This internal counter decrements during a threshold event sequence until it reaches the value of one. A new sequence cannot begin until the current one completes. Additional threshold start signals are ignored during a sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle in a current sequence, the stop terminates the current sequence and the start signals the beginning of a new one. However, if both signals are asserted

during the same cycle while not in a current event sequence, both signals are ignored. Figure 24-9 is a timing diagram for duration threshold event counting.

An illegal condition exists if the threshold value obtained from PMLCB*n*[THRESHOLD] and PMLCB*n*[TBMULT] is less than two. Under these conditions the intent of threshold counting is ambiguous.

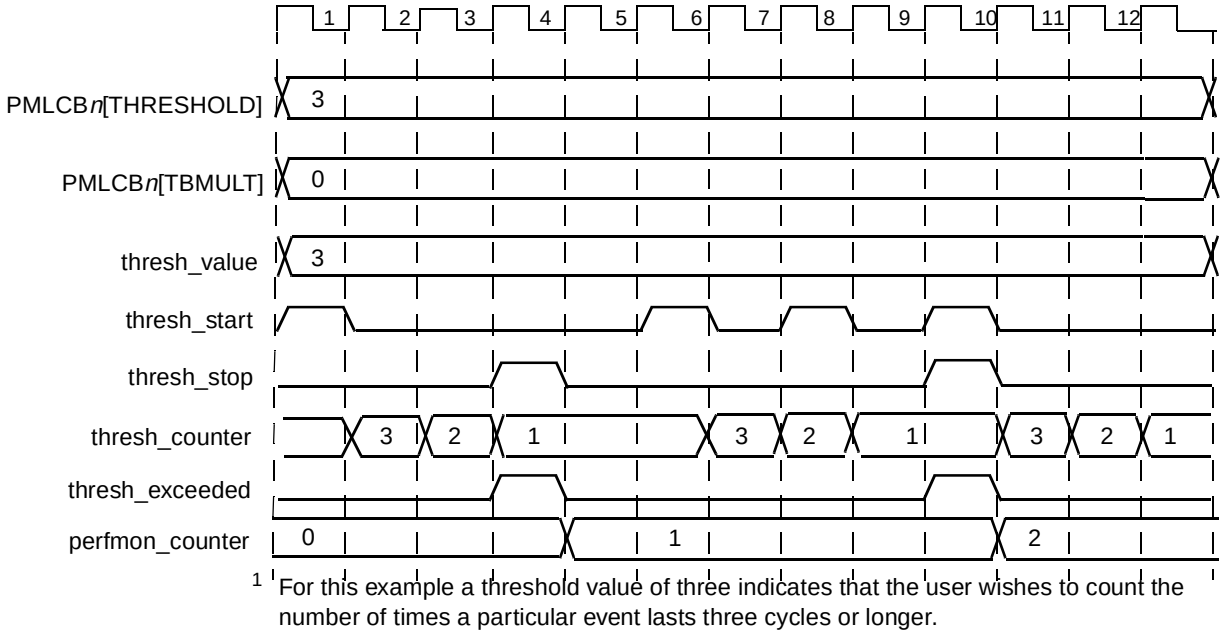


Figure 24-9. Duration Threshold Event Sequence Timing Diagram

The second type of threshold event is the quantity threshold event. For these types of threshold event sequences the performance monitor counter is only incremented when the specified threshold event exceeds the threshold value. These events do not use the multiplier register field (PMLCB*n*[TBMULT]) like the duration threshold events. This type of threshold event is generally used to monitor the usage of buffers and queues. For example, the usage of a specific queue could be characterized by measuring the amount of time the queue is completely full or partially full. For this example the threshold field would be used to specify how many entries are required to be valid in the queue for that event to be counted.

24.4.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in Table 24-10, are dedicated to specific registers.

Note that the chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register's msb is set. For this overflow to occur, PMLCA*n*[CE] should be cleared to avoid signalling an interrupt when the counter's most significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

24.4.5 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local control register of counter PMC1. This is done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs.

When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions.

Triggering is disabled when the counter's trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

24.4.6 Burstiness Counting

The burstiness counting feature makes it easier to characterize events that occur in rapid succession followed by a relatively long pause. As shown in Table 24-9, event bursts are defined by size, granularity, and distance.

Table 24-9. Burst Definition

Parameter	Description	Register Field
Size	The minimum number of events constituting a burst	PMLCA*n* [BSIZE]
Granularity	The maximum time between individual events counted as members of the same burst	PMLCA*n* [BGRAN]
Distance	The minimum time between bursts	PMLCA*n* [BDIST] x PMLCB*n*[TBMULT]

Figure 24-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

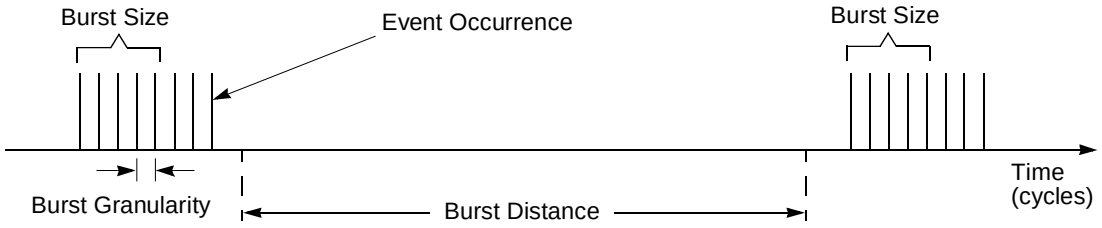


Figure 24-10. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA*n*[BSIZE]) specifies the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences equals or exceeds

PMLCA*n*[BSIZE]. Furthermore, these individual event occurrences must be separated by no more clock cycles than the value in the burstiness granularity field (PMLCA*n*[BGRAN]). Note that, although a burst is identified when the minimum number of events occurs, it is not counted until the burst sequence has ended. A burst sequence ends when the specified burstiness granularity is exceeded, at which point the last valid event has occurred for that sequence.

PMLCA*n*[BGRAN] specifies the maximum number of cycles between individual events for them to qualify as members of the same burst sequence.

The burstiness distance field (PMLCA*n*[BDIST]) and threshold/burstiness multiplier field (PMLCB*n*[TBMULT]) specify the acceptable number of cycles between the end of a burst sequence and the beginning of a new sequence for a group of event occurrences to be counted as an individual burst. The product of the burstiness distance field and the threshold/burstiness multiplier field determine the burstiness distance value used to determine when another burst sequence can begin. Note that the burst distance count begins when a new burst sequence ends and the PMC is incremented. No new burst sequence may begin until the burst distance count has reached zero. After the burst distance count reaches zero, it holds the zero value indicating that a new burst sequence can be counted. The burst distance count begins again when a new burst sequence is identified and counted.

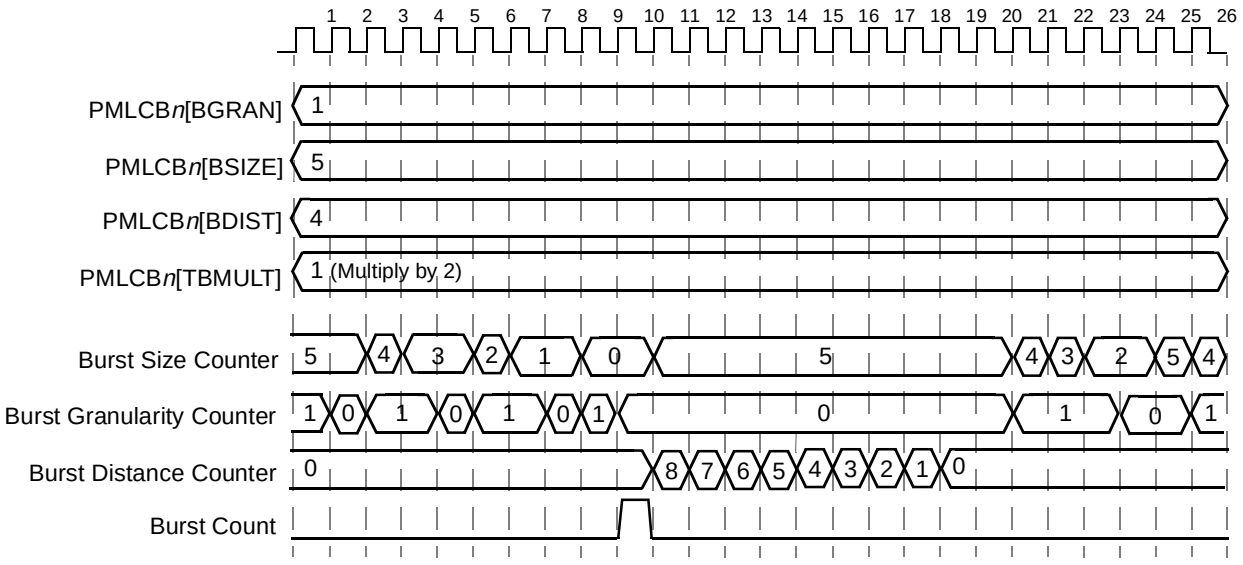
Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

Figure 24-10 shows that the burst distance is measured from the end of one burst sequence and that a new burst sequence may not begin until the burst distance count expires.

Three internal counters track the different values required for burstiness counting.

- Burstiness size is monitored by a counter. It is loaded with the value specified in the local control register when the burst granularity counter and the burst distance counters reach zero, and no new event is occurring. It always decrements when the following conditions occur: its value is not already zero, an event occurs, and the burst distance count equals zero.
- Burstiness granularity is monitored by a counter that is loaded with the specified value in the local control register on the rising edge of an event occurrence whenever the burst distance count equals zero. The granularity counter is decremented (if it has not already reached zero) when an event is not occurring and burst distance count equals zero.
- Burstiness distance is measured by a counter that is loaded with the product of PMLCB*n*[BDIST] and PMLCB*n*[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled (and the counter has not already reached zero).

A burst is counted at the end of a burst sequence when the three burst parameter counters are all equal to zero. Figure 24-11 shows a burstiness counting example.



1 For this example: count bursts of 5 event occurrences, burst granularity of 1, and acceptable distance between bursts of 8 (product of TBMULT and BDIST).

Figure 24-11. Burstiness Counting Timing Diagram

24.4.7 Performance Monitor Events

Table 24-10 lists performance monitor events specified in PMLCA1–PMLC11.

The event assignment column indicates the event's type and number, using the following formats:

- Ref:#—Reference events are shared across counters PMC1–PMC11. The number indicates the event. For example, Ref:6 means that PMC1–PMC11 share reference event 6.
- C[0–11]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:62 means PMC8 is assigned event 62 (PIC interrupt wait cycles).

Counter events not specified in Table 24-10 are reserved.

Note that some functional blocks may operate at a frequency other than that of the performance monitor. In these cases, the count represented by the performance monitor counters may be inaccurate.

Table 24-10. Perfor.mance Monitor Events

Event Counted	Number	Description of Event Counted
General Events		
Nothing	Ref:0	Register counter holds current value
System cycles	C0 and Ref:63	CCB (platform) clock cycles

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
DDR Memory Controller 1 Events		
Cycles a read is returning data from DRAM	Ref:19	Each data beat returned to the memory controller on the DRAM interface
Cycles a read or write transfers data from (or to) DRAM	Ref:27	Each data beat transferred to or from the DRAM
Pipelined read misses in the row open table	C1:121	Row open table read misses issued while a read is outstanding
Pipelined read or write misses in the row open table	C2:64	Row open table read or write misses issued while a read or write is outstanding
Non-pipelined read misses in the row open table	C3:124	Row open table read misses issued when no reads are outstanding
Non-pipelined read or write misses in the row open table	C4:64	Row open table read or write misses issued when no reads or writes are outstanding
Pipelined read hits in the row open table	C5:120	Row open table read hits issued when a read is outstanding
Pipelined read or write hits in the row open table	C6:64	Row open table read or write hits issued when a read or write is outstanding
Non-pipelined read hits in the row open table	C7:121	Row open table read hits issued when no reads are outstanding
Non-pipelined read or write hits in the row open table	C8:64	Row open table read or write hits issued when no reads or writes are outstanding
Forced page closings not caused by a refresh	C1:64	Precharges issued to the DRAM for any reason except refresh. The possibilities are as follows: • A new transaction must be issued to an already active bank and sub-bank that has a different row open. • A new transaction must be issued, but the row open table is full and there is no bank/sub-bank match between the current transaction and the row open table. • The BSTOPRE interval expired for an open row.
Row open table misses	C2:65 and C11:66	Transactions that miss in the row open table
Row open table hits	C3:64 and C10:68	Transaction that hit in the row open table
Force page closings	C4:65	Forced page closings including those due to refreshes
Read-modify-write transactions due to ECC	C5:64	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence is issued on the DRAM interface.
Forced page closings due to collision with bank and sub-bank	C9:64	Increments if a new transaction must be issued to an active bank and sub-bank that has a different row open
Reads or writes from core 0-1	Ref:13	—
Reads or writes from eTSEC 1-4	C3:65	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Reads or writes from high speed interfaces (PCI Express 1-3, and Serial RapidIO)	C4:67	—
Reads or writes from DMA 1-2	C5:66	—
Reads or writes from Security	C6:69	—
Row open table hits for reads or writes from core 0-1	Ref:14	—
Row open table hits for reads or writes from eTSEC 1-4	C6:65	—
Row open table hits for reads or writes from high speed interfaces (PCI Express 1-3, and Serial RapidIO)	C7:65	—
Row open table hits for reads or writes from DMA 1-2	C8:66	—
Row open table hits for reads or writes from Security	C7:68 and C10:65	—
DDR Memory Controller 2 Events		
Cycles a read is returning data from DRAM	Ref:28	Each data beat returned to the memory controller on the DRAM interface
Cycles a read or write transfers data from (or to) DRAM	Ref:29	Each data beat transferred to or from the DRAM
Pipelined read misses in the row open table	C5:65	Row open table read misses issued while a read is outstanding
Pipelined read or write misses in the row open table	C6:75	Row open table read or write misses issued while a read or write is outstanding
Non-pipelined read misses in the row open table	C7:73	Row open table read misses issued when no reads are outstanding
Non-pipelined read or write misses in the row open table	C8:70	Row open table read or write misses issued when no reads or writes are outstanding
Pipelined read hits in the row open table	C1:72	Row open table read hits issued when a read is outstanding
Pipelined read or write hits in the row open table	C2:73	Row open table read or write hits issued when a read or write is outstanding
Non-pipelined read hits in the row open table	C3:77	Row open table read hits issued when no reads are outstanding
Non-pipelined read or write hits in the row open table	C4:78	Row open table read or write hits issued when no reads or writes are outstanding

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Forced page closings not caused by a refresh	C9:74	Precharges issued to the DRAM for any reason except refresh. The possibilities are as follows: • A new transaction must be issued to an already active bank and sub-bank that has a different row open. • A new transaction must be issued, but the row open table is full and there is no bank/sub-bank match between the current transaction and the row open table. • The BSTOPRE interval expired for an open row.
Row open table misses	C9:73 and C10:67	Transactions that miss in the row open table
Row open table hits	C9:65 and C11:67	Transaction that hit in the row open table
Force page closings	C10:66	Forced page closings including those due to refreshes
Read-modify-write transactions due to ECC	C9:69	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence is issued on the DRAM interface.
Forced page closings due to collision with bank and sub-bank	C11:65	Increments if a new transaction must be issued to an active bank and sub-bank that has a different row open
Reads or writes from core 0-1	Ref:30	—
Reads or writes from eTSEC 1-4	C7:75	—
Reads or writes from high speed interfaces (PCI Express 1-3, and Serial RapidIO)	C9:71	—
Reads or writes from DMA 1-2	C10:69	—
Reads or writes from Security	C11:68	—
Row open table hits for reads or writes from core 0-1	Ref:31	—
Row open table hits for reads or writes from eTSEC 1-4	C10:70	—
Row open table hits for reads or writes from high speed interfaces (PCI Express, and Serial RapidIO)	C2:75	—
Row open table hits for reads or writes from DMA 1-2	C3:78	—
Row open table hits for reads or writes from Security	C4:80	—
Memory Target Queue 1 Events		
MEM TQ read/write address collision	C5:69	—
Memory Target Queue 2 Events		
MEM TQ read/write address collision	C1:74	—
DMA Controller 1 Events		

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Channel 0 read request	C1:66	DMA channel 0 read request active in the system
Channel 1 read request	C2:69	DMA channel 1 read request active in the system
Channel 2 read request	C3:68	DMA channel 2 read request active in the system
Channel 3 read request	C4:70	DMA channel 3 read request active in the system
Channel 0 write request	C1:67	DMA channel 0 write request active in the system
Channel 1 write request	C2:70	DMA channel 1 write request active in the system
Channel 2 write request	C3:69	DMA channel 2 write request active in the system
Channel 3 write request	C4:71	DMA channel 3 write request active in the system
Channel 0 descriptor request	C5:105	DMA channel 0 descriptor request active in the system
Channel 1 descriptor request	C6:108	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C7:105	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C8:105	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:68 and C5:117	DMA channel 0 read double word valid
Channel 1 read DW or less	C2:71 and C6:122	DMA channel 1 read double word valid
Channel 2 read DW or less	C3:70 and C7:118	DMA channel 2 read double word valid
Channel 3 read DW or less	C4:72 and C8:116	DMA channel 3 read double word valid
Channel 0 write DW or less	C1:69	DMA channel 0 write double word valid
Channel 1 write DW or less	C2:72	DMA channel 1 write double word valid
Channel 2 write DW or less	C3:71	DMA channel 2 write double word valid
Channel 3 write DW or less	C4:73	DMA channel 3 write double word valid
DMA Controller 2 Events		
Channel 0 read request	C8:86	DMA channel 0 read request active in the system
Channel 1 read request	C5:94	DMA channel 1 read request active in the system
Channel 2 read request	C10:75	DMA channel 2 read request active in the system
Channel 3 read request	C11:73	DMA channel 3 read request active in the system
Channel 0 write request	C8:90	DMA channel 0 write request active in the system
Channel 1 write request	C5:104	DMA channel 1 write request active in the system
Channel 2 write request	C10:76	DMA channel 2 write request active in the system
Channel 3 write request	C11:74	DMA channel 3 write request active in the system
Channel 0 descriptor request	C4:101	DMA channel 0 descriptor request active in the system

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Channel 1 descriptor request	C5:85	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C6:79	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C7:91	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:78 and C8:91	DMA channel 0 read double word valid
Channel 1 read DW or less	C2:80 and C9:108	DMA channel 1 read double word valid
Channel 2 read DW or less	C3:82 and C10:77	DMA channel 2 read double word valid
Channel 3 read DW or less	C4:107 and C11:75	DMA channel 3 read double word valid
Channel 0 write DW or less	C5:95	DMA channel 0 write double word valid
Channel 1 write DW or less	C6:92	DMA channel 1 write double word valid
Channel 2 write DW or less	C7:92	DMA channel 2 write double word valid
Channel 3 write DW or less	C8:92	DMA channel 3 write double word valid
e500 Coherency Module (ECM) Events		
ECM request wait core 0	C8:77	Asserted for every cycle core 0 request occurs
ECM request wait core 1	C11:98	Asserted for every cycle core 1 request occurs
ECM request wait I^2C/Security/FEC/Test Port	C7:77	Asserted for every cycle I2C/Security/FEC/Test Port request occurs
ECM request wait PEX1-3/DMA1-2/SRIO	C5:80	Asserted for every cycle PEX1-3/DMA1-2/SRIO request occurs
ECM request wait eTSEC1-4	C6:80	Asserted for every cycle eTSEC1-4 request occurs
ECM request wait SAP/PME/TLU0-1	C4:84	Asserted for every cycle SAP/PME/TLU0-1 request occurs
ECM dispatch	Ref:15	ECM dispatch (includes address only's) Note: all ECM dispatch events are for committed dispatches
ECM dispatch from core 0	C1:80	ECM dispatch from core 0 (includes address only's)
ECM dispatch from core 1	C10:95	ECM dispatch from core 1 (includes address only's)
ECM dispatch from eTSEC1	C4:85	—
ECM dispatch from eTSEC2	C8:80	—
ECM dispatch from eTSEC3	C9:80	—
ECM dispatch from eTSEC4	C5:81	—
ECM dispatch from SRIO	C10:96	—
ECM dispatch from TLU0	C6:81	—
ECM dispatch from TLU1	C7:80	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM dispatch from PEX1	C8:81	—
ECM dispatch from PEX2	C10:98	—
ECM dispatch from PEX3	C7:78	—
ECM dispatch from DMA1	C9:82	—
ECM dispatch from DMA2	C10:99	—
ECM dispatch from Security	C10:100	—
ECM dispatch from RIO Message Unit, Door Bell, or Port Write	C8:78	—
ECM dispatch from Pattern Matcher	C4:86	—
ECM dispatch from other	C6:82	—
ECM dispatch to DDR1	C7:79	—
ECM dispatch to DDR2	C10:101	—
ECM dispatch to L2	C10:102	—
ECM dispatch to SRAM	C8:79	—
ECM dispatch to LBIU	C9:83	—
ECM dispatch to RIO	C1:85	—
ECM dispatch to PEX1	C2:84	—
ECM dispatch to PEX2	C1:81	—
ECM dispatch to PEX3	C2:85	—
ECM dispatch to CCSR	C9:87	—
ECM dispatch write	C7:81	—
ECM dispatch write allocate	C1:82	—
ECM dispatch read	C2:86	—
ECM dispatch read atomic	C3:86	—
ECM data bus grant DDR1	C1:83	—
ECM data bus grant DDR2	C2:87	—
ECM data bus grant I^2C/Security/FEC/Test Port	C3:87	—
ECM data bus grant PEX1-3/DMA1-2/SRIO	C4:89	—
ECM data bus grant LBC	Ref:16	—
ECM data bus grant eTSEC1-4	Ref:17	—
ECM data bus grant SAP/PME/TLU0-1	Ref:18	—
ECM global data bus beat	C5:86	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM e500 direct read bus beat	C11:95	—
ECM e500 direct read bus beat forwarded	C7:84	ECM direct read bus beat forwarded directly to e500 R1 data bus
ECM cancel	C8:83	—
Mag2Cu (I2C/Security/FEC/Test Port) magenta requests	C1:119	Number of magenta requests (total)
Mag2Cu (I2C/Security/FEC/Test Port) magenta read requests	C2:76	Number of magenta read requests
Mag2Cu (I2C/Security/FEC/Test Port) magenta data beats	C3:75	Number of magenta data beats (total)
Mag2Cu (I2C/Security/FEC/Test Port) magenta read data beats	C4:79	Number of magenta read data beats
Mag2Cu (I2C/Security/FEC/Test Port) magenta request less than 32 bytes	C7:76	Number of magenta requests less than 32 bytes
Mag2Cu (SAP/PME/TLUs) magenta requests	C1:99	Number of magenta requests (total)
Mag2Cu (SAP/PME/TLUs) magenta read requests	C10:78	Number of magenta read requests
Mag2Cu (SAP/PME/TLUs) magenta data beats	C11:76	Number of magenta data beats (total)
Mag2Cu (SAP/PME/TLUs) magenta read data beats	C5:96	Number of magenta read data beats
Mag2Cu (SAP/PME/TLUs) magenta request less than 32 bytes	C3:101	Number of magenta requests less than 32 bytes
Interrupt Controller (PIC) Events		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt wait cycles	C8:126	Counts cycles when an interrupt waits to be acknowledge
PIC interrupt service cycles	C2:83	Number of cycles there is an interrupt currently being serviced.
PIC interrupt select 0 (Duration threshold)	C1:120	THRESHOLD: select 0–3: interrupt count over threshold. (Note: only unmasked, nonzero priority requests are acknowledged). The four interrupts are selected through register pairs, PM0MR*n*–PM3MR*n*. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs).”
PIC interrupt select 1 (Duration threshold)	C3:123	
PIC interrupt select 2 (Duration threshold)	C5:119	
PIC interrupt select 3 (Duration threshold)	C6:124	
eTSEC 1 Events		
DMA write data beats	C3:109	DMA write data beats
DMA read data beats	C4:110	DMA read data beats

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
DMA Write Request	C5:106	DMA Write Request
DMA Read Request	C6:109	DMA Read Request
Number of dropped frames	C9:88	Number of dropped frames
TxBD read lifetime (Duration Thresh-old)	Ref:34	TxBD read lifetime
RxBD read lifetime (Duration Thresh-old)	Ref:38	RxBD read lifetime
TxBD write lifetime (Duration Thresh-old)	Ref:42	TxBD write lifetime
RxBD write lifetime (Duration Thresh-old)	Ref:46	RxBD write lifetime
Read data lifetime (Duration Thresh-old)	Ref:50	Read data lifetime
Rx IP packets checked for checksum	C9:92	Rx IP packets checked for checksum
TX IP packet with checksum	C1:105	TX IP packet with checksum
TX TCP/UDP packet with checksum	C2:113	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C3:114	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C4:115	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C5:111	Number of rejected frames by filer
Number of rejected frames due to filer error	C6:114	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C5:110	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C6:113	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C7:110	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C8:110	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:110	—
Number of accepted frames to station address	C10:80	—
Number of accepted unicasted frames via hash	C11:78	—
Number of accepted group frames via hash	C6:96	—
Number of accepted frames via exact match	C7:100	—
Number of rejected frames at layer 2	C9:111	—
Number of RX interrupts signalled	C8:94	—
Number of TX interrupts signalled	C9:112	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
RX data write lifetime (Duration Threshold)	C10:81	—
Number of RX packets received while RX FIFO is full	C11:79	—
eTSEC 2 Events		
DMA write data beats	C5:107	DMA write data beats
DMA read data beats	C6:110	DMA read data beats
DMA Write Request	C7:107	DMA Write Request
DMA Read Request	C8:107	DMA Read Request
Number of dropped frames	C2:110	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:35	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:39	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:43	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:47	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:51	Read data lifetime
Rx IP packets checked for checksum	C3:115	Rx IP packets checked for checksum
TX IP packet with checksum	C4:116	TX IP packet with checksum
TX TCP/UDP packet with checksum	C5:101	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C6:115	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C7:111	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C8:111	Number of rejected frames by filer
Number of rejected frames due to filer error	C9:93	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C7:113	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C8:113	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C9:96	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C1:108	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:113	—
Number of accepted frames to station address	C10:82	—
Number of accepted unicasted frames via hash	C11:80	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Number of accepted group frames via hash	C6:97	—
Number of accepted frames via exact match	C7:101	—
Number of rejected frames at layer 2	C9:114	—
Number of RX interrupts signalled	C10:83	—
Number of TX interrupts signalled	C11:81	—
RX data write lifetime (Duration Threshold)	C8:96	—
Number of RX packets received while RX FIFO is full	C9:115	—
eTSEC 3 Events		
DMA write data beats	C7:108	DMA write data beats
DMA read data beats	C8:108	DMA read data beats
DMA Write Request	C9:90	DMA Write Request
DMA Read Request	C1:103	DMA Read Request
Number of dropped frames	C4:112	Number of dropped frames
TxBD read lifetime (Duration Threshold)	Ref:36	TxBD read lifetime
RxBD read lifetime (Duration Threshold)	Ref:40	RxBD read lifetime
TxBD write lifetime (Duration Threshold)	Ref:44	TxBD write lifetime
RxBD write lifetime (Duration Threshold)	Ref:48	RxBD write lifetime
Read data lifetime (Duration Threshold)	Ref:52	Read data lifetime
Rx IP packets checked for checksum	C6:116	Rx IP packets checked for checksum
TX IP packet with checksum	C7:112	TX IP packet with checksum
TX TCP/UDP packet with checksum	C8:112	TX TCP/UDP packet with checksum
TCP/UDP packets checked for c.s.	C9:94	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C1:106	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C2:114	Number of rejected frames by filer
Number of rejected frames due to filer error	C3:116	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C9:97	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C1:109	Number of cycles Rx FIFO > 1/2 full

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Number of cycles Rx FIFO > 3/4 full	C2:116	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C3:118	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:116	—
Number of accepted frames to station address	C10:84	—
Number of accepted unicasted frames via hash	C11:82	—
Number of accepted group frames via hash	C6:98	—
Number of accepted frames via exact match	C7:102	—
Number of rejected frames at layer 2	C9:117	—
Number of RX interrupts signalled	C8:97	—
Number of TX interrupts signalled	C9:118	—
RX data write lifetime (Duration Threshold)	C10:85	—
Number of RX packets received while RX FIFO is full	C11:83	—
eTSEC 4 Events		
DMA write data beats	C9:91	DMA write data beats
DMA read data beats	C1:104	DMA read data beats
DMA Write Request	C2:112	DMA Write Request
DMA Read Request	C3:112	DMA Read Request
Number of dropped frames	C6:104	Number of dropped frames
TxBD read lifetime (Duration Thresh-old)	Ref:37	TxBD read lifetime
RxBD read lifetime (Duration Thresh-old)	Ref:41	RxBD read lifetime
TxBD write lifetime (Duration Thresh-old)	Ref:45	TxBD write lifetime
RxBD write lifetime (Duration Thresh-old)	Ref:49	RxBD write lifetime
Read data lifetime (Duration Thresh-old)	Ref:53	Read data lifetime
Rx IP packets checked for checksum	C9:95	Rx IP packets checked for checksum
TX IP packet with checksum	C1:107	TX IP packet with checksum
TX TCP/UDP packet with checksum	C2:115	TX TCP/UDP packet with checksum

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
TCP/UDP packets checked for c.s.	C3:117	TCP/UDP packets checked for c.s.
IP or TCP/UDP Rx checksum error	C4:117	IP or TCP/UDP Rx checksum error
Number of rejected frames by filer	C5:102	Number of rejected frames by filer
Number of rejected frames due to filer error	C6:106	Number of rejected frames due to filer error
Number of cycles Rx FIFO > 1/4 full	C8:104	Number of cycles Rx FIFO > 1/4 full
Number of cycles Rx FIFO > 1/2 full	C9:98	Number of cycles Rx FIFO > 1/2 full
Number of cycles Rx FIFO > 3/4 full	C1:110	Number of cycles Rx FIFO > 3/4 full
Number of cycles Rx FIFO = full	C2:117	Number of cycles Rx FIFO = full
Number of accepted frames match	C9:119	—
Number of accepted frames to station address	C10:86	—
Number of accepted unicasted frames via hash	C11:84	—
Number of accepted group frames via hash	C6:99	—
Number of accepted frames via exact match	C7:95	—
Number of rejected frames at layer 2	C9:120	—
Number of RX interrupts signalled	C10:87	—
Number of TX interrupts signalled	C11:85	—
RX data write lifetime (Duration Threshold)	C8:98	—
Number of RX packets received while RX FIFO is full	C9:121	—
PCI Express 1 to OCeaN Gasket Events		
Inbound G2PI read	C8:119	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C9:101	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:124	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:126	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:125	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:124	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Inbound Static Queue 0 start (Duration Threshold)	Ref:54	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:55	Lifetime of OSQ entry 0.
PCI Express 2 to OCeaN Gasket Events		
Inbound G2PI read	C3:102	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C4:103	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C5:97	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C1:95	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C2:101	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C11:87	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:32	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:33	Lifetime of OSQ entry 0.
PCI Express 3 to OCeaN Gasket Events		
Inbound G2PI read	C9:124	A single pulse to indicate an inbound G2PI read has occurred.
Inbound G2PI write	C10:90	A single pulse to indicate an inbound G2PI write has occurred.
Inbound G2PI data	C11:89	A level signal to indicate the amount of data transferred if any for inbound G2PI request. Active for every beat of G2PI data.
Outbound G2PI read	C6:102	A single pulse to indicate an outbound G2PI read has occurred.
Outbound G2PI write	C7:98	A single pulse to indicate an outbound G2PI write has occurred.
Outbound G2PI data	C8:127	A level signal to indicate the amount of data transferred if any for outbound G2PI request. Active for every beat of G2PI data.
Inbound Static Queue 0 start (Duration Threshold)	Ref:61	Lifetime of ISQ entry 0 or 6.
Outbound Static Queue 0 start (Duration Threshold)	Ref:62	Lifetime of OSQ entry 0.
Serial RapidIO Events		
Inbound packet accepted	C1:124	Packet accepted on RIO
Inbound priority 0 packet accepted	C2:126	Packet accepted from RIO of priority 0

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Inbound non-idles received	C4:126	Non idles received. This can be used to determine the RIO link utilization. This is actually 1/2 of the actual count (please see section 4.14.1 for more information).
Inbound packet retry occurred	C5:83	Packet retry occurred, any cause
Inbound buffer is full	C6:85	Clock cycle occurred in which the inbound buffer is full to any priority (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.
Inbound buffer is full to priority 0	C7:123	Clock cycle occurred in which the inbound buffer is full to priority 0 (measured from when EOP received to EOP transferred on OCN). Event asserted for as many clock cycles as this is true.
Outbound non-idles transmitted	Ref:59	Non idles transmitted. This can be used to determine the RIO link utilization. This is actually 1/2 of the actual count (please see section 4.14.1 for more information).
Outbound packet sent to RapidIO interface	C2:127	Outbound packet sent to RapidIO interface
Outbound packet was misaligned	C4:76	Outbound packet was misaligned
Outbound packet was retried	C5:84	Outbound packet was retried
Outbound packet was reordered	C6:73	Outbound packet was reordered
Outbound packet sent to RIO of priority 0	C7:124	Outbound packet sent to RIO of priority 0
Outbound buffer is full	C8:123	Clock cycle occurred in which the outbound buffer is full to any priority. Event asserted for as many clock cycles as this is true.
Outbound buffer is full to priority 0	C9:103	Clock cycle occurred in which the outbound buffer is full to priority 0. Event asserted for as many clock cycles as this is true.
RapidIO Message Unit Events		
Inbound packet received	Ref:60	Packet received of any priority
Inbound priority 0 packet received	C5:77	Packet received of priority 0
Inbound buffer is full	C6:72	Clock cycle occurred in which the inbound buffer is full to any priority (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.
Inbound buffer is full for priority 0	C7:72	Clock cycle occurred in which the inbound buffer is full to priority 0 (measured from when SOP received to buffer release transferred on OCN). Event asserted for as many clock cycles as this is true.
Packet sent to OCeaN	C8:102	Packet sent to OCN
Packet sent to OCeaN of priority 0	C11:94	Packet sent to OCN of priority 0
Protocol Converter 1 Events		

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PEX Symbol Gain of 1 when SKP detected	C2:118	Number of times the Protocol Converter Elastic Buffer gains 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Gain of 2 when SKP detected	C3:95	Number of times the Protocol Converter Elastic Buffer gains 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Gain of 3 when SKP detected	C4:123	Number of times the Protocol Converter Elastic Buffer gains 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 1 when SKP detected	C5:118	Number of times the Protocol Converter Elastic Buffer loss 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Loss of 2 when SKP detected	C6:123	Number of times the Protocol Converter Elastic Buffer loss 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 3 when SKP detected	C7:119	Number of times the Protocol Converter Elastic Buffer loss 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Disparity Error	C8:117	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:99	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
Protocol Converter 2 Events		
PEX Symbol Gain of 1 when SKP detected	C9:126	Number of times the Protocol Converter Elastic Buffer gains 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Gain of 2 when SKP detected	C10:92	Number of times the Protocol Converter Elastic Buffer gains 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Gain of 3 when SKP detected	C11:91	Number of times the Protocol Converter Elastic Buffer gains 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 1 when SKP detected	C6:105	Number of times the Protocol Converter Elastic Buffer loss 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Loss of 2 when SKP detected	C7:94	Number of times the Protocol Converter Elastic Buffer loss 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 3 when SKP detected	C1:97	Number of times the Protocol Converter Elastic Buffer loss 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Disparity Error	C1:98	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C9:127	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
Protocol Converter 3 Events		
PEX Symbol Gain of 1 when SKP detected	C6:86	Number of times the Protocol Converter Elastic Buffer gains 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Gain of 2 when SKP detected	C7:126	Number of times the Protocol Converter Elastic Buffer gains 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Gain of 3 when SKP detected	C2:108	Number of times the Protocol Converter Elastic Buffer gains 3 PEX Symbols before receiving a SKP ordered-set

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PEX Symbol Loss of 1 when SKP detected	C9:109	Number of times the Protocol Converter Elastic Buffer loss 1 PEX Symbol before receiving a SKP ordered-set
PEX Symbol Loss of 2 when SKP detected	C10:93	Number of times the Protocol Converter Elastic Buffer loss 2 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Loss of 3 when SKP detected	C11:92	Number of times the Protocol Converter Elastic Buffer loss 3 PEX Symbols before receiving a SKP ordered-set
PEX Symbol Disparity Error	C10:94	Number of times a receive PEX Symbol Disparity Error was detected
PEX Symbol Decode Error	C11:93	Number of times an unrecognizable PEX Symbol (Decode Error) was received and detected
Local Bus Events		
Bank 1 hits (chip-select)	C1:115	—
Bank 2 hits (chip-select)	C2:120	—
Bank 3 hits (chip-select)	C3:119	—
Bank 4 hits (chip-select)	C4:118	—
Bank 5 hits (chip-select)	C5:112	—
Bank 6 hits (chip-select)	C6:117	—
Bank 7 hits (chip-select)	C7:114	—
Bank 8 hits (chip-select)	C8:114	—
Number of correctable FCM ECC errors	C3:121	—
Number of uncorrectable FCM ECC errors	C6:119	—
Requests granted to TLU port	C1:116	—
Requests granted to ECM port	C2:121	—
Cycles atomic lock for TLU port is enabled	C3:120	—
Cycles atomic reservation for ECM port is enabled	C4:119	—
Atomic reservation time-outs for TLU port	C5:113	—
Atomic reservation time-outs for ECM port	C6:118	—
Cycles for a read to FCM buffer RAM	C7:115	—
Cycles for a write to FCM buffer RAM	C8:115	—
Cycles a read is taking in GPCM	C1:117	—
Cycles a read is taking in UPM	C2:122	—
Cycles a write is taking in GPCM	C4:120	—

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Cycles a write is taking in UPM	C5:114	—
L2 Cache/SRAM Events		
Core instruction accesses to L2 that hit	Ref:22	—
Core instruction accesses to L2 that miss	C2:123	—
Core data accesses to L2 that hit	Ref:23	—
Core data accesses to L2 that miss	C4:121	—
Non-core burst write to L2 (cache external write or SRAM)	C5:115	—
Non-core non-burst write to L2	C6:120	—
Noncore write misses cache external write window and SRAM memory range	C7:116	—
Non-core read hit in L2	Ref:24	—
Non-core read miss in L2	C1:118	—
L2 allocates, from any source	Ref:25	—
L2 retries due to full write queue	C2:124	—
L2 retries due to address collision	C3:122	—
L2 failed lock attempts due to full set	C4:122	—
L2 victimizations of valid lines	C5:116	—
L2 invalidations of lines	C6:121	—
L2 clearing of locks	C7:117	—
Pattern Matching Engine (PME) Events		
DXE Execute Sequencer is at busy2 activity level	C11:118	Number of cycles in which the DXE Execute Sequencer is at busy2 activity level.
DXE Execute Sequencer is at busy1 activity level	C10:118	Number of cycles in which the DXE Execute Sequencer is at busy1 activity level.
DXE Prefetch Sequencer is at busy3 activity level	C3:108	Number of cycles in which the DXE Prefetch Sequencer is at busy3 activity level.
DXE Prefetch Sequencer is at busy2 activity level	C2:105	Number of cycles in which the DXE Prefetch Sequencer is at busy2 activity level.
DXE Prefetch Sequencer is at busy1 activity level	C1:73	Number of cycles in which the DXE Prefetch Sequencer is at busy1 activity level.
SRE Execute Sequencer is at busy3 activity level	C5:87	Number of cycles in which the SRE Execute Sequencer is at busy3 activity level.
SRE Execute Sequencer is at busy2 activity level	C11:117	Number of cycles in which the SRE Execute Sequencer is at busy2 activity level.

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
SRE Execute Sequencer is at busy1 activity level	C10:117	Number of cycles in which the SRE Execute Sequencer is at busy1 activity level.
SRE Prefetch Sequencer is at busy3 activity level	C5:82	Number of cycles in which the SRE Prefetch Sequencer is at busy3 activity level.
SRE Prefetch Sequencer is at busy2 activity level	C11:116	Number of cycles in which the SRE Prefetch Sequencer is at busy2 activity level.
SRE Prefetch Sequencer is at busy1 activity level	C10:116	Number of cycles in which the SRE Prefetch Sequencer is at busy1 activity level.
Free Buffer Manager (FBM) is at busy3 activity level	C11:119	Number of cycles in which the Free Buffer Manager (FBM) is at busy3 activity level.
Free Buffer Manager (FBM) is at busy2 activity level	C10:119	Number of cycles in which the Free Buffer Manager (FBM) is at busy2 activity level.
Free Buffer Manager (FBM) is at busy1 activity level	C4:109	Number of cycles in which the Free Buffer Manager (FBM) is at busy1 activity level.
Key Element Scanner (KES) is at busy2 activity level	C9:105	Number of cycles in which the Key Element Scanner (KES) is at busy2 activity level.
Key Element Scanner (KES) is at busy1 activity level	C8:125	Number of cycles in which the Key Element Scanner (KES) is at busy1 activity level.
Data Expansion Engine (DEE) is at busy2 activity level	C4:90	Number of cycles in which the Data Expansion Engine (DEE) is at busy2 activity level.
Data Expansion Engine (DEE) is at busy1 activity level	C6:66	Number of cycles in which the Data Expansion Engine (DEE) is at busy1 activity level.
Huffman Decoding Engine (HDE) is at busy2 activity level	C11:115	Number of cycles in which the Huffman Decoding Engine (HDE) is at busy2 activity level.
Huffman Decoding Engine (HDE) is at busy1 activity level	C10:115	Number of cycles in which the Huffman Decoding Engine (HDE) is at busy1 activity level.
DMA Engine (DE) is at busy3 activity level	C3:127	Number of cycles in which the DMA Engine (DE) is at busy3 activity level.
DMA Engine (DE) is at busy2 activity level	C2:104	Number of cycles in which the DMA Engine (DE) is at busy2 activity level.
DMA Engine (DE) is at busy1 activity level	C1:126	Number of cycles in which the DMA Engine (DE) is at busy1 activity level.
Memory Interface Arbiter (MIA) is busy	C9:81	Number of cycles in which the Memory Interface Arbiter (MIA) is busy. Asserted if there are any outstanding read or write transactions remaining in MIA\x{2019}s internal queues.
Write transactions issued on the system bus	C8:71	Number of write transactions issued on the system bus. Asserts for one cycle for every write transaction issued.
Write transactions pending on the system bus	C4:87	Number of write transactions (from 0 to 4) currently pending on the system bus.
	C6:87	
	C11:114	

Table 24-10. Perfor.mance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Read transactions issued on the system bus	C10:114	Number of cycles a read transactions is issued on the system bus.
Read transactions pending on the system bus	C3:126	Number of read transactions (from 0 to 4) currently pending on the system bus.
	C2:103	
	C1:123	
Debug Events		
External event	C3:125	Number of cycles trig_in pin is asserted
Watchpoint monitor hits	C2:125	—
Trace buffer hits	C1:122	—
DUART Events		
UART0 baud rate	C1:127	—
UART1 baud rate	C5:127	—
Chaining Events		
PMC0 carry-out	Ref:1	PMC0[0] 1-to-0 transitions.
PMC1 carry-out	Ref:2	PMC1[0] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[0] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[0] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[0] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[0] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[0] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[0] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[0] 1-to-0 transitions. Reserved for PMC8.
PMC9 carry-out	Ref:10	PMC9[0] 1-to-0 transitions. Reserved for PMC9.
PMC10 carry-out	Ref:11	PMC10[0] 1-to-0 transitions. Reserved for PMC10.
PMC11 carry-out	Ref:12	PMC11[0] 1-to-0 transitions. Reserved for PMC11.

24.4.8 Performance Monitor Examples

Table 24-12 contains sample register settings for the four supported modes.

- Simple event performance monitoring example
- Triggering event performance monitoring example
- Threshold event performance monitoring example
- Burstiness event performance monitoring example

The settings in Table 24-11 are identical for all four examples.

Table 24-11. PMGC0 and PMLCA*n* Settings

Field	Setting	Reason
PMGC0[FAC]	0	Counters must not be frozen.
PMGC0[PMIE]	1	Performance monitor interrupts are enabled
PMGC0[FCECE]	1	Counters should be frozen when an interrupt is signalled.
PMLCA*n*[FC]	0	Counters cannot be frozen for counting.
PMLCA*n*[CE]	1	Overflow condition enable is required to allow interrupt signalling.

For simple event counting, a non-threshold event is selected in PMLCA*n*[EVENT] and all other features are disabled by clearing all register fields except for CE.

For the triggering example any event can be selected in PMLCA*n*[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signalling. If PMLCB*n*[TRIGONSEL] is 3 and PMLCB*n*[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCB*n*[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCB*n*[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCA*n*[CE] for this counter must be cleared so that interrupt signalling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT]. For this example, the duration threshold value is scaled by two because PMLCB*n*[TBMULT] is one. All other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values for PMLCA*n*[BSIZE,BGRAN,BDIST] and PMLCB*n*[TBMULT] must be specified.

Table 24-12. Register Settings for Counting Examples

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMGC0	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1
PMLCA*n*	FC	0	0	0	0
	CE	1	1	1	1
	EVENT	89	68	39	2
	BSIZE	0	0	0	5
	BGRAN	0	0	0	1
	BDIST	0	0	0	8

Table 24-12. Register Settings for Counting Examples (continued)

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMLCB*n*	TRIGONSEL	0	3	0	0
	TRIGOFFSEL	0	5	0	0
	TRIGONCNTL	0	1	0	0
	TRIGOFFCNTL	0	2	0	0
	TBMULT	0	0	0	0
	THRESHOLD	0	0	3	0

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers. Counters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.þ

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

Chapter 25
Debug Features and Watchpoint Facility

This chapter describes all customer-visible debug modes of the MPC8572E integrated device. The debug features on the MPC8572E pertain to these interfaces: the local bus controller (LBC), and the DDR SDRAM interface. In addition to the external interfaces, the MPC8572E provides triggering capabilities based on user-programmable events. The watchpoint and trace buffer also provide some visibility to internal buses. This chapter also describes context ID registers, useful for software debug, and describes the JTAG access port signals that comply with the IEEE 1149.1 boundary-scan specification.

25.1 Introduction

As shown in the block diagram of Figure 25-1, the MPC8572E device provides the following debug features (listed with references to sections of this chapter that describe them):

- DDR SDRAM interface debug (Section 25.4.2, “DDR SDRAM Interface Debug”)
- Local bus controller (LBC) debug (Section 25.4.3, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 25.4.4, “Watchpoint Monitor,” and Section 25.4.5, “Trace Buffer”)

25.1.1 Overview

As shown in Figure 25-1, debug information is provided through the following interfaces: LBC and DDR SDRAM. Limited visibility, through a 256 x 64 trace buffer, is also provided for the processor core interface. This visibility into internal device operation is useful for debugging application software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID[0:4]) and a data-valid signal (MDVAL) indicates that meaningful debug information is visible on either the local bus or DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID[0:4] and MDVAL.

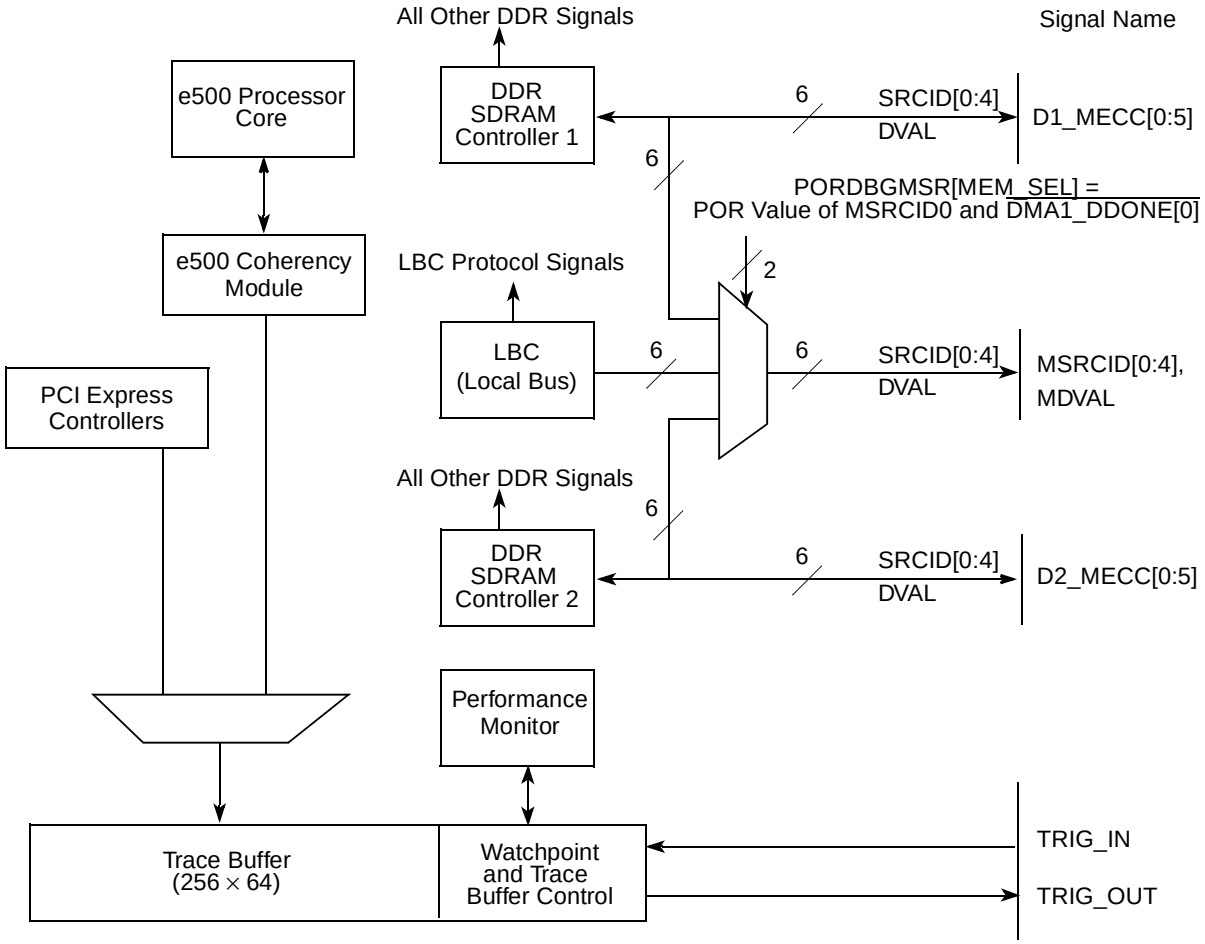


Figure 25-1. Debug and Watchpoint Monitor Block Diagram

Other system debugging is supported by the programmable triggering of the watchpoint monitor and trace buffer. Both can be triggered from one of the following three sources:

- Each other
- A performance monitor event
- An external source (through TRIG_IN).

The watchpoint monitor can be configured to assert TRIG_OUT when a programmed event occurs. The two context ID registers, described in Section 25.3.3, “Context ID Registers,” are useful for software debug.

25.1.2 Features

The principal features of the debug modes and the watchpoint monitor are as follows:

- LBC and DDR interface source ID and data-valid indicators
  - LBC or DDR SDRAM source ID can be selected to be driven onto MSRCID[0:4]
  - Source ID and data-valid indicators can be selected to be driven onto the error correcting code (ECC) pins of the DDR interface
- Watchpoint monitor that supports
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
- Trace buffer features that support
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocked with performance monitor to use its large number of counters
  - 256-entry trace buffer, 64 bits each
  - Programmable trace start and stop
  - Can function as a second watchpoint monitor
- Context ID registers that can be programmed to trigger events

25.1.3 Modes of Operation

The LBC, and DDR SDRAM interfaces all have debug modes, which are controlled by values on configuration inputs during the power-on reset (POR) sequence, as shown in Table 25-1.The DDR controller can also drive debug information on either MSRCID[0:4] or MECC[0:5]. See Section 25.4.1, “Source and Target ID,” for additional information about the source ID information driven on the debug signals in these modes.

Note that both the watchpoint monitor and trace buffer also operate in a variety of modes.

Table 25-1. POR Configuration Settings and Debug Modes

Configuration Signal	POR Value	Effect	Reference
$\overline{\text{DMA2\_DACK}}$[0], $\overline{\text{DMA1\_DDONE}}$[0] Default (11)	00	Local bus SDRAM information appears on MSRCID[0:4] and MDVAL.	25.1.3.1/25-4
	01	Reserved	
	10	Debug information from the DDR SDRAM controller 1 appears on MSRCID[0:4] and MDVAL.	
	11	Debug information from the DDR SDRAM controller 2 appears on MSRCID[0:4] and MDVAL.	

Table 25-1. POR Configuration Settings and Debug Modes (continued)

Configuration Signal	POR Value	Effect	Reference
DMA2_DDONE[0]	0	MECC[0:4] operate in debug mode and provide memory debug source ID and MECC5 provides data-valid information.	25.1.3.2/25-4
	1	Default value (internal pull-up resistor). MECC[0:4] operate in normal mode and provide DDR SDRAM error correcting code information.	

25.1.3.1 Memory Debug Mode (Local Bus and DDR)

The LBC and the DDR SDRAM controller can drive debug information (source ID and data-valid indicator) onto MSRCID[0:4] and MDVAL. As shown in Table 25-1, the values of DMA2_DACK[0] and DMA1_DDONE[0] during POR control this multiplexing. If DMA2_DACK[0] and DMA1_DDONE[0] are low when sampled during POR, the local bus SDRAM information appears on MSRCID[0:4] and MDVAL. If DMA2_DACK[0] is high when sampled during POR, the debug information from one of DDR SDRAM controllers is presented on MSRCID[0:4] and MDVAL. In this case, the value of DMA1_DDONE[0] determines which DDR controller debug information appears on these debug signals.

25.1.3.2 DDR SDRAM Interface Debug Mode

DMA2_DDONE[0] is sampled during POR to multiplex either ECC or debug information on the ECC pins of the DDR SDRAM interfaces. As shown in Table 25-1, if DMA2_DDONE[0] is low during POR, the ECC pins operate in debug mode and provide memory debug source ID and data-valid information. DMA2_DDONE[0] must be pulled low during POR to use the ECC pins in debug mode. If DMA2_DDONE[0] is unconnected, an internal pull-up resistor ensures the ECC pins always source DDR SDRAM error correcting code information as their default power-on reset configuration.

NOTE

If the DDR ECC pins are in debug mode (configured for debug during POR), ECC checking is disabled in the memory controller. In this case, MECC[0:4] do not provide ECC information and must not be connected to SDRAM devices.

25.1.3.3 Watchpoint Monitor Modes

The watchpoint monitor supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The watchpoint monitor triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The watchpoint monitor waits for a specific event before enabling (arming) the trigger logic. The monitor does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Assert TRIG_OUT on hit—The debug block can be programmed to assert the TRIG_OUT signal when a programmed watchpoint monitor event occurs. This signal can be used to trigger a logic analyzer.

25.1.3.4 Trace Buffer Modes

The trace buffer supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The trace buffer triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The trace buffer waits for a specific event before enabling (arming) the trigger logic. The trace buffer does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Specific interface selection—The trace buffer can be programmed to trace one of several internal interfaces.
- Specific event selection—The trace buffer can be programmed to trace on the occurrence of one or several concurrent events.
- Specific trace selection—To facilitate trace data filtering, the trace buffer can be configured to capture data under the following conditions:
  - On every cycle in which a valid transaction is present on the selected interface
  - Only when the programmed trace event is detected
- Programmable trace stop—The trace buffer may be programmed to stop tracing when a programmed stop-tracing event occurs or when the 256-entry buffer is full.

25.2 External Signal Description

This section provides information about all the external signals associated with the various MPC8572E debug functions.

As shown in Table 25-1, the MPC8572E has several signals that are sampled during POR to determine the configuration of the phase-locked loop clock mode and the ROM, flash, and dynamic memory. See Section 4.4.3, “Power-On Reset Configuration.”

To facilitate system testing, the MPC8572E provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes JTAG TAP signals.

25.2.1 Overview

All the signals associated with device debug features are summarized in Table 25-2, listed with a reference to the page number of the section with more information. The detailed descriptions are contained in Table 25-2. Some signals (the MECC bus for example) are additionally described in other chapters, but are described here also for completeness, with emphasis on their debugging utility.

Table 25-2. Debug, Watchpoint and Test Signal Summary

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
MDVAL	Memory data-valid	Debug	Selectable data-valid signal from either DDR SDRAM controller or LBC.	1	O	25-7
MECC[0:7]	DDR error correcting code	DDR SDRAM	In debug mode, the high-order six bits carry debug information (transaction source ID and data-valid indication).	0x08	O[1]	25-7
MSRCID[0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM controllers or local bus controller.	Reset_cfg	O	25-7
MSRCID[2:4]				111	O	25-7
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer.	1	I	25-8
TRIG_OUT	Trigger out	Debug	Can be used externally for triggering a logic analyzer. Additionally, it can be used for observing system ready indication. Functions are multiplexed onto this signal depending on TOSR[SEL] (see Table 25-25).	1	O	25-8
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up.	1	I	25-8
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up.	1	I	25-8
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data.	Hi Z	O	25-8
TMS	Test mode select	Debug	Carries commands to the TAP controller for boundary scan operations. Internally pulled up.	1	I	25-8
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously.	—	I	25-8
TEMP_ ANODE	Temperature diode anode	Test	This pin ties directly to the anode of an internal diode used to asses die temperature.	—	I	25-8
TEMP_ CATHODE	Temperature diode cathode	Test	This pin ties directly to the cathode of an internal diode used to asses die temperature.	—	O	25-8
$\overline{\text{TEST\_SEL}}$	Test select 1	Test	Factory test. Must be negated (pulled high) for normal operation.	—	I	25-8
$\overline{\text{LSSD\_ MODE}}$	Test	Test	Factory Test. Refer to the *MPC8572E Integrated Host Processor Hardware Specifications* for proper treatment.		I	25-8
L1_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8572E Integrated Host Processor Hardware Specifications* for proper treatment.		I	25-8
L2_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8572E Integrated Host Processor Hardware Specifications* for proper treatment.		I	25-8

1 While these signals are normally bidirectional, when sourcing debug information they are output only.

25.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals

25.2.2.1 Debug Signals—Details

Table 25-3 describes all signals associated with device debug modes.

Table 25-3. Debug Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDVAL	O	Memory data-valid. Indicates when valid data is available. May be used by a logic analyzer to capture the data on the data bus.	
		State Meaning	Asserted—Indicates that data is valid on the data bus during the current clock cycle. When the DDR SDRAM interface is selected to source information on MDVAL, this signal is valid for every cycle that data is driven or received on the DDR SDRAM interface. When the LBC is selected, this signal is valid for every cycle that data is driven or received on the local bus interface. The assertion of this signal may be used by a logic analyzer to capture data.
		Timing	Asserted/Negated—Referenced to the selected interface, (DDR or local bus). Asserts when data is valid. Assertions are held for the duration of the transfer. Read data timing is similar to MA. Write data timing is similar to the output MDQ.
MECC[0:7]	O	Memory ECC. DDR error checking and correcting. The normally bidirectional operation of the memory ECC (MECC) bus is described in Section 9.5.11, “Error Checking and Correcting (ECC).” This bus is used for debug functions when MSRCID1 is sampled low during POR. In debug mode, the high-order 5 bits (MECC[0:4]) may be used to provide the transaction source ID and MECC5 can be used as the data-valid indicator. In debug mode, MECC[0:5] is constantly driven with debug information and must be disconnected from the DDR memory’s ECC pins.	
		State Meaning	Asserted/Negated—In debug mode, MECC[0:5] is always driven. The source ID values appear during RAS and CAS cycles. A value of 0x1F (all ones) is driven during cycles other than RAS and CAS. The data-valid indicator appears when data is being received or driven on the pins.
		Timing	Driven every cycle in debug mode.
MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in Table 25-1. Two of these signals serve as reset configuration input signals.	
		State Meaning	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than RAS and CAS. The encodings shown in Table 25-26 provide detailed information about a memory transaction.
		Timing	Driven every cycle in debug mode. Similar timing to MA.

25.2.2.2 Watchpoint Monitor Trigger Signals—Details

Table 25-4 shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 25-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TRIG_IN	I	Trigger in. Can be used to trigger the watchpoint and trace buffers. Note this is an active-high (rising-edge triggered) signal.	
		State Meaning	Asserted—Indicates that a programmed/armed external event has been detected. Assertion may be used internally to trigger trace buffers and watchpoint mechanisms.
		Timing	Assertion/Negation—The MPC8572E interprets TRIG_IN as asserted on detection of the rising edge. It may occur at any time. Must remain asserted for at least 3 system clocks to be recognized internally.
TRIG_OUT	O	Trigger out. Function determined by TOSR[SEL]. When TOSR[SEL] is non-zero, it can be used for triggering external devices, like a logic analyzer, with either the watchpoint monitor, the trace buffer, or the performance monitor as trigger sources. When TOSR[SEL] is cleared, TRIG_OUT is multiplexed with READY, which indicates the operational readiness of the device (running or in low-power or debug modes). See Chapter 4, "Reset, Clocking, and Initialization," and Chapter 23, "Global Utilities," for more details about reset, low-power, and debug states.	
		State Meaning	Asserted—When TOSR[SEL] is all zeros, serves as the READY signal, indicating that the device is not in a low-power or debug mode and that it has emerged from reset. SEL ≠ 0 indicates that a programmed trigger event has occurred. Negation—No final watchpoint match condition
		Timing	Assertion may occur at any time. Remains asserted for at least 3 system clocks

25.2.2.3 Test Signals—Details

Table 25-5 shows detailed descriptions of the JTAG test signals.

Table 25-5. JTAG Test and Other Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TCK	I	JTAG test clock.	
		State Meaning	Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
TDI	I	JTAG test data input.	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.

Table 25-5. JTAG Test and Other Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TDO	O	JTAG test data output.	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 standard for more details.
TMS	I	JTAG test mode select.	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details.
$\overline{\text{TRST}}$	I	JTAG test reset.	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the MPC8572E. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation.
		Timing	See IEEE 1149.1 standard for more details.
$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the *MPC8572E Integrated Processor Hardware Specifications* for proper treatment.	
L1_TSTCLK	I	Used for factory test. Refer to the *MPC8572E Integrated Processor Hardware Specifications* for proper treatment.	
L2_TSTCLK	I	Used for factory test. Refer to the *MPC8572E Integrated Processor Hardware Specifications* for proper treatment.	
TEMP_ ANODE	I	This signal provides access to the anode of a diode used to assess die temperature. See the *Integrated Host Processor Hardware Specifications* for more information on how to accurately measure the junction temperature of a device. Note that this thermal diode is intended for engineering development only.	
TEMP_ CATHODE	I	This signal provides access to the cathode of a diode used to assess die temperature. See the *Integrated Host Processor Hardware Specifications* for more information on how to accurately measure the junction temperature of a device. Note that this thermal diode is intended for engineering development only.	
$\overline{\text{TEST\_SEL}}$	I	Used for factory test. Should be negated (pulled high) for normal operation.	

25.3 Memory Map/Register Definition

Table 25-6 shows the memory-mapped debug and watchpoint registers of the MPC8572E. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 25-6. Debug and Watchpoint Monitor Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	25.3.1.1/25-11
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	25.3.1.1/25-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	25.3.1.2/25-13
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	25.3.1.3/25-14
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	25.3.1.4/25-14
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	25.3.1.5/25-16
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	25.3.2.1/25-16
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	25.3.2.1/25-16
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	25.3.2.2/25-19
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	25.3.2.3/25-19
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	25.3.2.4/25-20
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	25.3.2.5/25-20
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	25.3.2.6/25-21
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	25.3.2.7/25-22
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	25.3.2.8/25-22
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	25.3.3.1/25-23
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	25.3.3.2/25-23

Table 25-6. Debug and Watchpoint Monitor Memory Map (continued)

Local Memory Offset	Register	Access	Reset	Section/Page
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	25.3.4.1/25-24

25.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

25.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1) shown in Figure 25-2 and Figure 25-3 control the specification of watchpoint monitor events.

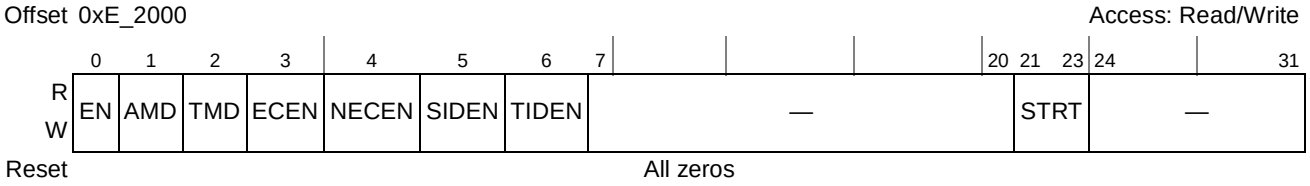


Figure 25-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 25-7 describes WMCR0 fields.

Table 25-7. WMCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 Watchpoint monitor events are not flagged. 1 A watchpoint monitor event is flagged.
1	AMD	Address match disable. Qualifies address match as a watchpoint event criterion. 0 Address matching is used to recognize a watchpoint event. 1 Address matching does not affect watchpoint event detection.
2	TMD	Transaction match disable. Qualifies transaction type match (as defined in WMCR1[IFSEL] and WMTMR) as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match does not affect watchpoint event detection.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 25.3.3, “Context ID Registers.” 0 Current context match does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).

Table 25-7. WMCR0 Field Descriptions (continued)

Bits	Name	Description
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 25.3.3, "Context ID Registers." 0 The failure of a current context match does not affect watchpoint event detection 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Source ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 Target ID does not affect watchpoint event detection. 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the watchpoint monitor to start looking for the programmed event. 000 No event. Armed immediately 001 Trace buffer event is detected 010 Performance monitor signals overflow 011 TRIG_IN transitions from 0 to 1 100 TRIG_IN transitions from 1 to 0 101 Current context ID equals programmed context ID 110 Current context ID is not equal to programmed context ID 111 Reserved
24–31	—	Reserved

Figure 25-3 shows the WMCR1.

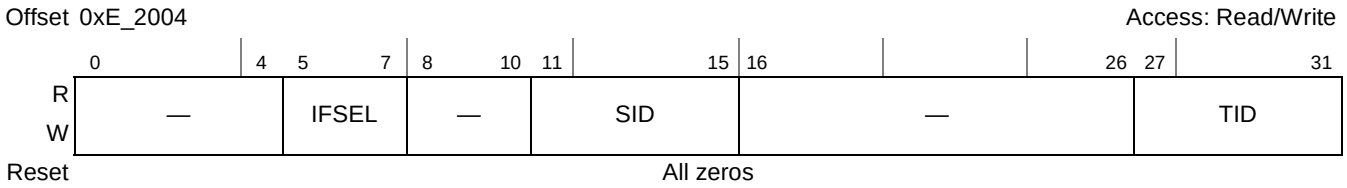


Figure 25-3. Watchpoint Monitor Control Register 1 (WMCR1)

Table 25-8 describes the WMCR1 fields.

Table 25-8. WMCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Selects the address, transaction type (as defined in WMTMR), and other attributes to be used for comparison 000 Selects e500 coherency module (ECM) dispatch interface 001 Reserved 010 Reserved 011 Reserved 100 Selects internal PCI Express 1 outbound interface 101 Selects internal PCI Express 2 outbound interface 110 Selects internal PCI Express 3 outbound interface 111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with WMCR0[SIDEN]. For a definition of the source ID, see Table 25-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with WMCR0[TIDEN]. For a definition of the target ID see Table 25-26.

25.3.1.2 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR) shown in Figure 25-4 contains the address to match against if WMCR[AMD] is clear. Note that this address may be further qualified with the bits described in Section 25.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR).”

Table 25-9 describes the WMAR fields.

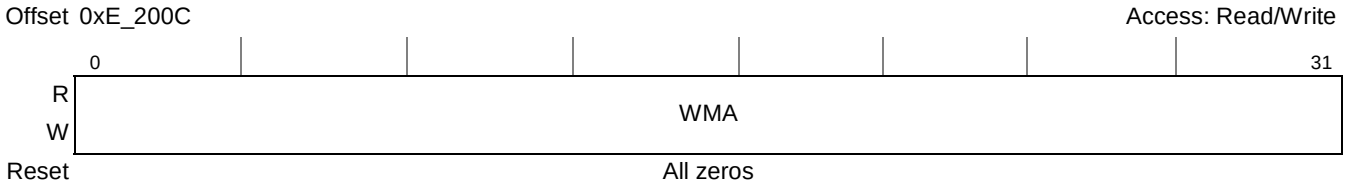


Figure 25-4. Watchpoint Monitor Address Register (WMAR)

Table 25-9. WMAR Field Descriptions

Bits	Name	Description
0–31	WMA	Watchpoint monitor address.

25.3.1.3 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in Figure 25-5 contains the mask for the address in the WMAR.

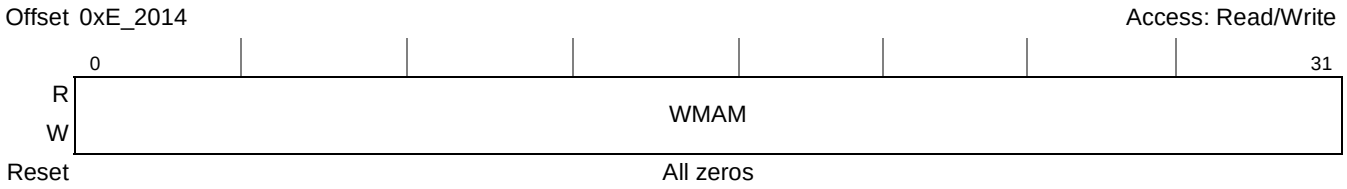


Figure 25-5. Watchpoint Monitor Address Mask Register (WMAMR)

Table 25-10 describes the WMAMR fields.

Table 25-10. WMAMR Field Descriptions

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. A value of zero masks the address comparison for the corresponding address bit. These bits only mask the address bits generated by the hardware, but do not affect the bits specified in WMAR. A bit that is masked from the comparison should be set to 0 in WMAR.

25.3.1.4 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in Figure 25-6, specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in Table 25-12, each bit represents as many as four separate transaction types; one for each interface. Setting a bit enables watchpoint monitoring for the corresponding transaction types.

Because the supported transaction types vary by interface, the type designated by a WMTMR field also depends on the interface specified by WMCR1[IFSEL]. Table 25-12 lists transaction types associated with each WMTMR bit by interface.

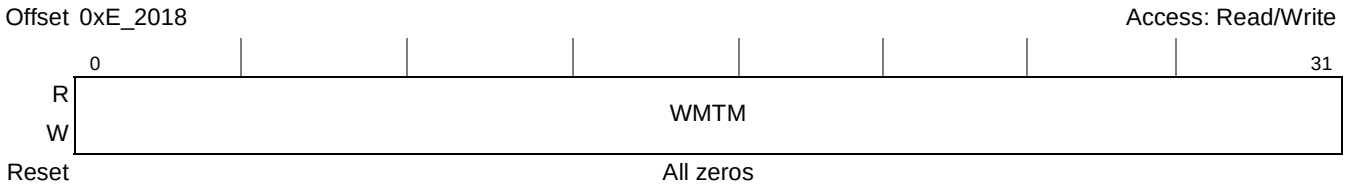


Figure 25-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 25-11 describes the WMTMR fields.

Table 25-11. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 25-12. The transaction associated with any particular bit may be different depending on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when WMCR0[TMD]=0.

The following table, Table 25-12, defines the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

Table 25-12. Transaction Types By Interface

Bit	Description	
	e500 Coherency Module Dispatch	PCI Express Outbound Transaction
0	Write with local processor snoop	Posted Write
1	Write with no local processor snoop	Non-posted Write
2	Write with allocate(L2 stashing)	—
3	Write with allocate and lock (L2 stashing with locking)	—
4	Reserved	—
5	Reserved	—
6	Reserved	—
7	Reserved	—
8	Read with local processor snoop	Read
9	Read with no local processor snoop	—
10	Read with unlock	—
11	Reserved	Read Response
12	Reserved	—
13–15	Reserved	—
16	ATOMIC clear	—
17	ATOMIC set	—
18	ATOMIC decrement	—
19	ATOMIC increment	—
20–24	Reserved	—
25	Address only transaction	—
26–31	Reserved	—

25.3.1.5 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR) shown in Figure 25-7 indicates the state of the watchpoint monitor.

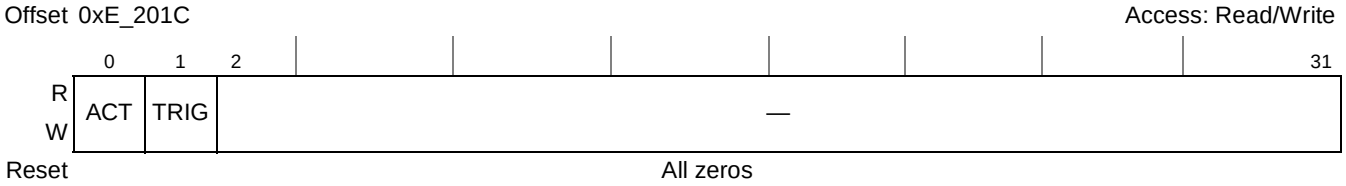


Figure 25-7. Watchpoint Monitor Status Register (WMSR)

Table 25-13 describes the WMSR fields.

Table 25-13. WMSR Field Descriptions

Bits	Name	Description
0	ACT	Active 0 The start triggering event has not occurred; watchpoint monitor is not armed. 1 The start triggering event has occurred; watchpoint monitor is armed.
1	TRIG	Triggered 0 The programmed event in WMCR0 has not yet been triggered. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

25.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

25.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 25-8 and Figure 25-9, specify trace buffer events.

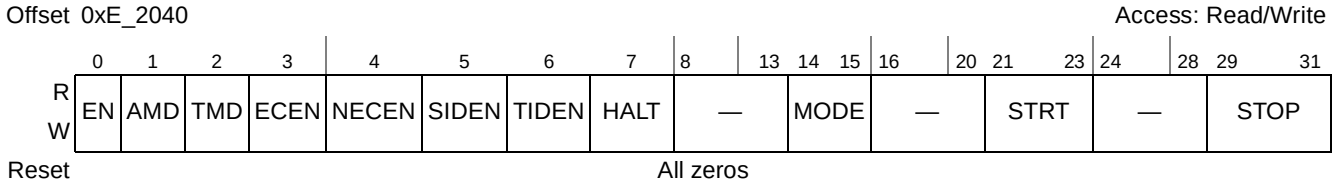


Figure 25-8. Trace Buffer Control Register 0 (TBCR0)

Table 25-14 describes the TBCR0 fields.

Table 25-14. TBCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 The trace buffer facility is disabled. 1 The trace buffer facility is enabled.
1	AMD	Address match disable 0 The address match is used to qualify a trace buffer event. 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 The transaction type match is used to qualify a trace buffer event. 1 The transaction type match is ignored when detecting a trace buffer event.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 25.3.3, “Context ID Registers.” 0 Current context match does not affect trace buffer event detection 1 Trace buffer events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 25.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect trace buffer event detection 1 trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Trace buffer events ignore the programmed source ID value. 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 Trace buffer events ignore the programmed TID event value. 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the ECM is selected for tracing (TBCR1[IFSEL] is all zeros).
7	HALT	Halt causes the trace buffer to stop tracing immediately. TBSR[ACT] remains set when this bit is set.
8–13	—	Reserved
14–15	MODE	Trace mode. Specifies one of two trace modes. 00 Trace every valid transaction 01 Reserved 10 Trace only cycles in which a trace event is detected. Note that if EN and other TBCR0 fields are not properly programmed to specify a traceable event, tracing occurs for every valid address. 11 Reserved
16–20	—	Reserved

Table 25-14. TBCR0 Field Descriptions (continued)

Bits	Name	Description
21–23	STRT	Start condition. Specifies the event that arms the trace buffer to start looking for the programmed event 000 No event. Armed immediately 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. Specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stops after it has been triggered at least once. 000 Buffer is full 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1 101 TRIG_IN transitions from 1 to 0 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID

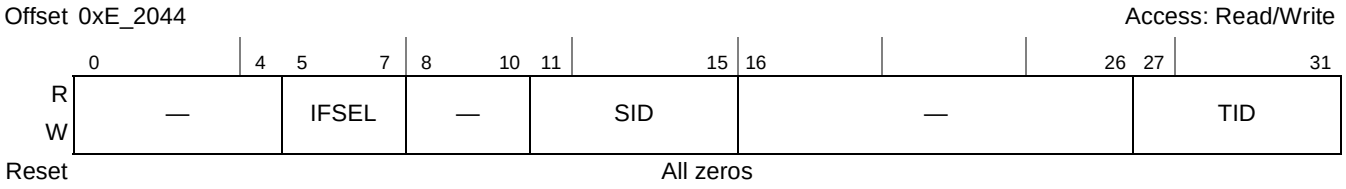


Figure 25-9. Trace Buffer Control Register 1 (TBCR1)

Table 25-15 describes the TBCR1 fields.

Table 25-15. TBCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Specifies the interface that sources information for both comparison/buffer control and buffer data capture. 000 Selects e500 coherency module (ECM) dispatch interface 001 Reserved 010 Reserved 011 Reserved 100 Selects internal PCI Express 1 outbound interface 101 Selects internal PCI Express 2 outbound interface 110 Selects internal PCI Express 3 outbound interface 111 Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with TBCR0[SIDEN]. The source ID is defined in Table 25-26.

Table 25-15. TBCR1 Field Descriptions

Bits	Name	Description
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in Table 25-26.

25.3.2.2 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR) shown in Figure 25-10 contains the address to match against (if TBCR0[AMD] is zero). This address may be further qualified by the mask bits defined in Section 25.3.2.3, "Trace Buffer Address Mask Register (TBAMR)."

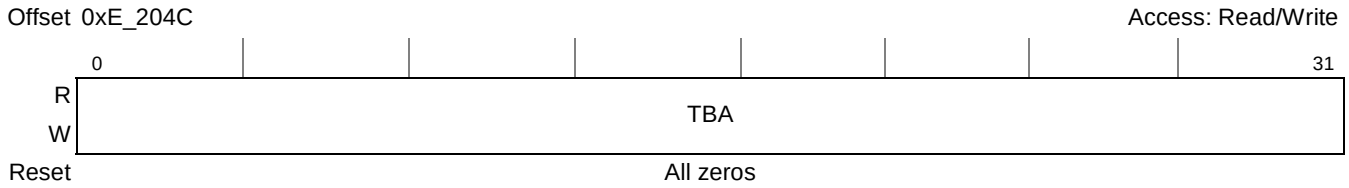


Figure 25-10. Trace Buffer Address Register (TBAR)

Table 25-16 describes the TBAR field.

Table 25-16. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address.

25.3.2.3 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR) shown in Figure 25-11 contains a mask for the TBAR, which allows excluding address bits from the comparison.

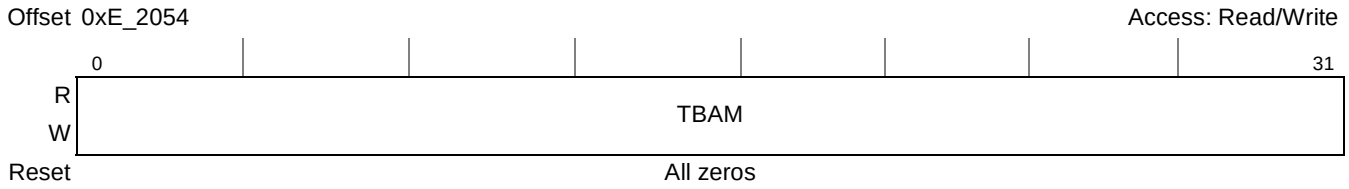


Figure 25-11. Trace Buffer Address Mask Register (TBAMR)

Table 25-17 describes the TBAMR field.

Table 25-17. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask.A value of zero masks the address comparison for the corresponding address bit. These bits only mask the address bits generated by the hardware, but do not affect the bits specified in TBAR. A bit that is masked from the comparison should be set to 0 in TBAR.

25.3.2.4 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR) shown in Figure 25-12 specifies which transaction types to monitor. Each bit in the TBTMR represents a transaction type on the selected interface. The transaction associated with any particular bit depends on the interface being monitored as specified by TBCR1[IFSEL]. Note that the transactions used for defining trace buffer events are the same as those defined for watchpoint monitor events. Table 25-12 defines the transaction types associated with each interface. Setting a bit enables a hit when this transaction is matched (provided all other match criteria are met and TBCR0[TMD] is clear).

Different interfaces support different transaction types, and the same bit may represent different transaction types depending on the interface.

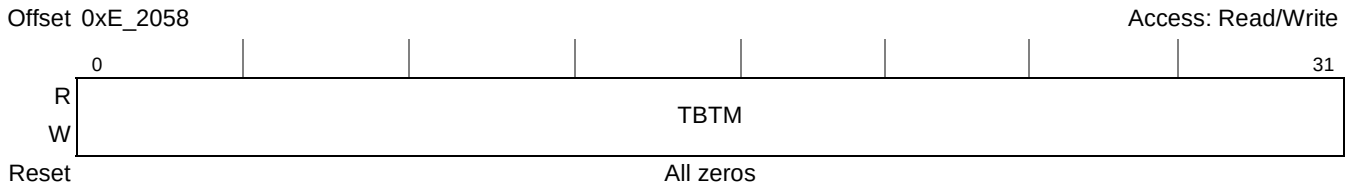


Figure 25-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 25-18 describes the TBTMR field.

Table 25-18. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 25-12. The transaction associated with a bit depends on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when TBCR0[TMD]=0.

25.3.2.5 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR) shown in Figure 25-13 indicates the operational state of the trace buffer.

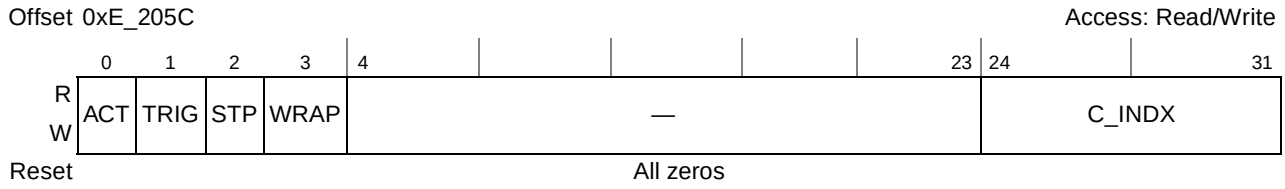


Figure 25-13. Trace Buffer Status Register (TBSR)

Table 25-19 describes the TBSR fields.

Table 25-19. TBSR Field Descriptions

Bits	Name	Description
0	ACT	Active. Indicates trace buffer activity. 0 The start triggering event has not yet occurred. Trace buffer is not armed. 1 The start triggering event has occurred. Trace buffer is armed.
1	TRIG	Triggered. Indicates whether or not a programmed event has been triggered. 0 The programmed event in TBCR0 has not yet been triggered. 1 The programmed event in TBCR0 has been triggered at least once.
2	STP	Stopped. Indicates whether or not a trace buffer stop condition has been detected. 0 No stop condition yet detected. 1 The trace buffer has detected a stop condition and is no longer capturing events.
3	WRAP	Wrapped. Indicates that the trace buffer write pointer has wrapped to the beginning of the buffer at least once. Set when the last entry of the trace buffer is written. 0 Pointer has not yet wrapped. 1 Pointer has wrapped to the beginning at least once.
4–23	—	Reserved
24–31	C_INDX	Current index. Represents the current value of the write pointer at the time TBSR was read. This value may be written by software to initialize the write pointer; however, software is not allowed to write the write pointer while the trace buffer is active. Writes are ignored while the trace buffer is active. It is recommended to write the status register before enabling the trace buffer in order to zero out any bits that might have been set during a prior run and to initialize the write pointer to zero.

25.3.2.6 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR) enables software to read or write the trace buffer. Each entry is 64 bits; therefore, it takes one write of TBACR and two reads of the access data register (TBADR and TBADHR) to read one 256-entry array entry. Similarly, it takes one write of TBACR and two writes of TBADR and TBADHR to write one array entry. Software can access any entry by writing the appropriate index into TBACR[INDX]. To read or write the buffer sequentially, starting with entry 0, the index must start with a value of 0 and increment every time a new entry is accessed.

TBACR is shown in Figure 25-14.

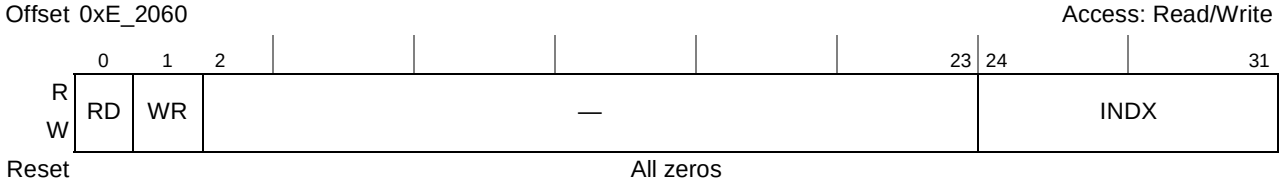


Figure 25-14. Trace Buffer Access Control Register (TBACR))

Table 25-20 describes the TBACR fields.

Table 25-20. TBACR Field Descriptions

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. This bit is automatically cleared when the read is performed.
1	WR	Write command. When set, a trace buffer write is performed using the value of TBACR[INDX]. This bit is automatically cleared when the write is performed. A write occurs only if the trace buffer is not active: write requests are ignored while the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer index to read from or write into (0–255). Used in conjunction with TBACR[RD] and TBACR[WR].

25.3.2.7 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 25-15, contains the high-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]), or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

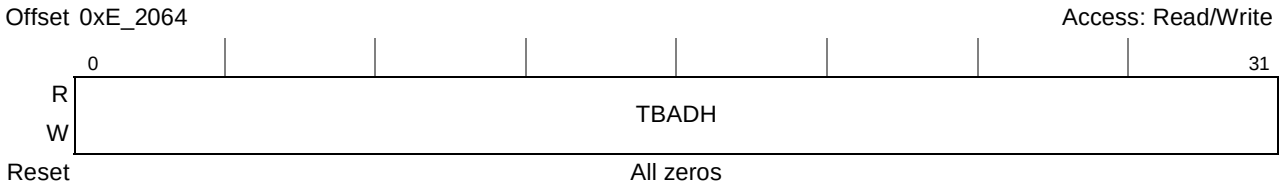


Figure 25-15. Trace Buffer Read High Register (TBADHR)

Table 25-21 describes TBADHR.

Table 25-21. TBADHR Field Descriptions

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. The higher 32 bits of the data read from or to be written into the trace buffer, depending on whether the array is accessed with a read or a write.

25.3.2.8 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 25-16, contains the low-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]) or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

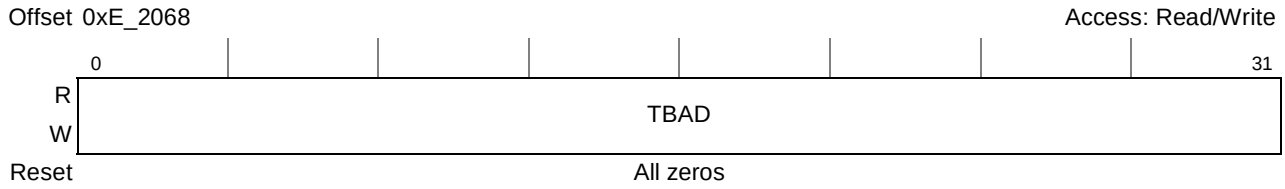


Figure 25-16. Trace Buffer Access Data Register (TBADR)

Table 25-22 describes the TBADR field.

Table 25-22. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. Corresponds to the lower 32 bits of the data read from the trace buffer or to be written into the trace buffer, depending on whether software is accessing the array with a read or a write.

25.3.3 Context ID Registers

This section describes the context ID registers. The current context ID register (CCIDR) and programmed context ID registers (PCIDR) are set by software and facilitate debugging complex software.

25.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 25-17, contains the user-programmed context ID. This register can be configured to trigger watchpoint events when its value matches the current context ID register (CCIDR), as controlled by WMCR0[ECEN] and WMCR0[NECEN]. See Section 25.3.1.1, “Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1),” for more information.

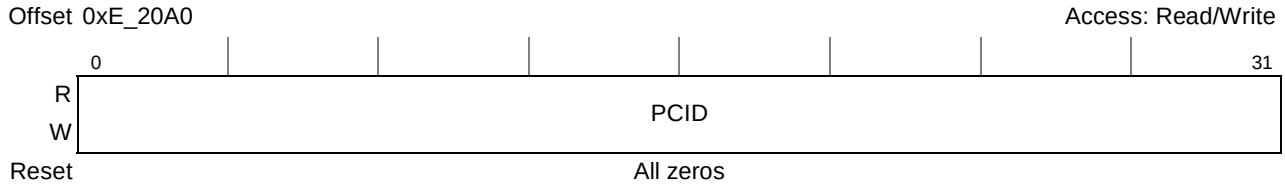


Figure 25-17. Programmed Context ID Register (PCIDR)

Table 25-23 describes the PCIDR field.

Table 25-23. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed context ID. Contains the user-programmed context ID. Compared with current context ID for context-sensitive event triggering

25.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR) shown in Figure 25-18 contains the current context ID. This register is written by software after a context switch and can be used to trigger events when compared with the programmed context ID register (PCIDR).

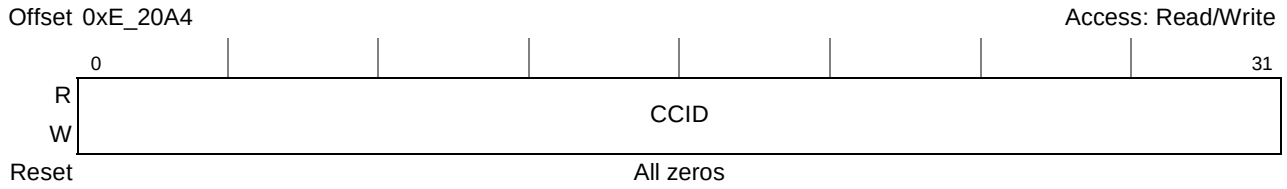


Figure 25-18. Current Context ID Register (CCIDR)

Table 25-24 describes the CCIDR field.

Table 25-24. CCIDR Field Descriptions

Bits	Name	Description
0–31	CCID	Current context ID. Set by user software. Typically loaded immediately following a context switch. Compared with user-programmed context ID for context-sensitive event triggering

25.3.4 Trigger Out Function

TRIG_OUT provides a convenient mechanism for triggering external system monitors and diagnostic equipment such as logic analyzers. Note that READY is multiplexed with TRIG_OUT. See the last paragraph of Section 4.4.2, “Power-On Reset Sequence,” for more information about READY functionality.

When the trace buffer hit is selected by TOSR[SEL], TRIG_OUT is only meaningful if the trace buffer control register 0 (TBCR0) is properly configured to hit on a traceable event. The same holds true for the watchpoint monitor when the watchpoint monitor is selected by TOSR[SEL].

25.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in Figure 25-19 specifies the source for TRIG_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor

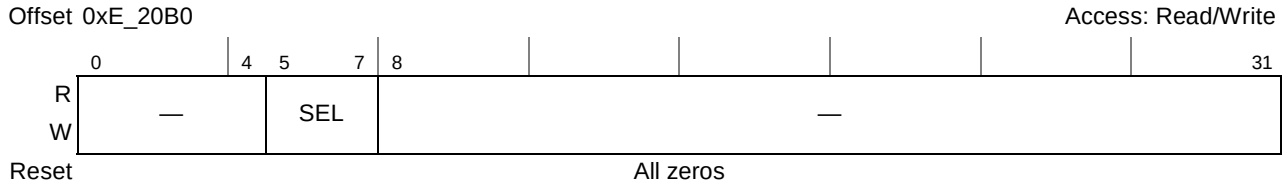


Figure 25-19. Trigger Out Source Register (TOSR)

Table 25-25 describes the TOSR fields.

Table 25-25. TOSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	SEL	Select. Selects the source for TRIG_OUT 000 READY signal. Multiplexed with TRIG_OUT. Basic device state indicator. READY asserts whenever the device is not in reset or not asleep. See Chapter 4, “Reset, Clocking, and Initialization,” for more details about the reset sequence, and Chapter 23, “Global Utilities,” for more information about power management states. 001 Selects the watchpoint monitor hit indication 010 Selects the trace buffer hit indication 011 Selects the performance monitor overflow indication
8–31	—	Reserved

25.4 Functional Description

The debug features on the MPC8572E use the LBC interfaces and the DDR SDRAM interfaces.

25.4.1 Source and Target ID

Debug information that is common to all the interfaces is the source ID (SID). The transaction source ID provides enough information to determine which block or port originated a transaction including the distinction between instruction and data fetches from the processor core. Table 25-26 shows the values and interpretation for the 5-bit SID field. Note that the table also includes ports that are only slaves, such as local memory. These ports are always targets. As such, the value shown represents a target ID (TID) and not a source ID. For ports that can function in both capacities, the value indicates source ID when mastering transactions, and target ID when responding as slave. The TID field is only meaningful when one of the following participates in the transaction:

- The e500 coherency module (ECM) dispatch bus
- The watchpoint monitor (WMCR1[IFSEL] = 000)
- The trace buffer (TBCR1[IFSEL] = 000)

Table 25-26. Source and Target ID Values

ID Value (Hex)	Transaction Source	Transaction Target
00	PCI Express 3	PCI Express 3
01	PCI Express 2	PCI Express 2
02	PCI Express 1	PCI Express 1
03	Reserved	Reserved
04	Pattern Matching Engine (PME)	Local Bus Controller
05	Reserved	Reserved
06	Reserved	Memory Controller 2

Table 25-26. Source and Target ID Values (continued)

ID Value (Hex)	Transaction Source	Transaction Target
07	Security	Reserved
08	Fast Ethernet maintenance interface (FEC)	Reserved
09	Reserved	Reserved
0A	Boot sequencer	Reserved
0B	Reserved	Interleaved Memories 1-2 or 1-4
0C	Serial RapidIO	Serial RapidIO
0D	Reserved	Reserved
0E	Table lookup unit 1 (TLU1)	Reserved
0F	Table lookup unit 2 (TLU2)	Memory Controller 1 / Local Address Space
10	Processor 1(instruction fetch)	Reserved
11	Processor 1(data fetch)	Reserved
12	Processor 2(instruction fetch)	Reserved
13	Processor 2(data fetch)	Reserved
14	Reserved	Reserved
15	Direct memory access controller 1 (DMA 1)	Reserved
16	Direct memory access controller 2 (DMA 2)	Reserved
17	System access port (SAP)	Reserved
18	Enhanced three-speed Ethernet controller 1 (eTSEC 1)	Reserved
19	Enhanced three-speed Ethernet controller 2 (eTSEC 2)	Reserved
1A	Enhanced three-speed Ethernet controller 3 (eTSEC 3)	Reserved
1B	Enhanced three-speed Ethernet controller 4 (eTSEC 4)	Reserved
1C	Serial RapidIO message unit	Reserved
1D	Serial RapidIO doorbell unit	Reserved
1E	Serial RapidIO port-write unit	Reserved
1F	Reserved	Reserved

25.4.2 DDR SDRAM Interface Debug

The DDR interfaces have two debug modes distinguished by which pins drive the debug information. In one mode, debug information (source ID, data valid) is multiplexed onto the ECC pins; the other mode uses the debug pins.

25.4.2.1 Debug Information on Debug Pins

If MSRCID0 is high and $\overline{\text{DMA1\_DDONE}}$[0] is low when sampled during POR, the debug information from the DDR SDRAM controller 1 is driven on MSRCID[0:4] and MDVAL. If both MSRCID0 and $\overline{\text{DMA1\_DDONE}}$[0] are high when sampled during POR, the debug information from the DDR SDRAM controller 2 is driven on MSRCID[0:4] and MDVAL. This POR value is captured in PORDBGMSR[MEM_SEL] as described in Section 23.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)." In this mode, the source ID appears on MSRCID[0:4] during a RAS or CAS cycle. During any other cycle, the value of MSRCID[0:4] is all ones, which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

25.4.2.2 Debug Information on ECC Pins

If MSRCID1 is low when sampled during POR, debug information from the DDR SDRAM interface is selected to appear on MECC[0:5] as shown in Figure 25-1. In this mode, the ID value of the source port, (the source ID), appears on MECC[0:4] during a RAS or CAS cycle. During any other cycle the value of MECC[0:4] is all ones. A data-valid signal (DVAL) is driven on MECC5 during valid DDR SDRAM data cycles.

NOTE

In this mode, MECC[0:5] must be disconnected from all SDRAM devices to prevent contention on those lines.

25.4.3 Local Bus Interface Debug

If both MSRCID0 and $\overline{\text{DMA1\_DDONE}}$[0] are low when sampled during POR, the LBC is selected as the source for the debug information appearing on MSRCID[0:4] and MDVAL. For more information on this mode, see Section 14.1.3.2, "Source ID Debug Mode."

25.4.4 Watchpoint Monitor

The watchpoint monitor (WM) can be programmed to arm and trigger on many different events including any of the following:

- External event (through TRIG_IN)
- A trace buffer event
- A performance monitor overflow event
- A comparison of the current and programmed context ID registers

A watchpoint event can be used in the following ways:

- Trigger a logic analyzer (using TRIG_OUT)
- Arm or trigger the trace buffer
- Trigger a performance monitor event

The large counters available in the performance monitor block and the interlock between it and the watchpoint monitor support sophisticated debug scenarios.

A WM trigger event may be composed of several events programmed in the watchpoint monitor control registers (WMCR0–WMCR1). Because the watchpoint monitor is disabled by default during POR, these registers must be initialized to make use of this debug feature. Note that the WM address mask register (WMAMR) and the type mask register (WMTMR) are cleared during POR. This means that the watchpoint monitor's default behavior following a power-on reset is to trigger on any address and no transaction type. The reset value of WMCR0[TMD] is 0 which means transaction matching is enabled but since no transaction is selected (WMTMR=0), a match will never occur. Either the transaction matching must be disabled by setting WMCR0[TMD] to a value of 1, or valid transactions must be selected by setting one or more of the WMTMR bits to a value of 1.

25.4.4.1 Watchpoint Monitor Performance Monitor Events

The WM can produce a performance monitor (PM) event with every trigger. This is accomplished by configuring the performance monitor to count WM events. For more information on this configuration see the events named 'Number of watchpoint monitor hits' and 'Number of trace buffer hits' in Table 24-10.

Multi-level triggers can be created using the watchpoint monitor, the performance monitor, and the trace buffer combined. For example, the WM can be programmed to trigger on events that also increment a PM counter (the performance monitor must also be programmed to respond to this event), the output of which (perfmon_overflow) could trigger the start of tracing in the trace buffer.

25.4.5 Trace Buffer

The trace buffer is a 256×64 array that can capture information about the internal processing of transactions to selected interfaces. The trace buffer controls are a superset of those for the watchpoint monitor. Close inspection of the trace buffer control registers (TBCR*n*) and the WM control registers (WMCR*n*) shows that trace buffer controls not needed for the WM are marked reserved in WMCR*n*. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer provides great flexibility about when to start tracing, when to stop tracing, and what to trace. The trace mode field, TBCR0[MODE], indicates when to trace: on every valid cycle, on a watchpoint monitor event, or when all the programmed events in the TBCR are met. This permits a user to program the trace condition in the watchpoint monitor and to program a start or stop condition in the trace buffer control register. The user can also program the TBCR with the conditions in which to stop tracing: on an event, or when the buffer is full. TBCR0[IFSEL] specifies which interface transactions are being captured.

The trace buffer can be programmed to trace the dispatch bus from any of the following:

- e500 coherency module (ECM)

Transactions come into the ECM, arbitrate for common resources, and get dispatched to the target port. Information such as transaction types, source ID, and other attributes can be captured in any of the selected interfaces.

25.4.5.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 25-20 shows the trace buffer entry format for an ECM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

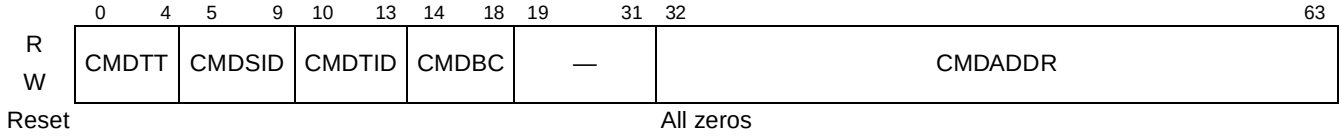


Figure 25-20. e500 Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 25-27 describes the fields of CMD trace buffer entries.

Table 25-27. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)

Bits	Name	Function
0–4	CMDTT	Transaction type. Specifies the transaction type as shown in Table 25-12. For example, a value of zero indicates a write with local processor snoop condition.
5–9	CMDSID	Source ID. Identifies the source of the transaction as shown in Table 25-26. For example, a value of 010101 indicates that DMA is the transaction source.
10–13	CMDTID	Target ID. Identifies the target of the transaction as shown in Table 25-26. For example, a value of 010101 indicates that DMA is the transaction target.
14–18	CMDBC	Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes. 00000 = 32 bytes 00001 = 1 byte 00010 = 2 bytes . . . 11110 = 30 bytes 11111 = 31 bytes
19–31	—	Reserved
32–63	CMDADDR	Address bits 0–31

Figure 25-21 shows the PCI Express trace buffer entry format when TBCR1[IFSEL] = 100 or 101 or 110.

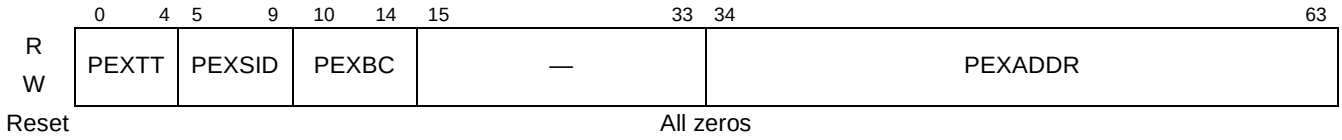


Figure 25-21. PCI Express Trace Buffer Entry

Table 25-28 describes the fields of PCI Express trace buffer entries when TBCR1[IFSEL] = 100 or 101 or 110.

Table 25-28. PCI Express Trace Buffer Entry Field Descriptions

Bits	Name	Function
0–4	PEXTT	Transaction type. Specifies the transaction type as shown in. For example, a value of all zeros maps to write.
5–9	PEXSID	Source ID. Identifies the source of the transaction as shown in Table 25-26. For example, a value of 010101 identifies DMA as the transaction source. For responses, this corresponds to Requestor's ID's bus number bits 3–7.
10–14	PCIBC	Byte count. The size of the transaction. 00000 4 bytes 00001 8 bytes 00010 12 bytes ... 11111 256 bytes
15–33	—	Reserved
34–63	PEXADDR	Address bits 31–2

25.5 Initialization

Configuring the appropriate control register must be the last step in the initialization sequence for either the watchpoint or trace buffer. That is, all required registers except the corresponding control register must be configured before any control register bits that enable watchpoint or trace events are set.

Appendix A
Revision History

This appendix provides a list of major differences among revisions of the *MPC8572E PowerQUICC™ III Integrated Host Processor Family Reference Manual*, as described in Section A.1, “Changes From Revision 1 (05/2008) to Revision 2 (05/2008),” and Section A.2, “Changes From Revision 0 (07/2007) to Revision 1 (05/2008).”

A.1 Changes From Revision 1 (05/2008) to Revision 2 (05/2008)

Major changes to the *MPC8572E PowerQUICC™ III Integrated Host Processor Family Reference Manual*, from Revision 1 to Revision 2, are as follows:

Section, Page	Changes
Chapter 14	Remove non-applicable sections/references (only visible in RM rev1) to the LTE signal and associated functionality.
15.7.1.9, 15-238	Corrected first row of “Additional SerDes setup as required” portion of Table 15-195, “SGMII Mode Register Initialization Steps,” as follows... Formerly read, “the Control Register (TBICON) is at offset address 0x00...” Now reads, “the control register (CR) is at offset address 0x00...”
15.7.1.9, 15-238	Corrected eleventh row of Table 15-195, “SGMII Mode Register Initialization Steps,” as follows... Formerly read, “Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.” Now reads, “Set up the MII Mgmt for a read cycle to TBI’s Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The control register (CR) is at offset address 0x00 from the TBI’s address.”
21.3.6.5, 21-37	Revised sections describing PEX_ERR_CAP_R0.
21.3.6.6, 21-38	Revised sections describing PEX_ERR_CAP_R1.
21.3.6.7, 21-40	Revised sections describing PEX_ERR_CAP_R2.
21.3.6.8, 21-42	Revised sections describing PEX_ERR_CAP_R3.

A.2 Changes From Revision 0 (07/2007) to Revision 1 (05/2008)

Major changes to the *MPC8572E PowerQUICC™ III Integrated Host Processor Family Reference Manual*, from Revision 0 to Revision 1, are as follows:

Section, Page	Changes
2.4, 2-17	In Table 2-1, “Memory Map,” remove PEX_LTSSM_STAT register row. In addition, update PIC BRR1 reset value to “0x0040_0301.”
3.1, 3-1	In MPC8572E Signal Grouping figures (Figure 3-1, Figure 3-2, and Figure 3-3), and Table 3-1, “Signal Reference By Functional Block,” change THERM[0:1] signal names to TEMP_ANODE and TEMP_CATHODE. In addition, change DMA controller 1 signals; only channels 0 and 1 are externally controllable. In addition, retask cfg_eng_use[0] POR configuration signal to become cfg_fcm_ecc.
3.1, 3-1	In Table 3-1, “MPC8572E Signal Reference By Functional Block,” remove references to D1_MDVAL, D1_MSRCID[0:4], D2_MDVAL, and D2_MSRCID[0:4].
3.1, 3-1	In Table 3-1, “MPC8572E Signal Reference by Functional Block,” change DMA controller 1 signals; only channels 0 and 1 are externally controllable.
3.1, 3-1	In Table 3-1, “MPC8572E Signal Reference by Functional Block,” retask cfg_eng_use[0] POR configuration signal to become cfg_fcm_ecc.
3.1, 3-1	In Table 3-1, “MPC8572E Signal Reference by Functional Block,” relocate UDE0 and UDE1 signal listing from PIC to Debug.
4.2.1, 4-2	In Table 4-2, “System Control Signals—Detailed Signal Descriptions,” change minimum assertion time for SRESET to correspond to Electrical Characteristics. In addition, clarify the asserted state meaning of $\overline{\text{HRESET\_REQ}}$
4.4.1.2, 4-10	In third paragraph, add cross reference to TCR[WRC] field regarding configuration of the e500 watchdog timer causing assertion of HRESET_REQ signal.
4.4.3.3, 4-14	In Table 4-11, “e500 Core0 Clock PLL Ratios,” make values 000, 001, and 010 Reserved.
4.4.3.19, 4-27	Modify second paragraph
4.4.3.24, 4-29	Add Section 4.4.3.24, “eLBC FCM ECC Configuration.”
4.4.3.27, 4-30	In Table 4-39, “Engineering Use POR Config Signal,” modifiy signal to reflect the retasking of MSRCID[0] to become cfg_fcm_ecc
5.2, 5-4	In Table 5-1, “Device Revision Level Cross-Reference,” update SVR values, as follows: 0x80E8_0011 for MPC8572E (with security), 0x80E0_0011 for MPC8572 (without security). In addition, update the core revision (3.0) and PVR (0x8021_0030).
5.3, 5-5	Modify Figure 5-3, “MU Pipeline, Showing Divide Bypass,” as follows:

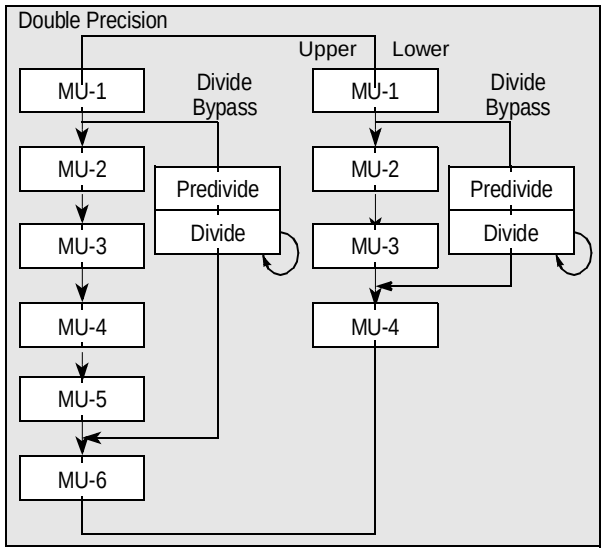


Figure 5-3. MU Pipeline, Showing Divide Bypass

5.3, 5-5	Add the following text to bulleted list below figure: "Four-cycle latency for multiplication, including SPE integer and fractional multiply instructions and embedded scalar and vector single-precision floating-point multiply instructions. Six-cycle latency for double-precision multiplication."
5.14, 5-31	In Table 5-8, "Differences Between the e500 Core and the PowerQUICC III Core Implementation," update descriptions for "HID1 Implementation" and "PIR value" features.
6.5.2, 6-13	In Table 6-7, "Processor ID Register (PIR)," clarify that reset value is 0x0 in core0 and 0x1 in core1.
6.5.4, 6-14	Update Table 6-8, "SVR Field Descriptions."
6.10.1, 6-25	Modify bit field descriptions in Table 6-18, "HID0 Field Descriptions."
6.10.2, 6-26	In Figure 6-33, "Hardware Implementation-Dependent Register 1 (HID1)," and Table 6-19, "HID1 Field Descriptions," add MID bit field (bits 60–63).
6.10.2, 6-26	In Table 6-19, "HID1 Field Descriptions," changed reset value for PLL_MODE from 01 to 11.
7.3, 7-8	In Table 7-3, "L2/SRAM Memory-Mapped Registers," in Figure 7-24, "L2 Error Address Capture Register (L2ERRADDRL)," and in Figure 7-25, "L2 Error Address Capture Register (L2ERRADDRH)," change offset of L2ERRADDRH to 0x20E5 and L2ERRADDRL to 0x20E50.
8.2.1.2, 8-4	In Table 8-3, "EEBPCR Field Descriptions," in CPU1_PRI bit field description (bits 26–27), change "00" to "10" for "Highest priority level." In addition, update CPU0_EN bit field description
Chapter 9	Change "D*n*_MAPAR_IN" to "D*n*_MAPAR_ERR" throughout.

9.3.1, 9-3	Under Table 9-1, “DDR Memory Interface Signal Summary,” add the following text: “Note that some devices implementing two DDR controllers may share one set of MDVAL and MSRCID[0:4] signals between them. Please refer to the signals and debug chapters for clarification on implementation.”
9.3.2.2, 9-11	In Table 9-4, “Clock Signals—Detailed Signal Descriptions,” update MCKE description
9.4, 9-11	In Table 9-5, “DDR Memory Controller Memory Map,” change reset value for DDR_IP_REV2 to 0x00*nn*_00*nn*.
9.4.1.2, 9-14	In Table 9-7, “CS*n*_CONFIG Field Descriptions,” modify sentence in ODT_RD_CFG and ODT_WR_CFG field descriptions, as follows: “ODT should only be used with DDR2 or DDR3.”
9.4.1.6, 9-21	In Table 9-11, “TIMING_CFG_1 Field Descriptions,” and in Table 9-12, “TIMING_CFG_2 Field Descriptions,” modify for the following bit field descriptions: WRREC, WRTORD, and RD_TO_PRE to add requirement for on-the-fly burst chop mode.
9.4.1.7, 9-23	In Figure 9-8, “DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2),” and Table 9-12, “TIMING_CFG_2 Field Descriptions,” change bits 4–8 CPO to Reserved.
9.4.1.9, 9-28	In Table 9-13, “DDR_SDRAM_CFG Field Descriptions,” add the following note to bit fields RD_EN and 2T_EN: “Note that RD_EN and 2T_EN must not both be set at the same time.”
9.4.1.9, 9-28	In Table 9-13, “DDR_SDRAM_CFG Field Descriptions,” add new programming requirement for HSE bit field description
9.4.1.14, 9-35	Change introduction to Figure 9-15, “DDR SDRAM Data Initialization Configuration Register (DDR_DATA_INIT),” to say the following: “The DDR SDRAM data initialization register, shown in Figure 9-15, provides the value that is used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.”
9.4.1.24, 9-46	Remove last sentence of second paragraph. In addition, in the list of 4-bit impedance settings in this section, place text “default full-strength impedance” next to the “1010” for DDR2 and the “0111” for DDR3; add note after the DDR3 list stating that “0000” should be used for the default half-strength value. In addition, modify the last paragraph (before DDRCDR_1 register figure) to say the following: “A value of 0000 should be used for default half-strength mode when driver calibration is not used. Note that the drivers may either be calibrated to full-strength or half-strength.”

9.4.1.27, 9-50 — In Figure 9-28, “DDR IP Block Revision 2 (DDR_IP_REV2),” change reset value to “0000_0000_*nnnn_nnnn*_0000_0000_*nnnn_nnnn*.”

9.4.1.33, 9-53 — In Figure 9-34, “Memory Data Path Read Capture ECC Register (CAPTURE_ECC),” and Table 9-49, “CAPTURE_ECC Field Descriptions,” extend bit field for ECE from 24–31 to 16–31; modify bit field description, as follows:

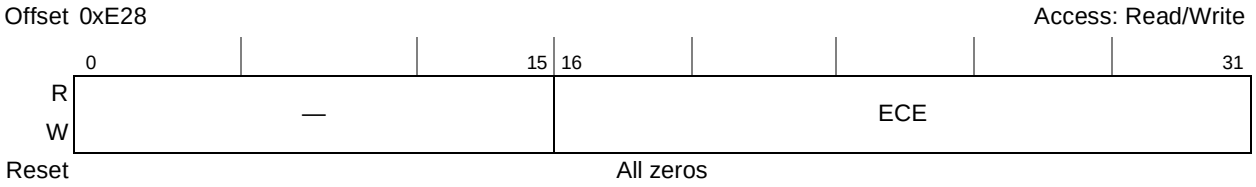


Figure 9-24. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-29. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 16:23—8-bit ECC code for 1st 32 bits 24:3—8-bit ECC code for 2nd 32 bits **Note:** In 64-bit mode, only 24:31 should be used, although 16:23 will show the 8-bit ECC code replicated.

9.4.1.37, 9-57 — In Table 9-43, “CAPTURE_ATTRIBUTES Field Descriptions,” add bit field description for TSIZ, as follows:

000 4 double words

001 1 double word

010 2 double words

011 3 double words

Others Reserved

9.5.3, 9-70 — Add sentence to 6th bullet, Mode register set: “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.”

9.5.6, 9-77 — Add the following note after the first paragraph:

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.

9.6, 9-86	Add the following in the Name column to Table 9-64, “Memory Interface Configuration Register Initialization Parameters”: TIMING_CFG_4, TIMING_CFG_5, DDR_ZQ_CNTL, DDR_WRLVL_CNTL, DDRCDR_1, and DDRCDR_2
9.6.1, 9-88	In Table 9-65, “Programming Differences Between Memory Types,” add “Differences” column. In addition, remove all references to DDR1.
9.6.1, 9-88	In Table 9-65, “Programming Differences Between Memory Types,” for ODT_PD_EXIT, change it to be set to 0001 for DDR1, and for FOUR_ACT, change it to be set for 00001 for DDR1.
Chapter 10	Remove all references to RCWL and RCWH.
10.3, 10-10	In Table 10-4, “PIC Register Address Map,” change BRR1 reset value to 0x0040_0301.
10.3.1.3, 10-21	In Figure 10-5, “Feature Reporting Register (FRR),” change reset value to 0x0040_0301.
10.3.1.6, 10-23	Add the following sentence to the first paragraph: “However, if one core is used to reset another one, the core being reset can effectively be held off indefinitely from issuing its initial boot vector fetch to the platform by leaving its appropriate PIR[Px] bit asserted. Clearing it releases the core to fetch.”
10.3.6.3, 10-40	In Figure 10-33, “Shared Message Signaled Interrupt Index Register (MSIIR),” and in Table 10-35, “MSIIR Field Descriptions,” change SRS field to bits 0–2 and IBS field to bits 3–7.
11.1.4.7, 11-12	Modify section
11.7.8.4, 11-162	In Figure 11-101, “RNGU Status Register,” change reset value from 0x003 to 0x01.
11.7.8.9, 11-166	Add the following note to end of section: **NOTE** Host reads of the RNGB FIFO should be performed on an 8-byte basis, regardless of how many bits of random number is actually required. Partial host reads can leave the RNGB FIFO in a state that results in a channel error.
Chapter 12	Change all instances of “E^2PROM” to “EEPROM.”
12.3.1.2, 12-6	In Figure 12-3, “I2C Frequency Divider Register (I2CFDR),” changed reset value to All zeros; in Table 12-5, “I2CFDR Field Descriptions,” revise FDR (bits 2–7) field description.
12.3.1.4, 12-9	In Table 12-7, “I2CSR Field Descriptions,” change I2CSR to I2CDR for the received case conditions in MCF bit field description.
12.3.1.5, 12-10	In Table 12-8, “I2CDR Field Descriptions,” modify last sentence of DATA bit field description to say, “Note that in both master receive and slave receive modes, the very first read is always a dummy read.”

12.4.5, 12-17	Modify first paragraph as follows: "If boot sequencer mode is selected on POR (by the settings on the cfg_boot_seq[0:1] reset configuration signals, as described in Section 4.4.3.8, "Boot Sequencer Configuration"), the I^2C1 module communicates with one or more EEPROMs through the I2C interface on IIC1_SCL and IIC1_SDA. The boot sequencer accesses the I^2C1 serial ROM device at a serial bit clock frequency equal to the platform (CCB) clock frequency divided by 2560. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device."
12.4.5, 12-17	Remove sentence, "For 1-byte transfers, a dummy read should be performed by the interrupt service routine."
Chapter 13	Replace "CCB clock" with "platform clock," throughout.
13.1.2, 13-2	Remove the DUART Signal Overview table.
Chapter 14	Remove section, "Multiplexed Address/Data Bus for 26-Bit Addressing." In addition, add overbar all active low signals $\overline{\text{LCS}n}$.
14.1.3, 14-3	Modify the section
14.2, 14-4	In Table 14-1, "Enhanced Local Bus Controller Detailed Signal Descriptions," change wording of LGPL*n* timing description to say its value is driven.
14.2, 14-4	In Table 14-1, "Enhanced Local Bus Controller Detailed Signal Descriptions," modify the signal description of LBCTL to say the following: "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 OR*n*[BCTLD]." In addition, modify LA signal addressing and change state meaning to the following: "LA is the address bus used to transmit addresses to external RAM devices. Refer to Section 14.5, "Initialization/Application Information," for address signal multiplexing."
14.3.1.1, 14-10	In Table 14-4, "BR*n* Field Descriptions," add the following sentence to DECC (bits 21–22) field description: "The reset value of this field is configured during power-on reset by the signal, cfg_fcm_ecc."
14.3.1.1, 14-10	In Table 14-4, "BR*n* Field Descriptions," clarify PS field description with respect to FCM capabilities, as follows: "10 16-bit (not supported for FCM) 11 32-bit (not supported for FCM)"
14.3.1.2, 14-12	Move Table 14-5, "Reset value of OR0 Register," to Section 14.3.1.2, "Option Registers (OR0–OR7)."
14.3.1.4, 14-21	In Table 14-11, "MxMR Field Descriptions," modify AM (bits 5–7) field description.
14.3.1.7, 14-24	Update second part of first paragraph, as follows:

	“To avoid race conditions between software and a busy eLBC, registers that affect currently running special operation and LSOR must not be re-written 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.”
14.3.1.9, 14-26	Change first sentence to say, “The transfer error status register (LTESR) indicates the cause of an error or event.”
14.3.1.15, 14-33	In Table 14-22, “LCRR Field Descriptions,” revise description for CLKDIV to say system clock is equivalent to ccb_clk and deleted statement about twice the csb_clk. In addition, modify PBYP (bit 0) field description
14.3.1.16, 14-34	In Table 14-22, “FMR Field Descriptions,” in CWTO (bits 16–19) field description, change “(CW0–CW3)” to “(CW0, CW1, RBW and RSW).”
14.4.2.1, 14-47	In Table 14-30, “GPCM Read Control Signal Timing,” update tARCS and tCSRP field so they match the RTL behavior.
14.4.2.2, 14-49	In Table 14-30, “GPCM Write Control Signal Timing,” update tAWCS, tCSWP and tWC so they match the RTL behavior.
14.4.2.3.2, 14-51	Modify sentence in first number of numbered list, as follows: “OR*n*[CSNT], along with OR*n*[TRLX], control the timing for the appropriate strobe negation in write cycles.”
14.4.3.4.2, 14-71	Revise steps 3 and 4 in the numbered list.
14.4.4.4.1, 14-78	In Figure 14-65, “RAM Word Field Descriptions,” add the following note to LOOP and AMX bit field descriptions: “AMX must not change values in any RAM word which begins a loop.”
14.4.4.4.5, 14-83	Add the following to the bulleted list: “Loop start word should not have an AMX change with regard to the previous word.”
14.4.4.4.7, 14-84	Modify the note to say: “AMX must not change values in any RAM word which begins a loop.”
14.4.4.4.7, 14-84	Modify last two sentences of first paragraph to say the following: “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.”
14.4.4.5, 14-87	Add the following conditions to the end of the second paragraph: “The conditions are as follows: • The PLL must be enabled, that is, LCRR[PBYP] = 0. • DLT3 bit must be cleared in the same RAM word to avoid mid-sampling of read data. • LBCR[LPBSE] = 0 and MXMR[GPL4] = 1

	• The combination WAEN=1 and UTA=1 should be in the RAM word next to the word which gets frozen by LUPWAIT assertion. This condition limits the use of this mode to cases where the exact cycle of LUPWAIT assertion is predictable.”
14.5.1, 14-88	Remove the following text: “The local bus can be used either with separate address and data buses or with a multiplexed address/data bus. A full 32-bit address or the lower 26 bits of the address can be provided according to the size of memory devices and pin count limitations.”
14.5.1.3, 14-89	Change the last sentence in the first paragraph below Figure 14-71, “GPCM Address Timings,” to say the following: “Typical values for the two propagation delays are in the order of 3–6 ns, so for a 133-MHz bus frequency, LCS should arrive on the order of 3 bus clocks later.”
14.5.4.5, 14-95	Add the following text to the beginning of the second paragraph: “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.”
14.5.4.6, 14-96	Add the following text to the beginning of the second paragraph: “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.”
14.5.8, 14-106	Remove Section 14.5.8, “Interfacing to DSP Host Ports.”
15.3, 15-5	Modify first bullet’s content under “Ethernet and FIFO operation”
15.4, 15-6	In Table 15-1, “eTSEC*n* Network Interface Signal Properties,” and in Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” change “EC2_MDC” and “EC2_MDIO” to “EC3_MDC” and “EC3_MDIO.” In addition, add MII to designated modes during which TXD[7:4] and RXD[7:4] are unused. In addition, for RGMII and RTBI protocols, change description of TSEC_RX_ER from “Unused, output driven low” to “Unused.” In addition, remove references to TSEC_1588_PP3 signal and to TMR_TEMASK[PP3EN].
15.4.1, 15-9	In Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” change the State Meaning description for TSEC*n*_CRS
15.4.1, 15-9	In Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” remove the following sentence from TSEC*n*_GTX_CLK description: “In RMII mode, TSEC*n*_GTX_CLK feeds back the effective transmit clock according to the interface, 100Base-T is 25 MHz and 10Base-T is 2.5 MHz.”
15.4.1, 15-9	In Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” modify TSEC*n*_RX_CLK signal description

15.4.1, 15-9	In Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” modify TSEC*n*_TX_CLK signal description
15.4.1, 15-9	In Table 15-2, “eTSEC Signals—Detailed Signal Descriptions,” change TSEC*n*_TX_EN signal description from “rising and falling edges of the TSEC*n*_TX_CLK, respectively” to “rising and falling edges of the TSEC*n*_GTX_CLK, respectively.”
15.5.3.1.3, 15-28	In Figure 15-4, “IEVENT Register Definition,” and in Table 15-7, “IEVENT Field Descriptions,” add FGPI (bit 27) field.
15.5.3.1.4, 15-32	In Figure 15-5, “IMASK Register Definition,” and in Table 15-8, “IMASK Field Descriptions,” add FGPIEN (bit 27) field.
15.5.3.2.1, 15-41	In Table 15-15, “TCTRL Field Descriptions,” modify TFC_PAUSE (bit 28) field description
15.5.3.2.2, 15-42	Modify section
15.5.3.2.4, 15-47	In Table 15-18, “TXIC Field Descriptions,” add the following footnote to ICCS bit field description : “The term ‘system clock’ refers to CCB clock/2.”
15.5.3.3.1, 15-54	In Figure 15-24, “RCTRL Register Definition” and Table 15-28, “RCTRL Field Descriptions,” add PRSFM (bit 26) field
15.5.3.3.1, 15-54	In Table 15-28, “RCTRL Field Descriptions,” modify bit field description for TS (bit 7)”
15.5.3.3.1, 15-54	In Table 15-28, “RCTRL Field Descriptions,” modify RSF (bit 29) field description
15.5.3.3.3, 15-59	In Table 15-30 “RXIC Field Descriptions,” add the following footnote to ICCS bit field description : “The term ‘system clock’ refers to CCB clock/2.”
15.5.3.3.5, 15-61	In Table 15-32, “RBIFX Field Descriptions,” modify [BnCTL]=01 field descriptions to clarify that arbitrary extraction of preamble is not supported in FIFO modes
15.5.3.3.7, 15-64	In Figure 15-30, “Receive Queue Filer Table Control Register Definition,” and in Table 15-34, “RQFCR Field Descriptions,” add GPI (bit 0) field.
15.5.3.3.8, 15-65	In Table 15-35, “RQFPR Field Descriptions,” add the following text to IPF (bit 20) field description: “See the descriptions of receive FCB fields IP and PRO in Section 18.6.4.3, “Receive Path Off-Load,” for more information on determining the status of received packets for which IPF is set.”
15.5.3.3.8, 15-65	In Table 15-35, “RQFPR Field Descriptions,” ETY (bits 16–31) field description
15.5.3.5.2, 15-76	In Table 15-43, “MACCFG2 Field Descriptions,” modify Huge Frame (bit 26) bit field description and MPEN (bit 28) bit field description; clarify that TX preamble length may be from 0x3 to 0xF.
15.5.3.6.26, 15-101	In Table 15-83, “TPKT Field Descriptions,” change TPKT size to bits 10–31.
15.5.3.8.1, 15-117	In Table 15-108, “FIFOCFG Field Descriptions,” modify IPG (bits 8–15) field description, as follows:

	"Minimum inter packet gap. This sets the minimum number of cycles inserted between back-to-back frames transmitted over the FIFO interface. The minimum required is 3 cycles if CRCAPP=0, 5 cycles for 16-bit interfaces if CRCAPP=1 and 7 cycles for 8-bit interfaces if CRCAPP=1."
15.5.3.9.2, 15-120	In Figure 15-107, "ATTRELI Register Definition," modify the size of EI (bits 18–25), and in Table 15-110, "ATTRELI Field Descriptions," modify EI bit field description, as follows: "Extracted index. Points to the first byte, as a multiple of 64 bytes, within the receive frame from which to begin extracting data."
15.5.3.10.1, 15-121	Modified first paragraph
15.5.3.11, 15-122	Modify section
15.5.3.11.1, 15-122	In Table 15-113, "TMR_CTRL Register Field Descriptions," clarifiy description of CIPH (bit 25) field description In addition, modify ETEP2 field description to say "External trigger 2 edge polarity" (was "External trigger 1 edge polarity"), and ETEP1 field description to say "External trigger 1 edge polarity" (was "External trigger 2 edge polarity").
15.5.3.11.1, 15-122	In Table 15-113, "TMR_CTRL Register Field Descriptions," add the following note to TMSR (bit 26) bit field description: "**Note:** Prior to initiating timer reset (setting TMSR), must gracefully stop receiver (See MACCFG1[RX_EN] description). User programmable registers are not reset by the soft reset; for example, TMR_CTRL, TMR_TEMASK, TMR_PEMASK, TMR_ADD, TMR_PRSC, TMROFF_H/L, TMR_ALARM*n*, and TMR_FIPER*n*."
15.5.3.11.3, 15-125	Remove references to TSEC_1588_PP3 signal and to TMR_TEMASK[PP3EN].
15.5.3.11.11, 15-131	Add the following note: **NOTE** All TMROFF_H registers in a device should be set to the same value, and all TMROFF_L registers in a device should be set to the same value. Otherwise, the precision time protocol may not work. In addition, remove the following sentence: "This allows each Ethernet channel to have a different timebase offset from the central TMR_CNT_H/L."
15.5.3.11.12, 15-131	In Table 15-126, "TMR_ALARM*n*_H/L Register Field Descriptions," modify the first paragraph of ALARM_H/L bit field description
15.5.4.3.10, 15-145	In Table 15-140, "TBICON Field Descriptions," modify Clock Select (bit 10) field description
15.6.1.8, 15-157	Add Section 15.6.1.8, "SGMII Interface"
15.6.2, 15-157	Modify last bulleted item in section as follows:

	"On transmission, the minimum inter-packet gap (set in FIFOCFG[IPG]) is three cycles if CRC is not automatically appended. Each CRC data beat adds to this requirement. For 16-bit FIFO interfaces the minimum Tx IPG is 5 cycles and for 8-bit FIFO interfaces the minimum is 7 cycles."
15.6.3.2, 15-166	Change step 3 to say the following: "Set SOFT_RESET bit in MACCFG1 register (Note that SOFT_RESET must remain set for at least 3 TX clocks before proceeding.)"
15.6.3.8, 15-175	Add sentence clarifying that sleep mode is not supported for Magic Packet in the first paragraph.
15.6.3.8, 15-175	Modify last sentence, as follows: "Only frames addressed specifically to the MAC's station address or a valid multicast or broadcast address can be examined for the Magic Packet sequence."
15.6.3.11, 15-179	Revise section
15.6.3.13, 15-179	In Table 15-160, "Reception Errors," remove the following note: **"Note:** Any values in the length/type field between 1500 and 1536 will be treated as a length, however, only illegal packets exist with this length/type since these are not valid lengths and not valid types. These are treated by the MAC logic as out of range. Software must confirm the parser and filer results by checking the type/length field after the packet has been written to memory to see if it falls in this range."
15.6.4.2, 15-182	In Table 15-161, "Tx Frame Control Block Description," add the following sentence to PTP (bit 15) field description: "Must be clear if TMR_CTRL[TE] is clear."
15.6.4.3, 15-184	In Table 15-162, "Rx Frame Control Block Descriptions," add the following text to IP bit field descriptions (bit 1): "If S/W is relying on the RxFCB for the parse results, any RxFCB[IP] bits set with the corresponding RxFCB[PRO] = 0xFF indicates a fragmented packet (or that this packet had a back-to-back IPv6 routing extension header). Additionally, RQFPR[IPF] (see ") indicates that the packet was fragmented."
15.6.4.3, 15-184	In Table 15-162, "Rx Frame Control Block Descriptions," PRO bit field description (bits 8–15)
15.6.4.3, 15-184	In Table 15-162, "Rx Frame Control Block Descriptions," replace "Layer 4 protocol identified according to IANA specification. Valid only if IP = 1," in PRO bit field description with the following: If IP = 1, PRO is set as follows: • PRO=0xFF for a fragment header or a back to back route header • PRO=0x*nn* for an unrecognized header, where nn is the next protocol field • PRO=(TCP/UDP header), as defined in the IANA specification, if TCP or UDP header is found If IP = 0, PRO is undefined."

15.6.5.1.1, 15-187	Add the following statement to the bulleted list: “The GPI field offers the user the ability to interrupt the core upon matching a rule that causes a frame to be filed to memory. Once the last RxBD corresponding to that frame is written to memory, the IEVENT[FGPI] event will be asserted. This bit will be set regardless of any interrupt coalescing that may be set.”
15.6.5.1.4, 15-189	Add the following paragraphs to end of section: “A functional interrupt is provided via use of the general purpose interrupt (GPI) bit in the filer table. When a property matches the value in the RQPROP entry at this index, and REJ = 0 and AND = 0, the filer will set IEVENT[FGPI] when the corresponding receive frame is written to memory. This allows the user to set up a filer rule where the core will be interrupted upon the reception of ‘special’ frames. If the timer is enabled (TMR_CTRL[TE] = 1), then the interrupt dedicated for timer events (in addition to the usual recieve, transmit and error interrupts) will be asserted.”
15.6.6.2.1, 15-196	Modify the last sentence of the note at the end of the section to say: “As soon as the hardware consumes a BD (by writing it back to memory), RBPTR*n* advances and the free BD count reflects the correct number of available free BDs.”
15.6.7.4.1, 15-189	Remove Section 15.6.7.4.1, “Rx FCB Change.”
15.6.7.4.1, 15-200	Modify section Section 15.6.5.1.1, “Filing Rules.”
15.6.8.1, 15-201	Replace sentence in first paragraph with the following: “Because of pre-fetching, a minimum of four buffer descriptors per ring are required. This applies to both the transmit and the receive descriptor rings.” removed statements about supporting 1588 on a per-eTSEC basis
15.6.8.2, 15-202	In Table 15-169, “Transmit Data Buffer Descriptor (TxBD) Field Descriptions,” modify TR bit field description.
15.6.8.3, 15-205	In Table 15-170, “Receive Buffer Descriptor Field Descriptions,” add the following sentence to Rx Data Buffer Pointer (bits 0–31) field description: “For best performance, use 64-byte aligned receive buffer pointer addresses. The buffer must reside in memory external to the eTSEC.”
15.6.8.3, 15-205	In Table 15-170, Receive Buffer Descriptor Field Descriptions,” modify bit field descriptions for TR and LG.
15.7.1.7, 15-232	Remove references to TSEC4_RX_ER.
15.7.1.7, 15-232	In Table 15-189, “8-Bit FIFO Interface Mode Signal Configurations, eTSEC1/2,” in Table 15-190, “8-Bit FIFO Interface Mode Signal Configurations, eTSEC3/4,” and in Table 15-192, “16-Bit FIFO Interface Mode Signal Configuration (eTSECs1 and 2),” remove support for 3.3-V FIFO interface.

15.7.1.9, 15-238	Add Table 15-194, “SGMII Interface Signal Configuration (4-Wire),” to Section 15.7.1.9, “SGMII Interface Support.”
15.7.1.9, 15-238	In Table 15-195, “SGMII Mode Register Initialization Steps,” remove references to TBICON[Enable Wrap] and TBICON[Comma Detect].
15.7.1.9, 15-238	In Table 15-195, “SGMII Mode Register Initialization Steps,” modify row to say the following: “Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBICON register, MIIMCON[0000_0000_0000_0000_0000_0000_0010_0000] This sets TBI in single clock mode and MII Mode off to enable communication with SerDes.”
Chapter 16	Replace “EC3_MDC” and “EC3_MDIO” with “EC5_MDC” and “EC5_MDIO” throughout.
16.4, 16-6	In second paragraph, change the signal count from “1” to “18.”
17.3.7.4, 17-55	In Table 17-59, “MIA_CR Field Descriptions,” update the bit field descriptions.
17.3.7.5, 17-56	In Table 17-60, “ MIA_ACR Field Descriptions,” update the bit field descriptions.
17.4.5.6, 17-174	Add Section 17.4.5.6, “Error Handling” “
19.2.1, 19-5	Add the following note to Figure 19-3, “DMA Signal Summary”: “Note that this device does not support external control for every DMA channel.”
19.2.2, 19-5	In Table 19-3, “DMA Signals—Detailed Signal Descriptions,” clarify offering of external DMA signals.
19.3.1.1, 19-9	In Table 19-5, “MR*n* Field Descriptions,” remove the following sentence for bit field 4–7 BWC: “Defined only if MR*n*[EMP_EN] is set.”
19.3.1.1, 19-9	In Table 19-5, “MR*n* Field Descriptions,” modify CS (bit 31) field description
19.3.1.4, 19-15	Remove references to ATMU bypass mode for RapidIO.
19.3.1.6, 19-17	Remove references to ATMU bypass mode for RapidIO.
19.4.1.3, 19-31	Modify third paragraph, as follows: “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.”
19.4.1.4, 19-32	Modify the first bullet in the bulleted list, as follows: • $\overline{\text{DMA\_DREQ}}$ - Asserting edge triggers a DMA transfer start or restart from a pause request. Sets MR*n*[CS]. (Note that negating $\overline{\text{DMA\_DREQ}}$ does NOT clear MR*n*[CS].) Modify second paragraph
19.4.3, 19-35	Remove references to ATMU bypass mode for RapidIO.
19.4.5, 19-38	Remove references to ATMU bypass mode for RapidIO.
20.6.1.2, 20-11	In Table 20-2, “DICAR Field Description,” update DR field description

20.6.2.4, 20-23	In Table 20-19, “GCCSR Field Descriptions,” add the following parenthetical note to bit fields H, M, and D: “(Note that although this status bit is R/W, manually changing its value does not affect logical operation.)”
20.6.4.3, 20-37	In Table 20-33, “ECACSR Field Descriptions,” modify ECI (bits 8–23) field description
20.6.5.9, 20-47	In Table 20-48, “Physical Configuration Register (PCR), make bit 27 CCP read/write.
20.6.7.9, 20-59	In Table 20-62, “RIWAR*n* Field Descriptions,” modify TGINT bit field description to make 0011–1110 Reserved.
20.7.1.7, 20-69	In Table 20-72, “OM*n*DATR Field Descriptions,” references to “OM*n*MR[MM]” should instead reference “OM*n*DATR[MM].”
20.7.1.10, 20-71	References to “OM*n*MR[MM]” should instead reference “OM*n*DATR[MM].”
20.9.4.1, 20-141	References to “OM*n*MR[MM]” should instead reference “OM*n*DATR[MM].”
20.9.4.2.7, 20-153	References to “OM*n*MR[MM]” should instead reference “OM*n*DATR[MM].”
20.10.2.1.7, 20-171	In Table 20-130, “Outbound Doorbell Hardware Errors,” replace “ODMR[EIE]” with “OM*n*MR[EIE].”
21.1.1.1, 21-2	Add the following text to the end of the section: “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.”
21.3.2.2, 21-11	Add the following text to the end of the first paragraph: “Also note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 21.4.1.2.1, “Byte Order for Configuration Transactions,” for more information.”
21.3.2.3, 21-11	In Table 21-6, “PEX_OTB_CPL_TOR Field Descriptions,” modify TC field description
21.3.2.4, 21-12	In Table 21-7, “PEX_CONF_RTY_TOR Field Descriptions,” modify TC field description.
21.3.2.5, 21-13	In Figure 21-6, “PCI Express Configuration Register (PEX_CONFIG),” and Table 21-8, “PEX_CONFIG Field Descriptions,” change IPOL (bit 16) to Reserved.
21.3.5.2.6, 21-28	In Table 21-22, “PCI Express Inbound Window Attributes Registers Field Descriptions,” modify TRGT bit field description to make 1101–1110 Reserved.
21.3.6.1, 21-30	In Table 21-23, “PCI Express Error Detect Register Field Descriptions,” add the following sentence to PCT (bit 8) field description: “Hot reset is recommended to restore stability of the system.”
21.3.6.4, 21-36	In Table 21-26, “PCI Express Error Capture Status Register Field Descriptions,” update GSID field description

21.3.6.4, 21-36	Revise register descriptions for PEX_ERR_CAP_R*n* registers, Figure 21-28, Figure 21-29, and Figure 21-30.
21.3.6.6, 21-38	For all applicable registers, change descriptions for OD0, OD1, and OD2 to say: “Internal platform transaction information. Reserved for factory debug.”
21.3.7.1, 21-43	Add the following text to the end of the section: “Note that 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.”
21.3.7.1, 21-43	Add the following text to the end of the first paragraph: “Note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 21.4.1.2.1, “Byte Order for Configuration Transactions,” for more information.’
21.3.9.11, 21-75	In Table 21-86, “PCI Express Link Control Register Field Description ,” modify RL (bit 5) field description
21.3.10, 21-83	Add the following footnote to Figure 21-101, “PCI Express Extended Configuration Space”: “Note that the PCI Express Controller Internal CSRs are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers will return all 0s.” In addition, change range for Internal CSR space in to 0x400–0x6FF.
21.3.10.2, 21-84	In Table 21-99, “PCI Express Uncorrectable Error Status Register Field Description,” add the following sentence to CTO (bit 14) field description: “Hot reset is recommended to restore stability of the system.”
21.3.10.18, 21-96	Add the following sentence: Note that the state of PEX_CFG_READY[CFG_READY] is dependent upon the POR configuration settings described in Section 4.4.3.5, “Host/Agent Configuration” and Section 4.4.3.7, “CPU Boot Configuration”
21.4.1.2, 21-100	Replace section “Outbound Byte Swapping,” section “Inbound Byte Swapping,” and section “PEX_CONFIG_DATA Byte Swapping,” with Section 21.4.1.2, “Byte Ordering.”
21.4.1.9.1, 21-104	Remove references to data invariance.
21.4.1.9.1, 21-104	In Table 21-122, “PCI Express ATMU Outbound Messages,” change entries for Assert INTA, INTB, INTC, INTD, and Deassert INTA, INTB, INTC, INTD messages to be supported in EP column; add “Sent upstream by endpoint” to description column.
21.4.1.10.3, 21-112	In Table Table 21-126, “Error Conditions,” modify entry for Outbound response/Internal platform response (last row in table) with the following error:

	"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."
21.4.2.1.2, 21-114	Revise entire section.
21.4.2.1.4, 21-114	Revise section, as follows: "Software can generate outbound assert or deassert INTx message transactions by using the outbound ATMU mechanism described in Section 21.4.1.9.1, "Outbound ATMU Message Generation.'"
21.4.6, 21-117	Add new section, "Link Down."
23.4.1.1, 23-4	In Table 23-4, "PORPLLSR Field Descriptions," modify field descriptions for e500_1_Ratio (bits 2–7) and e500_0_Ratio (bits 10–15). In addition, remove 1:1 as a valid core/platform frequency combination.
23.4.1.6, 23-12	In Table 23-9, "PORDEVSR2 Field Descriptions," modify SRDS_SGMII_REFCLK bit field descriptionSection 4.4.3.12, "SGMII SerDes Reference Clock Configuration
23.4.1.6, 23-12	In Figure 23-6, "POR Device Status Register 2 (PORDEVSR2), and in Table 23-9, "PORDEVSR2 Field Descriptions," add FCM_ECC bit field (bit 26). Section 4.4.3.24, "eLBC FCM ECC Configuration
23.4.1.10, 23-16	In Table 23-13, "POWMGTCSR Field Descriptions," add the following note to SLP (bit 14) field description: "Note that Sleep mode must not be requested while either core is in boot hold-off mode."
23.4.1.12, 23-20	In Table 23-15, "RSTRSCR Field Descriptions, modify WDT0_RR (bit 11) and WDT1_RR (bit 13) field descriptions
23.4.1.19, 23-24	Change introduction to the DDRCLKDR register to say the following: "Shown in Figure 23-19, the DDRCLKDR contains bits that allow disabling the clocks of the DDR SDRAM controller."
23.4.1.21, 23-27	In Table 23-24, "SRDS1CR Field Descriptions," modify state definitions (swapped 0 and 1 descriptions) for XMITEQAD and XMITEQEH.
23.4.1.22, 23-28	Add description for the SDRS2CR, as follows: "Shown in Figure 23-22, the SRDS2CR contains the control bits used for adjusting the transmit equalization of the SerDes 2 signals. Note that reserved fields with non-zero reset values are for internal use only and their values should be preserved."
23.5.1.9, 23-37	Modify second paragraph to say the following: "The MPC8572E does not perform dynamic bus snooping as described in the *e500 Reference Manual*. That is, when the e500 core is in the core-stopped state the core is not awakened to perform snoops on global transactions. Therefore, before entering nap mode, the L1 caches should be flushed if coherency is required during this power-down mode."
24.4.7, 24-16	Update Table 24-10, "Performance Monitor Events."

25.1.3, 25-3	In Table 25-1, “POR Configuration Settings and Debug Modes,” for POR Value of 10 and 11, replace the descriptions with the following sentences, respectively: “10 Debug information from the DDR SDRAM controller 1 appears on MSRCID[0:4] and MDVAL.” “11 Debug information from the DDR SDRAM controller 2 appears on MSRCID[0:4] and MDVAL.”
25.1.3.1, 25-4	Rename section, formerly “Local Bus (LBC) Debug Mode” to “Memory Debug Mode (Local Bus and DDR”; change section to properly reference $\overline{\text{DMA2\_DACK[0]}}$ instead of MSRCID0.
25.1.3.2, 25-4	Change section to properly reference $\overline{\text{DMA2\_DDONE[0]}}$ instead of MSRCID1.
25.2.1, 25-6	In Table 25-2, “Debug, Watchpoint and Test Signal Summary,” and Table 25-5, “JTAG Test and Other Signals—Detailed Signal Descriptions,” change THERM[0:1] signal names to TEMP_ANODE and TEMP_CATHODE.
25.4.1, 25-25	In Table 25-26, “Source and Target ID Values,” modify the source and target ID table to have three distinct columns: ID Value (Hex), Transaction Source, and Transaction Target.
25.4.5, 25-28	In introduction sentences for Figure 25-21, “PCI Express Trace Buffer Entry,” and Table 25-28, “PCI Express Trace Buffer Entry Field Descriptions,” append the following sentence fragment to each: “when TBCR1[IFSEL] = 100 or 101 or 110.”

Glossary

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this reference manual.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Atomic access. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The Power Architecture technology implements atomic accesses through the **lwarx**/**stwcx.** instruction pair.

Autobaud. The process of determining a serial data rate by timing the width of a single bit.

B

Beat. A single state on the bus interface that may extend across multiple bus cycles. A transaction can be composed of multiple address or data *beats*.

Big-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *most-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the *most-significant byte*. See *Little-endian*.

Boundedly undefined. A characteristic of certain operation results that are not rigidly prescribed by the Power Architecture technology. Boundedly-undefined results for a given operation may vary among implementations and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be boundedly undefined are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

Breakpoint. A programmable event that forces the core to take a breakpoint exception.

Burst. A multiple-beat data transfer whose total size is typically equal to a cache block.

Bus clock. Clock that causes the bus state transitions.

Bus master. The owner of the address or data bus; the device that initiates or requests the transaction.

C

Cache. High-speed memory containing recently accessed data or instructions (subset of main memory).

Cache block. A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache block may vary among processors; the maximum block size is one *page*. In Power Architecture processors, *cache coherency* is maintained on a cache-block basis. Note that the term ‘cache block’ is often used interchangeably with ‘cache line.’

Cache coherency. An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor’s cache.

Cache flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

Cast out. A *cache block* that must be written to memory when a cache miss causes a cache block to be replaced.

Changed bit. One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. See also *Page access history bits* and *Referenced bit*.

Clean. An operation that causes a cache block to be written to memory, if modified, and then left in a valid, unmodified state in the cache.

Clear. To cause a bit or bit field to register a value of zero. See also *Set*.

Context synchronization. An operation that ensures that all instructions in execution complete past the point where they can produce an *exception*, that all instructions in execution complete in the context in which they began execution, and that all subsequent instructions are *fetched* and executed in the new context. Context synchronization may result from executing specific instructions (such as **isync** or **rfi**) or when certain events occur (such as an exception).

Copy-back operation. A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.

D

Direct-mapped cache. A cache in which each main memory address can appear in only one location within the cache; operates more quickly when the memory request is a cache hit.

Double data rate. Memory that allows data transfers at the start and end of a clock cycle. thereby doubling the data rate.

E

Effective address (EA). The 32-bit address specified for a load, store, or an instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* address or an I/O address.

Exclusive state. MEI state (E) in which only one caching device contains data that is also in system memory.

F

Fetch. Retrieving instructions from either the cache or main memory and placing them into the instruction queue.

Flush. An operation that causes a cache block to be invalidated and the data, if modified, to be written to memory.

Frame-check sequence (FCS). Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD, and CRC.

G

General-purpose register (GPR). Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.

Guarded. The guarded attribute pertains to out-of-order execution. When a page is designated as guarded, instructions and data cannot be accessed out-of-order.

H

Harvard architecture. An architectural model featuring separate caches and other memory management resources for instructions and data.

I

Illegal instructions. A class of instructions that are not implemented for a particular processor. These include instructions not defined by the architecture. In addition, for 32-bit implementations, instructions that are defined only for 64-bit implementations are considered to be illegal instructions. For 64-bit implementations instructions that are defined only for 32-bit implementations are considered to be illegal instructions.

Implementation. A particular processor that conforms to the architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features.

Inbound ATMU windows. Mappings that perform address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and map the transaction to its target interface.

In-order. An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model.

Integer unit. An execution unit in the core responsible for executing integer instructions.

Inter-packet gap. The gap between the end of one Ethernet packet and the beginning of the next transmitted packet.

Instruction latency. The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

K

Kill. An operation that causes a *cache block* to be invalidated without writing any modified data to memory.

L

L2 cache. Level-2 cache. See *Secondary cache*.

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

Least-significant bit (lsb). The bit of least value in an address, register, field, data element, or instruction encoding.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

Little-endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *least-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *most-significant byte*. See *Big-endian*.

Local access window. Mapping used to translate a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The local memory map is defined by a set of eight local access windows. The size of each window can be configured from 4 Kbytes to 2 Gbytes.

M

Media access control (MAC) sublayer. Sublayer that provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.

Media-independent interface (MII) sublayer. Sublayer that provides a standard interface between the MAC layer and the physical layer for 10/100-Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.

Medium-dependent interface (MDI) sublayer. Sublayer that defines different connector types for different physical media and PMD devices.

Memory access ordering. The specific order in which the processor performs load and store memory accesses and the order in which those accesses complete.

Memory-mapped accesses. Accesses whose addresses use the page or block address translation mechanisms provided by the MMU and that occur externally with the bus protocol defined for memory.

Memory coherency. An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.

Memory consistency. Refers to agreement of levels of memory with respect to a single processor and system memory (for example, on-chip cache, secondary cache, and system memory).

Memory management unit (MMU). The functional unit that is capable of translating an *effective* (logical) *address* to a physical address, providing protection mechanisms, and defining caching methods.

Modified/exclusive/invalid (MEI). *Cache coherency* protocol used to manage caches on different devices that share a memory system. Note that neither the PowerPC ISA nor the Power ISA definitions specifies the implementation of an MEI protocol to ensure cache coherency.

Modified state. MEI state (M) in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.

Most-significant bit (msb). The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

N

NaN. An abbreviation for not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs—signaling NaNs and quiet NaNs.

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

O

OCeaN. (On-chip network) Non-blocking crossbar switch fabric. Enables full duplex port connections at 128Gb/s concurrent throughput and independent per port transaction queuing and flow control. Permits high bandwidth, high performance, as well as the execution of multiple data transactions.

Outbound ATMU windows. Mappings that perform address translations from local 32-bit address space to the address spaces of, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

P

Packet. A unit of binary data that can be routed through a network. Sometimes packet is used to refer to the frame plus the preamble and start frame delimiter (SFD).

Page. A region in memory. The OEA defines a page as a 4-Kbyte area of memory aligned on a 4-Kbyte boundary.

Page access history bits. The *changed* and *referenced* bits in the PTE keep track of the access history within the page. The referenced bit is set by the MMU whenever the page is accessed for a read or write operation. The changed bit is set when the page is stored into. See *Changed bit* and *Referenced bit*.

Page fault. A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory*. A page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.

Page table. A table in memory is comprised of *page table entries*, or PTEs. It is further organized into eight PTEs per PTEG (page table entry group). The number of PTEGs in the page table depends on the size of the page table (as specified in the SDR1 register).

Page table entry (PTE). Data structures containing information used to translate *effective address* to physical address on a 4-Kbyte page basis. A PTE consists of 8 bytes of information in a 32-bit processor and 16 bytes of information in a 64-bit processor.

Physical coding sublayer (PCS). Sublayer responsible for encoding and decoding data stream to and from the MAC sublayer.

Physical medium attachment (PMA) sublayer. Sublayer responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SERDES) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers.

Physical medium dependent (PMD) sublayer. Sublayer responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.

Physical memory. The actual memory that can be accessed through the system's memory bus.

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

Primary opcode. The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction.

Program order. The order of instructions in an executing program. More specifically, this term is used to refer to the original order in which program instructions are fetched into the instruction queue from the cache.

Protection boundary. A boundary between *protection domains*.

Protection domain. A protection domain is a segment, a virtual page, a BAT area, or a range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

Q

Quad word. A group of 16 contiguous locations starting at an address divisible by 16.

Quiesce. To come to rest. The processor is said to quiesce when an exception is taken or a **sync** instruction is executed. The instruction stream is stopped at the decode stage and executing instructions are allowed to complete to create a controlled context for instructions that may be affected by out-of-order, parallel execution. See *Context synchronization*.

R

rA. The **r**A instruction field is used to specify a GPR to be used as a source or destination.

rB. The **r**B instruction field is used to specify a GPR to be used as a source.

rD. The **r**D instruction field is used to specify a GPR to be used as a destination.

rS. The **r**S instruction field is used to specify a GPR to be used as a source.

Record bit. Bit 31 (or the Rc bit) in the instruction encoding. When it is set, updates the condition register (CR) to reflect the result of the operation.

Reconciliation sublayer. Sublayer that maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.

Reduced instruction set computing (RISC). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

Referenced bit. One of two *page history bits* found in each *page table entry*. The processor sets the *referenced bit* whenever the page is accessed for a read or write. See also *Page access history bits*.

Reservation. The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.

Reservation station. A buffer between the dispatch and execute stages that allows instructions to be dispatched even though the results of instructions on which the dispatched instruction may depend are not available.

S

Secondary cache. A cache memory that is typically larger and has a longer access time than the primary cache. A secondary cache may be shared by multiple devices. Also referred to as L2, or level-2, cache.

Set (*v*)**.** To write a nonzero value to a bit or bit field; the opposite of *clear*. The term ‘set’ may also be used to generally describe the updating of a bit or bit field.

Set (*n*)**.** A subdivision of a *cache*. Cacheable data can be stored in a given location in one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was used least recently. See *Set-associative*.

Set-associative. Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

Slave. The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

Snooping. Monitoring addresses driven by a bus master to detect the need for coherency actions.

Snoop push. Response to a snooped transaction that hits a modified cache block. The cache block is written to memory and made available to the snooping device.

Stall. An occurrence when an instruction cannot proceed to the next stage.

Sticky bit. A bit that when *set* must be cleared explicitly.

Superscalar machine. A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.

Supervisor mode. The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

Synchronization. A process to ensure that operations occur strictly *in order*. See *Context synchronization*.

System memory. The physical memory available to a processor.

T

Tenure. The period of bus mastership. There can be separate address bus tenures and data bus tenures.

Throughput. The measure of the number of instructions that are processed per clock cycle.

Time-division multiplex (TDM). A single serial channel used by several channels taking turns.

Transaction. A complete exchange between two bus devices. A transaction is typically comprised of an address tenure and one or more data tenures, which may overlap or occur separately from the address tenure. A transaction may be minimally comprised of an address tenure only.

Transfer termination. Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure.

Translation lookaside buffer (TLB). A cache that holds recently-used *page table entries*.

U

User mode. The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

V

Virtual address. An intermediate address used in the translation of an *effective address* to a physical address.

Virtual memory. The address space created using the memory management facilities of the processor. Program access to *virtual memory* is possible only when it coincides with *physical memory*.

W

Way. A location in the cache that holds a cache block, its tags, and status bits.

Word. A 32-bit data element.

Write-back. A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache block is *cast out* to make room for newer data.

Write-through. A cache memory update policy in which all processor write cycles are written to both the cache and memory.

Index

Numerics

A

Accumulator (ACC), 6-47
Address mask (LBC), 14-13
Address translation and mapping units (ATMUs)
  inbound windows, 2-10
    illegal interactions between inbound ATMUs and local access windows, 2-11
    PCI Express, 21-25
      endpoint (EP) mode, 21-25
      root complex (RC) mode, 21-25
    RapidIO, 2-10, 20-97
  local access windows, 2-3–2-11
    see also Local access windows
  outbound windows, 2-10
    PCI Express, 21-20
    RapidIO, 20-95
AFEU
  context, 11-100
  context memory, 11-100, 11-101
  data size register, 11-94
  FIFOs, 11-101
  interrupt control register, 11-98
  interrupt status register, 11-97
  key registers, 11-101
  key size register, 11-94
  mode register, 11-93
  reset control register, 11-95
  status register, 11-70, 11-96, 11-142, 11-162
Applications
  examples, 1-14
Arbitration
  I^2C interface
    arbitration control, 12-15
    loss of arbitration—forcing of slave mode, 12-24
    procedure for arbitration, 12-15
ARC Four execution unit (AFEU), 11-92
Architecture
  overview, 1-3
ASLEEP (global utilities asleep) signal, 23-2, 23-33

B

Battery backup, self-refresh mode, 9-93
BBEAR (branch buffer address register), see e500 core, registers
BBTAR (branch buffer target address register), see e500 core, registers
Block diagrams
  DDR controller, 9-1, 9-60
  debug modes, watchpoint monitor, and trace buffer, 25-1
  DMA controller, 19-1
  DUART, 13-2
  e500 coherency module (ECM), 8-1
  eTSEC, 15-2
  GPIO*n* module, 22-1
  I^2C interface, 12-1
  interrupt controller (PIC), 10-1
  L2 cache/SRAM, 7-1
  local bus controller (LBC), 14-1
  PCI Express, 21-2
  performance monitor, 24-2
  RapidIO controller, 20-1
  TSEC, 16-4
Boot mode
  CPU holdoff (POR), 4-19, 8-5
  POR status register (PORBMSR), 23-6
Boot page translation, 4-8
Boot ROM location (POR), 4-15
Boot sequencer
  boot holdoff mode (POR), 4-20, 8-5
  boot page translation, 4-8
  I^2C interface, 12-2, 12-17–12-20
  overview, 1-6, 4-9
  POR configuration, 4-20
BUCSR (branch unit control and status register), see e500 core, registers
Buffer descriptors
  see TSEC, buffer descriptors
Buffer descriptors, *see* eTSEC, buffer descriptors

C

Chaining
  performance monitor events, 24-35
$\overline{\text{CKSTP\_IN}}$ (global utilities checkstop in) signal, 23-2
$\overline{\text{CKSTP\_OUT}}$ (global utilities checkstop out) signal, 23-3
CLK_OUT (global utilities clock out) signal, 23-3, 23-26
Clocks
  DDR clock distribution, 9-75
  DDR controller clock disable, 23-24

device clock signals summary, 4-3
see also Signals, clock
device clocking operation, 4-30–4-33
CCB (platform) clock, 4-30
minimum CCB frequency equations, 4-32
Ethernet clocks, 4-32
RapidIO clocks, 4-31
system clock/PCI clock, 4-30
eTSEC
inputs and outputs, 15-10
management clock out (EC_MDC), 15-11, 15-81
I^2C
clock stretching, 12-17
clock synchronization, 12-16
input synchronization and digital filter, 12-16
LBC bus clocks and clock ratios, 14-3
clock ratio register (LCRR), 14-33
overview, 1-12
POR settings
e500 core PLL ratio, 4-14
system/CCB PLL ratio, 4-13
TSEC
inputs and outputs, 16-7
management clock out (EC_MDC), 16-42
Coherency rules
L2 cache, 7-29
Configuration
DDR, 9-13–9-50, 9-62
ECM
CCB address configuration register (EEBACR), 8-3
CCB port configuration register (EEBPCR), 8-4
eTSEC interfaces, 15-207–15-237
LBC
configuration register (LBCR), 14-32
PIC
global configuration register, 10-22
POR, *see* Power-on-reset (POR)
RapidIO, 20-94
TSEC interfaces, 16-72–16-75
Configuration space
PCI Express, 21-43
Configuration, control, and status
accessing CCSR memory from external masters, 2-12, 2-13
accessing CCSRs, 4-5
alternate configuration space (ALTBAR and ALTCAR), 4-6
boot page translation, 4-8
CCSR and RapidIO registers, 2-16
CCSR memory map, 2-11–2-17
CCSRBAR update guidelines, 4-5
memory map/register definition, 4-4
organization of CCSR memory, 2-13
Context ID registers, 25-23–25-24
Controller registers, 11-52
CR (condition register), see e500 core, registers
Crypto-channel
configuration register, 11-41
Crypto-channel registers, 11-40
CSRR0–1 (critical save/restore registers 0–1), see e500 core, registers
CTR (count register), see e500 core, registers
CTS, see DUART_CTS[0:1]

D

DACn (data address compare registers 1–2), see e500 core, registers
Data cache
see L2 cache/SRAM
Data encryption standard execution units (DEU), 11-62, 11-112, 11-121
DBCRn (debug control register 0–2), see e500 core, registers
DBSR (debug status register), see e500 core, registers
DDR controller
address signal mappings, 9-6
battery-backed RAM and forced self-refresh mode, 9-93
block diagram, 9-1, 9-60
clock distribution, 9-75
clocks
disabling, 23-24
configuration, example, 9-62
data beat ordering, 9-82
debug mode
signal selection (POR), 4-28, 4-29
source and target ID, 25-4, 25-25
driver impedance calibration, 9-10
error checking and correcting (ECC), 7-41, 9-83
testing ECC with error injection, 9-51–9-52
error handling, 9-52, 9-85
features, 9-2
functional description, 9-60
initialization/application information, 9-86
programming different memory types, 9-88
interrupts, 9-56
memory map/register definition, 9-11
modes of operation, 9-3
on-die termination for CSs, 9-10
overview, 1-4
page mode and logical bank retention, 9-82
performance monitor events, 24-17, 24-18
register descriptions, 9-13
by acronym, *see* Register Index
configuration registers, 9-13–9-50
error handling registers, 9-52–9-60

error injection registers, 9-51–9-52
SDRAM operation, 9-64
address multiplexing, 9-66
initialization sequence, 9-93
JEDEC standard interface commands, 9-70
mode-set command timing, 9-76
organizations supported, 9-64
refresh operation, 9-78
power-saving modes, 9-79
timing, 9-79
registered DIMM mode, 9-77
timing, 9-72
write timing adjustments, 9-77
self-refresh
operation in sleep mode, 9-81
using forced mode for battery backup, 9-93
signals summary, 9-3
see also Signals, DDR
DEAR (data exception address register), see e500 core, registers
Debug modes
and watchpoint monitor signals summary, 25-5
see also Signals, debug
and watchpoint monitor/trace buffer block diagram, 25-1
DDR signal selection (POR)
ECC pins used for debug, 4-28, 4-29
DDR source ID debug modes, 25-4, 25-25
source ID on debug signals, 25-27
source ID on ECC pins, 25-27
DDR/LBC signal selection (POR), 4-28
e500 core registers, 6-39–6-45
features, 25-3
functional description, 25-25
LBC source ID debug mode, 14-4, 25-4, 25-25
memory map/register definition, 25-10
modes of operation (set at POR), 25-3
overview, 25-1
PCI/PCI-X
source ID debug mode, 25-25
performance monitor events, 24-35
POR status (global utilities), 23-11
READY negation, 4-3
software debug
context ID registers, 25-23
trace buffer, see Trace buffer
watchpoint, see Watchpoint monitor
DEC (decrementer register), see e500 core, registers
DECAR (decrementer auto-reload register), see e500 core, registers
Descriptor structure, 11-22
DEU
FIFOs, 11-121
interrupt control register, 11-73, 11-118
interrupt status register, 11-71, 11-117
IV register, 11-121
key registers, 11-121
key size register, 11-67, 11-113, 11-114
mode register, 11-63, 11-112
reset control register, 11-68, 11-115
DMA channel 2 and 3 signal select, 23-13
DMA controller
block diagram, 19-1
channel operation, 19-27
bandwidth control, 19-33
channel abort, 19-33
channel state, 19-34
stride size and distance, 19-34
descriptor formats, 19-35
error handling, 19-35
features, 19-2
functional description, 19-27
interrupts, 19-9–19-13, 19-15, 19-21, 19-26, 19-35
limitations and restrictions, 19-38
memory map/register definition, 19-6
modes of operation, 19-2
basic mode transfer, 19-27
basic chaining mode, 19-29
basic chaining single-write start mode, 19-29
basic direct mode, 19-28
basic direct single-write start mode, 19-28
channel continue mode for cascading transfer chains, 19-32
basic channel continue mode, 19-33
extended mode, 19-33
extended DMA mode transfer, 19-30
extended chaining mode, 19-30
extended chaining single-write start mode, 19-31
extended direct mode, 19-30
extended direct single-write start mode, 19-30
external control mode transfer, 19-31
overview, 19-2
performance monitor events, 24-19, 24-20
register descriptions, 19-9–19-26
by acronym, see Register Index
signal select—channel 2 and 3, 23-13
signals summary, 19-5
see also Signals, DMA controller
system considerations, 19-39
unusual scenarios, 19-41
DMA to configuration and control registers, 19-41
DMA to DUART, 19-42
DMA to e500 core, 19-41
DMA to Ethernet, 19-41
DMA to I2C, 19-42

transfer interfaces, 19-35
DMA_DACK[0:3] (DMA acknowledge) signals, 19-6
DMA_DDONE[0:3] (DMA done) signals, 19-6
DMA_DREQ[0:3] (DMA request) signals, 19-5
Doorbell and port-write controller, *see* RapidIO controller, message unit
Doze mode, 1-12, 23-32
see also Global utilities, power management
DUART
asynchronous communication bits, 13-1
parity bit, 13-20
START bit, 13-19
STOP bit, 13-20
baud-rate generator logic, 13-20
block diagram, 13-2
divisor latch access bit (ULCRn[DLAB]), 13-4, 13-11
error handling, 13-21
framing error, 13-8, 13-14, 13-20, 13-21
overrun error, 13-21
parity error, 13-21
errors detected, 13-2
features, 13-1
functional description, 13-18
initialization/application information, 13-23
interrupts
interrupt control logic, 13-22
interrupt enable and control registers, 13-8–13-10
memory map/register definition, 13-3
modes of operation, 13-2
DMA mode selection, 13-22
FIFO mode, 13-21
interrupts, 13-22
local loopback mode, 13-21
overview, 1-6, 13-1
PC16450 UART compatibility, 13-1
performance monitor events, 24-35
register descriptions, 13-5–13-18
UART0 register offsets, 13-4
UART1 register offsets, 13-4
serial interface data format, 13-2
serial interface operation, 13-19–13-20
data transfer, 13-20
START bit, 13-19
STOP bit, 13-20
transaction protocol example, 13-19
signals summary, 13-3
see also Signals, DUART

E

e500 coherency module (ECM)
block diagram, 8-1
CCB arbiter, 8-10
CCB interface, 8-11
configuration
CCB address configuration register (EEBACR), 8-3
CCB port configuration register (EEBPCR), 8-4
error handling
error handling registers, 8-6–8-9
features, 8-2
functional description, 8-10
global data multiplexor, 8-11
I/O arbiter, 8-10
initialization/application information, 8-11
interrupts
ECM error enable register (EEER), 8-7
memory map/register definition, 8-3
overview, 1-4, 8-2
performance monitor events, 24-21
register descriptions, 8-3
by acronym, see Register Index
transaction queue, 8-10
e500 core
boot mode (POR), 4-19
branch operations
registers, 6-9–6-11
branch target buffer (BTB)
registers, 6-23–6-25
computational operations
registers, 6-8–6-9
debug registers, 6-39–6-45
hardware implementation-dependent registers (HID0–1), 6-25–6-28
interrupts
registers, 6-17–6-22
sources, 10-4
L1 caches
registers, 6-28–6-32
memory management unit (MMU)
registers, 6-32–6-39
MMU assist registers (MAS0–MAS4, MAS6), 6-34–6-38
MMU assist registers (MAS0–MAS4, MAS6–MAS7), 6-34–6-39
overview, 1-3
performance monitor
registers, 6-48–6-52
processor control registers, 6-11–6-14
registers
BBEAR (branch buffer address register), 6-23
BBTAR (branch buffer target address register), 6-23
BUCSR (branch unit control and status register), 6-24
CR (condition register), 6-9
CSRR0–1 (critical save/restore registers 0–1), 6-17
CTR (count register), 6-11

DACn (data address compare registers 1–2), 6-45
DBCRn (debug control register 0–2), 6-39–6-43
DBSR (debug status register), 6-43
DEAR (data exception address register 0), 6-18
DEC (decrementer register), 6-16
DECAR (decrementer auto-reload register), 6-17
ESR (exception syndrome register), 6-19
GPRs (general-purpose registers), 6-8
HID0–1 (hardware implementation-dependent registers 0–1), 6-25
IACn (instruction address compare registers 1–2), 6-45
IVORn (interrupt vector offset registers), 6-18
IVPR (interrupt vector prefix register), 6-18
L1CFG0–1 (L1 cache configuration registers 0–1), 6-30
L1CSR0–1 (L1 cache status and control registers 0–1), 6-28
LR (link register), 6-11
MAS0–MAS6 (MMU assist registers 0–6), 6-34–6-39
MCAR (machine check address register), 6-21
MCSR (machine check syndrome register), 6-21
MCSRR0–1 (machine check save/restore registers 0–1), 6-20
MMUCFG (MMU configuration register), 6-32
MMUCSR0 (MMU control and status register 0), 6-32
MSR (machine state register), 6-11
PIDn (process ID registers 0–2), 6-32
PIR (processor ID register), 6-13
PMCn (performance monitor counter registers 0–3), 6-52
PMGC0 (performance monitor global control register 0), 6-49
PMLCa*n* (performance monitor local control registers a0–a3), 6-50
PMLCb*n* (performance monitor local control registers b0–b3), 6-51
PVR (processor version register), 6-13
SPEFSCR (signal processing and embedded floating-point status and control register), 6-45
SPRGn (software-use registers 0–7), 6-22
SRR0–1 (save/restore registers 0–1), 6-17
SVR (system version register), 6-14, 23-23
TBL (time base lower register), 6-16
TBU (time base upper register), 6-16
TCR (timer control register), 6-14
TLB0CFG (TLB0 configuration register), 6-33
TLB1CFG (TLB1 configuration register), 6-34
TSR (timer status register), 6-15
UPMCn (user performance monitor counter registers 0–3), 6-52
UPMGC0 (user performance monitor global control register 0), 6-49
UPMLCan (user performance monitor local control registers a0–a3), 6-50
UPMLCbn (user performance monitor local control registers b0–b3), 6-51
USPRG0 (user software-use register 0), 6-22
XER (integer exception register), 6-8
signal processing engine (SPE)
registers, 6-45
software-use SPRs, 6-22
time base
RTC (real time clock) signal options, 4-4, 4-32
timer registers, 6-14–6-17
EC_GTX_CLK125 (eTSEC gigabit transmit 125 MHz source) signal, 15-11
EC_MDC (eTSEC management data clock) signal, 15-11
EC_MDC (TSEC management data clock) signals, 16-7
EC_MDIO (eTSEC management data input/output, BIDI) signal, 15-11
EC_MDIO (TSEC management data input/output) signals, 16-7
Error handling
DDR, 9-52–9-60, 9-85
DMA, 19-35
DUART, 13-2, 13-21
framing error, 13-8, 13-14, 13-20, 13-21
overrun error, 13-21
parity error, 13-21
ECM
error handling registers, 8-6–8-9
eTSEC, 15-179–15-181
I^2C interface
boot sequencer mode, 12-19
L2 cache/SRAM
error handling registers, 7-19
error injection, 7-20
LBC
transfer error registers, 14-26–14-31
PCI Express registers, 21-30–21-43
RapidIO, 20-102
error types, 20-102
logical layer errors detected, 20-106
message unit
doorbell error handling, 20-170–20-181
inbound error handling, 20-160–20-167
outbound error handling, 20-143–20-149, 20-154–20-156
port-write error handling, 20-184–20-188
physical layer errors detected, 20-103
TSEC, 16-65–16-66
ESR (exception syndrome register), see e500 core, registers
eTSEC
block diagram, 15-2

buffer descriptors, 15-200–15-207
receive buffer descriptors (RxBD), 15-205
transmit buffer descriptors (TxBD), 15-202
clocks
inputs and outputs, 15-10
management clock out (EC_MDC), 15-11, 15-81
operation, 4-32
configuration of interfaces, 15-207–15-237
16-bit FIFO mode, 15-235
8-bit FIFO mode, 15-232
GMII interface mode, 15-212
MAC configuration, 15-72
MII interface mode, 15-208
RGMII interface mode, 15-220
RMII interface mode, 15-224
RTBI interface mode, 15-228
TBI interface mode, 15-216
data width (POR), 4-21, 4-23, 4-24
error-handling, 15-179–15-181
eTSEC1 protocol (POR), 4-24
eTSEC2 protocol (POR), 4-25, 4-26
eTSEC3 protocol (POR), 4-25
eTSEC4 protocol (POR), 4-26
features, 15-2
FIFO interface connections, 15-157
16-bit encoded packet FIFO mode, 15-163
16-bit GMII-style packet FIFO mode, 15-161
8-bit encoded packet FIFO mode, 15-161
8-bit GMII-style packet FIFO mode, 15-160
CRC appending and checking, 15-159
flow control, 15-159
signal summary, 15-164
functional description, 15-146
gigabit Ethernet channel operation, 15-165
flow control, 15-175
frame reception, 15-168
frame recognition, 15-171
frame transmission, 15-167
initialization sequence, 15-165
soft reset and reconfiguring procedure, 15-166
internal and external loop back, 15-179
inter-packet gap time, 15-179
Magic Packet mode, 15-175
preamble customization, 15-169
RMON support, 15-171
hash function
algorithm, 15-173
registers, 15-115–15-117
initialization/application information, 15-207–15-237
gigabit Ethernet channel, 15-165
soft reset and reconfiguring procedure, 15-166
see also eTSEC, configuration
interrupts, 15-176–15-179
interrupt coalescing, 15-177
by frame count threshold, 15-177
by timer threshold, 15-178
interrupt registers, 15-28–15-33
lossless flow control, 15-194
back pressure determination and free buffers, 15-194
software use of hardware-initiated back pressure, 15-196
MAC functionality, 15-72–15-87
configuration, 15-72
CSMA/CD control, 15-72
handling packet collisions, 15-72
packet flow control, 15-73
PHY links control, 15-74
registers, 15-75–15-87
memory map/register definition, 15-15
detailed memory map, 15-16–15-26
eTSEC2–4 controller offsets, 15-26
top-level module map, 15-15
modes of operation, 15-5
RMON support, 15-87
overview, 1-7, 15-1
physical interface connections, 15-146
gigabit media-independent interface (GMII), 15-149
media-independent interface (MII), 15-147
reduced gigabit media-independent interface (RGMII), 15-149
reduced media-independent interface (RMII), 15-148
reduced ten-bit interface (RTBI), 15-151
ten-bit interface (TBI), 15-150
quality of service (QoS) support, 15-186–15-194
receive queue filer, 15-186
transmission scheduling, 15-192
register descriptions, 15-26–15-146
by acronym, *see* Register Index
DMA attribute registers, 15-119–15-120
FIFO registers, 15-117–15-118
general control and status registers, 15-26–15-41
hash function registers, 15-115–15-117
lossless flow control registers, 15-120–15-122
MAC registers, 15-75–15-87
MIB registers, 15-87–15-115
receive control and status registers, 15-53–15-71
ten-bit interface registers, 15-135–15-146
transmit control and status registers, 15-41–15-52
signals, 15-9
FIFO interface signal summary, 15-164
see also Signals, eTSEC
summary, 15-6
signals<$startmode, 15-9
TCP/IP off-load, 15-181–15-186
frame control blocks, 15-182

receive path off-load, 15-184
transmit path off-load, 15-182
EU
access, 11-51
assignment status register, 11-52
External system configuration
POR (LAD[0:31]) status, 4-28, 23-13
External writes, *see* L2 cache/SRAM, stashing

F

Features
distinctive, 17-4
Fetch register, 11-47
Full-on mode (power), 1-12

G

General Purpose I/O module (GPIO), *see* GPIO
Global utilities
clock out
CLK_OUT signal, 23-3, 23-26
clock out control register (CLKOCR), 23-26
overview, 23-1
DDR controller
clock disable, 23-24
DMA signal multiplex control register (PMUXCR), 23-13
features, 23-1
functional description, 23-30
interrupt and local bus signal multiplexing, 23-13
interrupts and power management, 23-36
LBC voltage select, 23-24
machine check summary
sources of *mcp* (MCPSUMR), 23-18
memory map/register definition, 23-3
overview, 23-1
POR configuration
boot mode status register (PORBMSR), 23-6
debug mode status register (PORDBGMSR), 23-11
device status register (PORDEVSR), 23-7
I/O impedance status register (PORIMPSCR), 23-7
LAD[0:31] external system configuration (GPPORCR), 4-28, 23-13
PLL status register (PORPLLSR), 23-4
see also Power-on reset (POR)
power management
and interrupts, 23-36
and snooping, 23-37
block disable (DEVDISR), 23-14, 23-32
CKSTP_IN and core_stopped mode, 23-31
core and device control bits, 23-33
core and device modes, 23-30
device mode control and status register (POWMGTCSR), 23-16
doze mode, 23-32
dynamic power management, 23-32
features, 23-1
functional description, 23-30
nap mode, 23-33
power-down sequence, 23-34
sleep mode, 23-33, 23-37
software considerations, 23-37
processor version register (PVR), 23-22
register descriptions, 23-4
by acronym, *see* Register Index
reset
HRESET_REQ control, 23-23
RapidIO and PCI Express reset requests (RSTRSCR), 23-20, 23-21
signals summary, 23-1
see also Signals, global utilities
snooping and power management, 23-37
system version register (SVR), 23-23
GPCM (LBC general-purpose chip-select machine), 14-46
see also Local bus controller (LBC)
GPIO
block diagram, 22-1
features, 22-1
memory map/register definition, 22-2
overview, 22-1
registers, 22-2–22-5
signals, 22-2
GPRs (general-purpose registers), see e500 core, registers

H

Hash function, *see* eTSEC, hash function
Hash table algorithm, see TSEC, hash function
HID0–1 (hardware implementation-dependent registers 0–1), see e500 core, registers
HRESET (hard reset) signal, 4-2, 4-10
HRESET_REQ (hard reset request) signal, 4-2, 12-18, 12-19

I

I/O impedance
LBC and PCI/PCI-X signals
control and status register (global utilities), 23-7
I/O requirements, 23-37
I^2C interface
arbitration
arbitration control, 12-15
loss of arbitration—forcing of slave mode, 12-24
procedure for arbitration, 12-15
block diagram, 12-1

boot sequencer
POR configuration, 4-20
boot sequencer mode, 12-2, 12-17–12-20
error condition behavior, 12-19
calling address match condition, 12-6
clock control, 12-16
clock stretching, 12-17
clock synchronization, 12-16
input synchronization and digital filter, 12-16
master mode, 12-16
slave mode, 12-16
data transfer, 12-13
error handling
boot sequencer mode, 12-19
features, 12-2
frequency divider
frequency divider register (I2CFDR), 12-6
functional description, 12-11
handshaking, 12-16
implementation details, 12-13
address compare, 12-15
control transfer, 12-14
transaction monitoring, 12-14
initialization/application information, 12-21–12-25
boot sequencer mode, see I2C interface, boot sequencer mode
generation of SCL when SDA low, 12-23
initialization sequence, 12-21
post-transfer software response, 12-22
repeated START generation, 12-23
START generation, 12-12, 12-22
STOP generation, 12-13, 12-23
interrupts
calling address match condition, 12-6
flowchart for interrupt service routine, 12-24
interrupt after transfer, 12-22
interrupt enable bit (I2CCR[MIEN]), 12-8
interrupt on START, 12-22
interrupt pending status bit (I2CSR[MIF]), 12-10
interrupt-driven byte-to-byte transfers, 12-2
read of last byte, 12-23
slave mode interrupt service routine guidelines, 12-23
for slave transmitter routine, 12-24
loss of arbitration, 12-24
memory map/register definition, 12-4
modes of operation, 12-2
boot sequencer mode, 12-2, 12-17–12-20
interrupt-driven byte-to-byte data transfer, 12-2
master mode, 12-2
slave mode, 12-2
overview, 12-2
register descriptions, 12-5
by acronym, see Register Index
signals summary, 12-3
see also Signals, I2C
transaction protocol, 12-11
handshaking, 12-16
repeated START condition, 12-3, 12-13
slave address transmission, 12-12
START condition, 12-3, 12-12, 12-22
STOP condition, 12-3, 12-13, 12-23
IACn (instruction address compare registers 1–2), see e500 core, registers
ID register, 11-59
Initialization
DDR (initialization and application information), 9-86
programming different memory types, 9-88
ECM (initialization and application information), 8-11
eTSEC (initialization and application information), 15-165, 15-207–15-237
see also eTSEC, configuration
I2C interface (initialization and application information), 12-21–12-25
boot sequencer mode, see I2C interface, boot sequencer mode
generation of SCL when SDA low, 12-23
initialization sequence, 12-21
post-transfer software response, 12-22
repeated START generation, 12-23
START generation, 12-12, 12-22
STOP generation, 12-13, 12-23
PIC (initialization and application information), 10-60
TSEC (initialization and application information)
see also TSEC, configuration
TSEC (initialization and application), 16-72–16-75
TSEC (initialization and application) information, 16-72
watchpoint monitor and trace buffer, 25-30
Initialization/application information
LBC, 14-88
Initiator write, 11-51
Interrupt
clear register, 11-57
mask register, 11-53
status register, 11-56
Interrupt controller (PIC)
block diagram, 10-1
configuration (global), 10-22
critical interrupts, 10-6, 10-31, 10-33
destination (interrupt routing)
IRQ_OUT, 10-6
end of interrupt (EOI), 10-53, 10-57
external interrupts
routed to critical interrupt (cint), 10-33
routed to IRQ_OUT, 10-32

features, 10-2
flow (interrupt processing), 10-54
functional description, 10-54
global timers, 10-25, 10-59
cascading of timers, 10-28, 10-30
clocking of timers, 10-26, 10-30
RTC (real time clock) signal options, 4-4, 4-32, 10-28, 10-30
initialization/application information, 10-60
interprocessor interrupts, 10-57
interrupt acknowledge (IACK) signaling, 10-52, 10-57
interrupt routing (mixed mode), 10-6
interrupt source priorities, 10-56
memory map/register definition, 10-10
messaging interrupts, 10-32, 10-33, 10-58
modes of operation, 10-5, 10-22
mixed mode, 10-5
pass-through mode (to support external interrupt controllers), 10-5
nesting of interrupts, 10-56
overview, 1-5, 10-1
performance monitor events, 24-23
processor core interrupt sources, 10-4
critical interrupt (cint) sources, 10-31
programming guidelines, 10-60
changing interrupt source configuration, 10-62
register descriptions, 10-20
by acronym, see Register Index, 10-20
global registers, 10-20–10-25
global timer registers, 10-25–10-30
interrupt source configuration registers, 10-25–10-28, 10-42, 10-48
message registers, 10-36, 10-38
non-accessible registers
interrupt pending register (IPR), 10-54
interrupt request register (IRR), 10-55
per-CPU registers, 10-49–10-53
performance monitor mask registers, 10-34–10-36
summary registers, 10-30, 10-33
reset of PIC, 10-22, 10-60
reset processor from software, 10-23, 10-60
signals summary, 10-9
see also Signals, PIC
simultaneous interrupts, priorities, 10-56
sources of interrupts, 10-6
internal (to PIC) interrupt destinations, 10-31, 10-32, 10-33, 10-34
internal (to PIC) interrupt sources, 10-7
spurious vector generation, 10-25, 10-57
vendor identification, 10-23
Interrupts
channel done, 11-40
channel error, 11-40
DDR, 9-56
DMA, 19-9–19-13, 19-15, 19-21, 19-26, 19-35
DUART
interrupt control logic, 13-22
interrupt enable and control registers, 13-8–13-10
e500 core
registers, 6-17–6-22
ECM interrupt register (ECM error enable register—EEER), 8-7
eTSEC, 15-176–15-179
interrupt registers, 15-28–15-33
general, 11-39
I^2C interface
calling address match condition, 12-6
flowchart for interrupt service routine, 12-24
interrupt after transfer, 12-22
interrupt enable bit (I2CCR[MIEN]), 12-8
interrupt on START, 12-22
interrupt pending status bit (I2CSR[MIF]), 12-10
interrupt-driven byte-to-byte transfers, 12-2
read of last byte, 12-23
slave mode interrupt service routine guidelines, 12-23
for slave transmitter routine, 12-24
loss of arbitration, 12-24
IRQ[9:11] signal select, 23-13
LBC interrupt register, 14-29
performance monitor (PIC), 24-23
power management and interrupts (global utilities), 23-36
RapidIO
message unit
doorbell, 20-170, 20-177
inbound, 20-160
outbound, 20-142, 20-153
port-write controller, 20-170, 20-177, 20-183
see also Interrupt controller (PIC)
TSEC, 16-63
interrupt registers, 16-12–16-17
IRQ[0:11] (interrupt request 0–11) signals, 10-9
IRQ[9:11] signal select
global utilities, 23-13
IRQ_OUT (interrupt request out) signal, 10-9, 10-30
IVORn (interrupt vector offset registers), see e500 core, registers
IVPR (interrupt vector prefix register), see e500 core, registers

J

JTAG test access port
signals summary, 25-5
see also Signals, JTAG, 25-5

L

L1CFG0–1 (L1 cache configuration registers 0–1), see e500 core, registers
L1CSR0–1 (L1 cache status and control registers 0–1), see e500 core, registers
L2 cache/SRAM
  allocation of lines, 7-34
  block diagram, 7-1
  coherency rules, 7-29
  error handling registers, 7-19
  error injection, 7-20
  external writes, *see* stashing
  flash clearing, instruction and data locks, 7-32
  locking
    clearing locks on selected lines, 7-32
    entire, 7-31
    programmed memory ranges, 7-32
    selected lines, 7-32
    with stale data, 7-33
  memory map/register definition, 7-8
  memory-mapped SRAM
    coherency rules, 7-31
  memory-mapped windows, 2-4
  operation, 7-35
  overview, 7-1
  performance monitor events, 24-33
  PLRU bit update considerations, 7-34
  register descriptions, 7-9–7-27
  replacement policy, 7-33
  SRAM features, 7-1
  stashing, 7-27
  state transitions, 7-37
    due to core-initiated transactions, 7-37
    due to system-initiated transactions, 7-40
  timing, 7-28
LA[27:31] (LBC non-multiplexed address) signals, 14-7
LAD[0:31] (LBC multiplexed address/data) signals, 14-7
LALE (LBC external address latch enable) signal, 14-5, 14-42
LBCTL (LBC data buffer control) signal, 14-7, 14-44
LBS[0:3] (LBC UPM byte select) signals, 14-6
LCK[0:2] (LBC clock) signals, 14-7
LCS[0:7] (LBC chip select) signals, 14-5
LCS[5:7] signal select
  global utilities, 23-13
LCS0 (LBC chip select 0) signal, 14-56, 14-70
LDP[0:3] (LBC data parity) signals, 14-7
LGPL0 (LBC GP line 0) signal, 14-6
LGPL1 (LBC GP line 1) signal, 14-6
LGPL2 (LBC GP line 2) signal, 14-6
LGPL3 (LBC GP line 3) signal, 14-6
LGPL4 (LBC GP line 4) signal, 14-6
LGPL5 (LBC GP line 5) signal, 14-7
LGTA (LBC GPCM transfer acknowledge) signal, 14-6, 14-55
Local access windows, 2-3–2-11
  ATMUs, *see* Address translation and mapping units (ATMUs)
  configuring local access windows, 2-9
  distinguishing local access windows from other mapping functions, 2-9
  illegal interactions
    between inbound ATMUs and local access windows, 2-11
    between local access windows and DDR SDRAM chip selects, 2-9
  L2 cache/SRAM window interactions, 2-4
  precedence if overlapping among themselves, 2-9
  precedence if overlapping with L2 cache/SRAM windows, 2-4
  registers, 2-7–2-8
    by acronym, *see* Register Index
Local address map, 1-4, 1-13
  see also Local access windows
  see also Local access windows
Local bus controller (LBC)
  address and address space checking, 14-42
  address mask field—option registers, 14-13
  atomic bus operations, 14-44
  block diagram, 14-1
  boot chip-select operation, 14-56, 14-70
  bus monitor, 14-45
  bus turnaround, 14-90
    additional address phases (UPM cycles), 14-91
    address following read, 14-90
    read data following address, 14-90
    read-modify-write cycle (parity), 14-91
  clocks and clock ratios, 14-3
    clock ratio register (LCRR), 14-33
  configuration
    LBC configuration register (LBCR), 14-32
  debug mode
    signal selection (POR), 4-28
    source and target ID, 25-4, 25-25
  error handling
    transfer error registers, 14-26–14-31
  external access termination (LGTA), 14-55
  features, 14-2
  functional description, 14-40
  general-purpose chip-select machine (GPCM), 14-46
    chip-select and write enable negation timing, 14-51
    chip-select assertion timing, 14-50
    extended hold time on read accesses, 14-54
    GPCM mode

registers, 14-14, 14-16
output enable timing, 14-54
programmable wait state configuration, 14-51
relaxed timing, 14-52
timing configuration, 14-47, 14-49, 14-66
initialization/application information, 14-88
interrupts
transfer error interrupt enable register (LTEIR), 14-29
LCS[5:7] signal select, 23-13
memory map/register definition, 14-8
memory refresh timer prescaler, 14-23
modes of operation, 14-3
bus clock and clock ratios, 14-3
GPCM mode, registers, 14-14, 14-16
source ID debug mode, 14-4
UPM mode, registers, 14-19
overview, 14-2
performance monitor events, 24-32
peripherals, 14-88
GPCM timing, 14-89
hierarchy for very high speeds, 14-89
multiplexed address/data, 14-88
port sizes, 14-91
register descriptions, 14-10
by acronym, see Register Index
signals, 14-4
see also Signals, LBC
UPM interfaces, 14-72–14-102
block diagram, 14-73
extended hold time (reads), 14-87
programming the UPMs, 14-76
RAM array, 14-78
address multiplexing, 14-84
byte select signal timing, 14-82
chip select signal timing, 14-82
data timing, 14-85
general purpose signal timing, 14-83
LGPL[0:5] timing (LAST), 14-86
loop control, 14-83
RAM word definition, 14-78
REDO, 14-83
wait mechanism (WAEN), 14-86
signal timing, 14-77
synchronous UPWAIT (early transfer acknowledge), 14-87
UPM mode
registers, 14-19, 14-20
UPM requests, 14-73
exception requests, 14-75
memory access requests, 14-74
refresh timer requests, 14-75
software requests, 14-75
voltage selection, 23-24
ZBT SRAM interface, 14-97, 14-102
LOE (LBC GPCM output enable) signal, 14-6
LPBSE (LBC parity byte select) signal, 14-6
LR (link register), see e500 core, registers
LSYNC_IN (LBC DLL synchronization in) signal, 14-8
LSYNC_OUT (LBC DLL synchronization out) signal, 14-8
LWE[0:3] (LBC GPCM write enable) signals, 14-6

M

MA[0:14] (DDR address bus) signals, 9-8
MAC functionality
see TSEC, MAC functionality
MAC functionality, *see* eTSEC, MAC functionality
Machine check
MCP (processor machine check) signal, 10-10
mcp summary register (MCPSUMR), 23-18
SRESET (soft reset) signal, 4-9
MAS0–MAS6 (MMU assist registers 0–6), *see* e500 core, registers
Master control register, 11-36, 11-60
MBA[0:1] (DDR logical bank address) signals, 9-8
MCAR (machine check address register), see e500 core, registers
MCAS (DDR column address strobe) signal, 9-9
MCK[0:5] (DDR clock output complement) signals, 9-11
MCK[0:5] (DDR clock output) signals, 9-11
MCKE[0:3] (DDR clock enable) signals, 9-11
MCP (processor machine check) signal, 10-10
MCS[0:3] (DDR chip select) signals, 9-9
MCSR (machine check syndrome register), see e500 core, registers
MCSRR0–1 (machine check save/restore registers 0–1), see e500 core, registers
MDEU
context registers, 11-147
data size register, 11-140, 11-146
FIFOs, 11-150
interrupt control register, 11-145
interrupt status register, 11-140, 11-143
key registers, 11-150
key size register, 11-140
mode register, 11-102, 11-135, 11-136
reset control register, 11-141
MDIC[0:1] (DDR driver impedance calibration) signals, 9-10
MDM[0:8] (DDR SDRAM data output mask) signals, 9-10
MDQ[0:8] (DDR data bus strobe) signals, 9-7, 9-61
MDVAL (DDR/LBC debug mode data valid) signal, 4-28, 14-8, 25-3, 25-7
MECC[0:5] (DDR error correcting code) signals as debug, 25-4, 25-7

MECC[0:7] (DDR error correcting code) signals, 4-28, 9-8
Memory maps
  CCSR memory, 2-4
    accessing CCSR memory from external masters, 2-12, 2-13
    CCSR and RapidIO registers, 2-16
    CCSR map, complete list of memory-mapped registers (by offset), 2-17
    CCSR organization, 2-13
    CCSR registers, 2-11–2-17
    device-specific utilities, 2-16
    general utilities registers, 2-14
    programmable interrupt controller (PIC) space, 2-15
  configuration, control, and status registers, 4-4
  DDR controller, 9-11
    illegal interaction between local access windows and DDR SDRAM chip selects, 2-9
  debug, watchpoint, and trace buffer registers, 25-10
  device memory map
    address translation and mapping, 2-3
    overview and example, 2-1
  DMA, 19-6
  DUART, 13-3
  ECM, 8-3
  eTSEC, 15-15
  global utilities, 23-3
  GPIO, 22-2
  I^2C, 12-4
  interrupt controller (PIC), 10-10
  L2 cache/SRAM, 7-8
  LBC, 14-8
  performance monitor, 24-3
  RapidIO
    endpoint, 20-4
    message unit, 2-63, 20-8
  TSEC, 16-8
Memory target queue
  performance monitor events, 24-19
Message digest execution unit (MDEU), 11-135
Message interrupts, see Interrupt controller (PIC), message interrupts
Message unit, *see* RapidIO controller, message unit
MMU assist registers (MAS0–MAS4, MAS6–MAS7), 6-34–6-39
MMUCFG (MMU configuration register), see e500 core, registers
MMUCSR0 (MMU control and status register 0), see e500 core, registers
MODT[0:3] (DDR on-die termination) signals, 9-10
MRAS (DDR row address strobe) signal, 9-9
MSR (machine state register), see e500 core, registers
MSRCID[0:4] (DDR/LBC debug source ID) signals, 4-28, 14-8, 25-3, 25-7
MWE (DDR write enable) signal, 9-9, 9-10

N

Nap mode, 1-12, 23-33
  see also Global utilities, power management

O

On-chip memory
  overview, 1-4

P

PCI Express Base Specification, Rev. 1.0a
  see PCI Express controller
PCI Express controller
  accessing configuration space
    endpoint (EP) mode, 21-45
    root complex (RC) mode, 21-43
  address translation and mapping unit (ATMU)
    inbound windows, 21-25
      endpoint (EP) mode, 21-25
      root complex (RC) mode, 21-25
    outbound windows, 21-20
  block diagram, 21-2
  clocks
    minimum CCB frequency equation, 4-32
  commands
    command register, 21-46
  configuration space accesses, 21-43
  error handling registers, 21-30–21-43
  features, 21-3
  latency timer, 21-51
  modes of operation, 21-4
    link width, 21-4
    root complex or endpoint mode, 21-4
  overview, 21-1
  POR configuration, 21-4
  power management, 21-13–21-19, 21-70–21-71
  register descriptions
    configuration header registers, 21-45–21-68
      32-bit memory base address register, 21-53
      64-bit high memory base address register, 21-54
      64-bit low memory base address register, 21-54
      base address registers, 21-52–21-54, 21-59
      bridge control register, 21-68
      bus status register, 21-48
      cache line size register, 21-50
      capabilities pointer register, 21-56, 21-66
      command register, 21-46

configuration and status register base address (PCSRBAR), 21-52, 21-59
device ID register, 21-46, 21-55
I/O base register, 21-61
I/O base upper 16 bits register, 21-65
I/O limit register, 21-61
I/O limit upper 16 bits register, 21-66
interrupt line register, 21-56, 21-67
interrupt pin register, 21-57, 21-67
latency timer register, 21-51, 21-61
maximum latency (EP-mode) register, 21-58
memory base register, 21-63
memory limit register, 21-63
minimum grant (EP-mode) register, 21-57
prefetchable base upper 32 bits register, 21-65
prefetchable limit upper 32 bits register, 21-65
prefetchable memory base register, 21-64
prefetchable memory limit register, 21-64
primary bus number register, 21-59
revision ID register, 21-49
secondary bus number register, 21-60
secondary status register, 21-62
subordinate bus number register, 21-60
subsystem vendor ID register, 21-55
vendor ID register, 21-46
device-specific configuration space registers, 21-69–21-82
capabilities register, 21-72
capability ID register, 21-72
data capabilities register, 21-73
device control register, 21-73
device status register, 21-74
link capabilities register, 21-75
link control register, 21-75
link status register, 21-76
MSI message address register, 21-81
MSI message capability ID register, 21-80
MSI message control register, 21-80
MSI message data register, 21-82
MSI message upper address register, 21-81
power management capabilities register, 21-70
power management capability ID register, 21-70
power management data register, 21-71
power management status and control register, 21-71
root control register, 21-79
root status register, 21-79
slot capabilities register, 21-77
slot control register, 21-77
slot status register, 21-78
extended configuration space registers, 21-83–21-91
advanced error capabilities and control register, 21-88
advanced error reporting capability ID register, 21-84
correctable error mask register, 21-87
correctable error source ID register, 21-91
correctable error status register, 21-87
error source ID register, 21-91
header log register, 21-88
root error command register, 21-89
root error status register, 21-90
uncorrectable error mask register, 21-85
uncorrectable error severity register, 21-86
uncorrectable error status register, 21-84
memory-mapped registers, 21-6
ATMU registers, 21-20–21-30
by acronym, *see* Register Index
configuration access registers, 21-10–21-13, 21-43–21-45
error management registers, 21-30–21-43
IP block revision registers, 21-19–21-20
pwr mgmt and message registers, 21-13–21-19
signals summary, 21-5
see also Signals, PCI Express
PCI/PCI-X controller
debug mode
source and target ID (PCI_AD[63:59]), 25-25
Performance monitor (device)
block diagram, 24-2
burstiness, 24-14, 24-36
control registers, 24-6–24-10
counters (PMCn)
chaining, 24-13
registers, 24-10
triggering, 24-14
event counting, 24-12
events, 24-16, 24-16
chaining, 24-35
DDR controller, 24-17, 24-18
debug, 24-35
DMA controller, 24-19, 24-20
DUART, 24-35
e500 coherency module (ECM), 24-21
interrupt controller (PIC), 24-23
L2 cache/SRAM, 24-33
local bus controller (LBC), 24-32
memory target queue, 24-19
events triggered by watchpoint monitor, 25-28
examples, 24-35
burstiness event, 24-14
burstiness event counting, 24-36
simple event counting, 24-36
threshold event counting, 24-36
triggering event counting, 24-36
external signals, 24-3
features, 24-3

functional description, 24-12
interrupts, 24-12
interrupts (from PIC) to generate events, 10-34
masking interrupts (from PIC), 10-34
memory map/register definition, 24-3
overflow indication on TRIG_OUT, 25-25
overview, 24-2
threshold events, 24-12
Phase-locked loops (PLLs)
POR status (global utilities), 23-4
PIDn (process ID registers 0–2), see e500 core, registers
PIR (processor ID register), see e500 core, registers
PKEU
data size register, 11-153
EU_GO register, 11-100, 11-120, 11-146, 11-158, 11-165
interrupt control register, 11-157
interrupt status register, 11-156
key size register, 11-151, 11-153
mode register, 11-151
parameter memory A, 11-159
parameter memory B, 11-159
parameter memory E, 11-159
parameter memory N, 11-159
reset control register, 11-154
status register, 11-116, 11-155
PMCn (performance monitor counter registers 0–3), see e500 core, registers
PMGC0 (performance monitor global control register 0), see e500 core, registers
PMLCa*n* (performance monitor local control registers a0–a3), *see* e500 core, registers
PMLCb*n* (performance monitor local control registers b0–b3), *see* e500 core, registers
Port-write controller, *see* RapidIO controller, message unit
Power management
block disable
block disable control (DEVDISR), 23-14, 23-32
DDR interface, 9-79
device low-power modes, 23-30–23-37
control and status register (POWMGTCSR), 23-16
READY negation, 4-3
overview, 1-12
PCI Express, 21-13–21-19, 21-70–21-71
see also Global utilities, power management
Power-on reset (POR)
configuration
boot ROM location, 4-15
boot sequencer configuration, 4-20
clock
e500 core PLL ratio, 4-14
system/CCB PLL ratio, 4-13
CPU boot configuration, 4-19
DDR debug mode (ECC pins used for debug), 4-28, 4-29, 25-3
eTSEC1 protocol, 4-24
eTSEC1–2 data width, 4-21, 4-23
eTSEC2 protocol, 4-25, 4-26
eTSEC3 protocol, 4-25
eTSEC3–4 data width, 4-24
eTSEC4 protocol, 4-26
general-purpose (external system) configuration—LAD[0:31] (GPPORCR), 4-28
memory debug select (DDR or LBC), 4-28, 25-3
PCI debug configuration, 25-3
RapidIO device ID, 4-27
configuration reporting
global utilities, 23-4, 23-6, 23-7, 23-11, 23-13
debug modes summary, 25-3
hard reset, 4-10
PCI Express, modes of operation, 21-4
reset configuration signals, 3-23
sequence of events, 4-10
and READY signal, 4-3, 4-11
Processor version (PVR), 23-22
Programmable interrupt controller (PIC), *see* Interrupt controller (PIC), 1-5
PVR (processor version register), see e500 core, registers

Q

Quality of service (QoS), *see* eTSEC

R

Random number generator (RNG)
overview, 11-159
RapidIO controller
address translation and mapping unit (ATMU)
inbound ATMU translation, 2-10
inbound windows, 20-97
crossed boundary errors, 20-97
hits to multiple windows, 20-97
outbound windows, 20-95
crossed boundary errors, 20-96
hits to multiple windows, 20-95
block diagram, 20-1
clocks
minimum CCB frequency equation, 4-32
operation, 4-31
configuration
accessing config. reg's with RapidIO packets, 20-94
control symbol summary, 20-92
control symbols
link-request/reset-device, 20-98
device ID (POR), 4-27

error handling, 20-102
error types, 20-102
logical layer errors detected, 20-106
message unit
doorbell errors, 20-170–20-181
inbound message errors, 20-160–20-167
outbound message errors, 20-143–20-149, 20-154–20-156
port-write errors, 20-184–20-188
physical layer errors detected, 20-103
features, 20-1
functional description, 20-89
hot-swap support, 20-100
interrupts
message unit
doorbell, 20-170, 20-177
inbound message controller, 20-160
outbound message controller, 20-142, 20-153
port-write, 20-170, 20-177, 20-183
maintenance accesses
inbound, 20-94
outbound, 20-94
memory map/register definition
endpoint, 20-4
message unit, 2-63, 20-8
message unit
doorbell controller operation, 20-168
command and status register (PWDCSR), 20-181
doorbell queue and pointer structure, 20-169, 20-176
enabling and disabling, 20-181
error handling, 20-170–20-181
inbound doorbell controller, 20-175
interrupts, 20-170, 20-177, 20-183
outbound doorbell controller, 20-169
retry response conditions, 20-177
features, 20-140
inbound message controller operation, 20-157
enabling and disabling, 20-167
error handling, 20-160–20-167
inbound message structure, 20-157
interrupts, 20-160
message steering, 20-159
retry response conditions, 20-159
mailbox command and status register (MCSR), 20-168
outbound message controller operation, 20-141
arbitration for multiple message units, 20-156
chaining mode operation, 20-149–20-156
descriptor format, 20-153
direct mode operation, 20-141
enabling and disabling, 20-144
error handling, 20-143–20-149, 20-154–20-156
interrupts, 20-142, 20-153
switching between direct and chaining modes, 20-152
outbound modes of operation, 20-141
overview, 20-139
port-write controller operation, 20-182
command and status register (PWDCSR), 20-188
discarding port-writes, 20-184
enabling and disabling, 20-188
error handling, 20-184–20-188
interrupts, 20-170, 20-177, 20-183
modes of operation, 20-3
message unit (RMU), 20-3
RapidIO port, 20-3
overview, 1-11, 20-1
packet format summary, 20-91
register descriptions
1x/4x LP-Serial registers, 20-22–20-28
architectural registers, 20-10–20-21
ATMU registers, 20-50–20-61
by acronym, *see* Register Index
error reporting logical registers, 20-30–20-35
error reporting physical registers, 20-36–20-42
implementation space registers, 20-42–20-49
message unit registers, 20-61–20-89
doorbellregisters, 20-78–20-87
port-write registers, 20-87–20-89
revision control registers, 20-49–20-50
signal summary, 20-4
see also Signals, RapidIO
transactions supported, 20-89
READY signal, 4-3, 4-11, 25-24, 25-25
Registers
AFEU
data size, 11-94
interrupt control, 11-98
interrupt status, 11-97
key, 11-101
key size, 11-94
mode, 11-93
reset control, 11-95
status, 11-70, 11-96, 11-142, 11-162
by acronym (memory-mapped registers)
see Register Index
configuration, control, and status, 2-11–2-17, 4-4
device-specific utilities, 2-16
general utilities, 2-14
programmable interrupt controller (PIC) space, 2-15
context ID, 25-23–25-24
crypto-channel
configuration, 11-41
general, 11-40
DDR
configuration registers, 9-13–9-50

error handling registers, 9-52–9-60
error injection registers, 9-51–9-52
DEU
interrupt control, 11-73, 11-118
interrupt status, 11-71, 11-117
IV, 11-121
key, 11-121
key size, 11-67, 11-113, 11-114
mode, 11-63, 11-112
reset control, 11-68, 11-115
e500 core, see e500 core, registers
ECM, 8-3
eTSEC, 15-26–15-146
DMA attribute registers, 15-119–15-120
FIFO registers, 15-117–15-118
general control and status registers, 15-26–15-41
hash function registers, 15-115–15-117
lossless flow control registers, 15-120–15-122
MAC registers, 15-75–15-87
MIB registers, 15-87–15-115
receive control and status registers, 15-53–15-71
ten-bit interface registers, 15-135–15-146
transmit control and status registers, 15-41–15-52
EU assignment status, 11-52
fetch, 11-47
global utilities, 23-4
POR boot mode status, 23-6
POR debug mode status, 23-11
POR device status, 23-7
POR external system configuration, 23-13
POR I/O impedance status, 23-7
POR PLL status, 23-4
GPIO, 22-2–22-5
I^2C interface, 12-5
ID, 11-59
interrupt
clear, 11-57
mask, 11-53
status, 11-56
L2 cache/SRAM registers, 7-8–7-27
LBC, 14-10
local access window registers
attributes registers (LAWAR0–LAWAR7), 2-8
base address registers (LAWBAR0–LAWBAR7), 2-7
master control, 11-36, 11-60
MDEU
context registers, 11-147
data size, 11-140, 11-146
interrupt control, 11-145
interrupt status, 11-140, 11-143
key, 11-150
key size, 11-140
mode, 11-102, 11-135, 11-136
reset control, 11-141
PCI Express
configuration header registers, 21-45–21-68
32-bit memory base address register, 21-53
64-bit high memory base address register, 21-54
64-bit low memory base address register, 21-54
base address registers, 21-52–21-54, 21-59
bridge control register, 21-68
bus status register, 21-48
cache line size register, 21-50
capabilities pointer register, 21-56, 21-66
command register, 21-46
configuration and status register base address (PCSRBAR), 21-52, 21-59
device ID register, 21-46, 21-55
I/O base register, 21-61
I/O base upper 16 bits register, 21-65
I/O limit register, 21-61
I/O limit upper 16 bits register, 21-66
interrupt line register, 21-56, 21-67
interrupt pin register, 21-57, 21-67
latency timer register, 21-51, 21-61
maximum latency (EP-mode) register, 21-58
memory base register, 21-63
memory limit register, 21-63
minimum grant (EP-mode) register, 21-57
prefetchable base upper 32 bits register, 21-65
prefetchable limit upper 32 bits register, 21-65
prefetchable memory base register, 21-64
prefetchable memory limit register, 21-64
primary bus number, 21-59
revision ID register, 21-49
secondary bus number, 21-60
secondary status register, 21-62
subordinate bus number, 21-60
subsystem vendor ID, 21-55
vendor ID, 21-46
device-specific configuration space registers, 21-69–21-82
capabilities register, 21-72
capability ID register, 21-72
device capabilities register, 21-73
device control register, 21-73
device status register, 21-74
link capabilities register, 21-75
link control register, 21-75
link status register, 21-76
MSI message address register, 21-81
MSI message capability ID register, 21-80
MSI message control register, 21-80
MSI message data register, 21-82

MSI message upper address register, 21-81
power management capabilities register, 21-70
power management capability ID register, 21-70
power management data register, 21-71
power management status and control register, 21-71
root control register, 21-79
root status register, 21-79
slot capabilities register, 21-77
slot control register, 21-77
slot status register, 21-78
extended configuration space registers, 21-83–21-91
advanced error capabilities and control register, 21-88
advanced error reporting capability ID register, 21-84
correctable error mask register, 21-87
correctable error source ID register, 21-91
correctable error status register, 21-87
error source ID register, 21-91
header log register, 21-88
root error command register, 21-89
root error status register, 21-90
uncorrectable error mask register, 21-85
uncorrectable error severity register, 21-86
uncorrectable error status register, 21-84
memory-mapped registers
ATMU registers, 21-20–21-30
configuration access registers, 21-10–21-13, 21-43–21-45
error management registers, 21-30–21-43
IP block revision registers, 21-19–21-20
pwr mgmt and message registers, 21-13–21-19
performance monitor, descriptions, 24-3
PIC, 10-20
global registers, 10-20–10-25
global timer registers, 10-25–10-30
interrupt source configuration registers, 10-25–10-28, 10-42
message registers, 10-36
non-accessible registers
interrupt pending register (IPR), 10-54
interrupt request register (IRR), 10-55
per-CPU registers, 10-49–10-53
performance monitor mask registers, 10-34–10-36
summary registers, 10-30, 10-30–10-33, 10-34
PKEU
EU_GO, 11-100, 11-120, 11-146, 11-158, 11-165
interrupt control, 11-157
interrupt status, 11-156
key size, 11-151, 11-153
reset control, 11-154
status, 11-116, 11-155
PKEU data size, 11-153
processor version register (PVR), 23-22
RapidIO
1x/4x LP-Serial registers, 20-22–20-28
architectural registers, 20-10–20-21
ATMU registers, 20-50–20-61
error reporting logical registers, 20-30–20-35
error reporting physical registers, 20-36–20-42
implementation space registers, 20-42–20-49
message unit registers, 20-61–20-89
doorbell registers, 20-78–20-87
port-write registers, 20-87–20-89
revision control registers, 20-49–20-50
RNG
data size, 11-161
interrupt control, 11-164
interrupt status, 11-163
mode, 11-160
reset control, 11-161
security engine (SEC)
KEU, 11-122
system version register (SVR), 23-23
three-speed Ethernet controller
interrupt event, 18-9, 18-11, 18-12
trace buffer, 25-16–25-23
trigger out source register, 25-24
TSEC
FIFO control and status registers, 16-20–16-23
general control and status registers, 16-12
hash function registers, 16-48–16-50
MAC registers, 16-33–16-48
receive control and status registers, 16-30–16-33
transmit control and status registers, 16-23–16-29
watchpoint monitor, 25-11–25-16
Reset
core reset through PIC register, 10-23, 10-60
hard reset actions, 4-10
HRESET_REQ control, 23-23
operations, 4-9
power-on reset (POR)
configuration, see Power-on reset (POR), configuration
sequence of events, 4-10
requests from RapidIO and PCI Express, 23-20, 23-21
signals summary, 4-2
see also Signals, reset
soft request, 23-20, 23-21
soft reset actions, 4-9
and reconfiguring the eTSEC, 15-166
RMON support, *see* eTSEC, modes of operation
RNG
data size register, 11-161
FIFO, 11-165
interrupt control register, 11-164
interrupt status register, 11-163

mode register, 11-160
reset control register, 11-161
RTC (real time clock) signal, 4-4, 4-32, 10-28, 10-30, 23-34
RTS, see DUART_RTS[0:1]

S

SCL (I^2C serial clock) signal, 12-3, 12-4
SD_RX[4:7]/$\overline{\text{SD\_RX}}$[4:7] (RapidIO serial data input and complement) signals, 20-4
SD_RX[7:0]/$\overline{\text{SD\_RX}}$[7:0] (PCI Express serial data input and complement) signals, 21-5
SD_TX[4:7]/$\overline{\text{SD\_TX}}$[4:7] (RapidIO serial data output and complement) signals, 20-4
SD_TX[7:0]/$\overline{\text{SD\_TX}}$[7:0] (PCI Express serial data output and complement) signals, 21-5
SDA (I^2C serial data) signal, 12-3, 12-4
Security engine (SEC)
  register descriptions
    KEU, 11-122
Serial data/clock, see I^2C interface, 12-1
Signals
  clock
    RTC (real time clock), 4-4, 4-32, 10-28, 10-30, 23-34
    SYSCLK (system clock input), 4-3, 4-4
  complete signal listing
    configuration signals, sampled at POR, 3-23
      see also Power-on reset (POR)
    figure showing groupings, 3-1
    reference by functional block, 3-6
  DDR
    MA[0:14] (address bus), 9-8
    MBA[0:1] (logical bank address), 9-8
    MCAS (column address strobe), 9-9
    MCK[0:5] (DDR clock output complements), 9-11
    MCK[0:5] (DDR clock outputs), 9-11
    MCKE[0:3] (DDR clock enables), 9-11
    MCS[0:3] (chip selects), 9-9
    MDIC[0:1] (driver impedance calibration), 9-10
    MDM[0:8] (SDRAM data output mask), 9-10
    MDQS[0:8] (data bus strobes), 9-7, 9-61
    MDVAL (debug mode data valid), 4-28, 25-3, 25-7
    MECC[0:5] (error correcting code)
      as debug signals, 25-4, 25-7
    MECC[0:7] (error correcting code), 4-28, 9-8
    MODT[0:3] (on-die termination), 9-10
    MRAS (row address strobe), 9-9
    MSRCID[0:4] (debug source ID), 4-28, 25-3, 25-7
    MWE (write enable), 9-9, 9-10
  DMA
    $\overline{\text{DMA\_DACK}}$[0:3] (DMA acknowledge), 19-6
    $\overline{\text{DMA\_DDONE}}$[0:3] (DMA done), 19-6
    $\overline{\text{DMA\_DREQ}}$[0:3] (DMA request), 19-5
  DUART
    UART_CTS[0:1] (DUART clear to send), 13-1, 13-3
    UART_RTS[0:1] (DUART request to send), 13-1, 13-3
    UART_SIN [0:1] (DUART transmitter serial data in), 13-2, 13-3
    UART_SOUT [0:1] (DUART transmitter serial data out), 13-2, 13-3
  eTSEC
    EC_GTX_CLK125 (eTSEC gigabit transmit 125 MHz source), 15-11
    EC_MDC (eTSEC management data clock), 15-11
    EC_MDIO (eTSEC management data input/output, BIDI), 15-11
    FIFO interface signal summary, 15-164
    TSEC*n*_COL (eTSEC 1–4 collision input), 15-10
    TSEC*n*_CRS (eTSEC 1–4 carrier sense input/FIFO receiver flow control), 15-10
    TSEC*n*_GTX_CLK (eTSEC 1–4 gigabit transmit clock), 15-10
    TSEC*n*_RX_CLK (eTSEC 1–4 receive clock), 15-11
    TSEC*n*_RX_DV (eTSEC 1–4 receive data valid), 15-12
    TSEC*n*_RX_ER (eTSEC 1–4 receive error), 15-12
    TSEC*n*_RXD[7:0] (eTSEC 1–4 receive data in), 15-12
    TSEC*n*_TX_CLK (eTSEC 1–4 transmit clock in), 15-13
    TSEC*n*_TX_EN (eTSEC 1–4 transmit data valid), 15-13
    TSEC*n*_TX_ER (eTSEC 1–4 transmit error), 15-14
    TSEC*n*_TXD[7:0] (eTSEC 1–4 transmit data out), 15-13
  global utilities
    ASLEEP, 23-2, 23-33
    $\overline{\text{CKSTP\_IN}}$ (checkstop in), 23-2
    $\overline{\text{CKSTP\_OUT}}$ (checkstop out), 23-3
    CLK_OUT, 23-3, 23-26
  GPIO, 22-2
  I^2C
    SCL (serial clock), 12-3, 12-4
    SDA (serial data), 12-3, 12-4
  JTAG
    TCK (JTAG test clock), 25-8
    TDI (JTAG test data input), 25-8
    TDO (JTAG test data output), 25-9
    TMS (JTAG test mode select), 25-9
    TRST (JTAG test reset), 25-9
  LBC
    LA[27:31] (non-multiplexed address), 14-7
    LAD[0:31] (multiplexed address/data), 14-7
    LALE (external address latch enable), 14-5, 14-42
    LBCTL (data buffer control), 14-7, 14-44
    LBS[0:3] (UPM byte select), 14-6
    LCK[0:2] (clock), 14-7
    LCS[0:7] (chip select), 14-5
    LCS0 (LBC chip select 0), 14-56, 14-70
    LDP[0:3] (data parity), 14-7

LGPL0 (GP line 0), 14-6
LGPL1 (GP line 1), 14-6
LGPL2 (GP line 2), 14-6
LGPL3 (GP line 3), 14-6
LGPL4 (GP line 4), 14-6
LGPL5 (GP line 5), 14-7
LGTA (GPCM transfer acknowledge), 14-6, 14-55
LOE (GPCM output enable), 14-6
LPBSE (parity byte select), 14-6
LSYNC_IN (DLL synchronization in), 14-8
LSYNC_OUT (DLL synchronization out), 14-8
LWE[0:3] (GPCM write enable), 14-6
MDVAL (debug mode data valid), 4-28, 14-8, 25-3, 25-7
MSRCID[0:4] (debug source ID), 4-28, 14-8, 25-3, 25-7
TA (data transfer acknowledge), 14-43
UPWAIT (UPM wait), 14-6, 14-73
other
TEST_SEL (factory test), 25-6
PCI Express
SD_RX[7:0]/$\overline{\text{SD\_RX}}$[7:0] (PCI Express serial data input and complement) signals, 21-5
SD_TX[7:0]/$\overline{\text{SD\_TX}}$[7:0] (PCI Express serial data output and complement) signals, 21-5
PIC
IRQ[0:11], 10-9
IRQ_OUT, 10-9, 10-30
MCP, 10-10
RapidIO
SD_RX[4:7]/$\overline{\text{SD\_RX}}$[4:7] (RapidIO serial data input and complement) signals, 20-4
SD_TX[4:7]/$\overline{\text{SD\_TX}}$[4:7] (RapidIO serial data output and complement) signals, 20-4
reset
HRESET (hard reset), 4-2, 4-10
HRESET_REQ (hard reset request), 4-2, 12-18, 12-19
READY, 4-3, 25-24, 25-25
SRESET (soft reset), 4-3, 4-9
TSEC
EC_MDC (TSEC management data clock), 16-7
EC_MDIO (TSEC management data input/output), 16-7
TSECn_COL (TSEC 1–2 collision input), 16-7
TSECn_CRS (TSEC 1–2 carrier sense input), 16-7
TSECn_RX_CLK (TSEC 1–2 receive clock), 16-7
TSECn_RX_DV (TSEC 1–2 receive data valid), 16-7
TSECn_RX_ER (TSEC 1–2 receive error), 16-7
TSECn_RXD[7:0] (TSEC 1–2 receive data in), 16-7
TSECn_TX_CLK (TSEC 1–2 transmit clock in), 16-7
TSECn_TX_EN (TSEC 1–2 transmit data valid in), 16-8
TSECn_TX_ER (TSEC 1–2 transmit error in), 16-8
TSECn_TXD[7:0] (TSEC 1–2 transmit data out), 16-7
watchpoint monitor
TRIG_IN (watchpoint trigger in), 25-8, 25-12, 25-18
TRIG_OUT (watchpoint trigger out), 25-8, 25-24
Sleep mode, 1-12, 23-33, 23-37
see also Global utilities, power management
Snooping
power management and snooping (global utilities), 23-37
Soft reset, 23-20, 23-21
Soft reset and reconfiguring for TSEC, 16-55
SPEFSCR (signal processing and embedded floating-point status and control register), see e500 core, registers
SPRGn (software-use registers 0–7), see e500 core, registers
SRAM, *see* L2 cache/SRAM, 7-29
SRESET (soft reset) signal, 4-3, 4-9
SRR0–1 (save/restore registers 0–1), see e500 core, registers
Stashing, *see* L2 cache/SRAM, stashing, 7-27
SVR (system version register), see e500 core, registers
SYSCLK (system clock input) signal, 4-3, 4-4

T

TA (LBC data transfer acknowledge) signal, 14-43
TBL (time base lower register), see e500 core, registers
TBU (time base upper register), see e500 core, registers
TCK (JTAG test clock) signal, 25-8
TCR (timer control register), see e500 core, registers
TDI (JTAG test data input) signal, 25-8
TDO (JTAG test data output) signal, 25-9
Test interface, see JTAG test access port
TEST_SEL (factory test) signal, 25-6
Three-speed Ethernet controller
detailed memory map—control/status registers, 18-5
functional description, 18-27
interrupt event register (IEVENT), 18-9, 18-11, 18-12
introduction, 18-1
memory map/register definition, 18-4
memory-mapped register descriptions, 18-8
modes of operation, 18-3
op-level module memory map, 18-4
TLB0CFG (TLB0 configuration register), see e500 core, registers
TLB1CFG (TLB1 configuration register), see e500 core, registers
TMS (JTAG test mode select) signal, 25-9
Trace buffer
and watchpoint monitor, block diagram, 25-1
as a second watchpoint monitor, 25-28
functional description, 25-28–25-30
initialization, 25-30
modes of triggering and arming, 25-5
overview, 25-1
register descriptions, 25-16–25-23
by acronym, see Register Index
see also Watchpoint monitor, 25-5
traced data formats relative to TBCR1[IFSEL]

ECM trace buffer entry, 25-29
PCI trace buffer entry, 25-29
TRIG_IN (watchpoint trigger in) signal, 25-8, 25-12, 25-18
TRIG_OUT (watchpoint trigger out) signal, 25-8, 25-24
TRST (JTAG test reset) signal, 25-9
TSEC
block diagram, 16-4
buffer descriptors, 16-66
receive buffer descriptor (RxBD), 16-69
transmit data buffer descriptor (TxBD), 16-67
clocks
inputs and outputs, 16-7
management clock out (EC_MDC), 16-42
configuration of interfaces, 16-72
MII interface mode, 16-72
error handling, 16-65–16-66
features, 16-5
functional description, 16-52
gigabit Ethernet channel operation
flow control, 16-62
frame recognition, 16-59
destination address recognition, 16-59
initialization sequence, 16-54
hardware controlled initialization, 16-54
user initialization, 16-54
inter-packet gap time, 16-64
interrupt handling, 16-63
hash function
hash table algorithm, 16-61
hash table effectiveness, 16-61
registers, 16-48
initialization/application information, 16-72
see also TSEC, configuration
interrupts, 16-63
interrupt registers, 16-12–16-17
MAC functionality, 16-33–16-48
configuration, 16-34
CSMA/CD controlling, 16-34
packet collisions, 16-34
packet flow, 16-35
PHY link control, 16-35
registers, 16-36
memory map/register definition, 16-8
detailed memory map, 16-9–16-12
top level module map, 16-8
modes of operation, 16-6
10 Mbps and 100 Mbps MII operation, 16-6
address recognition options, 16-6
full and half-duplex operation, 16-6
internal and external loop back, 16-64
overview, 16-5
physical interface connections, 16-52
media-independent interface (MII), 16-52
register descriptions, 16-12
by acronym, see Register Index, 16-12
FIFO control and status registers, 16-20–16-23
general control and status registers, 16-12
hash function registers, 16-48–16-50
MAC registers, 16-33–16-48
receive control and status registers, 16-30–16-33
transmit control and status registers, 16-23–16-29
signals, 16-6
see also Signals, TSEC
soft reset and reconfiguring procedure, 16-55
TSEC*n*_COL (eTSEC 1–4 collision input) signals, 15-10
TSECn_COL (TSEC 1–2 collision input) signals, 16-7
TSEC*n*_CRS (eTSEC 1–4 carrier sense input/FIFO receiver flow control) signals, 15-10
TSECn_CRS (TSEC 1–2 carrier sense input) signals, 16-7
TSEC*n*_GTX_CLK (eTSEC 1–4 gigabit transmit clock) signals, 15-10
TSEC*n*_RX_CLK (eTSEC 1–4 receive clock) signals, 15-11
TSECn_RX_CLK (TSEC 1–2 receive clock) signals, 16-7
TSEC*n*_RX_DV (eTSEC 1–4 receive data valid) signals, 15-12
TSECn_RX_DV (TSEC 1–2 receive data valid) signals, 16-7
TSEC*n*_RX_ER (eTSEC 1–4 receive error) signals, 15-12
TSECn_RX_ER (TSEC 1–2 receive error) signals, 16-7
TSEC*n*_RXD[7:0] (eTSEC 1–4 receive data in) signals, 15-12
TSECn_RXD[7:0] (TSEC 1–2 receive data in) signals, 16-7
TSEC*n*_TX_CLK (eTSEC 1–4 transmit clock in) signals, 15-13
TSECn_TX_CLK (TSEC 1–2 transmit clock in) signals, 16-7
TSEC*n*_TX_EN (eTSEC 1–4 transmit data valid) signals, 15-13
TSECn_TX_EN (TSEC 1–2 transmit data valid in) signals, 16-8
TSEC*n*_TX_ER (eTSEC 1–4 transmit error) signals, 15-14
TSECn_TX_ER (TSEC 1–2 transmit error in) signals, 16-8
TSEC*n*_TXD[7:0] (eTSEC 1–4 transmit data out) signals, 15-13
TSECn_TXD[7:0] (TSEC 1–2 transmit data out) signals, 16-7
TSR (timer status register), see e500 core, registers

U

UART_CTS[0:1] (DUART clear to send) signals, 13-1, 13-3
UART_RTS[0:1] (DUART request to send) signals, 13-1, 13-3
UART_SIN [0:1] (DUART transmitter serial data in) signals, 13-2, 13-3
UART_SOUT [0:1] (DUART transmitter serial data out) signals, 13-2, 13-3

Universal asynchronous receiver/transmitter, see DUART
UPMCn (user performance monitor counter registers 0–3), see e500 core, registers
UPMGC0 (user performance monitor global control register 0), see e500 core, registers
UPMLCan (user performance monitor local control registers a0–a3), see e500 core, registers
UPMLCbn (user performance monitor local control registers b0–b3), see e500 core, registers
UPWAIT (LBC UPM wait) signal, 14-6, 14-73
USPRG0 (user software-use register 0), see e500 core, registers

V

Voltage selection
  LBC signals, 23-24

W

Watchpoint monitor
  and trace buffer, block diagram, 25-1
  functional description, 25-27–25-28
  initialization, 25-30
  modes of triggering and arming, 25-5
  overview, 25-1
  performance monitor events, 25-28
  register descriptions, 25-11–25-16
    by acronym, see Register Index
  second WM by using trace buffer, 25-28
  see also Trace buffer, 25-5
  signals summary, 25-5
    see also Signals, watchpoint, 25-5

X

XER (integer exception register), see e500 core, registers

Z

ZBT SRAM interface (LBC), 14-97, 14-102

Part I—Overview I
Overview 1
Memory Map 2
Signal Descriptions 3
Reset, Clocking, and Initialization 4
Part II—e500 Core Complex and L2 Cache II
Core Complex Overview 5
Core Register Summary 6
L2 Look-Aside Cache/SRAM 7
Part III—Memory, Security, and I/O Interfaces III
e500 Coherency Module 8
DDR Memory Controllers 9
Programmable Interrupt Controller (PIC) 10
Security Engine (SEC) 3.0 11
I2C Interfaces 12
DUART 13
Enhanced Local Bus Controller 14
Enhanced Three-Speed Ethernet Controllers 15
10/100 Fast Ethernet Controller 16
Pattern Matcher (PM) 1.1 17
Table Lookup Unit 18
DMA Controllers 19
Serial RapidIO Interface 20
PCI Express Interface Controller 21
General Purpose I/O (GPIO) 22
Part IV—Global Functions and Debug IV
Global Utilities 22
Device Performance Monitor 23
Debug Features and Watchpoint Facility 24

Revision History A

Glossary GLO
Index IND

I **Part I—Overview**
1 Overview
2 Memory Map
3 Signal Descriptions
4 Reset, Clocking, and Initialization
II **Part II—e500 Core Complex and L2 Cache**
5 Core Complex Overview
6 Core Register Summary
7 L2 Look-Aside Cache/SRAM
III **Part III—Memory, Security, and I/O Interfaces**
8 e500 Coherency Module
9 DDR Memory Controllers
10 Programmable Interrupt Controller (PIC)
11 Security Engine (SEC) 3.0
12 I2C Interfaces
13 DUART
14 Enhanced Local Bus Controller
15 Enhanced Three-Speed Ethernet Controllers
16 10/100 Fast Ethernet Controller
17 Pattern Matcher (PM) 1.1
18 Table Lookup Unit
19 DMA Controllers
20 Serial RapidIO Interface
21 PCI Express Interface Controller
22 General Purpose I/O (GPIO)
IV **Part IV—Global Functions and Debug**
23 Global Utilities
24 Device Performance Monitor
25 Debug Features and Watchpoint Facility

A Revision History

GLO Glossary
IND Index