



MPC8540xxxx
2/2003
Rev. 3.0

MPC8540 Integrated Processor Preliminary Reference Manual

***Preliminary Draft—Subject to Change Without Notice
Motorola Confidential Proprietary***

HOW TO REACH US:**USA/EUROPE/LOCATIONS NOT LISTED:**

Motorola Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre, 2 Dai King Street
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

<http://www.motorola.com/semiconductors>

DOCUMENT COMMENTS:

FAX (512) 933-2625
Attn: RISC Applications Engineering

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Contents

Paragraph Number	Title	Page Number
	Part I	
	Overview	
	Chapter 1	
	Overview	
1.1	Introduction.....	1-1
1.2	MPC8540 Overview	1-1
1.2.1	Key Features	1-2
1.3	MPC8540 Architecture Overview.....	1-7
1.3.1	e500 Core Overview	1-7
1.3.2	On-Chip Memory Unit.....	1-12
1.3.2.1	On-Chip Memory as Memory-Mapped SRAM.....	1-12
1.3.2.2	On-Chip Memory as L2 Cache	1-12
1.3.3	e500 Coherency Module (ECM).....	1-13
1.3.4	DDR SDRAM Controller	1-13
1.3.5	Programmable Interrupt Controller (PIC).....	1-14
1.3.6	I2C Controller	1-14
1.3.7	Boot Sequencer	1-15
1.3.8	Dual Universal Asynchronous Receiver/Transmitter (DUART).....	1-15
1.3.9	Local Bus Controller.....	1-15
1.3.10	Three Speed Ethernet Controllers (10/100/1000).....	1-16
1.3.11	Integrated DMA	1-16
1.3.12	PCI Controller	1-17
1.3.13	RapidIO Controller	1-17
1.3.13.1	RapidIO Message Unit	1-18
1.3.14	Power Management	1-18
1.3.15	Clocking.....	1-18
1.3.16	Address Map	1-18
1.3.17	OceaN Switch Fabric	1-19
1.4	Processing Across the On-Chip Fabric	1-19
1.5	Data Processing with the e500 Coherency Module	1-20
1.6	MPC8540 Application Examples.....	1-20
1.6.1	MPC8540 Applications.....	1-20

Contents

Paragraph Number	Title	Page Number

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 Window x Base Address Registers (LAWBAR0– LAWBAR7)	2-6
2.2.3.3	Local Access Window x Attributes Registers (LAWAR0–LAWAR7)	2-7
2.2.3.4	Precedence of Local Access Windows	2-8
2.2.3.5	Configuring Local Access Windows	2-8
2.2.3.6	Distinguishing Local Access Windows from Other Mapping Functions	2-8
2.2.3.7	Illegal Interaction between Local Access Windows and DDR SDRAM Chip Selects	2-9
2.2.4	Outbound Address Translation and Mapping Windows	2-9
2.2.5	Inbound Address Translation and Mapping Windows	2-9
2.2.5.1	RapidIO Inbound ATMU	2-9
2.2.5.2	PCI/PCI-X Inbound ATMU	2-10
2.2.5.3	Illegal Interaction between Inbound ATMUs and Local Access Windows	2-10
2.3	Configuration, Control, and Status Register Map	2-10
2.3.1	Accessing CCSR Memory from External Masters	2-11
2.3.2	Organization of CCSR Memory	2-12
2.3.3	General Utilities Registers	2-12
2.3.4	Interrupt Controller and CCSR	2-14
2.3.5	RapidIO and CCSR	2-14
2.3.6	Device-Specific Utilities	2-15
2.4	Complete CCSR Map	2-16

Chapter 3 Signal Descriptions

3.1	Introduction	3-1
3.2	Signals Overview	3-1
3.3	Configuration Signals Sampled at Reset	3-15
3.4	Output Signal States During Reset	3-16

Contents

Paragraph Number	Title	Page Number
Chapter 4		
Reset, Clocking, and Initialization		
4.1	Introduction.....	4-1
4.1.1	Overview.....	4-1
4.2	External Signal Description	4-1
4.2.1	Detailed Signal Descriptions	4-1
4.2.1.1	System Control Signals.....	4-1
4.2.1.2	Clock Signals	4-2
4.3	Memory Map/Register Definition.....	4-3
4.4	Functional Description.....	4-3
4.4.1	Reset Operations	4-3
4.4.1.1	Hard Reset	4-3
4.4.1.2	Soft Reset.....	4-3
4.4.2	Power-On Reset Sequence	4-3
4.4.3	Power-On Reset (POR) Configuration.....	4-5
4.4.3.1	System PLL Ratio.....	4-6
4.4.3.2	e500 Core PLL Ratio	4-6
4.4.3.3	Boot ROM Location	4-7
4.4.3.4	Host/Agent Configuration.....	4-7
4.4.3.5	CPU Boot Configuration.....	4-8
4.4.3.6	Boot Sequencer Configuration.....	4-8
4.4.3.7	TSEC Width.....	4-9
4.4.3.8	TSEC1 Protocol	4-9
4.4.3.9	TSEC2 Protocol	4-10
4.4.3.10	RapidIO Transmit Clock Source.....	4-10
4.4.3.11	RapidIO Device ID	4-10
4.4.3.12	PCI Width Configuration	4-11
4.4.3.13	PCI I/O Impedance	4-11
4.4.3.14	PCI Arbiter Configuration.....	4-11
4.4.3.15	PCI Debug Configuration	4-12
4.4.3.16	PCI-X Configuration.....	4-12
4.4.3.17	Memory Debug Configuration.....	4-12
4.4.3.18	DDR Debug Configuration	4-13
4.4.3.19	PCI/PCI-X Output Hold Configuration	4-13
4.4.3.20	Local Bus Output Hold Configuration.....	4-14
4.4.4	Clocking.....	4-14
4.4.4.1	System Clock/PCI Clock	4-14
4.4.4.2	RapidIO Clocks	4-15
4.4.4.3	Ethernet Clocks.....	4-16
4.4.4.4	Real Time Clock	4-16
4.5	Initialization/Applications Information	4-16

Contents

4.5.1	Local Configuration Control	4-16
4.5.1.1	Accessing Configuration, Control, and Status Registers	4-17
4.5.1.1.1	Updating CCSRBAR	4-17
4.5.1.1.2	Configuration Control and Status Register Base Address Register (CCSRBAR)	4-18
4.5.1.2	Accessing Alternate Configuration Space	4-18
4.5.1.2.1	Alternate Configuration Base Address Register (ALTCBAR)	4-19
4.5.1.2.2	Alternate Configuration Attribute Register (ALTCAR)	4-20
4.5.1.3	Boot Page Translation	4-20
4.5.1.3.1	Boot Page Translation Register (BPTR)	4-21
4.5.2	Boot Sequencer	4-21

Part II **e500 Core Complex and L2 Cache**

Chapter 5 **e500 Core Complex Overview**

5.1	Overview	5-1
5.2	Features	5-5
5.3	Instruction Set	5-11
5.4	Instruction Flow	5-13
5.4.1	Branch Detection and Prediction	5-13
5.4.2	e500 Execution Pipeline	5-14
5.5	Programming Model	5-17
5.6	On-Chip Cache Implementation	5-19
5.7	Interrupts and Exception Handling	5-19
5.7.1	Exception Handling	5-19
5.7.2	Interrupt Classes	5-20
5.7.3	Interrupt Types	5-20
5.7.4	Upper Bound on Interrupt Latencies	5-21
5.7.5	Interrupt Registers	5-22
5.8	Memory Management	5-24
5.8.1	Address Translation	5-26
5.8.2	MMU Assist Registers (MAS1–MAS4 and MAS6)	5-26
5.8.3	Process ID Registers (PID0–PID2)	5-27
5.8.4	TLB Coherency	5-27
5.9	Memory Coherency	5-27
5.9.1	Atomic Update Memory References	5-28
5.9.2	Memory Access Ordering	5-28
5.9.3	Cache Control Instructions	5-28

Contents

Paragraph Number	Title	Page Number
5.9.4	Programmable Page Characteristics	5-28
5.10	Core Complex Bus (CCB)	5-29
5.11	Performance Monitoring	5-29
5.11.1	Global Control Register	5-29
5.11.2	Performance Monitor Counter Registers	5-30
5.11.3	Local Control Registers	5-30
5.12	Legacy Support of PowerPC Architecture	5-30
5.12.1	Instruction Set Compatibility	5-31
5.12.1.1	User Instruction Set	5-31
5.12.1.2	Supervisor Instruction Set.....	5-31
5.12.2	Memory Subsystem	5-31
5.12.3	Exception Handling	5-31
5.12.4	Memory Management.....	5-32
5.12.5	Reset.....	5-32
5.12.6	Other Differences.....	5-32

Chapter 6 e500 Register Summary

6.1	e500 Register Model	6-1
6.2	Register Model for 32-Bit Book E Implementations	6-3
6.2.1	Special-Purpose Registers (SPRs)	6-4
6.3	Registers for Integer 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	6-8
6.4.1	Condition Register (CR)	6-9
6.4.2	Link Register (LR).....	6-10
6.4.3	Count Register (CTR).....	6-10
6.5	Processor Control Registers	6-10
6.5.1	Machine State Register (MSR)	6-10
6.5.2	Processor ID Register (PIR).....	6-12
6.5.3	Processor Version Register (PVR)	6-12
6.6	Timer Registers	6-13
6.6.1	Timer Control Register (TCR).....	6-13
6.6.2	Timer Status Register (TSR).....	6-14
6.6.3	Time Base	6-15
6.6.4	Decrementer Register.....	6-15
6.6.5	Decrementer Auto-Reload Register (DECAR).....	6-15
6.7	Interrupt Registers.....	6-16
6.7.1	Interrupt Registers Defined by Book E.....	6-16
6.7.1.1	Save/Restore Register 0 (SRR0).....	6-16

Contents

Paragraph Number	Title	Page Number
6.7.1.2	Save/Restore Register 1 (SRR1).....	6-16
6.7.1.3	Critical Save/Restore Register 0 (CSRR0)	6-16
6.7.1.4	Critical Save/Restore Register 1 (CSRR1)	6-16
6.7.1.5	Data Exception Address Register (DEAR).....	6-17
6.7.1.6	Interrupt Vector Prefix Register (IVPR).....	6-17
6.7.1.7	Interrupt Vector Offset Registers (IVORs).....	6-17
6.7.1.8	Exception Syndrome Register (ESR)	6-18
6.8	Software-Use SPRs (SPRG0–SPRG7 and USPRG0).....	6-19
6.9	MMU Registers.....	6-20
6.9.1	Process ID Register (PID).....	6-20
6.9.2	MMU Control and Status Register 0 (MMUCSR0)	6-20
6.9.3	MMU Configuration Register (MMUCFG)	6-21
6.9.4	TLB Configuration Registers (TLBnCFG)	6-22
6.9.4.1	TLB0 Configuration Register (TLB0CFG)	6-22
6.9.4.2	TLB Configuration Register 1 (TLB1CFG)	6-22
6.9.5	MMU Assist Registers (MAS0–MAS6)	6-23
6.9.5.1	MAS Register 0 (MAS0)	6-23
6.9.5.2	MAS Register 1 (MAS1)	6-24
6.9.5.3	MAS Register 2 (MAS2)	6-25
6.9.5.4	MAS Register 3 (MAS3)	6-26
6.9.5.5	MAS Register 4 (MAS4)	6-27
6.9.5.6	MAS Register 6 (MAS6)	6-28
6.10	Debug Registers	6-28
6.10.1	Debug Control Registers.....	6-28
6.10.1.1	Debug Control Register 0 (DBCR0).....	6-28
6.10.1.2	Debug Control Register 1	6-30
6.10.1.3	Debug Control Register 2 (DBCR2).....	6-32
6.10.2	Debug Status Register (DBSR).....	6-34
6.10.3	Instruction Address Compare Registers (IAC1–IAC2).....	6-35
6.10.4	Data Address Compare Registers (DAC1–DAC2).....	6-35
6.11	Signal Processing Engine (SPE) APU Registers	6-35
6.11.1	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR).....	6-35
6.11.2	Accumulator (ACC)	6-37
6.12	Performance Monitor Registers	6-37
6.12.1	Global Control Register 0 (PMGC0)	6-38
6.12.2	User Global Control Register 0 (UPMGC0).....	6-39
6.12.3	Local Control A Registers (PMLCa0–PMLCa3).....	6-39
6.12.4	User Local Control A Registers (UPMLCa0–UPMLCa3)	6-40
6.12.5	Local Control B Registers (PMLCb0–PMLCb3)	6-40

Contents

Paragraph Number	Title	Page Number

6.12.6	User Local Control B Registers (UPMLCb0–UPMLCb3).....	6-41
6.12.7	Performance Monitor Counter Registers (PMC0–PMC3).....	6-41

Chapter 7 L2 Look-Aside Cache/SRAM

7.1	L2 Cache Overview	7-1
7.1.1	L2 Cache/SRAM Features	7-2
7.2	Cache Organization.....	7-2
7.3	Memory Map/Register Definition.....	7-5
7.3.1	L2/SRAM Register Descriptions	7-6
7.3.1.1	L2 Control Register (L2CTL).....	7-6
7.3.1.2	L2 Cache External Write Address Registers (L2CEWAR _n).....	7-9
7.3.1.3	L2 Cache External Write Control Registers (L2CEWCR0– L2CEWCR03)	7-10
7.3.1.4	L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR _n).....	7-11
7.3.1.5	L2 Error Registers.....	7-12
7.3.1.5.1	Error Injection Registers.....	7-12
7.3.1.5.2	Error Control and Capture Registers	7-14
7.4	L2 Cache Timing	7-21
7.5	SRAM Coherency	7-22
7.5.1	L2 Cache Coherency Rules.....	7-22
7.5.2	Memory-Mapped SRAM Coherency Rules	7-23
7.6	L2 Cache Locking.....	7-23
7.6.1	Locking the Entire L2 Cache.....	7-23
7.6.2	Locking Programmed Memory Ranges	7-24
7.6.3	Locking Selected Lines.....	7-24
7.6.4	Clearing Locks On Selected Lines.....	7-24
7.6.5	Flash Clearing of Instruction and Data Locks	7-25
7.6.6	Locks With Stale Data	7-25
7.7	L2 Replacement Policy	7-25
7.7.1	PLRU Bit Update Considerations	7-26
7.7.2	Allocation of Lines	7-27
7.8	L2 Cache Operation	7-27
7.8.1	L2 Cache States	7-27
7.8.2	L2 State Transitions	7-28

Contents

Paragraph Number	Title	Page Number

Part III Memory and I/O Interfaces

Chapter 8 e500 Coherency Module

8.1	Introduction.....	8-1
8.2	Overview	8-1
8.2.1	Features.....	8-2
8.3	Memory Map/Register Definition.....	8-2
8.3.1	Register Descriptions	8-3
8.3.1.1	ECM CCB Address Configuration Register (EEBACR)	8-3
8.3.1.2	ECM CCB Port Configuration Register (EEBPCR).....	8-4
8.3.1.3	ECM Error Detect Register (EEDR)	8-5
8.3.1.4	ECM Error Enable Register (EEER)	8-5
8.3.1.5	ECM Error Attributes Capture Register (EEATR).....	8-6
8.3.1.6	ECM Error Address Capture Register (EEADR)	8-8
8.4	Functional Description.....	8-9
8.4.1	I/O Arbiter.....	8-9
8.4.2	CCB Arbiter	8-9
8.4.3	Transaction Queue	8-9
8.4.4	Global Data Multiplexor.....	8-9
8.4.5	CCB Interface	8-9
8.5	Initialization/Application Information	8-10

Chapter 9 DDR Memory Controller

9.1	Features.....	9-2
9.1.1	Modes of Operation	9-3
9.2	External Signal Description	9-3
9.2.1	Signals Overview	9-3
9.2.2	Detailed Signal Descriptions	9-5
9.2.2.1	Memory Interface Signals.....	9-5
9.2.2.2	Clock Interface Signals.....	9-9
9.2.2.3	Debug Signals.....	9-10
9.3	Memory Map/Register Definition.....	9-10
9.3.1	Register Descriptions	9-11
9.3.1.1	Chip Select Memory Bounds (CS _n _BNDS).....	9-11
9.3.1.2	Chip Select Configuration (CS _n _CONFIG)	9-12
9.3.1.3	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1).....	9-13
9.3.1.4	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2).....	9-16

Contents

Paragraph Number	Title	Page Number
9.3.1.5	DDR SDRAM Control Configuration (DDR_SDRAM_CFG)	9-17
9.3.1.6	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)	9-18
9.3.1.7	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL).....	9-19
9.3.1.8	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	9-20
9.3.1.9	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	9-21
9.3.1.10	Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT).....	9-21
9.3.1.11	Memory Data Path Read Capture High (CAPTURE_DATA_HI).....	9-22
9.3.1.12	Memory Data Path Read Capture Low (CAPTURE_DATA_LO).....	9-23
9.3.1.13	Memory Data Path Read Capture ECC (CAPTURE_ECC).....	9-23
9.3.1.14	Memory Error Detect (ERR_DETECT)	9-24
9.3.1.15	Memory Error Disable (ERR_DISABLE).....	9-25
9.3.1.16	Memory Error Interrupt Enable (ERR_INT_EN).....	9-25
9.3.1.17	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES).....	9-26
9.3.1.18	Memory Error Address Capture (CAPTURE_ADDRESS)	9-28
9.3.1.19	Single-Bit ECC Memory Error Management (ERR_SBE)	9-29
9.4	Functional Description.....	9-30
9.4.1	DDR SDRAM Interface Operation.....	9-35
9.4.1.1	Supported DDR SDRAM Organizations	9-36
9.4.2	DDR SDRAM Address Multiplexing	9-37
9.4.3	JEDEC Standard DDR SDRAM Interface Commands	9-38
9.4.4	SDRAM Interface Timing	9-39
9.4.4.1	Clock Distribution.....	9-43
9.4.5	DDR SDRAM Mode-Set Command Timing.....	9-43
9.4.6	DDR SDRAM Registered DIMM Mode	9-44
9.4.7	DDR SDRAM Write Timing Adjustments	9-45
9.4.8	DDR SDRAM Refresh	9-46
9.4.8.1	DDR SDRAM Refresh Timing.....	9-47
9.4.8.2	DDR SDRAM Refresh and Power-Saving Modes	9-48
9.4.8.2.1	Self-Refresh in Sleep Modes	9-48
9.4.9	DDR Data Beat Ordering.....	9-50
9.4.10	Page Mode and Logical Bank Retention	9-50
9.4.11	Error Checking and Correcting (ECC)	9-51
9.4.12	Error Management	9-53
9.5	Initialization/Application Information	9-54
9.5.1	DDR SDRAM Initialization Sequence.....	9-55

Contents

Paragraph Number	Title	Page Number
Chapter 10		
Programmable Interrupt Controller		
10.1	Introduction.....	10-1
10.1.1	Overview.....	10-4
10.1.2	Features.....	10-4
10.1.3	Modes of Operation	10-5
10.1.3.1	Mixed Mode (GCR[M] Set)	10-5
10.1.3.2	Pass-Through Mode (GCR[M] Cleared)	10-5
10.1.4	Internal Interrupt Sources	10-6
10.2	External Signal Description	10-7
10.2.1	Signal Overview.....	10-7
10.2.2	Detailed Signal Descriptions	10-7
10.3	Memory Map/Register Definition.....	10-9
10.3.1	Global Registers.....	10-15
10.3.1.1	Feature Reporting Register (FRR).....	10-15
10.3.1.2	Global Configuration Register (GCR).....	10-16
10.3.1.3	Vendor Identification Register (VIR).....	10-16
10.3.1.4	Processor Initialization Register (PIR)	10-17
10.3.1.5	IPI Vector/Priority Registers (IPIVPR _n).....	10-18
10.3.1.6	Spurious Vector Register (SVR)	10-19
10.3.2	Global Timer Registers	10-20
10.3.2.1	Timer Frequency Reporting Register (TFRR).....	10-20
10.3.2.2	Global Timer Current Count Registers (GTCCR _n)	10-20
10.3.2.3	Global Timer Base Count Registers (GTBCR _n).....	10-21
10.3.2.4	Global Timer Vector/Priority Registers (GTVPR _n).....	10-22
10.3.2.5	Global Timer Destination Registers (GTDR)	10-23
10.3.2.6	Timer Control Register (TCR).....	10-24
10.3.3	Summary Registers	10-26
10.3.3.1	$\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0).....	10-26
10.3.3.2	$\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1).....	10-27
10.3.3.3	Critical Interrupt Summary Register 0 (CISR0)	10-27
10.3.3.4	Critical Interrupt Summary Register 1 (CISR1)	10-28
10.3.4	Performance Monitor Mask Registers (PMMRs).....	10-29
10.3.4.1	Performance Monitor Mask Register (Even) (PMMR0/2/4/6).....	10-29
10.3.4.2	Performance Monitor Mask Registers (Odd) (PMMR1/3/5/7).....	10-30
10.3.5	Message Registers.....	10-30
10.3.5.1	Message Registers (MSGR0–MSGR3)	10-31
10.3.5.2	Message Enable Register (MER).....	10-31
10.3.5.3	Message Status Register (MSR)	10-32
10.3.6	Interrupt Source Configuration Registers	10-33
10.3.6.1	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11).....	10-33

Contents

Paragraph Number	Title	Page Number
10.3.6.2	External Interrupt Destination Registers (EIDR0–EIDR11)	10-34
10.3.6.3	Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR31)	10-35
10.3.6.4	Internal Interrupt Destination Registers (IIDR0–IIDR31).....	10-36
10.3.6.5	Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3).....	10-37
10.3.6.6	Messaging Interrupt Destination Registers (MIDR0–MIDR3)	10-38
10.3.7	Per CPU Registers	10-39
10.3.7.1	Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)	10-41
10.3.7.2	Processor Current Task Priority Register (CTPR)	10-42
10.3.7.3	Who Am I Register (WHOAMI)	10-43
10.3.7.4	Processor Interrupt Acknowledge Register (IACK)	10-44
10.3.7.5	Processor End of Interrupt Register (EOI)	10-45
10.4	Functional Description.....	10-45
10.4.1	Interrupt Source Priority	10-46
10.4.2	Processor Current Task Priority	10-47
10.4.3	Interrupt Acknowledge.....	10-47
10.4.4	Nesting of Interrupts	10-47
10.4.5	Processor Initialization	10-47
10.4.6	Spurious Vector Generation	10-48
10.4.7	Messaging Interrupts.....	10-48
10.4.8	Timers	10-48
10.4.9	Reset.....	10-49
10.5	Initialization/Application Information	10-49
10.5.1	Programming Guidelines	10-49
10.5.1.1	PIC Registers	10-49
10.5.1.2	Changing Interrupt Source Configuration	10-51
10.5.1.3	MSG, IPI, and Timer Priority Levels.....	10-51

Chapter 11 I2C Interface

11.1	Overview.....	11-1
11.1.1	Features	11-2
11.1.2	Modes of Operation	11-3
11.2	External Signal Description	11-3
11.2.1	Signal Overview.....	11-3
11.2.2	Detailed Signal Descriptions	11-4
11.3	Memory Map/Register Definition.....	11-5
11.3.1	Register Descriptions	11-5
11.3.1.1	I2C Address Register (I2CADR)	11-5
11.3.1.2	I2C Frequency Divider Register (I2CFDR).....	11-6
11.3.1.3	I2C Control Register (I2CCR).....	11-7
11.3.1.4	I2C Status Register (I2CSR).....	11-9

Contents

Paragraph Number	Title	Page Number
11.3.1.5	I2C Data Register (I2CDR)	11-10
11.3.1.6	Digital Filter Sampling Rate Register (I2CDFSRR)	11-11
11.4	Functional Description	11-11
11.4.1	START Condition	11-12
11.4.2	Slave Address Transmission	11-12
11.4.3	Repeated START Condition	11-13
11.4.4	STOP Condition	11-13
11.4.5	Arbitration Procedure	11-14
11.4.6	Clock Synchronization	11-14
11.4.7	Handshaking	11-15
11.4.8	Clock Stretching	11-15
11.4.9	Description of I ² C Functional Blocks	11-15
11.4.9.1	Clock Control	11-15
11.4.9.2	Input Synchronization and Digital Filter	11-16
11.4.9.2.1	Input Signal Synchronization	11-16
11.4.9.2.2	Filtering of SCL and SDA Lines	11-16
11.4.9.3	Transaction Monitoring	11-16
11.4.9.4	Arbitration Control	11-16
11.4.9.5	Control Transfer	11-17
11.4.9.6	Address Compare	11-18
11.4.10	Boot Sequencer Mode	11-18
11.4.10.1	EEPROM Calling Address	11-19
11.4.10.2	EEPROM Data Format	11-19
11.5	Initialization/Application Information	11-22
11.5.1	Initialization Sequence	11-22
11.5.2	Generation of START	11-22
11.5.3	Post-Transfer Software Response	11-23
11.5.4	Generation of STOP	11-23
11.5.5	Generation of Repeated START	11-24
11.5.6	Generation of SCL When SDA Low	11-24
11.5.7	Slave Mode Interrupt Service Routine	11-24
11.5.7.1	Slave Transmitter and Received Acknowledge	11-25
11.5.7.2	Loss of Arbitration and Forcing of Slave Mode	11-25
11.5.8	Interrupt Service Routine Flowchart	11-25

Chapter 12 DUART

12.1	Overview	12-1
12.1.1	Features	12-2
12.1.2	Modes of Operation	12-3
12.2	External Signal Description	12-3

Contents

Paragraph Number	Title	Page Number
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	Receiver Buffer Registers (URBR0 and URBR1).....	12-6
12.3.1.2	Transmitter Holding Registers (UTHR0 and UTHR1).....	12-6
12.3.1.3	Divisor Most and Least Significant Byte Registers (UDMB, UDLB).....	12-7
12.3.1.4	Interrupt Enable Register (UIER).....	12-8
12.3.1.5	Interrupt ID Registers (UIIR0 and UIIR1)	12-9
12.3.1.6	FIFO Control Registers (UFCR0 and UFCR1)	12-11
12.3.1.7	Line Control Registers (ULCR0 and ULCR1)	12-12
12.3.1.8	MODEM Control Registers (UMCR0 and UMCR1).....	12-13
12.3.1.9	Line Status Registers (ULSR0 and ULSR1).....	12-14
12.3.1.10	MODEM Status Registers (UMSR0 and UMSR1)	12-16
12.3.1.11	Scratch Registers (USCR0 and USCR1)	12-16
12.3.1.12	Alternate Function Registers (UAFR0 and UAFR1).....	12-17
12.3.1.13	DMA Status Registers (UDSR0 and UDSR1).....	12-17
12.4	Functional Description.....	12-19
12.4.1	Serial Interface.....	12-19
12.4.1.1	START Bit	12-20
12.4.1.2	Data Transfer	12-20
12.4.1.3	Parity Bit.....	12-20
12.4.1.4	STOP Bit.....	12-21
12.4.2	Baud-Rate Generator Logic	12-21
12.4.3	Local Loop-Back Mode	12-21
12.4.4	Errors	12-22
12.4.4.1	Framing Error	12-22
12.4.4.2	Parity Error	12-22
12.4.4.3	Overrun Error.....	12-22
12.4.5	FIFO Mode	12-22
12.4.5.1	FIFO Interrupts	12-23
12.4.5.2	DMA Mode Select.....	12-23
12.4.5.3	Interrupt Control Logic.....	12-23
12.5	DUART Initialization Sequence	12-24

Chapter 13 Local Bus Controller

13.1	Introduction.....	13-1
13.1.1	Overview	13-2
13.1.2	Features.....	13-2
13.1.3	Modes of Operation	13-3

Contents

Paragraph Number	Title	Page Number
13.1.3.1	LBC Bus Clock and Clock Ratios	13-4
13.1.3.2	Source ID Debug Mode	13-4
13.1.4	Power-Down Mode	13-4
13.1.5	References.....	13-4
13.2	External Signal Description	13-5
13.3	Memory Map/Register Definition.....	13-10
13.3.1	Register Descriptions	13-11
13.3.1.1	Base Registers (BR0–BR7)	13-11
13.3.1.2	Option Registers (OR0–OR7).....	13-13
13.3.1.2.1	Address Mask	13-13
13.3.1.2.2	Option Registers (ORx)—GPCM Mode	13-14
13.3.1.2.3	Option Registers (ORx)—UPM Mode	13-17
13.3.1.2.4	Option Registers (ORx)—SDRAM Mode.....	13-18
13.3.1.3	UPM Memory Address Register (MAR)	13-20
13.3.1.4	UPM Mode Registers (MxMR)	13-20
13.3.1.5	Memory Refresh Timer Prescaler Register (MRTPR).....	13-23
13.3.1.6	UPM Data Register (MDR)	13-24
13.3.1.7	SDRAM Machine Mode Register (LSDMR)	13-25
13.3.1.8	UPM Refresh Timer (LURT).....	13-27
13.3.1.9	SDRAM Refresh Timer (LSRT).....	13-28
13.3.1.10	Transfer Error Status Register (LTESR)	13-29
13.3.1.11	Transfer Error Check Disable Register (LTEDR).....	13-31
13.3.1.12	Transfer Error Interrupt Enable Register (LTEIR).....	13-32
13.3.1.13	Transfer Error Attributes Register (LTEATR)	13-33
13.3.1.14	Transfer Error Address Register (LTEAR)	13-34
13.3.1.15	Local Bus Configuration Register (LBCR)	13-34
13.3.1.16	Clock Ratio Register (LCRR).....	13-35
13.4	Functional Description.....	13-37
13.4.1	Basic Architecture.....	13-38
13.4.1.1	Address and Address Space Checking.....	13-38
13.4.1.2	External Address Latch Enable Signal (LALE).....	13-38
13.4.1.3	Data Transfer Acknowledge (TA).....	13-40
13.4.1.4	Data Buffer Control (LBCTL).....	13-41
13.4.1.5	Atomic Operation	13-41
13.4.1.6	Parity Generation and Checking (LDP).....	13-42
13.4.1.7	Bus Monitor	13-42
13.4.2	General-Purpose Chip-Select Machine (GPCM).....	13-42
13.4.2.1	Timing Configuration	13-43
13.4.2.2	Chip-Select Assertion Timing.....	13-47
13.4.2.2.1	Programmable Wait State Configuration	13-48
13.4.2.2.2	Chip-Select and Write Enable Negation Timing	13-48

Contents

Paragraph Number	Title	Page Number
13.4.2.2.3	Relaxed Timing.....	13-49
13.4.2.2.4	Output Enable (LOE) Timing.....	13-52
13.4.2.2.5	Extended Hold Time on Read Accesses	13-52
13.4.2.3	External Access Termination (LGTA)	13-54
13.4.2.4	Boot Chip-Select Operation.....	13-54
13.4.3	SDRAM Machine	13-55
13.4.3.1	Supported SDRAM Configurations	13-55
13.4.3.2	SDRAM Power-On Initialization	13-56
13.4.3.3	Intel PC133 and JEDEC-Standard SDRAM Interface Commands	13-57
13.4.3.4	Page Hit Checking	13-58
13.4.3.5	Page Management.....	13-58
13.4.3.6	SDRAM Address Multiplexing	13-58
13.4.3.7	SDRAM Device-Specific Parameters.....	13-59
13.4.3.7.1	Precharge-to-Activate Interval	13-60
13.4.3.7.2	Activate to Read/Write Interval	13-60
13.4.3.7.3	Column Address to First Data Out—CAS Latency	13-60
13.4.3.7.4	Last Data In to Precharge—Write Recovery	13-61
13.4.3.7.5	Refresh Recovery Interval (RFRC)	13-61
13.4.3.7.6	External Address and Command Buffers (BUFCMD)	13-62
13.4.3.8	SDRAM Interface Timing	13-62
13.4.3.9	SDRAM Read/Write Transactions	13-64
13.4.3.10	SDRAM MODE-SET Command Timing.....	13-65
13.4.3.11	SDRAM Refresh.....	13-65
13.4.3.11.1	SDRAM Refresh Timing.....	13-65
13.4.4	User-Programmable Machines (UPMs).....	13-66
13.4.4.1	UPM Requests	13-67
13.4.4.1.1	Memory Access Requests	13-68
13.4.4.1.2	UPM Refresh Timer Requests	13-69
13.4.4.1.3	Software Requests—RUN Command	13-69
13.4.4.1.4	Exception Requests.....	13-70
13.4.4.2	Programming the UPMs	13-70
13.4.4.3	UPM Signal Timing.....	13-70
13.4.4.4	The RAM Array	13-71
13.4.4.4.1	RAM Words	13-72
13.4.4.4.2	Chip-Select Signal Timing (CST _n).....	13-76
13.4.4.4.3	Byte-Select Signal Timing (BST _n).....	13-76
13.4.4.4.4	General-Purpose Signals (GxTx, GOx).....	13-77
13.4.4.4.5	Loop Control (LOOP)	13-77
13.4.4.4.6	Repeat Execution of Current RAM Word (REDO)	13-78
13.4.4.4.7	Address Multiplexing (AMX)	13-78
13.4.4.4.8	Data Valid and Data Sample Control (UTA)	13-79

Contents

Paragraph Number	Title	Page Number
13.4.4.4.9	LGPL[0:5] Signal Negation (LAST)	13-80
13.4.4.4.10	The Wait Mechanism (WAEN)	13-80
13.4.4.5	Synchronous Sampling of UPWAIT for Early Transfer Acknowledge ...	13-81
13.4.4.6	Extended Hold Time on Read Accesses	13-82
13.4.4.7	Memory System Interface Example Using UPM	13-82
13.5	Initialization/Application Information	13-88
13.5.1	Interfacing to Peripherals	13-88
13.5.1.1	Multiplexed Address/Data Bus and Unmultiplexed Address Signals	13-88
13.5.1.2	Peripheral Hierarchy on the Local Bus	13-89
13.5.1.3	Peripheral Hierarchy on the Local Bus for Very High Bus Speeds	13-89
13.5.1.4	GPCM Timings	13-90
13.5.2	Bus Turnaround	13-91
13.5.2.1	Address Phase after Previous Read	13-91
13.5.2.2	Read Data Phase after Address Phase	13-92
13.5.2.3	Read-Modify-Write Cycle for Parity Protected Memory Banks	13-92
13.5.2.4	UPM Cycles with Additional Address Phases	13-92
13.5.3	Interface to Different Port-Size Devices	13-92
13.5.4	Interfacing to SDRAM	13-94
13.5.4.1	Basic SDRAM Capabilities of the Local Bus	13-94
13.5.4.2	Maximum Amount of SDRAM Supported	13-95
13.5.4.3	SDRAM Machine Limitations	13-96
13.5.4.3.1	Analysis of Maximum Row Number Due to Bank Select Mux	13-96
13.5.4.3.2	Bank Select Signals	13-96
13.5.4.3.3	SDRAM of 128 MBytes	13-96
13.5.4.3.4	256-MByte SDRAM	13-99
13.5.4.3.5	512-MByte SDRAM Memory	13-99
13.5.4.3.6	Power-Down Mode	13-100
13.5.4.3.7	Self- Refresh	13-101
13.5.4.3.8	SDRAM Timing	13-102
13.5.4.4	Parity Support for SDRAM	13-104
13.5.5	Interfacing to ZBT SRAM	13-104
13.5.6	Interfacing to DSP Host Ports	13-106
13.5.6.1	Interfacing to MSC8101 HDI16	13-106
13.5.6.1.1	HDI16 Peripherals	13-106
13.5.6.1.2	Physical Interconnections	13-108
13.5.6.1.3	Supporting Burst Transfers	13-110
13.5.6.1.4	Host 60x Bus:HDI16 Peripheral Interface Hardware Timings	13-110
13.5.6.2	Interfacing to MSC8102 DSI	13-111
13.5.6.2.1	DSI in Asynchronous SRAM-Like Mode	13-111
13.5.6.2.2	DSI in Synchronous Mode	13-114
13.5.6.2.3	Broadcast Accesses	13-121

Contents

Paragraph Number	Title	Page Number
13.5.6.3	Interfacing to EHPI from Texas Instruments TMS320Cxxxx DSPs	13-122
13.5.6.3.1	Expansion to Multiple DSPs.....	13-126

Chapter 14 Three-Speed Ethernet Controller

14.1	Introduction.....	14-1
14.1.1	Three-Speed Ethernet Controller Overview	14-7
14.2	Features	14-7
14.3	Modes of Operation	14-8
14.4	External Signals Description.....	14-9
14.4.1	Detailed Signal Descriptions	14-11
14.4.1.1	TSECn_COL—Collision (Input).....	14-11
14.4.1.2	TSECn_CRS—Carrier Sense (Input)	14-11
14.4.1.3	TSECn_GTX_CLK—Gigabit Transmit Clock (Output).....	14-11
14.4.1.4	EC_GTX_CLK125—Gigabit Transmit 125 MHz Source (Input)	14-12
14.4.1.5	EC_MDC—Management Data Clock (Output)	14-12
14.4.1.6	EC_MDIO—Management Data Input/Output, BIDI (Input/Output).....	14-12
14.4.1.7	TSECn_RX_CLK—Receive Clock (Input).....	14-12
14.4.1.8	TSECn_RX_DV—Receive Data Valid (Input).....	14-12
14.4.1.9	TSECn_RXD[7:0]—Receive Data (Input).....	14-12
14.4.1.10	TSECn_RX_ER—Receive Error (Input).....	14-13
14.4.1.11	TSECn_TX_CLK—Transmit Clock (Input)	14-13
14.4.1.12	TSECn_TXD[7:0]—Transmit Data (Output).....	14-13
14.4.1.13	TSECn_TX_EN—Transmit Data Valid (Output)	14-14
14.4.1.14	TSECn_TX_ER—Transmit Error (Output).....	14-14
14.5	Memory Map/Register Definition.....	14-14
14.5.1	Top-Level Module Memory Map	14-15
14.5.2	Detailed Memory Map—Control/Status Registers.....	14-15
14.5.3	Memory-Mapped Register Descriptions.....	14-21
14.5.3.1	TSEC General Control and Status Registers	14-21
14.5.3.1.1	Interrupt Event Register (IEVENT).....	14-21
14.5.3.1.2	Interrupt Mask Register (IMASK).....	14-24
14.5.3.1.3	Error Disabled Register (EDIS).....	14-25
14.5.3.1.4	Ethernet Control Register (ECNTRL).....	14-26
14.5.3.1.5	Minimum Frame Length Register (MINFLR).....	14-28
14.5.3.1.6	Pause Time Value Register (PTV)	14-28
14.5.3.1.7	DMA Control Register (DMACTRL)	14-29
14.5.3.1.8	TBI Physical Address Register (TBIPA)	14-31
14.5.3.2	TSEC FIFO Control and Status Registers	14-31
14.5.3.2.1	FIFO Transmit Threshold Register (FIFO_TX_THR)	14-32
14.5.3.2.2	FIFO Transmit Starve Register (FIFO_TX_STARVE)	14-33

Contents

Paragraph Number	Title	Page Number
14.5.3.2.3	FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)	14-33
14.5.3.3	TSEC Transmit Control and Status Registers	14-34
14.5.3.3.1	Transmit Control Register (TCTRL)	14-34
14.5.3.3.2	Transmit Status Register (TSTAT)	14-35
14.5.3.3.3	TxBD Data Length Register (TBDLEN)	14-36
14.5.3.3.4	Current Transmit Buffer Descriptor Pointer (CTBPTR)	14-36
14.5.3.3.5	Transmit Descriptor Base Address (TBASE)	14-37
14.5.3.3.6	Out-of-Sequence TxBD Register (OSTBD)	14-38
14.5.3.3.7	Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)	14-40
14.5.3.4	TSEC Receive Control and Status Registers	14-40
14.5.3.4.1	Receive Control Register (RCTRL)	14-41
14.5.3.4.2	Receive Status Register (RSTAT)	14-41
14.5.3.4.3	RxBD Data Length Register (RBDLEN)	14-42
14.5.3.4.4	Current Receive Buffer Descriptor Pointer (CRBPTR)	14-43
14.5.3.4.5	Maximum Receive Buffer Length Register (MRBLR)	14-43
14.5.3.4.6	Receive Buffer Descriptor Pointer (RBPTR)	14-44
14.5.3.4.7	Receive Descriptor Base Address Register (RBASE)	14-45
14.5.3.5	MAC Functionality	14-45
14.5.3.5.1	Configuring the MAC	14-46
14.5.3.5.2	Controlling CSMA/CD	14-46
14.5.3.5.3	Handling Packet Collisions	14-46
14.5.3.5.4	Controlling Packet Flow	14-47
14.5.3.5.5	Controlling PHY Links	14-48
14.5.3.6	MAC Registers	14-48
14.5.3.6.1	MAC Configuration 1 Register (MACCFG1)	14-48
14.5.3.6.2	MAC Configuration 2 Register (MACCFG2)	14-50
14.5.3.6.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)	14-51
14.5.3.6.4	Half-Duplex Register (HAFDUP)	14-52
14.5.3.6.5	Maximum Frame Length Register (MAXFRM)	14-54
14.5.3.6.6	MII Management Configuration Register (MIIMCFG)	14-54
14.5.3.6.7	MII Management Command (MIIMCOM)	14-55
14.5.3.6.8	MII Management Address Register (MIIMADD)	14-56
14.5.3.6.9	MII Management Control Register (MIIMCON)	14-57
14.5.3.6.10	MII Management Status Register (MIIMSTAT)	14-57
14.5.3.6.11	MII Management Indicator Register (MIIMIND)	14-58
14.5.3.6.12	Interface Status Register (IFSTAT)	14-58
14.5.3.6.13	Station Address Part 1 Register (MACSTNADDR1)	14-59
14.5.3.6.14	Station Address Part 2 Register (MACSTNADDR2)	14-60
14.5.3.7	MIB Registers	14-61
14.5.3.7.1	Transmit and Receive 64-Byte Frame Counter (TR64)	14-61

Contents

Paragraph Number	Title	Page Number
14.5.3.7.2	Transmit and Receive 65- to 127-Byte Frame Counter (TR127)	14-62
14.5.3.7.3	Transmit and Receive 128- to 255-Byte Frame Counter (TR255)	14-62
14.5.3.7.4	Transmit and Receive 256- to 511-Byte Frame Counter (TR511)	14-63
14.5.3.7.5	Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)	14-64
14.5.3.7.6	Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)	14-64
14.5.3.7.7	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)	14-65
14.5.3.7.8	Receive Byte Counter (RBYT)	14-66
14.5.3.7.9	Receive Packet Counter (RPKT)	14-66
14.5.3.7.10	Receive FCS Error Counter (RFCS)	14-67
14.5.3.7.11	Receive Multicast Packet Counter (RMCA)	14-68
14.5.3.7.12	Receive Broadcast Packet Counter (RBCA)	14-68
14.5.3.7.13	Receive Control Frame Packet Counter (RXCF)	14-69
14.5.3.7.14	Receive Pause Frame Packet Counter (RXPF)	14-70
14.5.3.7.15	Receive Unknown OPCode Packet Counter (RXUO)	14-70
14.5.3.7.16	Receive Alignment Error Counter (RALN)	14-71
14.5.3.7.17	Receive Frame Length Error Counter (RFLR)	14-72
14.5.3.7.18	Receive Code Error Counter (RCDE)	14-72
14.5.3.7.19	Receive Carrier Sense Error Counter (RCSE)	14-73
14.5.3.7.20	Receive Undersize Packet Counter (RUND)	14-74
14.5.3.7.21	Receive Oversize Packet Counter (ROVR)	14-74
14.5.3.7.22	Receive Fragments Counter (RFRG)	14-75
14.5.3.7.23	Receive Jabber Counter (RJBR)	14-76
14.5.3.7.24	Receive Dropped Packet Counter (RDRP)	14-76
14.5.3.7.25	Transmit Byte Counter (TBYT)	14-77
14.5.3.7.26	Transmit Packet Counter (TPKT)	14-78
14.5.3.7.27	Transmit Multicast Packet Counter (TMCA)	14-78
14.5.3.7.28	Transmit Broadcast Packet Counter (TBCA)	14-79
14.5.3.7.29	Transmit Pause Control Frame Counter (TXPF)	14-80
14.5.3.7.30	Transmit Deferral Packet Counter (TDFR)	14-80
14.5.3.7.31	Transmit Excessive Deferral Packet Counter (TEDF)	14-81
14.5.3.7.32	Transmit Single Collision Packet Counter (TSCL)	14-82
14.5.3.7.33	Transmit Multiple Collision Packet Counter (TMCL)	14-82
14.5.3.7.34	Transmit Late Collision Packet Counter (TLCL)	14-83
14.5.3.7.35	Transmit Excessive Collision Packet Counter (TXCL)	14-84
14.5.3.7.36	Transmit Total Collision Counter (TNCL)	14-84
14.5.3.7.37	Transmit Drop Frame Counter (TDRP)	14-85
14.5.3.7.38	Transmit Jabber Frame Counter (TJBR)	14-86
14.5.3.7.39	Transmit FCS Error Counter (TFCS)	14-86
14.5.3.7.40	Transmit Control Frame Counter (TXCF)	14-87
14.5.3.7.41	Transmit Oversize Frame Counter (TOVR)	14-88

Contents

Paragraph Number	Title	Page Number
14.5.3.7.42	Transmit Undersize Frame Counter (TUND)	14-88
14.5.3.7.43	Transmit Fragment Counter (TFRG)	14-89
14.5.3.7.44	Carry Register One (CAR1)	14-90
14.5.3.7.45	Carry Register Two (CAR2)	14-91
14.5.3.7.46	Carry Mask Register One (CAM1)	14-92
14.5.3.7.47	Carry Mask Register Two (CAM2)	14-93
14.5.3.8	Hash Function Registers	14-95
14.5.3.8.1	Individual Address Registers 0–7 (IADDR _n)	14-95
14.5.3.8.2	Group Address Registers 0–7 (GADDR _n)	14-95
14.5.3.8.3	Attribute Register (ATTR)	14-96
14.5.3.8.4	Attribute Extract Length and Extract Index Register (ATTRELI)	14-97
14.5.4	Ten-Bit Interface (TBI)	14-98
14.5.4.1	TBI MII Set Register Descriptions	14-98
14.5.4.2	Control Register (CR)	14-99
14.5.4.3	Status Register (SR)	14-100
14.5.4.4	AN Advertisement Register (ANA)	14-101
14.5.4.5	AN Link Partner Base Page Ability Register (ANLPBPA)	14-103
14.5.4.6	AN Expansion Register (ANEX)	14-104
14.5.4.7	AN Next Page Transmit Register (ANNPT)	14-105
14.5.4.8	AN Link Partner Ability Next Page Register (ANLPANP)	14-106
14.5.4.9	Extended Status Register (EXST)	14-107
14.5.4.10	Jitter Diagnostics Register (JD)	14-108
14.5.4.11	TBI Control Register (TBICON)	14-109
14.6	Functional Description	14-110
14.6.1	Connecting to Physical Interfaces	14-110
14.6.1.1	Media-Independent Interface (MII)	14-111
14.6.1.2	Gigabit Media-Independent Interface (GMII)	14-111
14.6.1.3	Reduced Gigabit Media-Independent Interface (RGMII)	14-112
14.6.1.4	Ten-Bit Interface (TBI)	14-113
14.6.1.5	Reduced Ten-Bit Interface (RTBI)	14-114
14.6.2	Gigabit Ethernet Channel Operation	14-118
14.6.2.1	Initialization Sequence	14-118
14.6.2.1.1	Hardware Controlled Initialization	14-118
14.6.2.1.2	User Initialization	14-118
14.6.2.2	Soft Reset and Reconfiguring Procedure	14-119
14.6.2.3	Gigabit Ethernet Frame Transmission	14-120
14.6.2.4	Gigabit Ethernet Frame Reception	14-122
14.6.2.5	RMON Support	14-123
14.6.2.6	Frame Recognition	14-123
14.6.2.6.1	Destination Address Recognition	14-123
14.6.2.6.2	Hash Table Algorithm	14-125

Contents

Paragraph Number	Title	Page Number
14.6.2.7	Flow Control	14-126
14.6.2.8	Interrupt Handling	14-126
14.6.2.9	Inter-Packet Gap Time	14-127
14.6.2.10	Internal and External Loop Back	14-128
14.6.2.11	Error-Handling Procedure.....	14-128
14.6.3	Buffer Descriptors.....	14-129
14.6.3.1	Data Buffer Descriptor.....	14-129
14.6.3.2	Transmit Data Buffer Descriptor (TxBD).....	14-131
14.6.3.3	Receive Buffer Descriptor (RxBD).....	14-133
14.7	Application Information.....	14-135
14.7.1	Interface Mode Configuration.....	14-135
14.7.1.1	MII Interface Mode.....	14-135
14.7.1.2	GMII Interface Mode.....	14-139
14.7.1.3	TBI Interface Mode	14-142
14.7.1.4	RGMII Interface Mode	14-145
14.7.1.5	RTBI Interface Mode	14-149

Chapter 15 DMA Controller

15.1	Introduction.....	15-1
15.1.1	Block Diagram	15-1
15.1.2	Overview	15-2
15.1.3	Features	15-2
15.1.4	Modes of Operation	15-2
15.2	External Signal Description	15-5
15.2.1	Signal Overview.....	15-5
15.2.2	Detailed Signal Descriptions	15-6
15.3	Memory Map/Register Definition.....	15-7
15.3.1	Module Memory Map.....	15-7
15.3.2	DMA Register Descriptions.....	15-10
15.3.2.1	Mode Registers (MR _n)	15-10
15.3.2.2	Status Registers (SR _n)	15-13
15.3.2.3	Current Link Descriptor Address Registers (CLNDAR _n)	15-14
15.3.2.4	Source Attributes Registers (SATR _n)	15-16
15.3.2.5	Source Address Registers (SAR _n)	15-18
15.3.2.5.1	Source Address Registers For RapidIO Maintenance Reads (SARM _n)	15-18
15.3.2.6	Destination Attributes Registers (DATR _n)	15-19
15.3.2.7	Destination Address Registers (DAR _n)	15-21
15.3.2.7.1	Destination Address Registers for RapidIO Maintenance Writes (DARM _n).....	15-22

Contents

Paragraph Number	Title	Page Number
15.3.2.8	Byte Count Registers (BCRn).....	15-23
15.3.2.9	Next Link Descriptor Address Registers (NLNDARn)	15-23
15.3.2.10	Current List Descriptor Address Registers (CLSDARn)	15-24
15.3.2.11	Next List Descriptor Address Registers (NLSDARn)	15-25
15.3.2.12	Source Stride Registers (SSRn)	15-26
15.3.2.13	Destination Stride Registers (DSRn)	15-26
15.3.2.14	DMA General Status Register (DGSR)	15-27
15.4	Functional Description	15-29
15.4.1	DMA Channel Operation	15-29
15.4.1.1	Basic DMA Mode Transfer	15-30
15.4.1.1.1	Basic Direct Mode	15-30
15.4.1.1.2	Basic Direct Single-Write Start Mode	15-31
15.4.1.1.3	Basic Chaining Mode	15-31
15.4.1.1.4	Basic Chaining Single-Write Start Mode	15-32
15.4.1.2	Extended DMA Mode Transfer	15-33
15.4.1.2.1	Extended Direct Mode	15-33
15.4.1.2.2	Extended Direct Single-Write Start Mode.....	15-33
15.4.1.2.3	Extended Chaining Mode	15-33
15.4.1.2.4	Extended Chaining Single-Write Start Mode	15-34
15.4.1.3	External Control Mode Transfer	15-34
15.4.1.4	Channel Continue Mode for Cascading Transfer Chains	15-35
15.4.1.4.1	Basic Mode	15-36
15.4.1.4.2	Extended Mode	15-36
15.4.1.5	Channel Abort	15-36
15.4.1.6	Bandwidth Control.....	15-37
15.4.1.7	Channel State	15-37
15.4.1.8	Illustration of Stride Size and Stride Distance.....	15-37
15.4.2	DMA Transfer Interfaces	15-38
15.4.3	DMA Errors	15-38
15.4.4	DMA Descriptors.....	15-38
15.4.5	Limitations and Restrictions	15-41
15.5	DMA System Considerations	15-42
15.5.1	Unusual DMA Scenarios	15-44
15.5.1.1	DMA to e500 Core	15-44
15.5.1.2	DMA to Ethernet	15-44
15.5.1.3	DMA to Configuration and Control Registers.....	15-45
15.5.1.4	DMA to I2C.....	15-46
15.5.1.5	DMA to DUART.....	15-46

Contents

Paragraph Number	Title	Page Number
Chapter 16		
PCI Bus Interface		
16.1	Introduction.....	16-1
16.1.1	Overview.....	16-2
16.1.1.1	MPC8540 as a PCI Initiator.....	16-4
16.1.1.2	MPC8540 as a PCI Target.....	16-4
16.1.2	Features.....	16-4
16.1.3	Modes of Operation	16-5
16.1.3.1	Host/Agent/Agent Lock Mode	16-5
16.1.3.1.1	PCI/PCI-X Signal Output Hold Timing.....	16-6
16.2	External Signal Descriptions.....	16-6
16.3	Memory Map/Register Definitions	16-13
16.3.1	PCI/PCI-X Memory Mapped Registers	16-14
16.3.1.1	PCI/X ATMU Outbound Registers	16-17
16.3.1.1.1	PCI/X Outbound Translation Address Registers (POTAR).....	16-17
16.3.1.1.2	PCI/X Outbound Translation Extended Address Registers (POTEAR)	16-18
16.3.1.1.3	PCI/X Outbound Window Base Address Registers (POWBAR).....	16-19
16.3.1.1.4	PCI/X Outbound Window Attributes Registers (POWAR).....	16-19
16.3.1.2	PCI/X ATMU Inbound Registers	16-21
16.3.1.2.1	PCI/X Inbound Translation Address Registers (PITAR).....	16-22
16.3.1.2.2	PCI/X Inbound Window Base Address Registers (PIWBAR).....	16-23
16.3.1.2.3	PCI/X Inbound Window Base Extended Address Registers (PIWBEAR).....	16-23
16.3.1.2.4	PCI/X Inbound Window Attributes Registers (PIWAR).....	16-24
16.3.1.3	PCI/X Error Management Registers	16-26
16.3.1.3.1	PCI/X Error Detect Register	16-27
16.3.1.3.2	PCI/X Error Capture Disable Register	16-28
16.3.1.3.3	PCI/X Error Enable Register	16-29
16.3.1.3.4	PCI/X Error Attributes Capture Register.....	16-29
16.3.1.3.5	PCI/X Error Address Capture Register.....	16-31
16.3.1.3.6	PCI/X Error Extended Address Capture Register.....	16-31
16.3.1.3.7	PCI/X Error Data Low Capture Register.....	16-32
16.3.1.3.8	PCI/X Error Data High Capture Register	16-32
16.3.1.4	PCI/X Interrupt Acknowledge Register	16-33
16.3.2	PCI/PCI-X Configuration Header	16-33
16.3.2.1	Accessing PCI/X Configuration Header Registers	16-34
16.3.2.2	PCI Agent Mode Configuration Register Access.....	16-35
16.3.2.3	PCI Bus Command Register—Offset 0x04	16-35
16.3.2.4	PCI Bus Status Register—Offset 0x06	16-37
16.3.2.5	PCI Bus Programming Interface Register—Offset 0x09.....	16-38

Contents

Paragraph Number	Title	Page Number
16.3.2.6	PCI Bus Base Class Code Register—Offset 0x0B	16-38
16.3.2.7	PCI Bus Cache Line Size Register—Offset 0x0C	16-39
16.3.2.8	PCI Bus Latency Timer Register—0x0D	16-39
16.3.2.9	PCI Base Address Registers.....	16-40
16.3.2.10	PCI Bus Capabilities Pointer Register	16-42
16.3.2.11	PCI Bus Interrupt Line Register	16-43
16.3.2.12	PCI Bus Interrupt Pin Register	16-43
16.3.2.13	PCI Bus Maximum Grant (MAX GNT) Register.....	16-44
16.3.2.14	PCI Bus Maximum Latency (MAX LAT) Register.....	16-44
16.3.2.15	PCI Bus Function Register (PBFR).....	16-45
16.3.2.16	PCI Bus Arbiter Configuration Register (PBACR).....	16-45
16.3.2.17	PCI-X Next Capabilities ID Register—0x60.....	16-46
16.3.2.18	PCI-X Capability Pointer Register—0x61	16-47
16.3.2.19	PCI-X Command Register—0x62	16-47
16.3.2.20	PCI-X Status Register—0x64.....	16-48
16.4	Functional Description.....	16-49
16.4.1	PCI/PCI-X Bus Arbitration.....	16-49
16.4.1.1	PCI Bus Arbiter Operation.....	16-50
16.4.1.2	PCI Bus Parking.....	16-51
16.4.1.3	Broken Master Lock-Out	16-51
16.4.1.4	Power-Saving Modes and the PCI Arbiter.....	16-52
16.4.2	PCI Bus Protocol	16-52
16.4.2.1	Basic Transfer Control	16-52
16.4.2.2	PCI Bus Commands.....	16-53
16.4.2.3	Addressing	16-54
16.4.2.3.1	Memory Space Addressing	16-55
16.4.2.3.2	I/O Space Addressing	16-56
16.4.2.3.3	Configuration Space Addressing	16-56
16.4.2.4	Device Selection	16-56
16.4.2.5	Byte Alignment.....	16-57
16.4.2.6	Bus Driving and Turnaround	16-57
16.4.2.7	PCI Bus Transactions.....	16-57
16.4.2.7.1	PCI Read Transactions.....	16-58
16.4.2.7.2	PCI Write Transactions.....	16-59
16.4.2.8	Transaction Termination	16-60
16.4.2.8.1	Master-Initiated Termination	16-60
16.4.2.8.2	Target-Initiated Termination	16-61
16.4.2.9	Fast Back-to-Back Transactions	16-64
16.4.2.10	Dual Address Cycles.....	16-64
16.4.2.11	Configuration Cycles	16-66
16.4.2.11.1	PCI Configuration Space Header.....	16-66

Contents

Paragraph Number	Title	Page Number
16.4.2.11.2	Accessing the PCI Configuration Space	16-68
16.4.2.11.3	Type 0 Configuration Translation	16-70
16.4.2.11.4	Type 1 Configuration Translation	16-72
16.4.2.12	Other Bus Transactions	16-72
16.4.2.12.1	Interrupt-Acknowledge Transactions	16-73
16.4.2.12.2	Special-Cycle Transactions	16-73
16.4.2.13	PCI Error Functions	16-74
16.4.2.13.1	PCI Parity	16-74
16.4.2.13.2	Error Reporting	16-76
16.5	Initialization/Application Information	16-76
16.5.1	Power-On Reset Configuration Modes	16-76
16.5.1.1	Host Mode	16-77
16.5.1.2	Agent Mode	16-77
16.5.1.3	Agent Config Lock Mode	16-77
16.5.2	Nonposted Writes in PCI-X	16-77
16.5.3	Bridging between RapidIO and PCI/PCI-X	16-78

Chapter 17 RapidIO Interface

17.1	Introduction	17-1
17.1.1	RapidIO Terminology	17-2
17.1.2	Block Diagram	17-3
17.1.3	RapidIO Interface Features	17-4
17.1.3.1	Supported Features	17-5
17.1.3.2	Features Not Implemented	17-5
17.1.4	RapidIO Modes of Operation	17-5
17.1.4.1	Transmit Clock-Select Mode	17-5
17.1.4.2	CRC Checking Modes	17-5
17.1.4.3	Error Checking Disable Mode	17-6
17.1.4.4	Output Port Enable Mode	17-6
17.1.4.5	Output Port Driver Disable Mode	17-6
17.1.4.6	Accept-All Mode	17-6
17.1.4.7	Packet Response Time-Out Disable Mode	17-6
17.1.4.8	Link Response Time-Out Disable Mode	17-6
17.2	External Signal Description	17-6
17.2.1	Overview	17-7
17.2.2	Detailed Signal Descriptions	17-8
17.3	Memory Map/Register Definition	17-9
17.3.1	RapidIO Registers	17-9
17.3.2	Architectural Registers	17-14
17.3.2.1	Device Identity Capability Register (DIDCAR)	17-14

Contents

Paragraph Number	Title	Page Number
17.3.2.2	Device Information Capability Register (DICAR)	17-14
17.3.2.3	Assembly Identity Capability Register (AIDCAR)	17-15
17.3.2.4	Assembly Information Capability Register (AICAR)	17-16
17.3.2.5	Processing Element Features Capability Register (PEFCAR)	17-16
17.3.2.6	Switch Port Information Capability Register (SPICAR)	17-17
17.3.2.7	Source Operations Capability Register (SOCAR)	17-18
17.3.2.8	Destination Operations Capability Register (DOCAR)	17-19
17.3.2.9	Mailbox Command and Status Register (MSR)	17-21
17.3.2.10	Port-Write and Doorbell Command and Status Register (PWDCSR)	17-22
17.3.2.11	Processing Element Logic Layer Control Command and Status Register (PELLCCSR)	17-23
17.3.2.12	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	17-23
17.3.2.13	Base Device ID Command and Status Register (BDIDCSR)	17-24
17.3.2.14	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	17-25
17.3.2.15	Component Tag Command and Status Register (CTCSR)	17-25
17.3.2.16	8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)	17-26
17.3.2.17	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	17-27
17.3.2.18	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	17-27
17.3.2.20	Port General Control Command and Status Register (PGCCSR)	17-28
17.3.2.21	Port Link Maintenance Request Command and Status Register (PLMREQCSR)	17-29
17.3.2.22	Port Link Maintenance Response Command and Status Register (PLMRESPCSR)	17-29
17.3.2.23	Port Local AckID Status Command and Status Register (PLASCSR)	17-30
17.3.2.24	Port Error and Status Command and Status Register (PESCSR)	17-31
17.3.2.25	Port Control Command and Status Register (PCCSR)	17-32
17.3.3	Implementation Registers	17-33
17.3.3.1	General Registers	17-33
17.3.3.1.1	Configuration Register (CR)	17-33
17.3.3.1.2	Port Configuration Register (PCR)	17-34
17.3.3.1.3	Port Error Injection Register (PEIR)	17-35
17.3.3.2	ATMU Registers	17-36
17.3.3.2.1	RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR _n)	17-36
17.3.3.2.2	RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR _n)	17-38

Contents

Paragraph Number	Title	Page Number
17.3.3.2.3	RapidIO Outbound Window Attributes Registers 0–8 (ROWAR _n).....	17-39
17.3.3.2.4	RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR _n)	17-41
17.3.3.2.5	RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR _n)	17-42
17.3.3.2.6	RapidIO Inbound Window Attributes Registers 0–4 (RIWAR _n).....	17-42
17.3.3.3	Error Management Registers	17-44
17.3.3.3.1	Port Notification/Fatal Error Detect Register (PNFEDR)	17-44
17.3.3.3.2	Port Notification/Fatal Error Disable Register (PNFEDiR)	17-47
17.3.3.3.3	Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)...	17-49
17.3.3.3.4	Port Error Control Register (PECR)	17-50
17.3.3.3.5	Port Error Packet/Control Symbol Register 0 (PEPCSR0)	17-51
17.3.3.3.6	Port Error Packet Register 1 (PEPR1)	17-52
17.3.3.3.7	RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1).....	17-52
17.3.3.3.8	RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2).....	17-53
17.3.3.3.9	RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5).....	17-53
17.3.3.3.10	RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6).....	17-54
17.3.3.3.11	RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ).....	17-55
17.3.3.3.12	RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)	17-55
17.3.3.3.13	RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10).....	17-56
17.3.3.3.14	RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11).....	17-57
17.3.3.3.15	RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13).....	17-57
17.3.3.3.16	RapidIO Port Error Packet Register 2 (PEPR2)	17-58
17.3.3.3.17	RapidIO Packet Error Capture Register 2, Types 1, 2, and 5 (PECR2T125)	17-58
17.3.3.3.18	RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6).....	17-59
17.3.3.3.19	RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8).....	17-60
17.3.3.3.20	RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)	17-60
17.3.3.3.21	RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10).....	17-61
17.3.3.3.22	RapidIO Packet Error Capture Register 2, Types 11 and 13 (PECR2T11).....	17-62
17.3.3.3.23	Port Recoverable Error Detect Register (PREDR)	17-62
17.3.3.3.24	Port Error Recovery Threshold Register (PERTR).....	17-65
17.3.3.3.25	Port Retry Threshold Register (PRTR).....	17-66
17.3.4	RapidIO Message Unit Registers.....	17-67
17.3.4.1	RapidIO Outbound Message Registers.....	17-67
17.3.4.1.1	Outbound Mode Register (OMR).....	17-67
17.3.4.1.2	Outbound Status Register (OSR).....	17-69

Contents

Paragraph Number	Title	Page Number
17.3.4.1.3	Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR).....	17-70
17.3.4.1.4	Outbound Unit Source Address Register (OSAR).....	17-71
17.3.4.1.5	Outbound Destination Port Register (ODPR).....	17-72
17.3.4.1.6	Outbound Destination Attributes Register (ODATR)	17-72
17.3.4.1.7	Outbound Double-Word Count Register (ODCR).....	17-73
17.3.4.1.8	Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR).....	17-74
17.3.4.2	RapidIO Inbound Message Registers	17-75
17.3.4.2.1	Inbound Mode Register (IMR)	17-75
17.3.4.2.2	Inbound Status Register (ISR)	17-77
17.3.4.2.3	Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR).....	17-79
17.3.4.2.4	Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)	17-79
17.3.4.3	RapidIO Doorbell Registers.....	17-80
17.3.4.3.1	Doorbell Mode Register (DMR).....	17-80
17.3.4.3.2	Doorbell Status Register (DSR).....	17-82
17.3.4.3.3	Doorbell Queue/Dequeue Pointer Address Registers (DQDPAR)	17-83
17.3.4.3.4	Doorbell Queue/Enqueue Pointer Address Registers (DQEPAR)	17-84
17.3.4.4	RapidIO Port-Write Registers.....	17-85
17.3.4.4.1	Port-Write Mode Register (PWMR).....	17-85
17.3.4.4.2	Port-Write Status Register (PWSR).....	17-86
17.3.4.4.3	Port-Write Queue Base Address Register (PWQBAR)	17-87
17.4	Functional Description.....	17-88
17.4.1	RapidIO Transaction, Packet, and Control Symbol Summary	17-88
17.5	RapidIO Functionality	17-94
17.5.1	General Device Functionality Checklists.....	17-94
17.5.1.1	General Specification Functionality List	17-94
17.5.1.2	8/16 LP-LVDS Layer Functionality Lists.....	17-95
17.5.1.3	Basic Functionality List	17-95
17.5.1.4	General Device Link Maintenance List	17-97
17.5.1.5	General Device Packet Transmission List	17-97
17.5.1.6	General Device Packet Reception List	17-98
17.5.1.7	General Device Recoverable Errors List	17-100
17.5.1.8	General Device Nonrecoverable Errors List.....	17-102
17.5.2	Common Transport Layer Functionality Lists.....	17-103
17.5.2.1	Common Transport Layer Basic Functionality List	17-103
17.5.2.2	Common Transport Layer Packet Transmission List.....	17-103
17.5.2.3	Common Transport Layer Packet Reception List.....	17-103
17.5.2.4	Common Transport Layer Detectable Errors List	17-104

Contents

Paragraph Number	Title	Page Number
17.5.3	Logical Layer Functionality Lists	17-104
17.5.3.1	Logical Layer Basic Functionality List	17-104
17.5.3.2	Logical Layer Target Transaction Support List	17-105
17.5.3.3	Logical Layer Source Transaction Support List	17-106
17.5.3.4	Logical Layer Detectable Errors List	17-106
17.5.3.5	Logical Layer Extended Functionality Checklist	17-106
17.6	RapidIO Errors.....	17-106
17.7	ATMU (Address Translation and Mapping Unit)	17-112
17.7.1	Outbound ATMU Translation	17-113
17.7.1.1	Outbound ATMU Bypass Mode	17-113
17.7.1.2	Outbound Special Transactions and Requirements	17-113
17.7.2	Inbound ATMU Translation.....	17-114
17.7.2.1	Inbound ATMU LCSBA1CSR Window	17-114
17.7.2.2	Inbound ATMU Bypass Mode	17-114
17.7.2.3	Inbound Special Transactions and Requirements	17-115
17.8	RapidIO Message Unit.....	17-116
17.8.1	Overview	17-116
17.8.2	Features.....	17-116
17.8.3	Modes of Operation	17-117
17.8.4	RapidIO Messaging Description.....	17-117
17.8.4.1	Data Message Controller	17-117
17.8.4.2	Outbox Controller Operation.....	17-118
17.8.4.2.1	Direct Mode Operation.....	17-118
17.8.4.2.2	Chaining Mode Operation	17-118
17.8.4.2.3	Switching Between Direct and Chaining Modes.....	17-122
17.8.4.2.4	Descriptor Format.....	17-122
17.8.4.2.5	Outbox Controller Interrupts	17-123
17.8.4.2.6	Special Error Case Condition	17-123
17.8.4.3	Inbox Controller Operation.....	17-123
17.8.4.3.1	Retry Response Conditions	17-125
17.8.4.3.2	Error Response Conditions	17-125
17.8.4.3.3	Inbox Controller Interrupts	17-125
17.8.4.3.4	Data Message Controller Limitations and Restrictions	17-125
17.8.4.4	Doorbell Message Controller.....	17-125
17.8.4.4.1	Inbound Doorbell Reception	17-126
17.8.4.4.2	Doorbell Queue Entry Format	17-127
17.8.4.4.3	Retry Response Conditions	17-127
17.8.4.4.4	Error Response Conditions	17-127
17.8.4.4.5	Doorbell Controller Interrupts	17-127
17.8.4.5	Port-Write Controller Structure	17-128
17.9	Initialization and Application Information.....	17-129

Contents

Paragraph Number	Title	Page Number
Part IV		
Global Functions and Debug		
Chapter 18		
Global Utilities		
18.1	Introduction.....	18-1
18.1.1	Overview	18-1
18.1.2	Global Utilities Features	18-1
18.1.2.1	Power Management and Device Disables.....	18-1
18.1.2.2	Software Access to Current POR Configuration Settings.....	18-1
18.1.2.3	General-Purpose I/O	18-2
18.1.2.4	Interrupt and Local Bus Signal Multiplexing	18-2
18.1.2.5	Clock Control.....	18-2
18.2	External Signal Description	18-2
18.2.1	Signals Overview	18-2
18.2.2	Detailed Signal Descriptions	18-3
18.3	Memory Map/Register Definition.....	18-4
18.3.1	Register Descriptions	18-5
18.3.1.1	POR PLL Status Register (PORPLLSR).....	18-5
18.3.1.2	POR Boot Mode Status Register (PORBMSR).....	18-6
18.3.1.3	POR I/O Impedance Status and Control Register (PORIMPSCR).....	18-8
18.3.1.4	POR Device Status Register (PORDEVSR).....	18-9
18.3.1.5	POR Debug Mode Status Register (PORDBGMSR)	18-10
18.3.1.6	General-Purpose POR Configuration Register (GPPORCR)	18-11
18.3.1.7	General-Purpose I/O Control Register (GPIOCR)	18-11
18.3.1.8	General-Purpose Output Data Register (GPOUTDR)	18-12
18.3.1.9	General-Purpose Input Data Register (GPINDR).....	18-13
18.3.1.10	Alternate Function Signal Multiplex Control Register (PMUXCR)	18-14
18.3.1.11	Device Disable Register (DEVDISR).....	18-15
18.3.1.12	Power Management Control and Status Register (POWMGTCSR)	18-17
18.3.1.13	Machine Check Summary Register (MCPSUMR).....	18-19
18.3.1.14	Processor Version Register (PVR)	18-20
18.3.1.15	System Version Register (SVR).....	18-20
18.3.1.16	Clock Out Control Register (CLKOCR).....	18-21
18.3.1.17	DDR DLL Control Register (DDRDLLCR)	18-21
18.3.1.18	Local Bus DLL Control Register (LBDLLCR).....	18-22
18.4	Functional Description.....	18-23
18.4.1	Power Management	18-23
18.4.1.1	Dynamic Power Management.....	18-24
18.4.1.2	Shutting Down Unused Blocks.....	18-24
18.4.1.3	Software-Controlled Power-Down States	18-24

Contents

Paragraph Number	Title	Page Number
18.4.1.4	Doze Mode.....	18-24
18.4.1.5	Nap Mode	18-25
18.4.1.6	Sleep Mode	18-25
18.4.1.7	Power Management Control Bits.....	18-25
18.4.1.8	Interrupts and Power Management	18-26
18.4.1.9	Snooping in the Power-Down State	18-26
18.4.1.10	Software Considerations for Power Management	18-26
18.4.2	General-Purpose I/O Signals	18-27
18.4.3	Interrupt and Local Bus Signal Multiplexing	18-27

Chapter 19 Performance Monitor

19.1	Introduction.....	19-1
19.1.1	Overview	19-1
19.1.2	Features.....	19-2
19.2	External Signals	19-3
19.3	Memory Map and Register Definition	19-3
19.3.1	Register Summary.....	19-3
19.3.2	Control Registers	19-4
19.3.2.1	Performance Monitor Global Control Register (PMGC0)	19-5
19.3.2.2	Performance Monitor Local Control Registers (PMLCAn and PMLCBn).....	19-5
19.3.3	Counter Registers.....	19-10
19.3.3.1	Performance Monitor Counters (PMC0, PMC1, PMC2, PMC3, PMC4, PMC5, PMC6, PMC7, PMC8)	19-10
19.4	Functional Description.....	19-12
19.4.1	Performance Monitor Interrupt.....	19-12
19.4.2	Event Counting	19-12
19.4.3	Threshold Events	19-13
19.4.4	Chaining.....	19-14
19.4.5	Triggering	19-14
19.4.6	Burstiness Counting.....	19-15
19.4.7	Performance Monitor Events	19-17
19.4.8	Performance Monitor Examples	19-38

Chapter 20 Debug Features and Watchpoint Facility

20.1	Introduction.....	20-1
20.1.1	Overview	20-2
20.1.2	Features.....	20-3

Contents

Paragraph Number	Title	Page Number
20.1.3	Modes of Operation	20-3
20.1.3.1	Local Bus Interface Debug Modes of Operation	20-4
20.1.3.2	DDR SDRAM Interface Debug Modes of Operation.....	20-4
20.1.3.3	PCI/PCI-X Interface Debug Modes of Operation.....	20-4
20.1.3.4	Watchpoint Monitor Modes of Operation.....	20-4
20.1.3.5	Trace Buffer Modes of Operation.....	20-5
20.2	External Signal Description	20-5
20.2.1	Overview	20-5
20.2.2	Detailed Signal Descriptions	20-6
20.2.2.1	Memory Data Valid (MDVAL)—Output	20-6
20.2.2.2	Memory ECC (MECC[0:7])—Output	20-6
20.2.2.3	Memory Source ID (MSRCID[0:4])—Output	20-7
20.2.2.4	Watchpoint Trigger Signals	20-7
20.2.2.4.1	Trigger In (TRIG_IN)—Input	20-7
20.2.2.4.2	Trigger Out (TRIG_OUT)—Output	20-8
20.2.2.5	PCI Address (PCI_AD[63:59])—Output.....	20-8
20.2.3	Test and Configuration Signals	20-8
20.2.3.1	JTAG Test Clock (TCK)—Input	20-8
20.2.3.2	JTAG Test Data Input (TDI)—Input.....	20-9
20.2.3.3	JTAG Test Data Output (TDO)—Output	20-9
20.2.3.4	JTAG Test Mode Select (TMS)—Input	20-9
20.2.3.5	JTAG Test Reset ($\overline{\text{TRST}}$)—Input.....	20-9
20.2.3.6	LSSD_MODE—Input	20-10
20.2.3.7	L1_TSTCLK—Input	20-10
20.2.3.8	L2_TSTCLK—Input	20-10
20.2.3.9	TEST_SEL—Input	20-10
20.3	Memory Map/Register Definition	20-10
20.3.1	Watchpoint Monitor Register Descriptions	20-11
20.3.1.1	Watchpoint Monitor Control Register 0 (WMCR0).....	20-11
20.3.1.2	Watchpoint Monitor Control Register 1 (WMCR1).....	20-12
20.3.1.3	Watchpoint Monitor Address Register (WMAR)	20-13
20.3.1.4	Watchpoint Monitor Address Mask Register (WMAMR).....	20-14
20.3.1.5	Watchpoint Monitor Transaction Mask Register (WMTMR)	20-15
20.3.1.6	Watchpoint Monitor Status Register (WMSR).....	20-17
20.3.2	Trace Buffer Register Descriptions.....	20-18
20.3.2.1	Trace Buffer Control Register (TBCR0)	20-18
20.3.2.2	Trace Buffer Control Register 1 (TBCR1)	20-20
20.3.2.3	Trace Buffer Address Register (TBAR).....	20-20
20.3.2.4	Trace Buffer Address Mask Register (TBAMR)	20-21
20.3.2.5	Trace Buffer Transaction Mask Register (TBTMR).....	20-22
20.3.2.6	Trace Buffer Status Register (TBSR)	20-22

Contents

Paragraph Number	Title	Page Number
20.3.2.7	Trace Buffer Access Control Register (TBACR).....	20-23
20.3.2.8	Trace Buffer Access Data High Register (TBADHR)	20-24
20.3.2.9	Trace Buffer Access Data Register (TBADR)	20-25
20.3.3	Context ID Registers.....	20-25
20.3.3.1	Programmed Context ID Register (PCIDR)	20-25
20.3.3.2	Current Context ID Register (CCIDR)	20-26
20.3.4	Trigger Out Function	20-27
20.3.4.1	Trigger Out Source Register (TOSR)	20-27
20.4	Functional Description.....	20-27
20.4.1	PCI/PCI-X Interface Debug	20-29
20.4.2	DDR SDRAM Interface Debug.....	20-29
20.4.2.1	Debug Information on Debug Pins	20-29
20.4.2.2	Debug Information on ECC Pins	20-30
20.4.3	Local Bus Interface Debug	20-30
20.4.4	Watchpoint Monitor	20-30
20.4.4.1	Watchpoint Monitor Performance Monitor Events.....	20-30
20.4.5	Trace Buffer	20-31
20.4.5.1	Traced Data (TBCR[TRSEL]).....	20-31
20.5	Initialization	20-33

Chapter 21 10/100 Ethernet Controller

21.1	Introduction.....	21-1
21.1.1	10/100 Ethernet Controller Overview.....	21-5
21.2	10/100 Ethernet Controller Features	21-6
21.3	Modes of Operation	21-6
21.4	External Signals Description.....	21-7
21.4.1	Signals Overview	21-7
21.4.2	Detailed Signal Descriptions	21-8
21.5	Memory Map/Register Definition.....	21-10
21.5.1	Top-Level Module Memory Map	21-11
21.5.2	Detailed Memory Map.....	21-11
21.5.3	Memory Mapped Register Descriptions	21-15
21.5.3.1	10/100 Controller General Control and Status Registers.....	21-15
21.5.3.1.1	Interrupt Event Register (IEVENT).....	21-15
21.5.3.1.2	Interrupt Mask Register (IMASK).....	21-17
21.5.3.1.3	Error Disabled Register (EDIS).....	21-18
21.5.3.1.4	Minimum Frame Length Register (MINFLR).....	21-19
21.5.3.1.5	Pause Time Value Register (PTV)	21-20
21.5.3.1.6	DMA Control Register (DMACTRL)	21-20
21.5.3.1.7	Transmit Control Register (TCTRL)	21-22

Contents

Paragraph Number	Title	Page Number
21.5.3.1.8	Transmit Status Register (TSTAT).....	21-23
21.5.3.1.9	TxBD Data Length Register (TBDLEN).....	21-24
21.5.3.1.10	Current Transmit Buffer Descriptor Pointer (CTBPTR)	21-24
21.5.3.1.11	Transmit Buffer Descriptor Pointer (TBPTR)	21-25
21.5.3.1.12	Transmit Descriptor Base Address (TBASE)	21-25
21.5.3.1.13	Out-of-Sequence TxBD Register (OSTBD)	21-26
21.5.3.1.14	Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)	21-28
21.5.3.2	10/100 Controller Receive Control and Status Registers	21-29
21.5.3.2.1	Receive Control Register (RCTRL)	21-29
21.5.3.2.2	Receive Status Register (RSTAT).....	21-30
21.5.3.2.3	RxBD Data Length Register (RBDLEN)	21-31
21.5.3.2.4	Current Receive Buffer Descriptor Pointer (CRBPTR).....	21-31
21.5.3.2.5	Maximum Receive Buffer Length Register (MRBLR)	21-32
21.5.3.2.6	Receive Buffer Descriptor Pointer (RBPTR)	21-33
21.5.3.2.7	Receive Descriptor Base Address (RBASE).....	21-33
21.5.3.3	MAC Functionality	21-34
21.5.3.3.1	Configuring the MAC	21-34
21.5.3.3.2	Controlling CSMA/CD	21-34
21.5.3.3.3	Handling Packet Collisions.....	21-35
21.5.3.3.4	Controlling Packet Flow	21-36
21.5.3.3.5	Controlling PHY Links.....	21-36
21.5.3.4	MAC Registers.....	21-37
21.5.3.4.1	MAC Configuration 1 Register (MACCFG1)	21-37
21.5.3.4.2	MAC Configuration 2 Register (MACCFG2)	21-38
21.5.3.4.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG).....	21-40
21.5.3.4.4	Half Duplex Register (HAFDUP).....	21-41
21.5.3.4.5	Maximum Frame Length Register (MAXFRM)	21-42
21.5.3.4.6	MII Management Configuration Register (MIIMCFG)	21-42
21.5.3.4.7	MII Management Command (MIIMCOM).....	21-43
21.5.3.4.8	MII Management Address Register (MIIMADD).....	21-44
21.5.3.4.9	MII Management Control Register (MIIMCON).....	21-44
21.5.3.4.10	MII Management Status Register (MIIMSTAT).....	21-45
21.5.3.4.11	MII Management Indicator Register (MIIMIND).....	21-46
21.5.3.4.12	Interface Status Register (IFSTAT).....	21-46
21.5.3.4.13	Station Address Register Part 1 (MACSTNADDR1)	21-47
21.5.3.4.14	Station Address Register Part 2 (MACSTNADDR2)	21-47
21.5.3.5	Hash Function Registers	21-48
21.5.3.5.1	Individual Address Registers 0–7 (IADDR _n).....	21-48
21.5.3.5.2	Group Address Registers 0–7 (GADDR _n)	21-49
21.5.3.5.3	Attribute Register (ATTR).....	21-50
21.5.3.5.4	Attribute Extract Length and Extract Index Register (ATTRELI).....	21-51

Contents

Paragraph Number	Title	Page Number
21.6	Functional Description.....	21-52
21.6.1	Connecting the 10/100 Controller to the Physical Interface.....	21-52
21.6.1.1	MII Interface.....	21-52
21.6.2	10/100 Controller Channel Operation.....	21-54
21.6.2.1	Initialization Sequence.....	21-54
21.6.2.1.1	Hardware Controlled Initialization.....	21-54
21.6.2.1.2	User Initialization.....	21-54
21.6.2.2	10/100 Ethernet Frame Transmission.....	21-55
21.6.2.3	Gigabit Ethernet Frame Reception.....	21-57
21.6.2.4	Frame Recognition.....	21-58
21.6.2.4.1	Destination Address Recognition.....	21-58
21.6.2.4.2	Hash Table Algorithm.....	21-60
21.6.2.5	Flow Control.....	21-61
21.6.2.6	Interrupt Handling.....	21-61
21.6.2.7	Inter-Packet Gap Time.....	21-63
21.6.2.8	Internal and External Loop Back.....	21-63
21.6.2.9	Error-Handling Procedure.....	21-63
21.6.3	Buffer Descriptors.....	21-64
21.6.3.1	Data Buffer Descriptor.....	21-65
21.6.3.2	Transmit Data Buffer Descriptor (TxBD).....	21-66
21.6.3.3	Receive Buffer Descriptor (RxBD).....	21-68

Contents

Paragraph Number	Title	Page Number

Part I

Overview

Part I describes the many features of the MPC8540 integrated processor at an overview level. It contains the following chapters:

- Chapter 1, “Overview”
- Chapter 2, “Memory Map”
- Chapter 3, “Signal Descriptions”
- Chapter 4, “Reset, Clocking, and Initialization”

Chapter 1, “Overview,” provides a high-level description of features and functionality of the MPC8540 integrated communications controller. It describes the MPC8540, its interfaces, and its programming model. This chapter describes the functional operation of the MPC8540 with emphasis on peripheral functions.

Chapter 2, “Memory Map,” describes the memory map of the MPC8540. 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 detailed descriptions of each.

Chapter 3, “Signal Descriptions,” provides a listing of all the external signals, cross-references for signals that serve multiple functions, a listing of 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 sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8540.

Chapter 1

Overview

The MPC8540 integrates a processor that implements the PowerPC architecture with system logic required for networking, storage, and general purpose embedded applications. The MPC8540 is a member of the growing family of products that combine system level support for industry standard interfaces to processors that implement the PowerPC architecture. This chapter provides a high-level description of features and functionality of the MPC8540 integrated microprocessor.

1.1 Introduction

The MPC8540 uses the e500 core and RapidIO interconnect technology to balance processor performance with I/O system throughput. The e500 core, with its 256 Kbytes of level-2 cache, implements the enhanced Book E instruction set architecture and provides unprecedented levels of hardware and software debugging support.

In addition, the MPC8540 offers two integrated 10/100 Mbps/1 Gbps three speed Ethernet controllers (TSECs), a 10/100 maintenance port, a DDR SDRAM memory controller, a 64-bit PCI/PCI-X controller, and an 8-bit RapidIO port, a programmable interrupt controller, an I²C controller, a 4-channel DMA controller, a general purpose I/O port, and two universal asynchronous receiver/transmitters (DUART). The high level of integration in the MPC8540 helps simplify board design and offers significant bandwidth and performance.

1.2 MPC8540 Overview

The following section provides a high-level overview of the MPC8540 features.

Figure 1-1 shows the major functional units within the MPC8540.

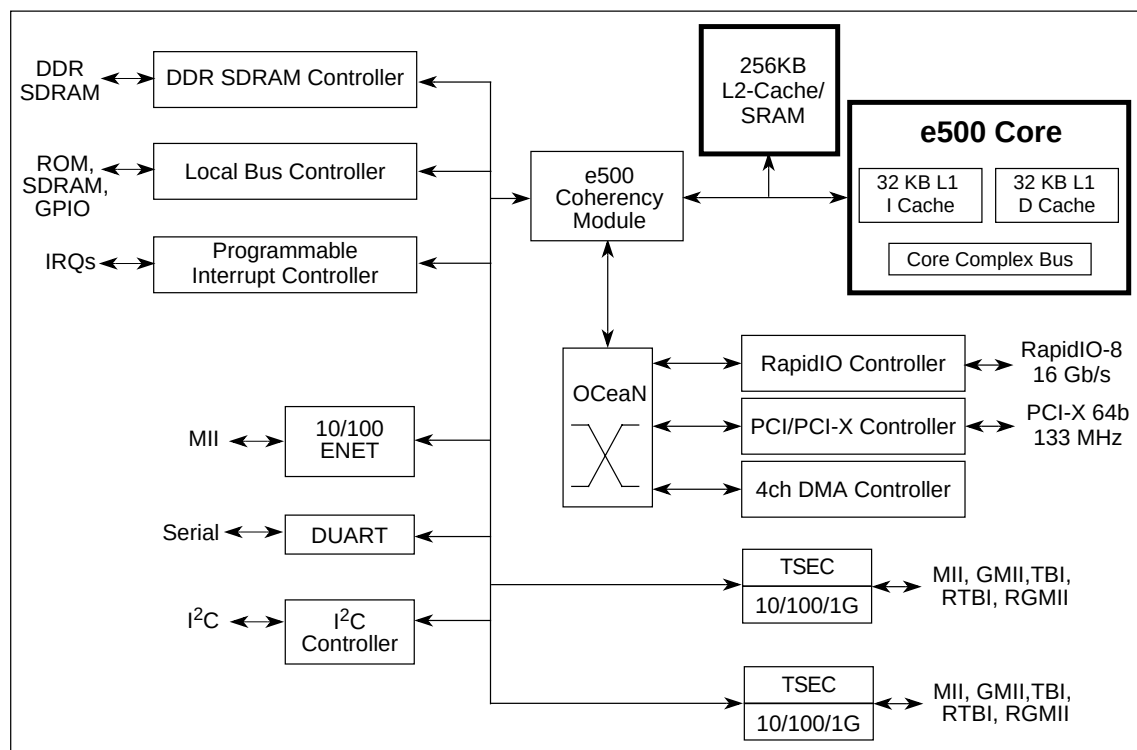


Figure 1-1. MPC8540 Block Diagram

1.2.1 Key Features

The following lists an overview of the MPC8540 feature set.

- Embedded e500 Book E-compatible core
 - High-performance, 32-bit Book E enhanced core that implements the PowerPC architecture
 - Dual issue superscalar, 7-stage pipeline design
 - 32-Kbyte L1 instruction cache and 32-Kbyte L1 data cache with parity protection
 - Lockable L1 caches—entire cache or on a per-line basis
 - Separate locking for instructions and data
 - Single processing engine auxiliary processing unit (SPE APU)
 - Single-precision floating-point single-instruction, multiple-data (SIMD) operations.
 - Memory management unit especially designed for embedded applications

- Enhanced hardware and software debug support
- Dynamic power management
- Performance monitor facility
- 256 Kbytes of on-chip memory
 - Can act as a 256-Kbyte Level 2 cache
 - Can act as a 256-Kbyte or two 128-Kbyte memory-mapped SRAM arrays
 - Can be partitioned into 128-Kbyte L2 cache plus 128-Kbyte SRAM
 - Full ECC support on 64-bit boundary in both cache and SRAM modes
 - SRAM operation supports relocation and is byte-accessible
 - Cache mode supports instruction caching, data caching, or both
 - External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types
 - Eight-way set-associative cache organization (1024 sets of 32-byte cache lines)
 - Supports locking the entire cache or selected lines
 - Individual line locks set and cleared through Book E instructions or by externally mastered transactions
 - Global locking and flash clearing done through writes to L2 configuration registers
 - Instruction and data locks can be flash cleared separately
 - Read and write buffering for internal bus accesses
- Address translation and mapping unit (ATMU)
 - Eight local access windows define mapping within local 32-bit address space
 - Inbound and outbound ATMU's map to larger external address spaces
 - Three inbound windows plus a configuration window on PCI/PCI-X
 - Four inbound windows plus a default and configuration window on RapidIO
 - Four outbound windows plus default translation for PCI
 - Eight outbound windows plus default translation for RapidIO
- DDR memory controller
 - Programmable timing supporting DDR-1 SDRAM
 - 64-bit data interface, up to 333MHz data rate
 - Four banks of memory, each up to 1 Gbyte
 - DRAM chip configurations from 64 Mbit to 1 Gbit with x8/x16 data ports
 - Full ECC support
 - Page mode support (up to 16 simultaneous open pages)
 - Contiguous or discontinuous memory mapping

- Read-modify-write support for RapidIO Atomic increment, decrement, set, and clear transactions
- Sleep mode support for self refresh SDRAM
- Supports auto refreshing
- On-the-fly power management using CKE
- Registered DIMM support
- Fast memory access via JTAG port
- 2.5V SSTL2 compatible I/O
- RapidIO interface unit
 - 8-bit RapidIO I/O and messaging protocols
 - Source synchronous double data rate (DDR) interfaces
 - Supports small type systems (small domain, 8-bit device ID)
 - Supports four priority levels (ordering within a level)
 - Reordering across priority levels
 - Maximum data payload of 256 bytes per packet
 - Packet pacing support at the physical layer
 - CRC protection for packets
 - Supports ATOMIC operations increment, decrement, set, and clear
 - LVDS signalling
- RapidIO compliant message unit
 - One inbound data message structure (inbox)
 - One outbound data message structure (outbox)
 - Supports chaining and direct modes in the outbox
 - Support of up to 16 packets per message
 - Support of up to 256 bytes per packet and up to 4 Kbytes of data per message
 - Supports one inbound doorbell message structure
- Programmable interrupt controller (PIC)
 - Based on the OpenPIC programming model
 - Supports 16 programmable interrupt and processor task priority levels
 - Supports 12 discrete external interrupts
 - Supports 4 message interrupts with 32-bit messages
 - Supports connection of an external interrupt controller such as the 8259
 - Four global high resolution timers/counters that can generate interrupts
 - Support 32 other internal interrupt sources

- Supports fully nested interrupt delivery
- Interrupts can be routed to external pin for external processing
- Interrupts can be routed to the e500 core's standard or critical interrupt inputs
- Interrupt summary registers allow fast identification of interrupt source
- I²C controller
 - Two-wire interface
 - Multiple master support
 - Master or slave I²C mode support
 - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
 - Optionally loads configuration data from serial ROM at reset via the I²C interface
 - Can be used to initialize configuration registers and/or memory
 - Supports extended I²C addressing mode
 - Data integrity checked with preamble signature and CRC
- DUART
 - Two 4-wire interfaces (RXD, TXD, RTS, CTS)
 - Program model compatible with the original 16450 UART and the PC16550D
- Local bus controller (LBC)
 - Multiplexed 32-bit address and data operating up to 166 MHz
 - Eight chip selects support 8 external slaves
 - Up to eight-beat burst transfers
 - The 32-, 16-, and 8-bit port sizes are controlled by an on-chip memory controller
 - Three protocol engines available on a per chip select basis:
 - General purpose chip select machine (GPCM)
 - Three user programmable machines (UPM)
 - Dedicated single data rate SDRAM controller
 - Parity support
 - Default boot ROM chip select with configurable bus width (8-,16-, or 32-bit)
- Three-speed (10/100/1000) Ethernet controllers (TSEC)
 - Dual IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3 AC compliant controllers
 - Support for different Ethernet physical interfaces:
 - 10/100/1000 Mbps IEEE 802.3 GMII
 - 10/100 Mbps IEEE 802.3 MII

- 10 Mbps IEEE 802.3 MII
 - 1000 Mbps IEEE 802.3z TBI
 - 10/100/1000 Mbps RGMII/RTBI
- Full and half-duplex support
- Buffer descriptors are backwards compatible with MPC8260 and MPC860T 10/100 programming models
- 9.6 KByte jumbo frame support
- RMON statistics support
- 2 KByte internal transmit and receive FIFOs
- MII management interface for control and status
- Programmable CRC generation and checking
- OCeaN switch fabric
 - Four-port crossbar packet switch
 - Reorders packets from a source based on priorities
 - Reorders packets to bypass blocked packets
 - Implements starvation avoidance algorithms
 - Supports packets with payloads up to 256 bytes
- Integrated DMA controller
 - 4-channel controller
 - All channels accessible by both the local and remote masters
 - Extended DMA functions (advanced chaining and striding capability)
 - Support for scatter and gather transfers
 - Unaligned transfer capability
 - Interrupt on completed segment, link, list, and error
 - Supports transfers to or from any local memory or I/O port
 - Selectable hardware-enforced coherency (snoop/no-snoop)
 - Ability to start and flow control each DMA channel from external 3-pin interface
 - Ability to launch DMA from single write transaction
- PCI/PCI-X functional unit
 - PCI 2.2 and PCI-X 1.0 compatible
 - 64-bit or 32-bit PCI support at 16 MHz to 66 MHz.
 - 64-bit PCI-X support up to 133 MHz
 - Host and agent mode support
 - 64-bit dual address cycle (DAC) support

- PCI-X supports multiple split transactions
- Supports PCI-to-memory and memory-to-PCI streaming
- Memory prefetching of PCI read accesses
- Support posting of processor-to-PCI and PCI-to-memory writes
- PCI 3.3V compatible
- Selectable hardware-enforced coherency
- Power management
 - Fully static 1.2 V CMOS design with 3.3V and 2.5 V I/O
 - Supports power save modes: nap, doze, and sleep
 - Employs dynamic power management
- System performance monitor
 - Support eight 32-bit counters that count the occurrence of selected events
 - Ability to count up to 512 counter specific events
 - Supports 64 reference events that can be counted on any of the eight counters
 - Supports duration and quantity threshold counting
 - Burstiness feature that permits counting of burst events with a programmable time between bursts
 - Triggering and chaining capability
 - Ability to generate an interrupt on overflow
- System access port
 - Uses JTAG interface and a TAP controller to access entire system memory map
 - Supports 32-bit accesses to configuration registers
 - Supports cache-line burst accesses to main memory
 - Supports large block (4 Kbyte) uploads and downloads
 - Supports continuous bit streaming of entire block for fast upload and download
- IEEE 1149.1-compliant, JTAG boundary scan
- 783 FC-PBGA package

1.3 MPC8540 Architecture Overview

The following section describe the major functional units of the MPC8540.

1.3.1 e500 Core Overview

The MPC8540 uses the e500 microprocessor core complex. The e500 core has an internal PLL, that allows independent optimization of the operating frequencies. The core

frequencies are derived from either the primary PCI clock input or an external oscillator. For information regarding the e500 core refer to the following specifications:

- *EREF: A Reference for Motorola Book E and the e500 Core*

The following is a brief list of some of the key features of the e500 core complex:

- Implements full Book E 32-bit architecture
- Cache structure
 - 32-Kbyte, 32-byte line, 8-way set associative instruction cache
 - 32-Kbyte, 32-byte line, 8-way set associative data cache
 - 1.5-cycle cache array access, 3-cycle load-to-use latency
 - Pseudo-LRU replacement algorithm
 - Copy-back data cache
 - Supports all Book E memory coherency modes*
- 256-Kbytes of on-chip memory
 - L2 cache partitioning is configurable
 - Can act as a 256-Kbyte L2 cache
 - 256-Kbyte array organized as 1024, eight-way sets of 32-byte cache lines
 - Array can be partitioned into 128-Kbyte L2 cache and 128-Kbyte memory mapped SRAM
 - Can act as two 128-Kbyte memory-mapped SRAM arrays or a 256-Kbyte SRAM region
 - SRAM operation is byte-accessible
 - Data ECC on 64-bit boundaries (single-error correction, double-error detection)
 - Tag parity (one bit covering all tag bits)
 - Cache mode supports instruction caching, data caching, or both
 - External masters can force data to be allocated into the cache through programmed memory ranges or special transaction types
 - Separate locking for instructions and data so that locks can be set and cleared separately
 - Supports locking the entire cache or selected lines
 - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges
 - Flash clearing done through writes to L2 configuration registers
 - Locks for the entire cache may be set and cleared by accesses to memory mapped control registers
- Dual-issue superscalar control

- Two-instructions-per-clock peak issue rate to any two distinct EUs
- Precise exception handling
- 7-stage pipeline control
- Decode unit
 - 12-entry instruction queue
 - Full hardware detection of interlocks
 - Decode up to two instructions per cycle
 - Decode serialization control
 - Register dependency resolution and renaming
- Branch processing unit (BPU)
 - Dynamic branch prediction
 - Two-entry branch instruction queue
 - Executes all branch and CR logical instruction
- Completion unit
 - As many as 14 instructions allowed in 14-entry completion queue
 - In-order retirement of up to two instructions per cycle
 - Completion and refetch serialization control
 - Synchronization for all instruction flow changes—interrupts and mispredicted branches
- Two simple units
 - Single-cycle add and subtract
 - Single-cycle shift and rotate
 - Single-cycle logical operations
 - Supports integer signal processing operations
- Multiple-cycle unit (MU)
 - Four-cycle latency for integer and SPE multiplication (including SPE vector and scalar, integer, fractional, and floating-point multiply instructions).
 - Variable-latency divide: 4, 11, 19, and 35 cycles for all Book E and SPE divide instructions. Note that the MU allows divide instructions to bypass the first two MU pipeline stages, freeing those stages for other MU instructions to execute in parallel. Two cycles before the divide instruction needs stage three, the MU signals the GIQ not to issue instructions to it, making stage three (MU-3) available for the divide instruction to reenter the MU pipeline.
 - Four-cycle floating-point multiply
 - Four-cycle floating-point add and subtract

- Signal processing engine APU (SPE APU). The SIMD capability provided by the double wide execution units (MIU, LSU, SIU1) is not a separate execution unit. The hardware that executes 32-bit Book E instructions also executes the lower half of 64-bit SPU instructions.
 - Single-cycle integer add and subtract
 - Single-cycle logical operations
 - Single-cycle shift and rotate
 - 4-cycle integer pipelined multiplies
 - 4-, 11-, 19-, and 35-cycle integer divides
 - 4-cycle single instruction multiple data (SIMD) pipelined multiply-accumulate (MAC)
 - 64-bit accumulator for no-stall MAC operations
 - Single precision floating-point single instruction multiple data (SIMD) operations
- Load/store unit (LSU)
 - 3-cycle load latency
 - Fully pipelined
 - Four-entry load queue allows up to four load misses before stalling
 - Can continue servicing load hits when load queue is full
 - Six-entry store queue allows full pipelining of stores
- Cache coherency
 - Supports both three-state MEI cache coherency and four-state cache coherency: invalid, shared, exclusive, exclusive-modified (MESI)
 - Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)
 - High-speed, on-chip local bus with data tagging
 - 32-bit address bus
 - 60x-like address protocol with address pipelining and retry/copyback
 - Two general-purpose read data, 1 write data bus
 - 128-bit data plus parity/tags (each data bus)
 - Supports out-of-order reads, in-order writes
 - Little to no data bus arbitration logic required for native systems
 - Easily adaptable to 60x-like environments
 - Supports one-level pipelining of addresses with address-retry responses
- Extended exception handling

- Supports Book E interrupt model
 - Interrupt vector prefix register (IVPR)
 - Vector offset registers (IVORs) 0–15 as defined in Book E, plus e500-defined IVORs 32–35
 - Exception syndrome register (ESR)
 - Book E-defined preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfci** instruction
- e500-specific interrupts not defined in Book E architecture
 - SPE APU unavailable exception
 - Vector floating-point data exception
 - Vector floating-point round exception
 - Performance monitor
- Memory management unit
 - 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
 - 256-entry, 2-way set-associative TLB for 4-Kbyte pages
 - Software reload for TLBs
 - Virtual memory support for as much as 4 Gbytes (2^{32}) of virtual memory
 - Real memory support for as much as 4 Gbytes (2^{32}) of physical memory
 - Support for big- and true little-endian memory on a per-page basis
- Power management
 - Low power design
 - 1.2 volt design
 - Internal clock multipliers of 2x, 2.5x, 3x from bus clock
 - Power-saving modes: standby and shutdown
 - Dynamic power management of execution units, caches, and MMUs
- Testability
 - LSSD scan design

- JTAG interface
- ESP support
- ABIST for arrays
- LBIST
- Reliability and serviceability
 - Internal code parity
 - Parity checking on e500 local bus

1.3.2 On-Chip Memory Unit

The MPC8540 contains an internal 256-Kbyte memory array that can be configured as memory-mapped SRAM or as a look-aside L2 cache. The array can be divided into two 128-Kbyte arrays, one of which may be used in cache mode.

The memory controller for this array connects to the core complex bus (CCB) and communicates via 128-bit read and write buses to the e500 core and the MPC8540 system logic.

1.3.2.1 On-Chip Memory as Memory-Mapped SRAM

When the on-chip memory is configured as an SRAM, the 256 Kbytes of memory can be configured to reside at any aligned location in the memory map. It is byte-accessible, and fully ECC protected using read-modify-write transactions for sub-cacheline transactions. I/O devices can access the SRAM by marking transactions global so that they are directed to the core complex bus (CCB).

1.3.2.2 On-Chip Memory as L2 Cache

The MPC8540 on-chip memory arrays include a 256-Kbyte data array, an address tag array and a status array.

The data array is organized as 1024 sets of 8 cache lines. Each cache line size is 32 bytes. Replacement policy within each 8-way set is governed by a pseudo-LRU algorithm.

The data is protected with ECC, while the tag array is protected by parity.

The L2 cache tags are non-blocking for efficient load/store and snooping operations. The L2 cache can be accessed internally while a load miss is pending (allowing hits under misses). Subsequent to a load miss updating the memory, loads and stores can occur to that line on the very next cycle.

The L2 status array stores five bits for each line that are used to determine the status of the line. Different combinations of these bits result in different L2 states. Note that for write-through accesses, there is no modified state. The status bits include:

V -> Valid

S -> Shared

IL -> Instruction Locked

DL -> Data Locked

All accesses to the L2 memory are fully pipelined so back-to-back loads and stores can have single-cycle throughput.

The cache can be configured to allocate instructions-only, data-only, or both. It can also be configured to allocate global I/O writes that correspond to a programmable address window or that use a special transaction type. In this way, DMA engines or I/O devices can force data into the cache.

Line locks can be set in a variety of ways. The Book E architecture defines instructions that explicitly set and clear locks in the L2. These instructions are supported by the core complex and the L2 controller. In addition, the L2 controller can be configured to lock all lines that fall into either of two specified address ranges when the line is allocated. Finally, the entire cache can be locked by writing to a configuration register in the L2 cache controller.

The status array tracks line locks as either instruction locks or data locks for each line, and the status array supports flash clearing of all instruction locks or data locks separately by writes to configuration registers in the cache controller.

1.3.3 e500 Coherency Module (ECM)

The e500 coherency module (ECM) provides a mechanism for I/O-initiated transactions to snoop the bus between the e500 core and the integrated L2 in order to maintain coherency across local cacheable memory. It also provides a flexible switch-type structure for core and I/O-initiated transactions to be routed or dispatched to target modules on the chip.

1.3.4 DDR SDRAM Controller

The MPC8540 supports DDR-I SDRAM up to 166 MHz (333 MHz data rate). The memory interface controls main memory accesses and provides for a maximum of 3.5 Gbytes of main memory. The memory controller can be configured to support the various memory sizes through software initialization of on-chip configuration registers.

The MPC8540 supports a variety of SDRAM configurations. SDRAM banks can be built using DIMMs or directly-attached memory devices. Fifteen multiplexed address signals provide for device densities of 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, and 1 Gbits. Four chip select signals support up to four banks of memory. The MPC8540 supports bank sizes from 64 Mbytes to 1 Gbyte. Nine column address strobes (MDM[0:8]) signals are used to provide byte selection for memory bank writes.

The MPC8540 can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to eight simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save three to four clock cycles from subsequent burst accesses that hit in an active page.

The MPC8540 supports error checking and correction (ECC) for system memory. Using ECC, the MPC8540 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

The MPC8540 can invoke a level of system power management by invoking on-the-fly the CKE SDRAM signal to put the memory into a low-power sleep mode.

1.3.5 Programmable Interrupt Controller (PIC)

The programmable interrupt controller implements the necessary functions to provide a flexible solution for a general purpose interrupt control. The interrupt controller unit implements the logic and programming structures of the OpenPIC architecture. The MPC8540 interrupt controller unit supports its single processor and provides for 12 external interrupts (with fully nested interrupt delivery), four message interrupts, internal-logic driven interrupts, and four global high resolution timers. Up to 16 programmable interrupt priority levels are supported.

The interrupt controller unit can be bypassed to allow use of an external interrupt controller. Inter-processor interrupt communication is supported through the external interrupt and soft reset signals of previous processors that implement the PowerPC architecture.

1.3.6 I²C Controller

The inter-IC (IIC or I²C) 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 I²C allows the MPC8540 to exchange data with other I²C 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. The synchronous, multimaster bus of the I²C allows the connection of additional devices to the bus for expansion and system development.

The I²C 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 I²C controller consists of a transmitter/receiver unit, a clocking unit, and a control unit. The I²C unit supports general broadcast mode and on-chip filtering rejects spikes on the bus.

1.3.7 Boot Sequencer

The MPC8540 provides an boot sequencer that uses the I²C interface to access an external serial ROM and loads the data into the MPC8540's configuration registers. The boot sequencer is enabled by a configuration pin sampled when the MPC8540's hardware reset signal is negated. If enabled, the boot sequencer holds The MPC8540's processor core in reset until the boot sequence is complete. If the boot sequencer is not enabled, the processor core exits reset and fetches boot code in default configurations.

1.3.8 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8540 includes a DUART intended for use in maintenance, bring up, and debug of systems. The MPC8540 provides a standard 4-wire handshake (TXD, RXD, RTS, 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 MPC8540's DUART supports full duplex operation. It is compatible with the PC16450 and PC16550 programming models. 16-byte FIFO's 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 bit, baud rate) are also software selectable.

1.3.9 Local Bus Controller

The MPC8540 local bus port allows connections with 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 SDRAM controller provides access to standard SDRAM. Each chip select can be configured so that the associated chip interface can be controlled by the GPCM, UPM, or SDRAM controller. All may exist in the same system.

The GPCM provides a flexible asynchronous interface to SRAM, EPROM, FEPRM, ROM, and other devices such as asynchronous DSP host interfaces and CAMs. Minimal glue logic is required. Handshake signals can be configured to transition on fractions of the system clock. The GPCM does not support bursting.

The UPM allows an extremely flexible interface in which the programmer configures each of a set of general purpose protocol signals by writing the transition pattern into a memory

array. The UPM supports synchronous and bursting interfaces. It also supports multiplexed addressing so that a simple DRAM interface can be implemented. The UPM is entirely flexible in order to provide a very high degree of customization with respect to both asynchronous and burst synchronous interfaces, which permits glueless or almost glueless connection to burst SRAM, custom ASIC, and synchronous DSP interfaces.

The local bus interface unit provides a synchronous DRAM (SDRAM) machine that provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices. An internal DLL (delay-locked loop) for bus clock generation ensures improved data setup margins for board designs. The SDRAM machine can optimize burst transfers and exploits interleaving to maximize data transfer bandwidth and minimize access latency. Programmable row and column address multiplexing allows a variety of SDRAM configurations and sizes to be supported without hardware changes.

1.3.10 Three Speed Ethernet Controllers (10/100/1000)

Each three speed Ethernet controller (TSEC) incorporates a media access control sublayer (MAC) that supports 10, 100, and 1000 Mbps Ethernet/802.3 networks with MII, GMII, RGMII, RTBI, and TBI physical interfaces. The TSEC includes 2-Kbyte receive and transmit FIFOs, and DMA functions.

The buffer descriptors are based upon the MPC8260 and MPC860T 10/100 programming models.

The MPC8540 TSEC supports programmable CRC generation and checking, RMON statistics, and jumbo frames up to 9.6 Kbytes.

1.3.11 Integrated DMA

The MPC8540 DMA engine is capable of transferring blocks of data from any legal address range to any other legal address range. Therefore, it can perform a DMA transfer between any of its I/O or memory ports or even between two devices or locations on the same port.

The DMA 4-channel controller allows chaining (both extended and direct) through local memory-mapped chain descriptors. Scattering, gathering, and unaligned 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 `DMA_DREQ`, `DMA_DACK`, and `DMA_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 directory 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).

1.3.12 PCI Controller

The MPC8540 64-bit PCI controller is compatible with the *PCI Local Bus Specification, Revision 2.2* and the *PCI-X Addendum, Revision 1.0*. The interface can function as a host or agent bridge interface in either PCI or PCI-X mode. Both PCI and PCI-X modes support 64-bit addressing and 32-bit or 64-bit data buses.

As a master, the MPC8540 supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI configuration space. Also, the MPC8540 can generate PCI special-cycle and interrupt-acknowledge commands. As a target, the MPC8540 supports read and write operations to system memory as well as configuration accesses.

PCI-X functionality includes split transaction support for four outstanding split transactions. Split response data is returned in order without interleaving. As a target, the MPC8540 supports all PCI-X sizes. As a master it will internally combine transactions up to 256 bytes.

An internal arbiter can be used to support up to 5 external masters. A round robin arbitration algorithm with two priority levels is used

1.3.13 RapidIO Controller

The RapidIO interconnect unit on the MPC8540 is based on the Rev. 1.1 “RapidIO Interconnect Specification.” 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 on the MPC8540 supports the I/O and message-passing logical specifications, the common transport specification, and the 8/16 LP-LVDS physical layer specification of the RapidIO interconnect specification. It does not support the globally shared memory logical specification.

Highlights of the implementation include: support for four priority levels and ordering within a priority level, CRC error management, 32- to 256-byte transactions and 8-bit data width ports.

The physical layer of the RapidIO unit can operate at up to 500 MHz. Since the interface is defined as a source synchronous, double data rate, LVDS signaling interconnect, the theoretical unidirectional peak bandwidth is 8 Gbps. Receive and transmit ports operate independently, resulting in an aggregate theoretical bandwidth of 16 Gbps.

1.3.13.1 RapidIO Message Unit

The MPC8540's RapidIO messaging supports one inbox/outbox structure for data and one doorbell structure for messages. 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.

1.3.14 Power Management

In addition to low voltage operation and dynamic power management in its execution units, the MPC8540 supports 4 power down modes. The doze, nap, and sleep modes can be entered under software control in the e500 core or by external masters accessing configuration registers.

Doze mode suspends execution of instructions in the e500 core. The core is left in a standby mode in which cache snooping and timebase interrupts are still enabled. System 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 timebase, which can be disabled separately. No snooping is performed in nap mode, but the system logic external to the processor core is fully functional.

Sleep mode shuts down not only the e500 core, but also all of the MPC8540 I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the part can be awakened.

1.3.15 Clocking

The MPC8540 takes in the PCI_CLK/SYSCLK signal as an input to the platform PLL, and multiplies it by an integer from 1 to 16 to generate the internal platform clock. This platform clock is the same frequency as the DDR DRAM data rate (e.g. 266 MHz, or 333 MHz). The core complex bus (CCB) and L2 cache also run at this frequency. The e500 core uses the platform clock as an input to its PLL, which multiplies it again by 2, 2.5, 3, or 3.5 to generate the core clock.

DLL's are used in the DDR DRAM controller and the local bus memory controller (LBC) to generate memory clocks. Six differential clock pairs are generated for DDR DRAMs. Two clock outputs are generated for the local bus controller.

The RapidIO transmit clock may be sourced from one of three locations: the platform clock, the RapidIO receive clock, or a special differential clock input. This input is designed to receive inputs from an external clock synthesis chip driving a clock up to 500 MHz.

1.3.16 Address Map

The MPC8540 supports a flexible physical address map. Conceptually, the address map consists of local space and external address space. The local address map is 4 Gbytes. The

MPC8540 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 unit (ATMU). Both inbound and outbound translation windows are provided. The ATMU allows the MPC8540 to be part of larger address maps such as the PCI 64-bit address environment and the RapidIO environment.

1.3.17 Ocean Switch Fabric

In order to reduce the strain on the core interconnects with the addition of new functional blocks in this generation of the PowerQUICC family, an on-chip non-blocking crossbar switch fabric called OCeaN (on-chip network) has been integrated to decrease contention, decrease latency, and increase bandwidth. This revolutionary non-blocking crossbar fabric allows for full-duplex port connections at 128 Gbps concurrent throughput and independent per-port transaction queuing and flow control.

1.4 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 MPC8540. 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 ATMU processes 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 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.

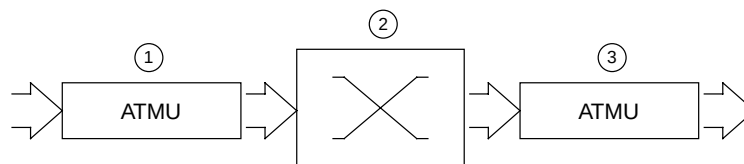


Figure 1-2. Processing Transactions Across the On-Chip Fabric

1.5 Data Processing with the e500 Coherency Module

Processing via the ECM is similar to processing between the CPM and local bus or 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 (except in the case where the CPM is directly accessing the local bus). However, simply because the ECM is used does not make transactions across it coherent. The e500 and L2 cache are snooped to maintain coherency only if the transaction across the ECM is recognized as global (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 transactions across the ECM are coherent transactions; all others (between the CPM and the local bus, and across the on-chip fabric) are non-coherent.

1.6 MPC8540 Application Examples

The following section provides block diagrams of different MPC8540 applications. The MPC8540 is a very flexible device and can be configured to meet many system application needs. In order to build a system, many factors should be considered.

1.6.1 MPC8540 Applications

The MPC8540 can be used for control processing in applications such as routers, switches, network memory, internet access devices, storage systems, and image display systems (IAD). Figure 1-3 shows an MPC8540 as a control plane processor in a high performance communication system using symmetric multi-processing (SMP).

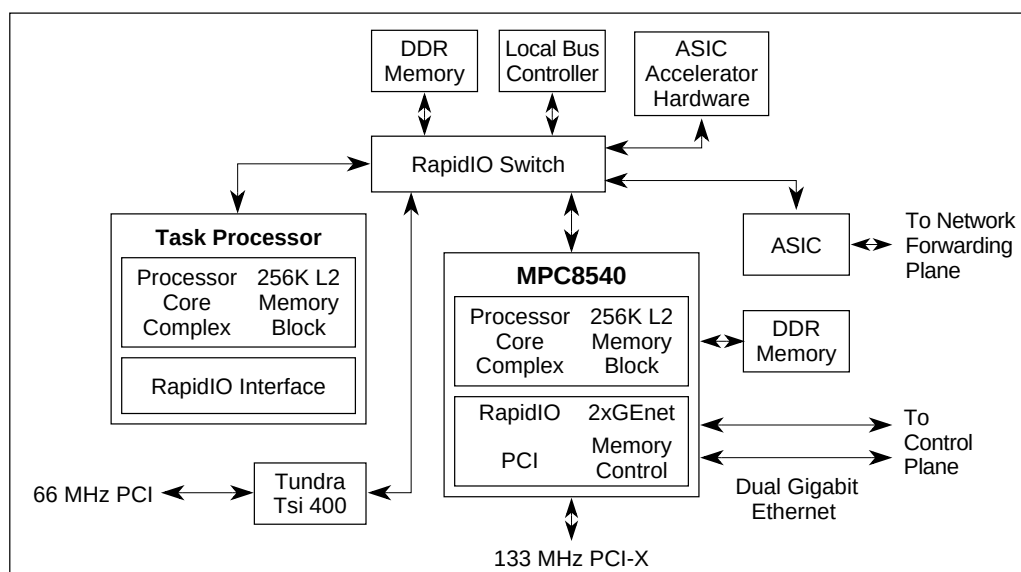


Figure 1-3. High Performance Communication System Using SMP

Figure 1-4 shows the MPC8540 as part of a high-end network card used in a system area network that is enabled by PCI/PCI-X.

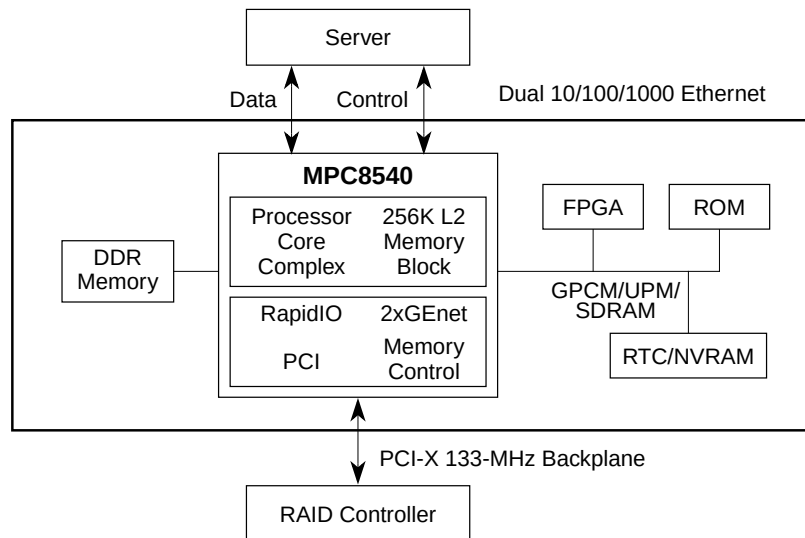


Figure 1-4. High-Performance Communication System Using MPC8540

Figure 1-5 shows the MPC8540 in a redundant arrays of independent disks (RAID) controller application.

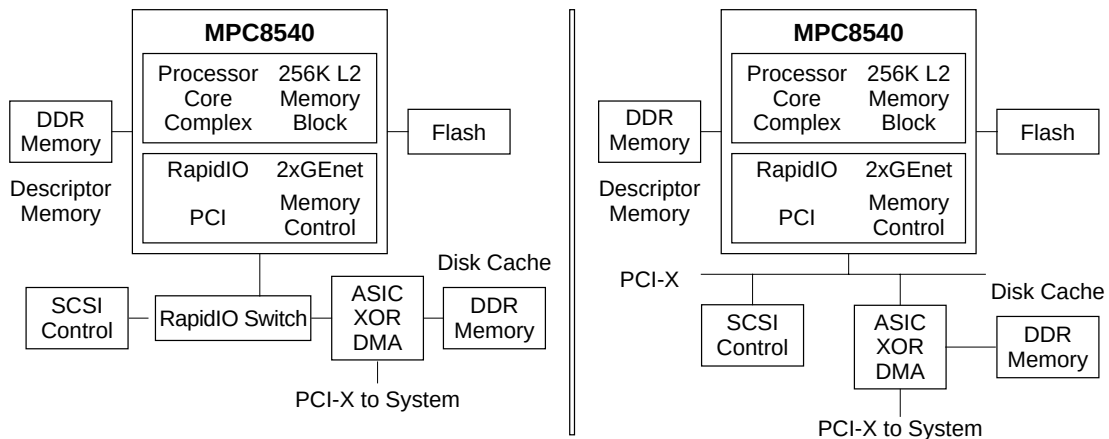


Figure 1-5. RAID Controller Application Using MPC8540

Figure 1-6 illustrates the MPC8540 in a virtual private network (VPN) access that is enabled through RapidIO and Ethernet.

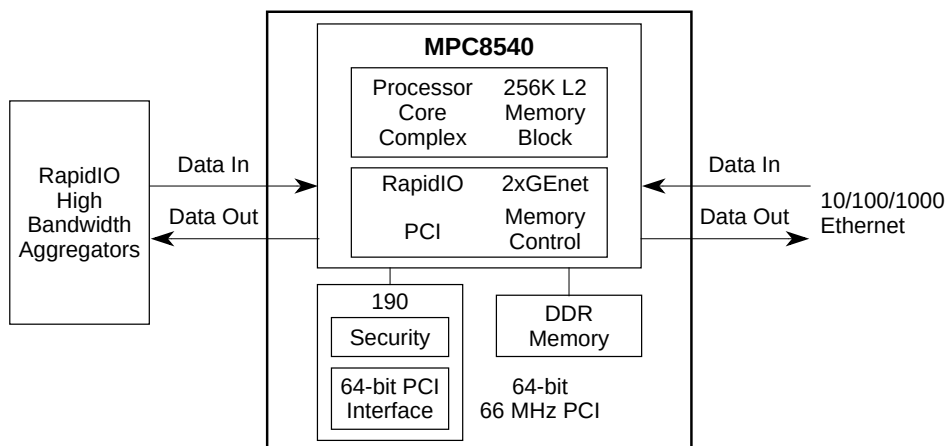


Figure 1-6. VPN Access Point Enabled by RapidIO and Ethernet

Figure 1-7 shows the MPC8540 in a serializer/deserializer (SERDES) data transfer mode application enabled by Ethernet.

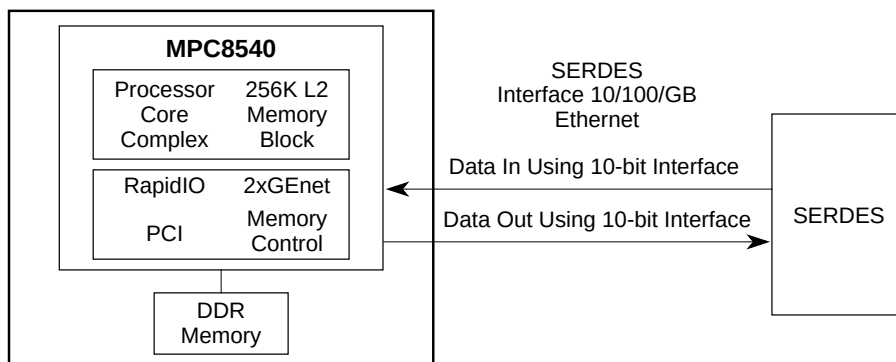


Figure 1-7. MPC8540 with SERDES

Figure 1-8 shows a DSP farm enabled by the MPC8540 using its local bus controller port.

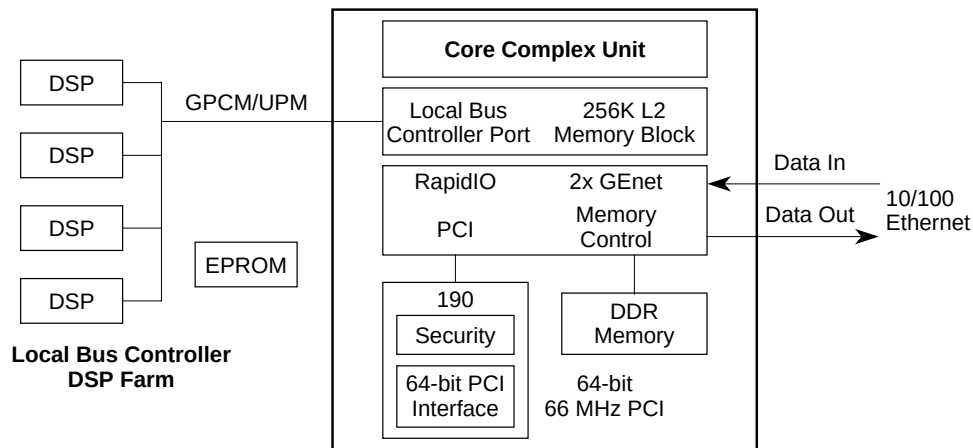


Figure 1-8. MPC8540 with a DSP Farm Enabled by General Purpose Chip Select Machine or User Programmable Machine on the Local Bus Controller Port

Chapter 2

Memory Map

This chapter describes the memory map of the MPC8540. 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 detailed descriptions of each.

2.1 Local Memory Map Overview and Example

The MPC8540 provides an extremely flexible local memory map. The local memory map refers to the 32-bit address space seen by the processor as it accesses memory and I/O space. Internal DMA engines also see this same local memory map. All memory accessed by the MPC8540's 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 eight local access windows. Each of these windows maps a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. Note that the local access windows do not perform any address translation. The size of each window can be configured from 4 Kbytes to 2 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/PCI-X	0000
Local bus	0100
RapidIO	1100
DDR SDRAM	1111

Figure 2-1 below shows an example memory map.

Local Memory Map Overview and Example

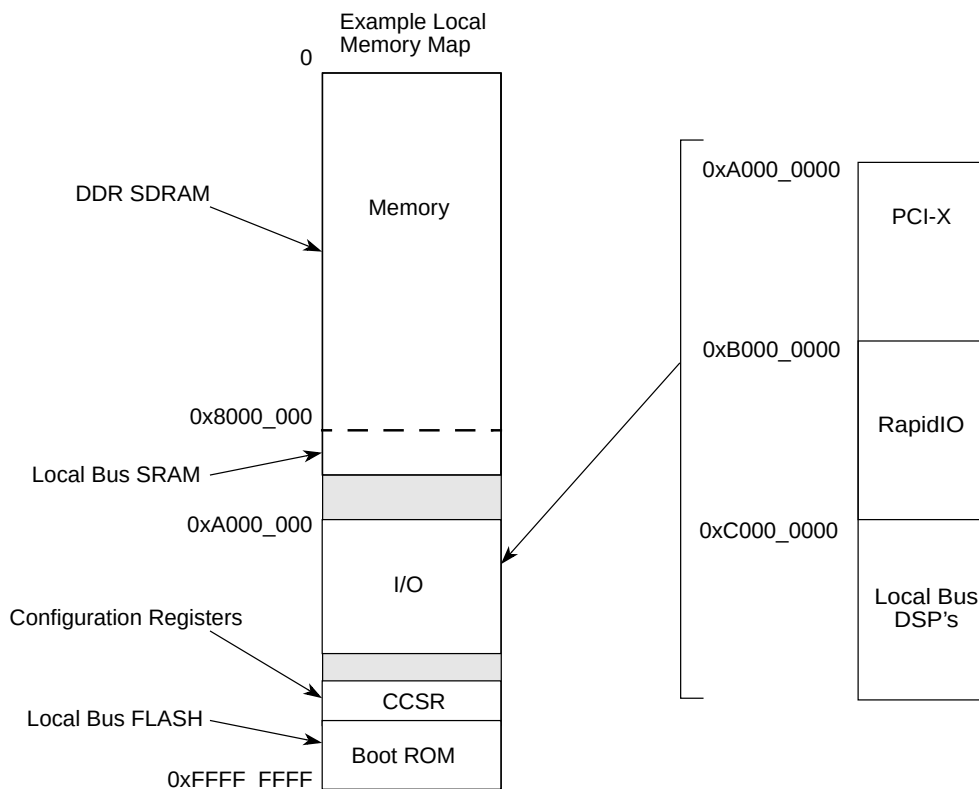


Figure 2-1. Example Local Memory Map

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

Table 2-2. Example Local Access Windows

Window	Base Address	Size	Target Interface
Window 0	0x0000_0000	2GB	0b1111 (DDR SDRAM)
Window 1	0x8000_0000	1MB	0b0100 (local bus)
Window 2	0xA000_0000	256MB	0b0000 (PCI/PCI-X)
Window 3	0xB000_0000	256MB	0b1100 (RapidIO)
Window 4	0xC000_0000	256MB	0b0100 (local bus)
Windows 5-7	Unused		

NOTE

In this example it is not necessary to use a local access window to specify the location of the boot ROM because the boot ROM is in its default location at the highest 8 Mbytes of memory. Neither is it required to define a local access window to describe the range of memory used for memory-mapped registers because this is a fixed 1-Mbyte space pointed to by the CCSRBAR register. See Section 4.5.1.1.2, “Configuration Control and Status Register Base Address Register (CCSRBAR).”

2.2 Address Translation and Mapping

Four distinct types of translation and mapping operations are performed on transactions in the MPC8540. These are as follows:

- Mapping a local address to a target interface
- Assigning attributes to transactions
- Translating the local 32-bit address to an external address space
- Translating external addresses to the local 32-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 32-bit address space to the address spaces of RapidIO or PCI/PCI-X, 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 windows size attribute dictates how many bits are translated.

2.2.1 SRAM Windows

The on-chip memory array of the MPC8540 can be configured as a memory-mapped SRAM of 128 or 256 Kbytes. Configuration registers in the L2 cache controller set the base addresses and sizes for these windows. When enabled, these windows supercede all other mappings of these addresses for processor and global (snooperable) 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 snooperable I/O transactions would map to the SRAM while non-snoopered I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snooperable will results be consistent if the SRAM window overlaps a local access window.

2.2.2 Window into Configuration Space

The configuration, control, and status registers base address register (CCSRBAR) defines a window that is used to access all memory-mapped configuration, control, and status registers. No address translation is done, so there is no associated translation address registers. The window is always enabled with a fixed size of 1 Mbyte, and no other attributes are attached; so there is no associated size/attribute register. This window always takes precedence over all local access windows. For more information, see Section 4.5.1.1.2, "Configuration Control and Status Register 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 32-bit address space with a particular target interface. This allows the internal interconnections of the

MPC8540 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 2.3.3, “General Utilities Registers.” A detailed description of the local access window registers is given below. 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.

Table 2-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0C08	Local access window 0 base address register (LAWBAR0)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C10	Local access window 0 attribute register (LAWSR0)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C28	Local access window 1 base address register (LAWBAR1)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C30	Local access window 1 attribute register (LAWAR1)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C48	Local access window 2 base address register (LAWBAR2)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C50	Local access window 2 attribute register (LAWAR2)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C68	Local access window 3 base address register (LAWBAR3)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C70	Local access window 3 attribute register (LAWAR3)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C88	Local access window 4 base address register (LAWBAR4)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C90	Local access window 4 attribute register (LAWAR4)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0CA8	Local access window 5 base address register (LAWBAR5)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CB0	Local access window 5 attribute register (LAWAR5)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0CC8	Local access window 6 base address register (LAWBAR6)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CD0	Local access window 6 attribute register (LAWSA6)	R/W	0x0000_0000	2.2.3.3/2-7

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

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0CE8	Local access window 7 base address register (LAWBAR7)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CF0	Local access window 7 attribute register (LAWSA7)	R/W	0x0000_0000	2.2.3.3/2-7

2.2.3.2 Local Access Window x Base Address Registers (LAWBAR0–LAWBAR7)

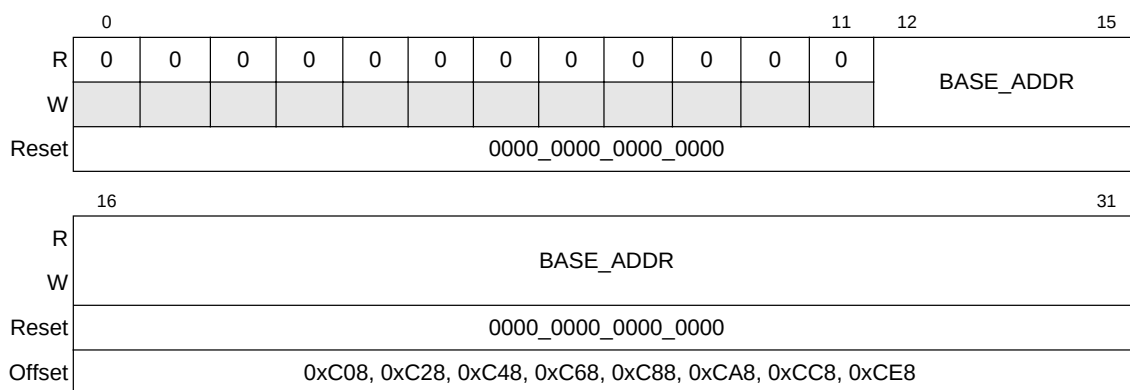
**Figure 2-2. Local Access Window x Base Address Registers (LAWBAR0–LAWBAR7)**

Table 2-5 describes LAWBARx bit settings.

Table 2-5. LAWBARx Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–31	BASE_ADDR	Identifies the 20 most-significant address bits of the base of local access window X. The specified base address should be aligned to the window size, as defined by LAWARN[SIZE].

2.2.3.3 Local Access Window x Attributes Registers (LAWAR0–LAWAR7)

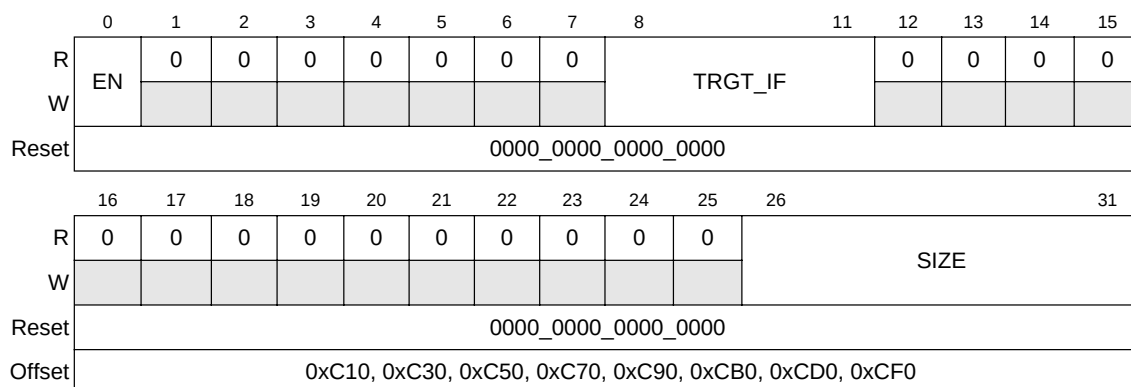


Figure 2-3. Local Access Window x Attributes Registers (LAWAR0–LAWAR7)

Table 2-6 describes LAWARx bit settings.

Table 2-6. LAWARx Bit Settings

Bits	Name	Description
0	EN	0 The local access window X (and all other LAWARX and LAWBARX fields) are disabled. 1 The local access window X is enabled and other LAWARX and LAWBARX fields combine to identify an address range for this window.
1–7	—	Write reserved, read = 0
8–11	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 the CCSRBAR and L2SRBAR registers. These mappings supercede local access window mappings, so configuration registers and SRAM do not appear as a target for local access windows. 0000 PCI/PCI-X 0001 Reserved 0010 Reserved 0011 Reserved 0100 Local bus memory controller 0101–1011 Reserved 1100 RapidIO 1101 Reserved 1110 Reserved 1111 DDR SDRAM
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 011110 2 Gbytes 011111–111111 Reserved

2.2.3.4 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 2-7, local access window 1 governs the mapping of the 1-Mbyte region from 0x7FF0_0000 to 0x7FFF_FFF, even though the window described in local access window 2 also encompasses that memory region.

Table 2-7. Overlapping Local Access Windows

Window	Base Address	Size	Target Interface
Window 1	0x7FF0_0000	1 Mbyte	0b0100 (local bus)
Window 2	0x0000_0000	2 Gbytes	0b1111 (DDR SDRAM)

2.2.3.5 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 the local e500 processor, the read of LAWAR3 should be followed by an **isync** instruction.

2.2.3.6 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 MPC8540's internal interconnects from the transactions source to its target. Once the transaction has arrived at its target interface, that interface controller may do additional mapping. For instance, the DDR SDRAM controller has 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 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.

2.2.3.7 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 controller, then there should not be a valid chip select configured for the same address in the DDR SDRAM controller. Because DDR SDRAM chip selects boundaries are defined by beginning address 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 32-bit address space to the external address space and attributes of a particular I/O interface. On the MPC8540, both the RapidIO and the PCI/PCI-X blocks have outbound address translation and mapping units (ATMUs).

The 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 17.3.3.2, "ATMU Registers," for a detailed description of the RapidIO outbound ATMU windows.

The PCI/PCI-X controller has four outbound ATMU windows plus a default window. The PCI/PCI-X outbound ATMU registers include an extended translation address register so that up to 64 bits of external address space can be supported. See Section 16.3.1.1, "PCI/X ATMU Outbound Registers," for a detailed description of the PCI/PCI-X 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 or RapidIO address space) to the local address space understood by the internal interfaces of the MPC8540. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Both the RapidIO controller and the PCI/PCI-X controller have inbound address translation and mapping units (ATMUs).

2.2.5.1 RapidIO Inbound ATMU

The 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 17.3.3.2, "ATMU Registers," for a detailed description of the RapidIO inbound ATMU windows.

2.2.5.2 PCI/PCI-X Inbound ATMU

The PCI/PCI-X controller has three general inbound ATMU windows plus a dedicated window for memory mapped configuration accesses (PCSRBAR). These windows have a one-to-one correspondence with the base address registers in the PCI/PCI-X programming model. Updating one automatically updates the other. There is no default inbound window; if a PCI/PCI-X address does not match one of the inbound ATMU windows, the MPC8540 does not respond with an assertion of $\overline{\text{PCI\_DEVSEL}}$. See Section 16.3.1.2, “PCI/X ATMU Inbound Registers,” for a detailed description of the PCI/PCI-X inbound ATMU windows.

2.2.5.3 Illegal Interaction between Inbound ATMUs and Local Access Windows

Since 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 the DDR SDRAM memory controller (target interface 0b1111) if the resulting translated local address is mapped to PCI (target interface 0b0000) 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 MPC8540 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 4.5.1.1.2, “Configuration Control and Status Register Base Address Register (CCSRBAR).” The default value for CCSRBAR is 4 Gbytes–9 Mbytes, or 0xFF70_0000.

NOTE

The configuration, control, and status window must not overlap a local access window that maps to the DDR controller. 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 2-4.

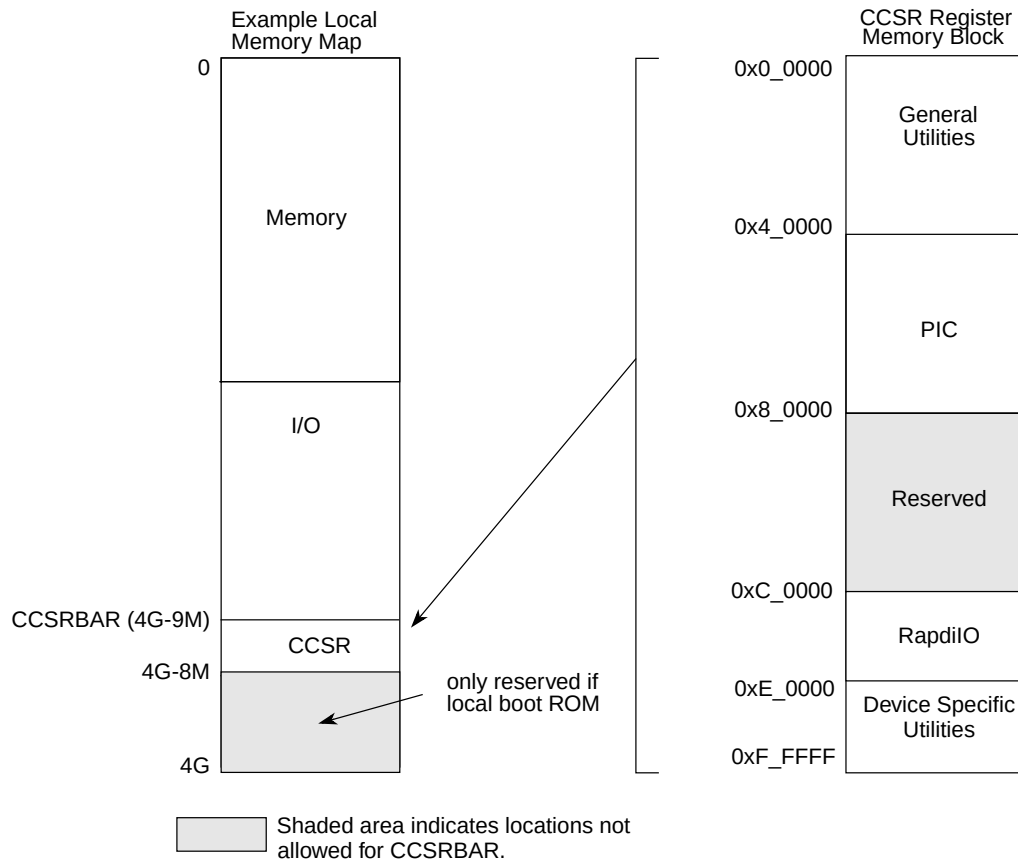


Figure 2-4. Example Top-Level Register Map

2.3.1 Accessing CCSR Memory from External Masters

In addition to being accessible by the e500 processor, the configuration, control, and status registers are accessible from external interfaces. This allows external masters on the I/O ports to configure the MPC8540.

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's programming model that is accessible to the external master from its external memory map.

The PCI base address for accessing the local CCSR memory is selectable through the PCI configuration and status register base address register (PCSRBAR), at offset 0x10, described in Section 16.3.2.9, "PCI Base Address Registers." An external PCI master sets this register by running a PCI configuration cycle to the MPC8540. Subsequent memory accesses by a PCI master to the PCI address range indicated by PCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

The RapidIO base address for accessing the local CCSR memory is selectable through the RapidIO LCSBA1CSR, defined in the RapidIO programming model. Section 17.3.2.12, “Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR).” An external 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.

2.3.2 Organization of CCSR Memory

The configuration, control, and status registers of the MPC8540 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.

Register that control functions that are not particular to a functional unit but to the device as a whole occupy the highest 256 Kbytes of CCSR memory. These are referred to as the 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 2-8. 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

2.3.3 General Utilities Registers

The overview of the general utilities registers is shown in Figure 2-5 below.

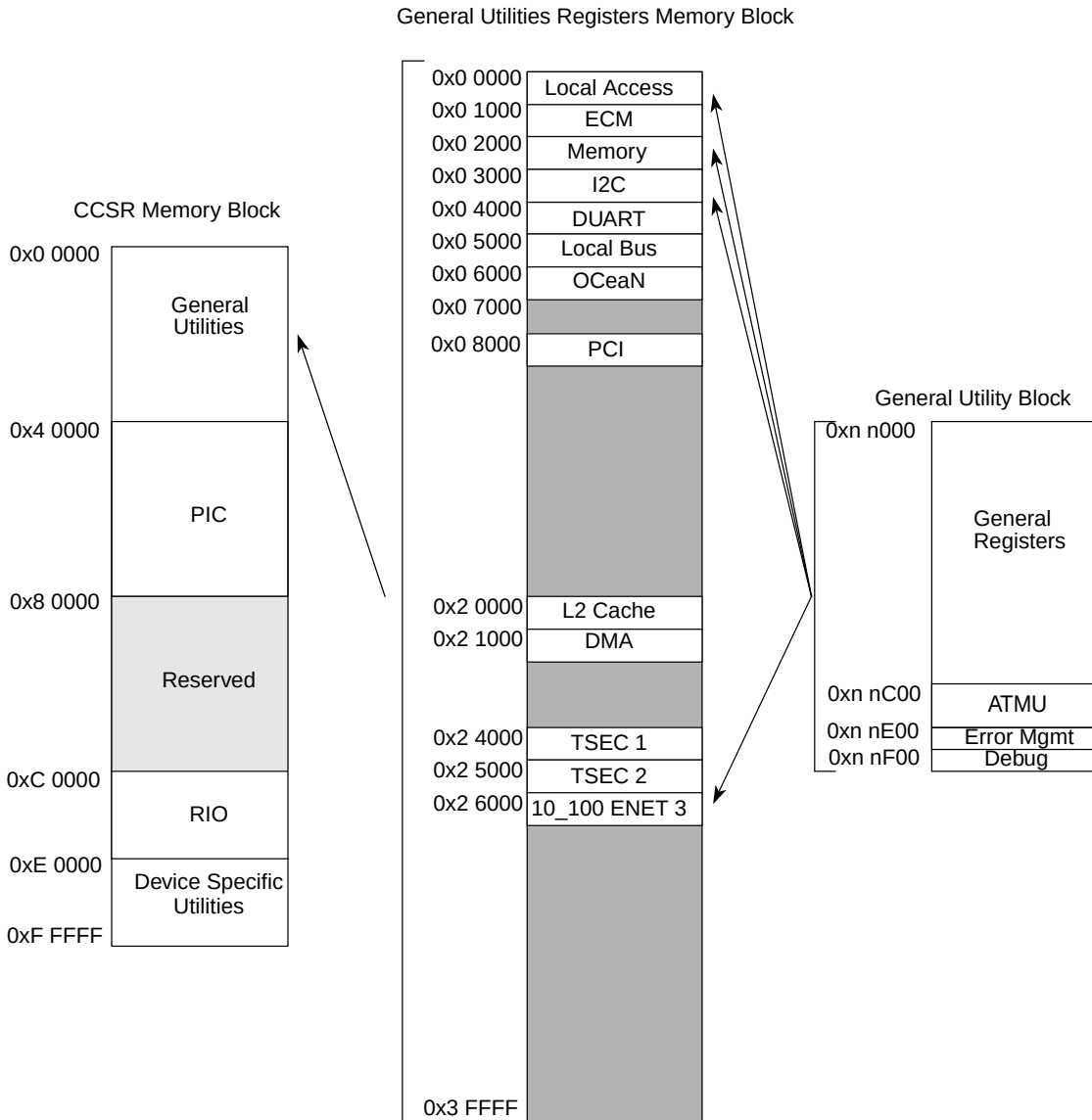


Figure 2-5. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

This figure also illustrates 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). 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 the DUART and I²C 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.

2.3.4 Interrupt Controller and CCSR

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

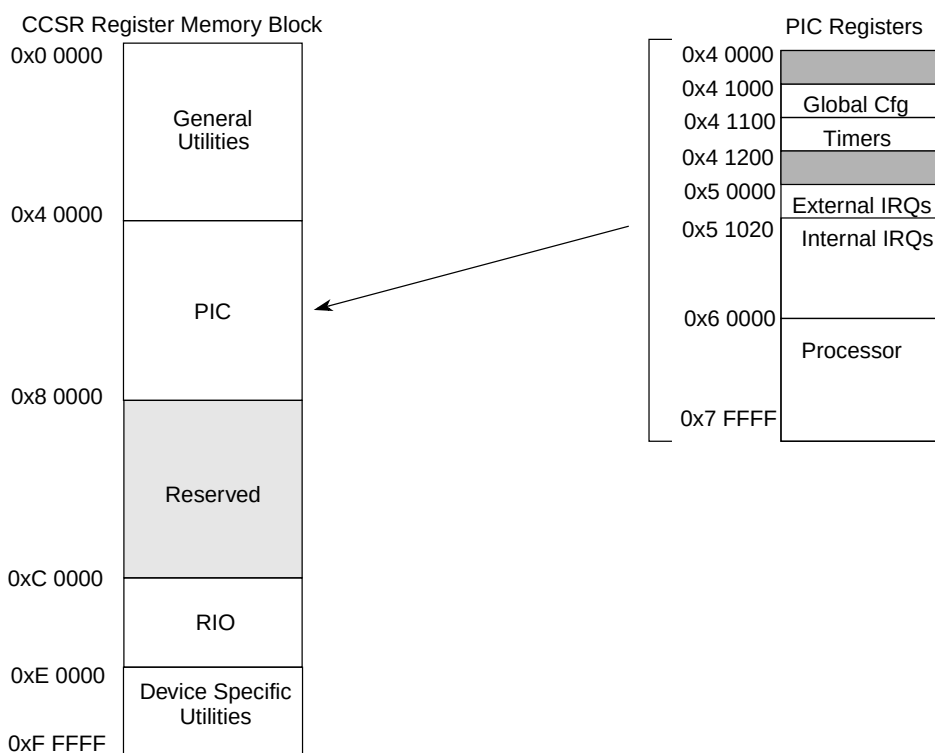


Figure 2-6. PIC Mapping to Configuration, Control, and Status Memory Block

2.3.5 RapidIO and CCSR

The RapidIO module uses 128 Kbytes of CCSR memory. 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.

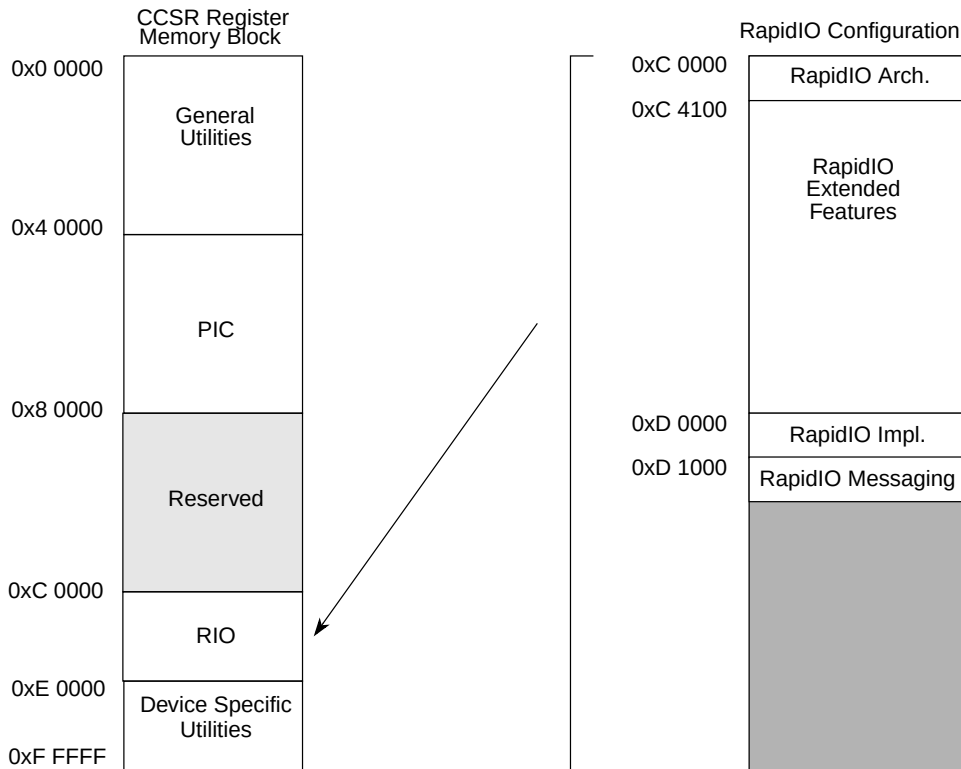


Figure 2-7. RapidIO Mapping to Configuration, Control, and Status Memory Block

2.3.6 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities. The 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.

Complete CCSR Map

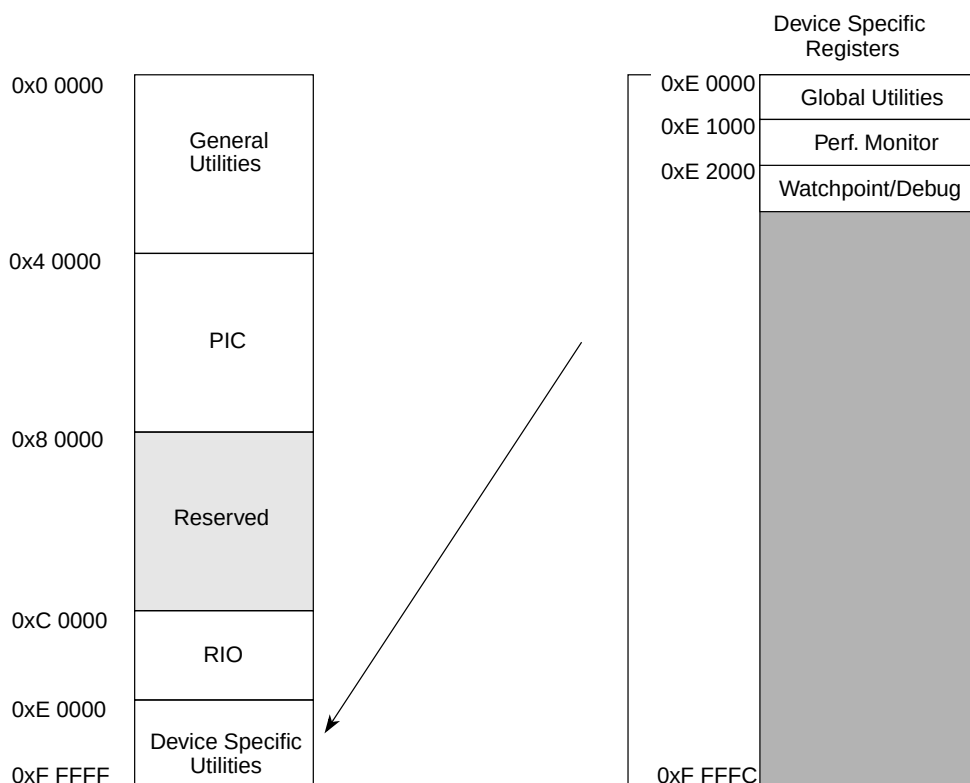


Figure 2-8. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.4 Complete CCSR Map

Table 2-9 lists the MPC8540 memory-mapped registers.

Table 2-9. Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
Local-Access Registers—Configuration, Control, and Status Registers				
0x0_0000	Control configuration status registers base address register (CCSRBAR)	R/W	0x000F_F700	4.5.1.1.2/4-18
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	0x0000_0000	4.5.1.2.1/4-19
0x0_0010	Alternate configuration size register (ALTCSR)	R/W	0x0000_0000	4.5.1.2.2/4-20
0x0_0020	Boot page translation register (BPTR)	R/W	0x0000_0000	4.5.1.3.1/4-21
Local-Access Registers—Local-Access Window Base and Size Registers				
0x0_0C08	Local access window 0 base address register (LAWBAR0)	R/W	0x0000_0000	2.2.3.2/2-6

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0C10	Local access window 0 attribute register (LAWSR0)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C28	Local access window 1 base address register (LAWBAR1)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C30	Local access window 1 attribute register (LAWAR1)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C48	Local access window 2 base address register (LAWBAR2)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C50	Local access window 2 attribute register (LAWAR2)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C68	Local access window 3 base address register (LAWBAR3)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C70	Local access window 3 attribute register (LAWAR3)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0C88	Local access window 4 base address register (LAWBAR4)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C90	Local access window 4 attribute register (LAWAR4)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0CA8	Local access window 5 base address register (LAWBAR5)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CB0	Local access window 5 attribute register (LAWAR5)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0CC8	Local access window 6 base address register (LAWBAR6)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CD0	Local access window 6 attribute register (LAWSA6)	R/W	0x0000_0000	2.2.3.3/2-7
0x0_0CE8	Local access window 7 base address register (LAWBAR7)	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CF0	Local access window 7 attribute register (LAWSA7)	R/W	0x0000_0000	2.2.3.3/2-7
Registers				
0x0_1000	ECM CCB address configuration register (EEBACR)	R/W	0x0000_0003	8.3.1.1/8-3
0x0_1010	ECM CCB port configuration register (EEBPCR)	R/W	0x0*00_0000	8.3.1.2/8-4
0x0_1E00	ECM error detect register (EEDR)	Special	0x0000_0000	8.3.1.3/8-5
0x0_1E08	ECM error enable register (EEER)	R/W	0x0000_0000	8.3.1.4/8-5
0x0_1E0C	ECM error attributes capture register (EEATR)	R	0x0000_0000	8.3.1.5/8-6
0x0_1E10	ECM error address capture register (EEADR)	R	0x0000_0000	8.3.1.6/8-8
DDR Memory Controller Memory Map				

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_2000	Chip Select 0 Memory Bounds (CS0_BNDS)	R/W	0x0000_0000	9.3.1.1/9-11
0x0_2008	Chip Select 1 Memory Bounds (CS1_BNDS)			
0x0_2010	Chip Select 2 Memory Bounds (CS2_BNDS)			
0x0_2018	Chip Select 3 Memory Bounds (CS3_BNDS)			
0x0_2080	Chip Select 0 Configuration (CS0_CONFIG)	R/W	0x0000_0000	9.3.1.2/9-12
0x0_2084	Chip Select 1 Configuration (CS1_CONFIG)			
0x0_2088	Chip Select 2 Configuration (CS2_CONFIG)			
0x0_208C	Chip Select 3 Configuration (CS3_CONFIG)			
0x0_2108	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)	R/W	0x0000_0000	9.3.1.3/9-13
0x0_210C	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)	R/W	0x0000_0000	9.3.1.4/9-16
0x0_2110	DDR SDRAM Control Configuration (DDR_SDRAM_CFG)	R/W	0x0200_0000	9.3.1.5/9-17
0x0_2118	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)	R/W	0x0000_0000	9.3.1.6/9-18
0x0_2124	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)	R/W	0x0000_0000	9.3.1.7/9-19
0x0_2E00	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	R/W	0x0000_0000	9.3.1.8/9-20
0x0_2E04	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	R/W	0x0000_0000	9.3.1.9/9-21
0x0_2E08	Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT)	R/W	0x0000_0000	9.3.1.10/9-21
0x0_2E20	Memory Data Path Read Capture High (CAPTURE_DATA_HI)	R/W	0x0000_0000	9.3.1.11/9-22
0x0_2E24	Memory Data Path Read Capture Low (CAPTURE_DATA_LO)	R/W	0x0000_0000	9.3.1.12/9-23
0x0_2E28	Memory Data Path Read Capture ECC (CAPTURE_ECC)	R/W	0x0000_0000	9.3.1.13/9-23
0x0_2E40	Memory Error Detect (ERR_DETECT)	Special	0x0000_0000	9.3.1.14/9-24
0x0_2E44	Memory Error Disable (ERR_DISABLE)	R/W	0x0000_0000	9.3.1.15/9-25
0x0_2E48	Memory Error Interrupt Enable (ERR_INT_EN)	R/W	0x0000_0000	9.3.1.16/9-25
0x0_2E4C	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)	R/W	0x0000_0000	9.3.1.17/9-26
0x0_2E50	Memory Error Address Capture (CAPTURE_ADDRESS)	R/W	0x0000_0000	9.3.1.18/9-28

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_2E54	Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS)	R/W	0x0000_0000	9.3.1.19/9-34
0x0_2E58	Single-Bit ECC Memory Error Management (ERR_SBE)	R/W	0x0000_0000	9.3.1.19/9-29
I²C				
0x0_3000	I ² C address register (I2CADR)	R/W	0x00	8.3.1.1/8-4
0x0_3004	I ² C frequency divider register (I2CFDR)	R/W	0x00	8.3.1.2/8-5
0x0_3008	I ² C control register (I2CCR)	R/W	0x00	8.3.1.3/8-6
0x0_300C	I ² C status register (I2CSR)	R/W	0x81	8.3.1.4/8-8
0x0_3010	I ² C data register (I2CDR)	R/W	0x00	8.3.1.5/8-10
0x0_3014	Digital filter sampling rate register (I2CDFSRR)	R/W	0x00	8.3.1.6/8-10
DUART Registers				
0x0_4500	ULCR[DLAB] = 0 UART0 receiver buffer register (URBR)	R	0x00	12.3.1.1/12-9
0x0_4500	ULCR[DLAB] = 0 UART0 transmitter holding register (UTHR)	W	0x00	12.3.1.2/12-10
0x0_4500	ULCR[DLAB] = 1 UART0 divisor least significant byte register (UDLB)	R/W	0x00	12.3.1.3/12-11
0x0_4501	ULCR[DLAB] = 0 UART0 interrupt enable register (UIER)	R/W	0x00	12.3.1.4/12-12
0x0_4501	ULCR[DLAB] = 1 UART0 divisor most significant byte register (UDMB)	R/W	0x00	12.3.1.3/12-11
0x0_4502	ULCR[DLAB] = 0 UART0 interrupt ID register (UIIR)	R	0x01	12.3.1.5/12-13
0x0_4502	ULCR[DLAB] = 0 UART0 FIFO control register (UFCR)	W	0x00	12.3.1.6/12-15
0x0_4502	ULCR[DLAB] = 1 UART0 alternate function register (UAFR)	R/W	0x00	12.3.1.12/12-21
0x0_4503	ULCR[DLAB] = x UART0 line control register (ULCR)	R/W	0x00	12.3.1.7/12-16
0x0_4504	ULCR[DLAB] = x UART0 MODEM control register (UMCR)	R/W	0x00	12.3.1.8/12-17
0x0_4505	ULCR[DLAB] = x UART0 line status register (ULSR)	R	0x60	12.3.1.9/12-18
0x0_4506	ULCR[DLAB] = x UART0 MODEM status register (UMSR)	R	0x00	12.3.1.10/12-20
0x0_4507	ULCR[DLAB] = x UART0 scratch register (USCR)	R/W	0x00	12.3.1.11/12-20

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_4510	ULCR[DLAB] = x UART0 DMA status register (UDSR)	R	0x01	12.3.1.13/12-22
0x0_4600	ULCR[DLAB] = 0 UART1 receiver buffer register (URBR)	R	0x00	12.3.1.1/12-9
0x0_4600	ULCR[DLAB] = 0 UART1 transmitter holding register (UTHR)	W	0x00	12.3.1.2/12-10
0x0_4600	ULCR[DLAB] =1 UART1 divisor least significant byte register (UDLB)	R/W	0x00	12.3.1.3/12-11
0x0_4601	ULCR[DLAB] = 0 UART1 interrupt enable register (UIER)	R/W	0x00	12.3.1.4/12-12
0x0_4601	ULCR[DLAB] =1 UART1 divisor most significant byte register (UDMB)	R/W	0x00	12.3.1.3/12-11
0x0_4602	ULCR[DLAB] = 0 UART1 interrupt ID register (UIIR)	R	0x01	12.3.1.5/12-13
0x0_4602	ULCR[DLAB] = 0 UART1 FIFO control register (UFCR)	W	0x00	12.3.1.6/12-15
0x0_4602	ULCR[DLAB] = 1 UART1 alternate function register (UAFR)	R/W	0x00	12.3.1.12/12-21
0x0_4603	ULCR[DLAB] = x UART1 line control register (ULCR)	R/W	0x00	12.3.1.7/12-16
0x0_4604	ULCR[DLAB] = x UART1 MODEM control register (UMCR)	R/W	0x00	12.3.1.8/12-17
0x0_4605	ULCR[DLAB] = x UART1 line status register (ULSR)	R	0x60	12.3.1.9/12-18
0x0_4606	ULCR[DLAB] = x UART1 MODEM status register (UMSR)	R	0x00	12.3.1.10/12-20
0x0_4607	ULCR[DLAB] = x UART1 scratch register (USCR)	R/W	0x00	12.3.1.11/12-20
0x0_4610	ULCR[DLAB] = x UART1 DMA status register (UDSR)	R	0x01	12.3.1.13/12-22
Local Bus Controller Registers				
0x0_5000	Base register 0 (BR0)	R/W	0x0000_RR01 ¹	13.3.1.1/13-11
0x0_5008	Base Register 1 (BR1)		0x0000_0000	
0x0_5010	Base Register 2 (BR2)			
0x0_5018	Base Register 3 (BR3)			
0x0_5020	Base Register 4 (BR4)			
0x0_5028	Base Register 5 (BR5)			
0x0_5030	Base Register6 (BR6)			
0x0_5038	Base Register7 (BR7)			

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_5004	Options register 0 (OR0)	R/W	0x0000_0FF7	13.3.1.2/13-20
0x0_500C	Options Register 1 (OR1)		0x0000_0000	
0x0_5014	Options Register 2 (OR2)			
0x0_501C	Options Register 3 (OR3)			
0x0_5024	Options Register 4 (OR4)			
0x0_502C	Options Register 5 (OR5)			
0x0_5034	Options Register6 (OR6)			
0x0_503C	Options Register7 (OR7)			
0x0_5068	UPM Address Register (MAR)	R/W	0x0000_0000	13.3.1.3/13-27
0x0_5070	UPMA Mode Register (MAMR)	R/W	0x0000_0000	13.3.1.4/13-27
0x0_5074	UPMB mode register (MBMR)	R/W	0x0000_0000	13.3.1.4/13-27
0x0_5078	UPMC mode register (MCMR)	R/W	0x0000_0000	13.3.1.4/13-27
0x0_5084	Memory refresh timer prescaler register (MRTPR)	R/W	0x0000_0000	13.3.1.5/13-30
0x0_5088	UPM data register (MDR)	R/W	0x0000_0000	13.3.1.6/13-30
0x0_5094	SDRAM mode register (LSDMR)	R/W	0x0000_0000	13.3.1.7/13-31
0x0_50A0	UPM refresh timer (LURT)	R/W	0x0000_0000	13.3.1.8/13-34
0x0_50A4	SDRAM refresh timer (LSRT)	R/W	0x0000_0000	13.3.1.9/13-34
0x0_50B0	Transfer error status register (LTESR)	Read/ Bit-reset	0x0000_0000	13.3.1.10/13-35
0x0_50B4	Transfer error disable register (LTEDR)	R/W	0x0000_0000	13.3.1.11/13-37
0x0_50B8	Transfer error interrupt register (LTEIR)	R/W	0x0000_0000	13.3.1.12/13-38
0x0_50BC	Transfer error attributes register (LTEATR)	R/W	0x0000_0000	13.3.1.13/13-33
0x0_50C0	Transfer error address register (LTEAR)	R/W	0x0000_0000	13.3.1.14/13-40

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_50D0	Configuration register (LBCR)	R/W	0x0000_0000	13.3.1.15/13-40
0x0_50D4	Clock ratio register (LCRR)	R/W	0x8000_0008	13.3.1.16/13-41
PCI/X Registers				
PCI/X Configuration Access Registers				
0x0_8000	PCI Configuration address access (PCI CFG_ADDR)	R/W	0x0000_0000	16.3.2.1/16-51
0x0_8004	PCI Configuration data access (PCI CFG_DATA)	R/W	0x0000_0000	16.3.2.1/16-51
0x0_8008	PCI/X Int_Ack access	R	0x0000_0000	16.3.1.4/16-50
0x0_800C - 0x0_8BFC	Reserved	—	—	—
PCI/X ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C—Outbound Window 0 (default)				
0x0_8C00	PCI/X Outbound window 0 (default) translation address register	R/W	0x0000_0000	16.3.1.1.1/16-34
0x0_8C04	PCI/X Outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-34
0x0_8C08	PCI/X outbound window 0 (default) base address register	R/W	0x0000_0000	16.3.1.1.3/16-35
0x0_8C0C	Reserved	—	—	
0x0_8C10	PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.1.5/16-36
0x0_8C14	Reserved	—	—	
0x0_8C18	Reserved	—	—	
0x0_8C1C	Reserved	—	—	
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	PCI/X Outbound window 1 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-34
0x0_8C24	PCI/X Outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-34
0x0_8C28	PCI/X Outbound window 1 base address register	R/W	0x0000_0000	16.3.1.1.3/16-35
0x0_8C2C	Reserved	—	—	
0x0_8C30	PCI/X Outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.1.5/16-36
0x0_8C34	Reserved	—	—	
0x0_8C38	Reserved	—	—	
0x0_8C3C	Reserved	—	—	

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	PCI/X Outbound window 2 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-34
0x0_8C44	PCI/X Outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-34
0x0_8C48	PCI/X Outbound window 2 base address register	R/W	0x0000_0000	16.3.1.1.3/16-35
0x0_8C4C	Reserved	—	—	
0x0_8C50	PCI/X Outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.1.5/16-36
0x0_8C54	Reserved	—	—	
0x0_8C58	Reserved	—	—	
0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	PCI/X Outbound window 3 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-34
0x0_8C64	PCI/X Outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-34
0x0_8C68	PCI/X Outbound window 3 base address register	R/W	0x0000_0000	16.3.1.1.3/16-35
0x0_8C6C	Reserved	—	—	
0x0_8C70	PCI/X Outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.1.5/16-36
0x0_8C74	Reserved	—	—	
0x0_8C78	Reserved	—	—	
0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	PCI/X Outbound window 4 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-34
0x0_8C84	PCI/X Outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-34
0x0_8C88	PCI/X Outbound window 4 base address register	R/W	0x0000_0000	16.3.1.1.3/16-35
0x0_8C8C	Reserved	—	—	
0x0_8C90	PCI/X Outbound window 4 attributes register	R/W	0x0000_0000	16.3.1.1.5/16-36
0x0_8C94	Reserved	—	—	
0x0_8C98	Reserved	—	—	
0x0_8C9C	Reserved	—	—	
0x0_8CA0 – 0x0_8D9C	Reserved	—	—	

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PCI/X Inbound window 3 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-39
0x0_8DA4	PCI/X Inbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-39
0x0_8DA8	PCI/X Inbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.3/16-40
0x0_8DAC	Reserved	—	—	
0x0_8DB0	PCI/X Inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.5/16-41
0x0_8DB4	Reserved	—	—	
0x0_8DB8	Reserved	—	—	
0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PCI/X Inbound window 2 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-39
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PCI/X Inbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.3/16-40
0x0_8DCC	PCI/X Inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.2.4/16-41
0x0_8DD0	PCI/X inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.5/16-41
0x0_8DD4	Reserved	—	—	
0x0_8DD8	Reserved	—	—	
0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PCI/X inbound window 1 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-39
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PCI/X Inbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.3/16-36
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PCI/X Inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.5/16-37
0x0_8DF4	Reserved	—	—	
0x0_8DF8	Reserved	—	—	
0x0_8DFC	Reserved	—	—	
PCI/X Error Management Registers				
0x0_8E00	PCI/X error detect register	Special	0x0000_0000	16.3.1.3/16-39

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_8E04	PCI/X error capture disabled register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E08	PCI/X error enable register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E0C	PCI/X error attributes capture register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E10	PCI/X error address capture register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E14	PCI/X error extended address capture register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E18	PCI/X error data low capture register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E1C	PCI/X error data high capture register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E20	PCI/X gasket timer register	R/W	0x0000_0000	16.3.1.3/16-39
0x0_8E24– 0x0_8EFC	Reserved	—	—	
0x0_8F00– 0x0_8FFC	Reserved for debug	—	—	
L2/SRAM Memory-Mapped Configuration Registers				
0x2_0000	L2 control register (L2CTL)	R/W	0x2000_0000	7.3.1.1/7-6
0x2_0010	L2 cache external write address register 0 (L2CEWAR0)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0018	L2 cache external write control register 0 (L2CEWCR0)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0020	L2 cache external write address register 1 (L2CEWAR1)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0028	L2 cache external write control register 1 (L2CEWCR1)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0030	L2 cache external write address register 2 (L2CEWAR2)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0038	L2 cache external write control register 2 (L2CEWCR2)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0040	L2 cache external write address register 3 (L2CEWAR3)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0048	L2 cache external write control register 3 (L2CEWCR3)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0100	L2 memory-mapped SRAM base address register 0 (L2SRBAR0)	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0108	L2 memory-mapped SRAM base address register 1 (L2SRBAR1)	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0E00	L2 error injection mask high register (L2ERRINJHI)	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E04	L2 error injection mask low register (L2ERRINJLO)	R/W	0x0000_0000	7.3.1.5.1/7-12

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_0E08	L2 error injection tag/ECC control register (L2ERRINJCTL)	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E20	L2 error data high capture register (L2CAPTDATAHI)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E24	L2 error data low capture register (L2CAPTDATALO)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E28	L2 error ECC capture register (L2CAPTECC)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E40	L2 error detect register (L2ERRDET)	Special	0x0000_0000	7.3.1.5.2/7-14
0x2_0E44	L2 error disable register (L2ERRDIS)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E48	L2 error interrupt enable register (L2ERRINTEN)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E4C	L2 error attributes capture register (L2ERRATTR)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E50	L2 error address capture register (L2ERRADDR)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E58	L2 error control register (L2ERRCTL)	R/W	0x0000_0000	7.3.1.5.2/7-14
DMA Registers				
DMA Registers—General Registers				
0x2_1100	DMA 0 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1104	DMA 0 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1108	Reserved	—	—	—
0x2_110C	DMA 0 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1110	DMA 0 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1114	DMA 0 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1118	DMA 0 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_111C	DMA 0 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1120	DMA 0 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1124	Reserved	—	—	—
0x2_1128	DMA 0 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_1130	Reserved	—	—	—
0x2_1134	DMA 0 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1138	Reserved	—	—	—
0x2_113C	DMA 0 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1140	DMA 0 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_1144	DMA 0 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1180	DMA 1 mode register	R/W	0x0000_0000	15.3.2.1/15-10

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_1184	DMA 1 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1188	Reserved	—	—	—
0x2_118C	DMA 1 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1190	DMA 1 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1194	DMA 1 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1198	DMA 1 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_119C	DMA 1 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_11A0	DMA 1 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	DMA 1 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_11B0	Reserved	—	—	—
0x2_11B4	DMA 1 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	DMA 1 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_11C0	DMA 1 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_11C4	DMA 1 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1200	DMA 2 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1204	DMA 2 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1208	Reserved	—	—	—
0x2_120C	DMA 2 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1210	DMA 2 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1214	DMA 2 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1218	DMA 2 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_121C	DMA 2 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1220	DMA 2 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1224	Reserved	—	—	—
0x2_1228	DMA 2 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_1230	Reserved	—	—	—
0x2_1234	DMA 2 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1238	Reserved	—	—	—
0x2_123C	DMA 2 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1240	DMA 2 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_1244	DMA 2 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_1280	DMA 3 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1284	DMA 3 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1288	Reserved	—	—	—
0x2_128C	DMA 3 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1290	DMA 3 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1294	DMA 3 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1298	DMA 3 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_129C	DMA 3 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_12A0	DMA 3 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	DMA 3 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_12B0	Reserved	—	—	—
0x2_12B4	DMA 3 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	DMA 3 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_12C0	DMA 3 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_12C4	DMA 3 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1300	DMA general status register	Read	0x0000_0000	15.3.2.14/15-27
TSEC1 General Control and Status Registers				
0x2_4000– 0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	Interrupt event register (IEVENT)	R/W	0x0000_0000	3.5.3.1.2/3-27
0x2_4014	Interrupt mask register (IMASK)	R/W	0x0000_0000	3.5.3.1.3/3-30
0x2_4018	Error Disabled register (EDIS)	R/W	0x0000_0000	3.5.3.1.4/3-32
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	Ethernet control register (ECNTRL)	R/W	0x0000_0000	3.5.3.1.5/3-33
0x2_4024	Minimum frame length register (MINFLR)	R/W	0x0000_0040	3.5.3.1.6/3-34
0x2_4028	Pause time value register (PTV)	R/W	0x0000_0000	3.5.3.1.7/3-35
0x2_402C	DMA control register (DMACTRL)	R/W	0x0000_0000	3.5.3.1.8/3-36
0x2_4030	TBI PHY address register (TBIPA)	R/W	0x0000_0000	/3-37
0x2_4034– 0x2_403C	Reserved	R	0x0000_0000	—
0x2_4040– 0x2_4088	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
TSEC FIFO Control and Status Registers				3.5.3.2/3-40
0x2_408C	FIFO transmit threshold register (FIFO_TX_THR)	R/W	0x0000_0080	3.5.3.2.20/3-56
0x2_4090– 0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO transmit starve register (FIFO_TX_STARVE)	R/W	0x0000_0020	3.5.3.2.7/3-46
0x2_409C	FIFO transmit Starve Shutoff register (FIFO_TX_STARVE_SHUTOFF)	R/W	0x0000_0080	3.5.3.2.8/3-47
0x2_40A0– 0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	Transmit control register (TCTRL)	R/W	0x0000_0000	3.5.3.3.1/3-60
0x2_4104	Transmit status register (TSTAT)	R/W	0x0000_0000	3.5.3.3.2/3-61
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TxBD data length (TBDLEN)	R/W	0x0000_0000	3.5.3.3.3/3-62
0x2_4110– 0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	Current TxBD pointer (CTBPTR)	R/W	0x0000_0000	3.5.3.3.5/3-63
0x2_4128– 0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TxBD pointer (TBPTR)	R/W	0x0000_0000	3.5.3.3.7/3-65
0x2_4188– 0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TxBD base address (TBASE)	R/W	0x0000_0000	3.5.3.3.9/3-66
0x2_4208– 0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	Out-of-sequence TxBD register (OSTBD)	R/W	0x0000_0000	3.5.3.3.10/3-67
0x2_42B4	Out-of-sequence Tx data buffer pointer register (OSTBDP)	R/W	0x0000_0000	13.5.3.3.11/13-69
0x2_42B8– 0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	Receive control register (RCTRL)	R/W	0x0000_0000	3.5.3.4.1/3-67
0x2_4304	Receive status register (RSTAT)	R/W	0x0000_0000	3.5.3.4.2/3-68
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RxBD data length (RBDLEN)	R/W	0x0000_0000	3.5.3.4.3/3-69

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_4310–0x2_431C	Reserved	R	0x0000_0000	—
0x2_4320	Reserved	R/W	0x0000_0000	3.5.3.4.4/3-70
0x2_4324	Current RxBD pointer (CRBPTR)	R/W	0x0000_0000	3.5.3.4.5/3-70
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	Maximum receive buffer length register (MRBL)	R/W	0x0000_0000	3.5.3.4.6/3-71
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RxBD pointer (RBPTR)	R/W	0x0000_0000	3.5.3.4.8/3-72
0x2_4388–0x2_4400	Reserved	R/W	0x0000_0000	3.5.3.4.9/3-73
0x2_4404	RxBD base address (RBASE)	R/W	0x0000_0000	3.5.3.4.10/3-74
0x2_4408–0x2_44FC	Reserved	R/W	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MAC configuration #1 (MACCFG1)	R/W, R	0x0000_0000	3.5.3.6.1/3-78
0x2_4504	MAC configuration #2 (MACCFG2)	R/W	0x0000_7000	3.5.3.6.2/3-79
0x2_4508	IPGIFGI	R/W	0x4060_5060	3.5.3.6.3/3-81
0x2_450C	Half-duplex (HAFDUP)	R/W	0x00A1_F037	3.5.3.6.4/3-82
0x2_4510	Maximum frame (MAXFRM)	R/W	0x0000_0600	3.5.3.6.5/3-83
0x2_4514	Reserved	R	0x0000_0000	—
0x2_4518	Reserved	R	0x0000_0000	—
0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MII Mgmt: configuration (MIIMCFG)	R/W	0x0000_0000	3.5.3.6.7/3-84
0x2_4524	MII Mgmt: command (MIICOM)	R/W	0x0000_0000	3.5.3.6.8/3-86
0x2_4528	MII Mgmt: address (MIIMADD)	R/W	0x0000_0000	3.5.3.6.9/3-86
0x2_452C	MII Mgmt: control (MIIMCON)	WO	0x0000_0000	3.5.3.6.10/3-87
0x2_4530	MII Mgmt: status (MIIMSTAT)	R	0x0000_0000	3.5.3.6.11/3-88
0x2_4534	MII Mgmt: indicators (MIIMIND)	R	0x0000_0000	3.5.3.6.12/3-88
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	Interface status (IFSTAT)	R,R/LH	0x0000_0001	3.5.3.6.14/3-91
0x2_4540	Station Address, part 1 (MACSTNADDR1)	R/W	0x0000_0000	3.5.3.6.15/3-92
0x2_4544	Station Address, part 2 (MACSTNADDR2)	R/W	0x0000_0000	3.5.3.6.16/3-93

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_4548–0x2_45FC	Reserved	R	0x0000_0000	—
0x2_4600–0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64 transmit and receive 64-byte frame counter	R/W	0x0000_0000	3.5.3.7.1/3-94
0x2_4684	TR127 transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	3.5.3.7.2/3-94
0x2_4688	TR255 transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	3.5.3.7.3/3-95
0x2_468C	TR511 transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	3.5.3.7.4/3-95
0x2_4690	TR1K transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	3.5.3.7.5/3-96
0x2_4694	TRMAX transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	3.5.3.7.6/3-97
0x2_4698	TRMGV transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	3.5.3.7.7/3-97
TSEC1 Receive Counters				
0x2_469C	RBYT receive byte counter	R/W	0x0000_0000	3.5.3.7.8/3-98
0x2_46A0	RPKT receive packet counter	R/W	0x0000_0000	3.5.3.7.9/3-99
0x2_46A4	RFCS receive FCS error counter	R/W	0x0000_0000	3.5.3.7.10/3-64
0x2_46A8	RMCA receive multicast packet counter	R/W	0x0000_0000	3.5.3.7.11/3-100
0x2_46AC	RBCA receive broadcast packet counter	R/W	0x0000_0000	3.5.3.7.12/3-101
0x2_46B0	RXCF receive control frame packet counter	R/W	0x0000_0000	3.5.3.7.13/3-101
0x2_46B4	RXPf receive PAUSE frame packet counter	R/W	0x0000_0000	3.5.3.7.14/3-102
0x2_46B8	RXUO receive unknown OP code counter	R/W	0x0000_0000	3.5.3.7.15/3-103
0x2_46BC	RALN receive alignment error counter	R/W	0x0000_0000	3.5.3.7.16/3-103
0x2_46C0	RFLR receive frame length error counter	R/W	0x0000_0000	3.5.3.7.17/3-104
0x2_46C4	RCDE receive code error counter	R/W	0x0000_0000	3.5.3.7.18/3-105
0x2_46C8	RCSE receive carrier sense error counter	R/W	0x0000_0000	3.5.3.7.19/3-105
0x2_46CC	RUND receive undersize packet counter	R/W	0x0000_0000	3.5.3.7.20/3-106
0x2_46D0	ROVR receive oversize packet counter	R/W	0x0000_0000	3.5.3.7.21/3-107
0x2_46D4	RFRG receive fragments counter	R/W	0x0000_0000	3.5.3.7.22/3-107

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_46D8	RJBR receive jabber counter	R/W	0x0000_0000	3.5.3.7.23/3-108
0x2_46DC	RDRP receive drop	R/W	0x0000_0000	3.5.3.7.24/3-109
Transmit Counters				
0x2_46E0	TBYT transmit byte counter	R/W	0x0000_0000	3.5.3.7.25/3-109
0x2_46E4	TPKT transmit packet counter	R/W	0x0000_0000	3.5.3.7.26/3-110
0x2_46E8	TMCA transmit multicast packet counter	R/W	0x0000_0000	3.5.3.7.27/3-111
0x2_46EC	TBCA transmit broadcast packet counter	R/W	0x0000_0000	3.5.3.7.28/3-111
0x2_46F0	TXPF transmit PAUSE control frame counter	R/W	0x0000_0000	3.5.3.7.29/3-112
0x2_46F4	TDFR transmit deferral packet counter	R/W	0x0000_0000	3.5.3.7.30/3-113
0x2_46F8	TEDF transmit excessive deferral packet counter	R/W	0x0000_0000	3.5.3.7.31/3-113
0x2_46FC	TSCL transmit single collision packet counter	R/W	0x0000_0000	3.5.3.7.32/3-114
0x2_4700	TMCL transmit multiple collision packet counter	R/W	0x0000_0000	3.5.3.7.33/3-115
0x2_4704	TLCL transmit late collision packet counter	R/W	0x0000_0000	3.5.3.7.34/3-115
0x2_4708	TXCL transmit excessive collision packet counter	R/W	0x0000_0000	3.5.3.7.35/3-116
0x2_470C	TNCL transmit total collision counter	R/W	0x0000_0000	3.5.3.7.36/3-117
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP transmit drop frame counter	R/W	0x0000_0000	3.5.3.7.38/3-118
0x2_4718	TJBR transmit jabber frame counter	R/W	0x0000_0000	3.5.3.7.39/3-119
0x2_471C	TFCS transmit FCS error counter	R/W	0x0000_0000	3.5.3.7.40/3-119
0x2_4720	TXCF transmit control frame counter	R/W	0x0000_0000	3.5.3.7.41/3-120
0x2_4724	TOVR transmit oversize frame counter	R/W	0x0000_0000	3.5.3.7.42/3-121
0x2_4728	TUND transmit undersize frame counter	R/W	0x0000_0000	3.5.3.7.43/3-121
0x2_472C	TFRG transmit fragments frame counter	R/W	0x0000_0000	3.5.3.7.44/3-122
General Registers				
0x2_4730	CAR1 carry register one register ²	R	0x0000_0000	3.5.3.7.45/3-123
0x2_4734	CAR2 carry register two register ²	R	0x0000_0000	3.5.3.7.46/3-124
0x2_4738	CAM1 carry register one mask register	R/W	0xFE01_FFFF	3.5.3.7.47/3-125
0x2_473C	CAM2 carry register two mask register	R/W	0x000F_FFFF	3.5.3.7.48/3-127
0x2_4740– 0x2_47FC	Reserved	R	0x0000_0000	—
Hash Function Registers				

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_4800	Individual address register 0 (IADDR0)	R/W	0x0000_0000	3.5.3.8.1/3-128
0x2_4804	Individual address register 1 (IADDR1)	R/W	0x0000_0000	
0x2_4808	Individual address register 2 (IADDR2)	R/W	0x0000_0000	
0x2_480C	Individual address register 3 (IADDR3)	R/W	0x0000_0000	
0x2_4810	Individual address register 4 (IADDR4)	R/W	0x0000_0000	
0x2_4814	Individual address register 5 (IADDR5)	R/W	0x0000_0000	
0x2_4818	Individual address register 6 (IADDR6)	R/W	0x0000_0000	
0x2_481C	Individual address register 7 (IADDR7)	R/W	0x0000_0000	
0x2_4820–0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	Group address register 0 (GADDR0)	R/W	0x0000_0000	3.5.3.8.2/3-129
0x2_4884	Group address register 1 (GADDR1)	R/W	0x0000_0000	
0x2_4888	Group address register 2 (GADDR2)	R/W	0x0000_0000	
0x2_488C	Group address register 3 (GADDR3)	R/W	0x0000_0000	
0x2_4890	Group address register 4 (GADDR4)	R/W	0x0000_0000	
0x2_4894	Group address register 5 (GADDR5)	R/W	0x0000_0000	
0x2_4898	Group address register 6 (GADDR6)	R/W	0x0000_0000	
0x2_489C	Group address register 7 (GADDR7)	R/W	0x0000_0000	
0x2_48A0–0x2_4BF4	Reserved	R	0x0000_0000	—
0x2_4BF8	Attribute register (ATTR)	R	0x0000_0000	3.5.3.9.6/3-160
0x2_4BFC	Attribute EL & EI register (ATTRELI)	R	0x0000_0000	3.5.3.8.4/3-131
TSEC1 Future Expansion Space				
0x2_4C00–0x2_4FFC	Reserved	R	0x0000_0000	—
0x2_5000–0x2_5FFF	TSEC2 REGISTERS ³			3.5.3/3-26
10/100 Ethernet Controller Registers				21.5.3/21-22
10/100 Ethernet Controller General Control and Status Registers				21.5.3.1/21-22
0x2_6000–0x2_600C	Reserved	R	0x0000_0000	—
0x2_6010	Interrupt event register (IEVENT)	R/W	0x0000_0000	21.5.3.1.2/21-22
0x2_6014	Interrupt mask register (IMASK)	R/W	0x0000_0000	21.5.3.1.3/21-25
0x2_6018	Error disabled register (EDIS)	R/W	0x0000_0000	21.5.3.1.4/21-26

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_601C–0x2_6020	Reserved	R	0x0000_0000	—
0x2_6024	Minimum frame length register (MINFLR)	R/W	0x0000_0040	21.5.3.1.5/21-27
0x2_6028	Pause time value register (PTV)	R/W	0x0000_0000	21.5.3.1.6/21-27
0x2_602C	DMA control register (DMACTRL)	R/W	0x0000_0000	21.5.3.1.7/21-28
0x2_6030–0x2_603C	Reserved	R	0x0000_0000	
0x2_6040–0x2_60FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Transmit Control and Status Registers				3.5.3.3/3-58
0x2_6100	Transmit control register (TCTRL)	R/W	0x0000_0000	21.5.3.2.25/21-51
0x2_6104	Transmit status register (TSTAT)	R/W	0x0000_0000	21.5.3.2.26/21-52
0x2_6108	Reserved	R	0x0000_0000	
0x2_610C	TxBD data length (TBDLEN)	R/W	0x0000_0000	21.5.3.2.27/21-53
0x2_6110–0x2_6120	Reserved	R	0x0000_0000	
0x2_6120	Current TxBD pointer high (CTBPTRH)	R/W	0x0000_0000	21.5.3.2.28/21-53
0x2_6124	Current TxBD pointer (CTBPTR)	R/W	0x0000_0000	21.5.3.2.29/21-54
0x2_6128–0x2_6180	Reserved	R	0x0000_0000	
0x2_6180	TxBD pointer high (TBPTRH)	R/W	0x0000_0000	21.5.3.2.30/21-54
0x2_6184	TxBD pointer (TBPTR)	R/W	0x0000_0000	21.5.3.2.31/21-55
0x2_6188–0x2_6200	Reserved	R	0x0000_0000	
0x2_6200	TxBD base address high (TBASEH)	R/W	0x0000_0000	21.5.3.2.32/21-56
0x2_6204	TxBD base address (TBASE)	R/W	0x0000_0000	21.5.3.2.33/21-56
0x2_6208–0x2_62AC	Reserved for more Tx base addresses	R	0x0000_0000	
0x2_62B0	Out-of-sequence TxBD register (OSTBD)	R/W	0x0000_0000	21.5.3.2.34/21-57
0x2_62B4	Out-of-sequence TxBD pointer register (OSTBDP)	R/W	0x0000_0000	21.5.3.2.35/21-59
0x2_62B8–0x2_62FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Receive Control and Status Registers				3.5.3.4/3-72
0x2_6300	Receive control register (RCTRL)	R/W	0x0000_0000	21.5.3.3.1/21-60
0x2_6304	Receive status register (RSTAT)	R/W	0x0000_0000	21.5.3.3.2/21-61

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_6308	Reserved	R	0x0000_0000	
0x2_630C	RxBD data length (RBDLEN)	R/W	0x0000_0000	21.5.3.3.3/21-62
0x2_6310– 0x2_6320	Reserved	R	0x0000_0000	
0x2_6320	Current RxBD pointer high (CRBPTRH)	R/W	0x0000_0000	21.5.3.3.4/21-62
0x2_6324	Current RxBD pointer (CRBPTR)	R/W	0x0000_0000	21.5.3.3.5/21-63
0x2_6328– 0x2_633C	Reserved	R	0x0000_0000	
0x2_6340	Maximum receive buffer length register (MRBLR)	R/W	0x0000_0000	21.5.3.3.6/21-64
0x2_6344	Reserved	R	0x0000_0000	
0x2_6348– 0x2_635C	Reserved for more Rx buffer length registers	R	0x0000_0000	
0x2_6360– 0x2_6380	Reserved	R	0x0000_0000	
0x2_6384	RxBD pointer (RBPTR)	R/W	0x0000_0000	21.5.3.3.9/21-67
0x2_6388– 0x2_639C	Reserved	R	0x0000_0000	
0x2_63A0– 0x2_63FC	Reserved for more Rx pointers	R	0x0000_0000	
0x2_6400	Reserved	R	0x0000_0000	
0x2_6404	RxBD base address (RBASE)	R/W	0x0000_0000	21.5.3.3.11/21-68
0x2_6408– 0x2_641C	Reserved	R	0x0000_0000	
0x2_6420– 0x2_64FC	Reserved for more Rx base addresses	R	0x0000_0000	
10/100 Ethernet Controller MAC registers				21.5.3.4/21-68
0x2_6500	MAC configuration #1 (MACCFGF1)	R/W, R	0x0000_0000	21.5.3.5.1/21-71
0x2_6504	MAC configuration #2 (MACCFGF2)	R/W	0x0000_7000	21.5.3.5.2/21-73
0x2_6508	IPGIFG	R/W	0x4060_5060	21.5.3.5.3/21-74
0x2_650C	Half-duplex (HAFDUP)	R/W	0x00A0_F037	21.5.3.5.4/21-75
0x2_6510	Maximum frame (MAXFRM)	R/W	0x0000_0600	21.5.3.5.5/21-76
0x2_6514	Reserved	R	0x0000_0000	
0x2_6518	Reserved	R	0x0000_0000	
0x2_651C	Reserved	R	0x0000_0000	
0x2_6520	MII management: configuration (MIIMCFG)	R/W	0x0000_0000	21.5.3.5.7/21-77
0x2_6524	MII management: command (MIIMCOM)	R/W	0x0000_0000	21.5.3.5.8/21-78

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_6528	MII management: address (MIIMADD)	R/W	0x0000_0000	21.5.3.5.9/21-79
0x2_652C	MII management: control (MIIMCON)	WO	0x0000_0000	21.5.3.5.10/21-80
0x2_6530	MII management: status (MIIMSTAT)	R	0x0000_0000	21.5.3.5.11/21-80
0x2_6534	MII management: indicators (MIIMIND)	R	0x0000_0000	21.5.3.5.12/21-81
0x2_6538	Reserved	R	0x0000_0000	
0x2_653C	Interface status (IFSTAT)	R,R/LH	0x0000_0001	21.5.3.5.14/21-83
0x2_6540	Station Address, part 1 (MACSTNADDR1)	R/W	0x0000_0000	21.5.3.5.15/21-84
0x2_6544	Station Address, part 2 (MACSTNADDR2)	R/W	0x0000_0000	21.5.3.5.16/21-85
0x2_6548– 0x2_65FC	Reserved	R	0x0000_0000	
0x2_6600– 0x2_667C	Reserved	R	0x0000_0000	
0x2_6680– 0x2_673C	Reserved	R	0x0000_0000	
0x2_6740– 0x2_67FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Hash Function Registers				3.5.3.7/3-125
0x2_6800	Individual address register 0 (IADDR0)	R/W	0x0000_0000	21.5.3.7.1/21-113
0x2_6804	Individual address register 1 (IADDR1)	R/W	0x0000_0000	
0x2_6808	Individual address register 2 (IADDR2)	R/W	0x0000_0000	
0x2_680C	Individual address register 3 (IADDR3)	R/W	0x0000_0000	
0x2_6810	Individual address register 4 (IADDR4)	R/W	0x0000_0000	
0x2_6814	Individual address register 5 (IADDR5)	R/W	0x0000_0000	
0x2_6818	Individual address register 6 (IADDR6)	R/W	0x0000_0000	
0x2_681C	Individual address register 7 (IADDR7)	R/W	0x0000_0000	
0x2_6820– 0x2_687C	Reserved	R	0x0000_0000	

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_6880	Group address register 0 (GADDR0)	R/W	0x0000_0000	21.5.3.7.2/21-114
0x2_6884	Group address register 1 (GADDR1)	R/W	0x0000_0000	
0x2_6888	Group address register 2 (GADDR2)	R/W	0x0000_0000	
0x2_688C	Group address register 3 (GADDR3)	R/W	0x0000_0000	
0x2_6890	Group address register 4 (GADDR4)	R/W	0x0000_0000	
0x2_6894	Group address register 5 (GADDR5)	R/W	0x0000_0000	
0x2_6898	Group address register 6 (GADDR6)	R/W	0x0000_0000	
0x2_689C	Group address register 7 (GADDR7)	R/W	0x0000_0000	
0x2_68A0–0x2_6BF4	Reserved	R	0x0000_0000	
0x2_6BF8	Attribute register (ATTR)	R	0x0000_0000	21.5.3.8.6/21-134
0x2_6BFC	Attribute EL & EI register (ATTRELI)	R	0x0000_0000	21.5.3.8.7/21-136
10/100 Ethernet Controller Future Expansion Space				
0x2_6C00–0x2_6FFC	Reserved	R	0x0000_0000	
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	Interprocessor interrupt 0 (IPI 0) dispatch register (IPIDR0)	W	0x0000_0000	10.3.7.1/10-47
0x4_0050	IPI 1 dispatch register (IPIDR1)			
0x4_0060	IPI 2 dispatch register (IPIDR2)			
0x4_0070	IPI 3 dispatch register (IPIDR3)			
0x4_0080	Current task priority register (CTPR)	R/W	0x0000_000F	10.3.7.2/10-49
0x4_0090	Who am I register (WHOAMI0)	R	0x0000_0000	10.3.7.3/10-50
0x4_00A0	Interrupt acknowledge register (IACK)	R	0x0000_0000	10.3.7.4/10-50
0x4_00B0	End of interrupt register (EOI)	W	0x0000_0000	10.3.7.5/10-51
0x4_00C–0x4_0FF0	Reserved	—	—	—
0x4_1000	Feature reporting register (FRR)	R	0x0037_0002	10.3.1.1/10-19
0x4_1010	Reserved	—	—	—
0x4_1020	Global configuration register (GCR)	R/W	0x0000_0000	10.3.1.2/10-20
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x4_1080	Vendor identification register (VIR)	R	0x0000_0000	10.3.1.3/10-21
0x4_1090	Processor initialization register (PIR)	R/W	0x0000_0000	10.3.1.4/10-22
0x4_10A0	IPI 0 vector/priority register (IPIVPR0)	R/W	0x8000_0000	10.3.1.5/10-24
0x4_10B0	IPI 1 vector/priority register (IPIVPR1)			
0x4_10C0	IPI 2 vector/priority register (IPIVPR2)			
0x4_10D0	IPI 3 vector/priority register (IPIVPR3)			
0x4_10E0	Spurious vector register (SVR)	R/W	0x0000_FFFF	10.3.1.6/10-24
0x4_10F0	Timer frequency reporting register (TFRR)	R/W	0x0000_0000	10.3.2.1/10-25
0x4_1100	Global timer 0 current count register (GTCCR0)	R	0x0000_0000	10.3.2.2/10-26
0x4_1110	Global timer 0 base count register (GTBCR0)	R/W	0x8000_0000	10.3.2.3/10-27
0x4_1120	Global timer 0 vector/priority register (GTVPR0)	R/W	0x8000_0000	10.3.2.4/10-28
0x4_1130	Global timer 0 destination register (GTDR0)	R/W	0x0000_0001	10.3.2.5/10-28
0x4_1140	Global timer 1 current count register (GTCCR1)	R	0x0000_0000	10.3.2.2/10-26
0x4_1150	Global Timer 1 Base Count Register (GTBCR1)	R/W	0x8000_0000	10.3.2.3/10-27
0x4_1160	Global timer 1 vector/priority register (GTVPR1)	R/W	0x8000_0000	10.3.2.4/10-28
0x4_1170	Global timer 1 destination register (GTDR1)	R/W	0x0000_0001	10.3.2.5/10-28
0x4_1180	Global timer 2 current count register (GTCCR2)	R	0x0000_0000	10.3.2.2/10-26
0x4_1190	Global timer 2 base count register (GTBCR2)	R/W	0x8000_0000	10.3.2.3/10-27
0x4_11A0	Global timer 2 vector/priority register (GTVPR2)	R/W	0x8000_0000	10.3.2.4/10-28
0x4_11B0	Global timer 2 destination register (GTDR2)	R/W	0x0000_0001	10.3.2.5/10-28
0x4_11C0	Global timer 3 current count register (GTCCR3)	R	0x0000_0000	10.3.2.2/10-26
0x4_11D0	Global timer 3 base count register (GTBCR3)	R/W	0x8000_0000	10.3.2.3/10-27
0x4_11E0	Global timer 3 vector/priority register (GTVPR3)	R/W	0x8000_0000	10.3.2.4/10-28
0x4_11F0	Global timer 3 destination register (GTDR3)	R/W	0x0000_0001	10.3.2.5/10-28
0x4_1200–0x4_12F0	Reserved	—	—	—
0x4_1300	Timer control register (TCR)	R/W	0x0000_0000	10.3.2.6/10-30
0x4_1310	IRQ_OUT summary register 0 (IRQSR0)	R	0x0000_0000	10.3.3.1/10-32
0x4_1320	IRQ_OUT summary register 1 (IRQSR1)	R	0x0000_0000	10.3.3.2/10-33
0x4_1330	Critical Interrupt summary register 0 (CISR0)	R	0x0000_0000	10.3.3.3/10-33
0x4_1340	Critical Interrupt summary register 1 (CISR1)	R	0x0000_0000	10.3.3.4/10-34
0x4_1350	Performance Monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-35
0x4_1360	Performance Monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-35

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x4_1370	Performance Monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-35
0x4_1380	Performance Monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-35
0x4_1390	Performance Monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-35
0x4_13A0	Performance Monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-35
0x4_13B0	Performance Monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-35
0x4_13C0	Performance Monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-35
0x4_13D0– 0x4_13F0	Reserved	—	—	—
0x4_1400	Message register 0 (MSGR0)	R/W	0x0000_0000	10.3.5.1/10-36
0x4_1410	Message register 1 (MSGR1)			
0x4_1420	Message register 2 (MSGR2)			
0x4_1430	Message register 3 (MSGR3)			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	Message enable register (MER)	R/W	0x0000_0000	10.3.5.2/10-37
0x4_1510	Message status register (MSR)	R/W	0x0000_0000	10.3.5.3/10-38
0x4_1520– 0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	External interrupt 0 (IRQ0) vector/priority register (EIVPR0)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0010	External interrupt 0 (IRQ0) destination register (EIDR0)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0020	External interrupt 1 (IRQ1) vector/priority register (EIVPR1)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0030	External interrupt 1 (IRQ1) destination register (EIDR1)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0040	External interrupt 2 (IRQ2) vector/priority register (EIVPR2)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0050	External interrupt 2 (IRQ2) destination register (EIDR2)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0060	External interrupt 3 (IRQ3) vector/priority register (EIVPR3)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0070	External interrupt 3 (IRQ3) destination register (EIDR3)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0080	External interrupt 4 (IRQ4) vector/priority register (EIVPR4)	R/W	0x8000_0000	10.3.6.1/10-39

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x5_0090	External interrupt 4 (IRQ4) destination register (EIDR4)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_00A0	External interrupt 5 (IRQ5) vector/priority register (EIVPR5)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_00B0	External interrupt 5 (IRQ5) destination register (EIDR5)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_00C0	External interrupt 6 (IRQ6) vector/priority register (EIVPR6)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_00D0	External interrupt 6 (IRQ6) destination register (EIDR6)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_00E0	External interrupt 7 (IRQ7) vector/priority register (EIVPR7)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_00F0	External interrupt 7 (IRQ7) destination register (EIDR7)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0100	External interrupt 8 (IRQ8) vector/priority register (EIVPR8)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0110	External interrupt 8 (IRQ8) destination register (EIDR8)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0120	External interrupt 9 (IRQ9) vector/priority register (EIVPR9)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0130	External interrupt 9 (IRQ9) destination register (EIDR9)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0140	External interrupt 10 (IRQ10) vector/priority register (EIVPR10)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0150	External interrupt 10 (IRQ10) destination register (EIDR10)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0160	External interrupt 11 (IRQ11) vector/priority register (EIVPR11)	R/W	0x8000_0000	10.3.6.1/10-39
0x5_0170	External interrupt 11 (IRQ11) destination register (EIDR11)	R/W	0x0000_0001	10.3.6.2/10-40
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	Internal interrupt 0 vector/priority register (IIVPR0)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0210	Internal interrupt 0 destination register (IIDR0)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0220	Internal interrupt 1 vector/priority register (IIVPR1)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0230	Internal interrupt 1 destination register (IIDR1)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0240	Internal interrupt 2 vector/priority register (IIVPR2)	R/W	0x8080_0000	10.3.6.3/10-41

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x5_0250	Internal interrupt 2 destination register (IIDR2)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0260	Internal interrupt 3 vector/priority register (IIVPR3)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0270	Internal interrupt 3 destination register (IIDR3)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0280	Internal interrupt 4 vector/priority register (IIVPR4)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0290	Internal interrupt 4 destination register (IIDR4)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_02A0	Internal interrupt 5 vector/priority register (IIVPR5)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_02B0	Internal interrupt 5 destination register (IIDR5)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_02C0	Internal interrupt 6 vector/priority register (IIVPR6)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_02D0	Internal interrupt 6 destination register (IIDR6)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_02E0	Internal interrupt 7 vector/priority register (IIVPR7)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_02F0	Internal interrupt 7 destination register (IIDR7)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0300	Internal interrupt 8 vector/priority register (IIVPR8)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0310	Internal interrupt 8 destination register (IIDR8)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0320	Internal interrupt 9 vector/priority register (IIVPR9)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0330	Internal interrupt 9 destination register (IIDR9)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0340	Internal interrupt 10 vector/priority register (IIVPR10)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0350	Internal interrupt 10 destination register (IIDR10)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0360	Internal interrupt 11 vector/priority register (IIVPR11)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0370	Internal interrupt 11 destination register (IIDR11)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0380	Internal interrupt 12 vector/priority register (IIVPR12)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0390	Internal interrupt 12 destination register (IIDR12)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_03A0	Internal interrupt 13 vector/priority register (IIVPR13)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_03B0	Internal interrupt 13 destination register (IIDR13)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_03C0	Internal interrupt 14 vector/priority register (IIVPR14)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_03D0	Internal interrupt 14 destination register (IIDR14)	R/W	0x0000_0001	10.3.6.4/10-43

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x5_03E0	Internal interrupt 15 vector/priority register (IIVPR15)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_03F0	Internal interrupt 15 destination register (IIDR15)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0400	Internal interrupt 16 vector/priority register (IIVPR16)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0410	Internal interrupt 16 destination register (IIDR16)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0420	Internal interrupt 17 vector/priority register (IIVPR17)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0430	Internal interrupt 17 destination register (IIDR17)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0440	Internal interrupt 18 vector/priority register (IIVPR18)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0450	Internal interrupt 18 destination register (IIDR18)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0460	Internal interrupt 19 vector/priority register (IIVPR19)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0470	Internal interrupt 19 destination register (IIDR19)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0480	Internal interrupt 20 vector/priority register (IIVPR20)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0490	Internal interrupt 20 destination register (IIDR20)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_04A0	Internal interrupt 21 vector/priority register (IIVPR21)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_04B0	Internal interrupt 21 destination register (IIDR21)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_04C0	Internal interrupt 22 vector/priority register (IIVPR22)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_04D0	Internal interrupt 22 destination register (IIDR22)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_04E0	Internal interrupt 23 vector/priority register (IIVPR23)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_04F0	Internal interrupt 23 destination register (IIDR23)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0500	Internal interrupt 24 vector/priority register (IIVPR24)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0510	Internal interrupt 24 destination register (IIDR24)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0520	Internal interrupt 25 vector/priority register (IIVPR25)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0530	Internal interrupt 25 destination register (IIDR25)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0540	Internal interrupt 26 vector/priority register (IIVPR26)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0550	Internal interrupt 26 destination register (IIDR126)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0560	Internal interrupt 27 vector/priority register (IIVPR27)	R/W	0x8080_0000	10.3.6.3/10-41

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x5_0570	Internal interrupt 27 destination register (IIDR27)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0580	Internal interrupt 28 vector/priority register (IIVPR28)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_0590	Internal interrupt 28 destination register (IIDR28)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_05A0	Internal interrupt 29 vector/priority register (IIVPR29)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_05B0	Internal interrupt 29 destination register (IIDR29)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_05C0	Internal interrupt 30 vector/priority register (IIVPR30)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_05D0	Internal interrupt 30 destination register (IIDR30)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_05E0	Internal interrupt 31 vector/priority register (IIVPR31)	R/W	0x8080_0000	10.3.6.3/10-41
0x5_05F0	Internal interrupt 31 destination register (IIDR31)	R/W	0x0000_0001	10.3.6.4/10-43
0x5_0600–0x5_15F0	Reserved	—	—	—
0x5_1600	Messaging interrupt 0 (MSG 0) vector/priority register (MIVPR0)	R/W	0x8000_0000	10.3.6.5/10-44
0x5_1610	Messaging interrupt 0 (MSG 0) destination register (MIDR0)	R/W	0x0000_0001	10.3.6.6/10-45
0x5_1620	Messaging interrupt 1 (MSG 1) vector/priority register (MIVPR1)	R/W	0x8000_0000	10.3.6.5/10-44
0x5_1630	Messaging interrupt 1 (MSG 1) destination register (MIDR1)	R/W	0x0000_0001	10.3.6.6/10-45
0x5_1640	Messaging interrupt 2 (MSG 2) vector/priority register (MIVPR2)	R/W	0x8000_0000	10.3.6.5/10-44
0x5_1650	Messaging interrupt 2 (MSG 2) destination register (MIDR2)	R/W	0x0000_0001	10.3.6.6/10-45
0x5_1660	Messaging interrupt 3 (MSG 3) vector/priority register (MIVPR3)	R/W	0x8000_0000	10.3.6.5/10-44
0x5_1670	Messaging interrupt 3 (MSG 3) destination register (MIDR3)	R/W	0x0000_0001	10.3.6.6/10-45
0x5_1680–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per CPU Registers				
0x6_0000–0x6_0030	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x6_0040	P0 IPI 0 dispatch register (IPIDR0)	W	all zeros	10.3.7.1/10-47
0x6_0050	P0 IPI 1 dispatch register (IPIDR1)			
0x6_0060	P0 IPI 2 dispatch register (IPIDR2)			
0x6_0070	P0 IPI 3 dispatch register (IPIDR3)			
0x6_0080	P0 current task priority register (CTPR0)	R/W	0x0000_000F	10.3.7.2/10-49
0x6_0090	P0 who am I register (WHOAMI0)	R	all zeros	10.3.7.3/10-50
0x6_00A0	P0 interrupt acknowledge register (IACK0)	R	all zeros	10.3.7.4/10-50
0x6_00B0	P0 end of interrupt register (EOI0)	W	all zeros	10.3.7.5/10-51
RapidIO Registers				
Architectural Registers				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0002_0002	17.3.2.1/17-14
0xC_0004	Device information capability register (DICAR)	R	0x8020_0010	17.3.2.2/17-14
0xC_0008	Assembly identity capability register (AIDCAR)	Special ⁴	0x0000_0000	17.3.2.3/17-15
0xC_000C	Assembly information capability register (AICAR)	Special ⁴	0x0000_0100	17.3.2.4/17-16
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE088_0009	17.3.2.5/17-16
0xC_0014	Switch port information capability register (SPICAR)	R	0x0000_0100	17.3.2.6/17-17
0xC_0018	Source Operations Capability Register (SOCAR)	R	0x0600_FCF0	17.3.2.7/17-18
0xC_001C	Destination operations capability register (DOCAR)	R	0x0600_FCF4	17.3.2.8/17-19
0xC_0040	Mailbox Command and Status Register (MSR)	R	0x0000_0000	17.3.2.9/17-21
0xC_0044	Port -Write and Doorbell Command and Status Register (PWDCSR)	R	0x0000_0020	17.3.2.10/17-22
0xC_004C	Processing Element Logical Layer Control Command and Status Register (PELLCCSR)	R	0x0000_0001	17.3.2.11/17-23
0xC_005C	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	R/W	0x0000_0000	17.3.2.12/17-23
0xC_0060	Base Device ID Command and Status Register (BDIDCSR)	R/W	0x00[cfg]_0000	17.3.2.13/17-24
0xC_0068	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	Special ⁴	0x0000_FFFF	17.3.2.14/17-25
0xC_006C	Component Tag Command and Status Register (CTCSR)	R/W	0x0000_0000	17.3.2.15/17-25
0xC_0100	8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)	R	0x0000_0002	17.3.2.16/17-26

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xC_0120	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	R/W	0xFFFF_FF00	17.3.2.17/17-27
0xC_0124	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	R/W	0xFFFF_FF00	17.3.2.18/17-27
0xC_013C	Port General Control Command and Status Register (PGCCSR)	R/W	0x[cfg]000_0000	17.3.2.19/17-28
0xC_0140	Port Link Maintenance Request Command and Status Register (PLMREQCSR)	R/W	0x0000_0000	17.3.2.19/17-28
0xC_0144	Port Link Maintenance Response Command and Status Register (PLMRESPCSR)	R	0x0000_0000	17.3.2.20/17-29
0xC_0148	Port Local Ackid Status Command and Status Register (PLASCSR)	Special ⁴	0x0000_0000	17.3.2.21/17-29
0xC_0158	Port Error and Status Command and Status Register (PESCSR)	Special ⁴	0x0000_0000	17.3.2.22/17-30
0xC_015C	Port Control Command and Status Register (PCCSR)	Special ⁴	0x4400_0000	17.3.2.23/17-31
Implementation Registers				
0xD_0000	Configuration Register (CR)	R/W	0x0000_0001	17.3.3.1.1/17-33
0xD_0010	Port Configuration Register (PCR)	R/W	0x0000_0010	17.3.3.1.2/17-34
0xD_0014	Port Error Injection Register (PEIR)	Special ⁴	0x0000_0000	17.3.3.1.3/17-35
ATMU Registers				
0xD_0C00	RapidIO Outbound Window Translation Address Register 0 (ROWTAR0)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C10	RapidIO Outbound Window Attributes Register 0 (ROWAR0)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C20	RapidIO Outbound Window Translation Address Register 1 (ROWTAR1)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C28	RapidIO Outbound Window Base Address Register 1 (ROWBAR1)	R/W	0x0000_0000	/17-38
0xD_0C30	RapidIO Outbound Window Attributes Register 1 (ROWAR1)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C40	RapidIO Outbound Window Translation Address Register 2 (ROWTAR2)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C48	RapidIO Outbound Window Base Address Register 2 (ROWBAR2)	R/W	0x0000_0000	/17-38
0xD_0C50	RapidIO Outbound Window Attributes Register 2 (ROWAR2)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C60	RapidIO Outbound Window Translation Address Register 3 (ROWTAR3)	R/W	0x0000_0000	17.3.3.2.1/17-36

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xD_0C68	RapidIO Outbound Window Base Address Register 3 (ROWBAR3)	R/W	0x0000_0000	/17-38
0xD_0C70	RapidIO Outbound Window Attributes Register 3 (ROWAR3)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C80	RapidIO Outbound Window Translation Address Register 4 (ROWTAR4)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C88	RapidIO Outbound Window Base Address Register 4 (ROWBAR4)	R/W	0x0000_0000	/17-38
0xD_0C90	RapidIO Outbound Window Attributes Register 4 (ROWAR4)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CA0	RapidIO Outbound Window Translation Address Register 5 (ROWTAR5)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CA8	RapidIO Outbound Window Base Address Register 5 (ROWBAR5)	R/W	0x0000_0000	/17-38
0xD_0CB0	RapidIO Outbound Window Attributes Register 5 (ROWAR5)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CC0	RapidIO Outbound Window Translation Address Register 6 (ROWTAR6)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CC8	RapidIO Outbound Window Base Address Register 6 (ROWBAR6)	R/W	0x0000_0000	/17-38
0xD_0CD0	RapidIO Outbound Window Attributes Register 6 (ROWAR6)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CE0	RapidIO Outbound Window Translation Address Register 7 (ROWTAR7)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CE8	RapidIO Outbound Window Base Address Register 7 (ROWBAR7)	R/W	0x0000_0000	/17-38
0xD_0CF0	RapidIO Outbound Window Attributes Register 7 (ROWAR7)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0D00	RapidIO Outbound Window Translation Address Register 8 (ROWTAR8)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0D08	RapidIO Outbound Window Base Address Register 8 (ROWBAR8)	R/W	0x0000_0000	/17-38
0xD_0D10	RapidIO Outbound Window Attributes Register 8 (ROWAR8)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0D60	RapidIO Inbound Window Translation Address Register 4 (RIWTAR4)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0D68	RapidIO Inbound Window Base Address Register 4 (RIWBAR4)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0D70	RapidIO Inbound Window Attributes Register 4 (RIWAR4)	R/W	0x0004_401F	17.3.3.2.6/17-42

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xD_0D80	RapidIO Inbound Window Translation Address Register 3 (RIWTAR3)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0D88	RapidIO Inbound Window Base Address Register 3 (RIWBAR3)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0D90	RapidIO Inbound Window Attributes Register 3 (RIWAR3)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DA0	RapidIO Inbound Window Translation Address Register 2 (RIWTAR2)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DA8	RapidIO Inbound Window Base Address Register 2 (RIWBAR2)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0DB0	RapidIO Inbound Window Attributes Register 2 (RIWAR2)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DC0	RapidIO Inbound Window Translation Address Register 1 (RIWTAR1)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DC8	RapidIO Inbound Window Base Address Register 1 (RIWBAR1)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0DD0	RapidIO Inbound Window Attributes Register1 (RIWAR1)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DE0	RapidIO Inbound Window Translation Address Register 0 (RIWTAR0)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DF0	RapidIO Inbound Window Attributes Register 0 (RIWAR0)	R/W	0x0004_401F	17.3.3.2.6/17-42
Error Management Registers				
0xD_0E00	Port Notification/Fatal Error Detect Register (PNFEDR)	Special ⁴	0x0000_0000	17.3.3.3.1/17-44
0xD_0E04	Port Notification/Fatal Error Disable Register (PNFEDiR)	R/W	0x0000_0000	17.3.3.3.2/17-47
0xD_0E08	Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)	R/W	0x0000_0000	17.3.3.3.3/17-49
0xD_0E0C	Port Error Control Register (PECR)	Special ⁴	0x0000_0000	17.3.3.3.4/17-50
0xD_0E10	Port Error Packet/Control Symbol Register 0 (PEPCSR0)	Special ⁴	0x0000_0000	17.3.3.3.5/17-51
0xD_0E14	Port Error Packet Register 1 (PEPR1)	Special ⁴	0x0000_0000	17.3.3.3.6/17-52
0xD_0E18	Port Error Packet Register 2 (PEPR2)	Special ⁴	0x0000_0000	17.3.3.3.16/17-58
0xD_0E20	Port Recoverable Error Detect Register (PREDR)	Special ⁴	0x0000_0000	17.3.3.3.23/17-62
0xD_0E28	Port Error Recovery Threshold Register (PERTR)	Special ⁴	0x00FF_0000	17.3.3.3.24/17-65
0xD_0E2C	Port Retry Threshold Register (PRTR)	Special ⁴	0x00FF_0000	17.3.3.3.25/17-66
RapidIO Message Unit				

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
RapidIO Outbound Message Registers				
0xD_1000	Outbound Mode Register (OMR)	R/W	0x0000_0000	17.3.4.1.1/17-67
0xD_1004	Outbound Status Register (OSR)	Special	0x0000_0000	17.3.4.1.2/17-69
0xD_100C	Outbound descriptor queue dequeue pointer address register (ODQDPAR)	R/W	0x0000_0000	17.3.4.1.3/17-70
0xD_1014	Outbound Source Address Register (OSAR)	R/W	0x0000_0000	17.3.4.1.4/17-71
0xD_1018	Outbound Destination Port Register (ODPR)	R/W	0x0000_0000	17.3.4.1.5/17-72
0xD_101C	Outbound Destination Attributes Register (ODATR)	R/W	0x0006_0000	17.3.4.1.6/17-72
0xD_1020	Outbound Double-word Count Register (ODCR)	R/W	0x0000_0000	17.3.4.1.7/17-73
0xD_1028	Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)	R/W	0x0000_0000	17.3.4.1.8/17-86
RapidIO Inbound Message Registers				
0xD_1060	Inbound Mailbox Mode Register (IMR)	R/W	0x0000_0000	17.3.4.2.1/17-75
0xD_1064	Inbound Mailbox Status Register (ISR)	Special	0x0000_0000	17.3.4.2.2/17-77
0xD_106C	Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)	R/W	0x0000_0000	17.3.4.2.3/17-91
0xD_1074	Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)	R/W	0x0000_0000	17.3.4.2.4/17-79
RapidIO Doorbell Registers				
0xD_1460	Doorbell Mode Register (DMR)	R/W	0x0000_0000	17.3.4.3.1/17-80
0xD_1464	Doorbell Status Register (DSR)	Special	0x0000_0000	17.3.4.3.2/17-82
0xD_146C	Doorbell Queue Dequeue Pointer Address Register (DQDPAR)	R/W	0x0000_0000	17.3.4.3.3/17-83
0xD_1474	Doorbell Queue Enqueue Pointer Address Register (DQEPAR)	R/W	0x0000_0000	17.3.4.3.4/17-99
RapidIO Port-Write Registers				
0xD_14E0	Port-write mode register (PWMR)	R/W	0x0000_0000	17.3.4.4.1/17-85
0xD_14E4	Port-write Status Register (PWSR)	Special	0x0000_0000	17.3.4.4.2/17-86
0xD_14EC	Port-write Queue Base Address Register (PWQBAR)	R/W	0x0000_0000	17.3.4.4.3/17-87
Global Utilities Registers				
Power-On Reset Configuration Values				
0xE_0000	POR PLL ratio status register (PORPLLSR)	R	0x00nn_00nn	18.3.1.1/18-10
0xE_0004	POR boot mode status register (PORBMSR)	R		18.3.1.2/18-12

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xE_0008	POR I/O impedance status and control register (PORIMPSCR)	R/W	0x000n_003F	18.3.1.3/18-13
0xE_000C	POR I/O device status register (PORDEVSR)	R		18.3.1.4/18-14
0xE_0010	POR debug mode status register (PORDBGMSR)	R		18.3.1.5/18-16
0xE_0020	General-purpose POR configuration register (GPPORCR)	R		18.3.1.7/18-18
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIO control register (GPIOCR)	R/W	0x0000_0000	18.3.1.8/18-18
0xE_0040	General-purpose output data register (GPOUTDR)	R/W	0x0000_0000	18.3.1.9/18-19
0xE_0050	General-purpose input data register (GPINDR)	R	0xn timer_nnnn	18.3.1.10/18-20
0xE_0060	Alternate function signal multiplex control (PMUXCR)	R/W	0x0000_0000	18.3.1.10/18-20
Device Disables				
0xE_0070	Device disable control (DEVDISR)	R/W	0x0000_0000	18.3.1.12/18-22
Power Management Registers				
0xE_0080	Power management status and control register (POWMGTCSR)	R/W	0x0000_0000	18.3.1.13/18-24
Interrupt Reporting				
0xE_0090	Machine check summary register (MCPSUMR)	Read/Clear	0x0000_0000	18.3.1.14/18-26
Version Registers				
0xE_00A0	Processor version register (PVR)	R	e500 Processor Version	18.3.1.15/18-27
0xE_00A4	System version register (SVR)	R	MPC8540 System Version	18.3.1.16/18-27
Debug Control				
0xE_0E00	Clock out select register (CLKOCR)	R/W	0x0000_0000	18.3.1.17/18-28
0xE_0E10	DDR DLL control register (DDRDLPCR)	R/W	0x0000_0000	18.3.1.18/18-29
0xE_0E20	LBC DLL control register (LBDLLCR)	R/W	0x0000_0000	18.3.1.20/18-31
Performance Monitor Control Registers				
0xE_1000	Global control register (PMGC0)	R/W	0x0000_0000	19.3.2.1/19-6
0xE_1010	Local control register A0 (PMLCA0)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1014	Local control register B0 (PMLCB0)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1018	Performance monitor counter 0 (PMC0)	R/W	0x0000_0000	19.3.3.1/19-12

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xE_1010	Performance monitor counter 0 Extended (PMC0 Ex)	R/W	0x0000_0000	
0xE_1020	Local control register A1 (PMLCA1)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1024	Local control register B1 (PMLCB1)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1028	Performance monitor counter 1 (PMC1)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1030	Local control register A2 (PMLCA2)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1034	Local control register B 2 (PMLCB2)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1038	Performance monitor counter 2 (PMC2)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1040	Local control register A3 (PMLCA3)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1044	Local control register B3 (PMLCB3)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1048	Performance monitor counter 3 (PMC3)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1050	Local control register A4 (PMLCA4)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1054	Local control register B4 (PMLCB4)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1058	Performance monitor counter 4 (PMC4)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1060	Local control register A5 (PMLCA5)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1064	Local control register B 5 (PMLCB5)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1068	Performance monitor counter 5 (PMC5)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1070	Local control register A6 (PMLCA6)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1074	Local control register B6 (PMLCB6)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1078	Performance monitor counter 6 (PMC6)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1080	Local control register A7 (PMLCA7)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1084	Local control register B7 (PMLCB7)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1088	Performance monitor counter 7 (PMC7)	R/W	0x0000_0000	19.3.3.1/19-12
0xE_1090	Local control register A8 (PMLCA8)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1094	Local control register B8 (PMLCB5)	R/W	0x0000_0000	19.3.2.2/19-7
0xE_1098	Performance monitor counter 8 (PMC8)	R/W	0x0000_0000	19.3.3.1/19-12
Debug and Watchpoint Monitor Registers				
Watchpoint Monitor Registers				
0xE_2000	Watchpoint Monitor Control Register 0 (WMCR0)	R/W	0x0000_0000	20.3.1.1/20-11
0xE_2004	Watchpoint Monitor Control Register 1 (WMCR1)	R/W	0x0000_0000	20.3.1.2/20-12
0xE_200C	Watchpoint Monitor Address Register (WMAR)	R/W	0x0000_0000	20.3.1.3/20-13
0xE_2014	Watchpoint Monitor Address Mask Register (WMAMR)	R/W	0x0000_0000	20.3.1.4/20-14

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0xE_2018	Watchpoint Monitor Transaction Mask Register (WMTMR)	R/W	0x0000_0000	20.3.1.5/20-15
0xE_201C	Watchpoint Monitor Status Register (WMSR)	R/W	0x0000_0000	20.3.1.6/20-17
Trace Buffer Registers				
0xE_2040	Trace Buffer Control Register (TBCR0)	R/W	0x0000_0000	20.3.1.1/20-11
0xE_2044	Trace Buffer Control Register (TBCR1)	R/W	0x0000_0000	20.3.1.2/20-12
0xE_204C	Trace Buffer Address Register (TBAR)	R/W	0x0000_0000	20.3.2.3/20-20
0xE_2054	Trace Buffer Address Mask Register (TBAMR)	R/W	0x0000_0000	20.3.2.4/20-21
0xE_2058	Trace Buffer Transaction Mask Register (TBTMR)	R/W	0x0000_0000	20.3.2.5/20-22
0xE_205C	Trace Buffer Status Register (TBSR)	R/W	0x0000_0000	20.3.2.6/20-22
0xE_2060	Trace Buffer Access Control Register (TBACR)	R/W	0x0000_0000	20.3.2.7/20-23
0xE_2064	Trace Buffer Access Data High Register (TBADHR)	R/W	0x0000_0000	20.3.2.8/20-24
0xE_2068	Trace Buffer Access Data Register (TBADR)	R/W	0x0000_0000	20.3.2.9/20-25
Context ID Registers				
0xE_20A0	Programmed Context ID Register (PCIDR)	R/W	0x0000_0000	20.3.3.1/20-25
0xE_20A4	Current Context ID Register (CCIDR)	R/W	0x0000_0000	20.3.3.2/20-26
Other Registers				
0xE_20B0	Trigger Output Source Register (TOSR)	R/W	0x0000_0000	20.3.4.1/20-27

¹ Port size for BR0 is configured from external pins during reset, hence 'RR' is either 0x08, 0x10, or 0x18..

² Cleared on read

³ TSEC2 has the same memory-mapped registers that are described for TSEC1 from 0x 2_4000 to 0x2_4FFF except the offsets are from 0x 2_5000 to 0x2_5FFF.

⁴ The bits of special registers are cleared by setting them.

Chapter 3

Signal Descriptions

3.1 Introduction

This chapter provides information about the MPC8540's external signals. It is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions. Two listings: one by functional block, and one alphabetical
- Listing of reset configuration signals.
- Listing of output signal states at reset

3.2 Signals Overview

MPC8540 signals are grouped as follows:

- DDR memory interface signals
- RapidIO interface signals
- PCI/PCI-X interface signals
- Ethernet management interface signals
- TSEC1 interface signals
- TSEC2 interface signals
- 10/100 FEC interface signals
- Local bus interface signals
- DMA interface signals
- PIC interface signals
- DUART interface signals
- I²C interface signals
- System control, power management, and debug signals
- Test, JTAG, and configuration signals
- Clock signals

Signals Overview

Figure 3-1 illustrates the external signals of the MPC8540, showing how the signals are grouped. Refer to the *MPC8540 Hardware Specification* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

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. Note that Figure 3-1 shows the individual signals grouped by functional block. In some cases, signals are multiplexed between the functional blocks. This is not shown in Figure 3-1, but the multiplexing is shown in the tables of Section 3.2.1, “Signal Cross Reference.”

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

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

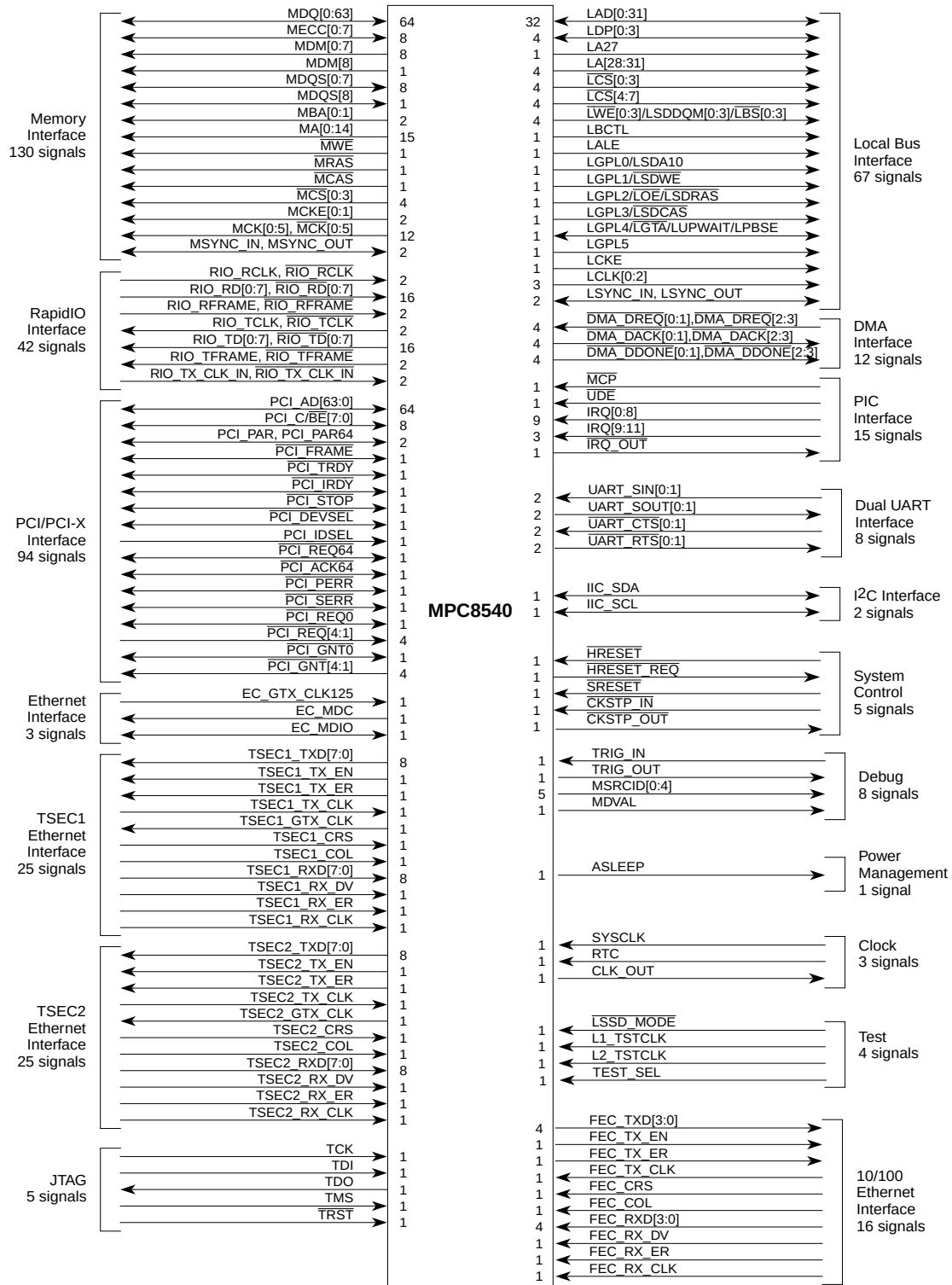


Figure 3-1. MPC8540 Signal Groupings

Signals Overview

The following tables provide summaries of signal functions. Table 3-1 provides an 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 chapter where the signal function is described.

Table 3-1. MPC8540 Signal Reference by Functional Block

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
MDQ[0:63]	DDR Data	DDR Memory	—	64	I/O	9-3/9-6
MECC[0:7]	DDR Error Correcting Code	DDR Memory	—	8	I/O	9-3/9-6
MDM[0:7]	DDR Data Mask	DDR Memory	—	8	O	9-3/9-6
MDM8	DDR ECC Data Mask	DDR Memory	—	1	O	9-3/9-6
MDQS[0:7]	DDR Data Strobe	DDR Memory	—	8	I/O	9-3/9-6
MDQS8	DDR ECC Data Strobe	DDR Memory	—	1	I/O	9-3/9-6
MBA[0:1]	DDR Bank Select	DDR Memory	—	2	O	9-3/9-6
MA[0:14]	DDR Address	DDR Memory	—	15	O	9-3/9-6
$\overline{\text{MWE}}$	DDR Write Enable	DDR Memory	—	1	O	9-3/9-6
$\overline{\text{MRAS}}$	DDR Row Address Strobe	DDR Memory	—	1	O	9-3/9-6
$\overline{\text{MCAS}}$	DDR Column Address Strobe	DDR Memory	—	1	O	9-3/9-6
$\overline{\text{MCS}}$ [0:3]	DDR Chip Select (2/DIMM)	DDR Memory	—	4	O	9-3/9-6
MCKE[0:1]	DDR Clock Enable	DDR Memory	—	2	O	9-4/9-9
MCK[0:5], MCK[0:5]	DDR Differential Clocks (3 pairs/DIMM)	DDR Memory	—	12	O	9-4/9-9
MSYNC_IN, MSYNC_OUT	DDR DLL Synchronization In/Out	DDR Memory	—	2	I/O	9-4/9-9
RIO_RCLK, $\overline{\text{RIO\_RCLK}}$	RapidIO Receive Clocks	RapidIO	—	2	I	17-4/17-8
RIO_RD[0:7], $\overline{\text{RIO\_RD}}$ [0:7]	RapidIO Receive Data	RapidIO	—	16	I	17-4/17-8
RIO_RFRAME, $\overline{\text{RIO\_RFRAME}}$	RapidIO Receive Frame	RapidIO	—	2	I	17-4/17-8
RIO_TCLK, $\overline{\text{RIO\_TCLK}}$	RapidIO Transmit Clock	RapidIO	—	2	O	17-4/17-8
RIO_TD[0:7], $\overline{\text{RIO\_TD}}$ [0:7]	RapidIO Transmit Data	RapidIO	—	16	O	17-4/17-8
RIO_TFRAME, $\overline{\text{RIO\_TFRAME}}$	RapidIO Transmit Frame	RapidIO	—	2	O	17-4/17-8

Table 3-1. MPC8540 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
RIO_TX_CLK_IN $\overline{\text{RIO\_TX\_CLK\_IN}}$	RapidIO Transmit Clock In	RapidIO	—	2	I	17-4/17-8
PCI_AD[63:0]	PCI Address/Data	PCI/X	—	64	I/O	16-1/16-7
PCI_C/ $\overline{\text{BE}}$ [7:0]	PCI Command/Byte Enable	PCI/X	—	8	I/O	16-1/16-7
PCI_PAR	PCI Parity	PCI/X	—	1	I/O	16-1/16-7
PCI_PAR64	PCI Parity 64	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_FRAME}}$	PCI Frame	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_TRDY}}$	PCI Target Ready	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_IRDY}}$	PCI Initiator Ready	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_STOP}}$	PCI Stop	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_DEVSEL}}$	PCI Device Select	PCI/X	—	1	I/O	16-1/16-7
PCI_IDSEL	PCI Initial Device Select	PCI/X	—	1	I	16-1/16-7
$\overline{\text{PCI\_REQ64}}$	PCI Request 64	PCI/X	cfg_pci_width	1	I/O	16-1/16-7
$\overline{\text{PCI\_ACK64}}$	PCI Acknowledge 64	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_PERR}}$	PCI Parity Error	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_SERR}}$	PCI System Error	PCI/X	—	1	I/O	16-1/16-7
PCI_REQ0	PCI Request 0	PCI/X	—	1	I/O	16-1/16-7
PCI_REQ[4:1]	PCI Request 4–1	PCI/X	—	4	I	16-1/16-7
$\overline{\text{PCI\_GNT0}}$	PCI Grant 0	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_GNT}}[4:1]$	PCI Grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	16-1/16-7
EC_GTX_CLK125	Gigabit Reference Clock	Gigabit Clock	—	1	I	14-1/14-10
EC_MDC	Ethernet Management Data Clock	Ethernet Management	cfg_tsec_reduce	1	O	14-1/14-10
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	14-1/14-10
TSEC1_TXD7	TSEC1 Transmit Data 7	TSEC1	deviceID0	1	O	14-1/14-10
TSEC1_TXD6	TSEC1 Transmit Data 6	TSEC1	deviceID1	1	O	14-1/14-10
TSEC1_TXD5	TSEC1 Transmit Data 5	TSEC1	deviceID2	1	O	14-1/14-10
TSEC1_TXD4	TSEC1 Transmit Data 4	TSEC1	deviceID3	1	O	14-1/14-10
TSEC1_TXD3	TSEC1 Transmit Data 3	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD2	TSEC1 Transmit Data 2	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD1	TSEC1 Transmit Data 1	TSEC1	—	1	O	14-1/14-10

Table 3-1. MPC8540 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
TSEC1_TXD0	TSEC1 Transmit Data 0	TSEC1	—	1	O	14-1/14-10
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	14-1/14-10
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	14-1/14-10
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	14-1/14-10
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	cfg_tsec1	1	O	14-1/14-10
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I	14-1/14-10
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	14-1/14-10
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	—	8	I	14-1/14-10
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	14-1/14-10
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	14-1/14-10
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	14-1/14-10
TSEC2_TXD7	TSEC2 Transmit Data 7	TSEC2	deviceID4	1	O	14-1/14-10
TSEC2_TXD6	TSEC2 Transmit Data 6	TSEC2	deviceID5	1	O	14-1/14-10
TSEC2_TXD5]	TSEC2 Transmit Data 5	TSEC2	deviceID6	1	O	14-1/14-10
TSEC2_TXD4	TSEC2 Transmit Data 4	TSEC2	deviceID7	1	O	14-1/14-10
TSEC2_TXD3]	TSEC2 Transmit Data 3	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD2	TSEC2 Transmit Data 2	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD1	TSEC2 Transmit Data 1	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD0	TSEC2 Transmit Data 0	TSEC2	—	1	O	14-1/14-10
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	14-1/14-10
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	—	1	O	14-1/14-10
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	14-1/14-10
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	cfg_tsec2	1	O	14-1/14-10
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I	14-1/14-10
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	14-1/14-10
TSEC2_RXD[0:7]	TSEC2 Receive Data	TSEC2	—	8	I	14-1/14-10
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	14-1/14-10
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	14-1/14-10
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	14-1/14-10
FEC_TXD[0:3]	FEC Transmit Data	10/100	—	4	O	21-1/21-8
FEC_TX_EN	FEC Transmit Enable	10/100	—	1	O	21-1/21-8
FEC_TX_ER	FEC Transmit Error	10/100	—	1	O	21-1/21-8
FEC_TX_CLK	FEC Transmit Clock In	10/100	—	1	I	21-1/21-8

Table 3-1. MPC8540 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
FEC_CRS	FEC Carrier Sense	10/100	—	1	I	21-1/21-8
FEC_COL	FEC Collision Detect	10/100	—	1	I	21-1/21-8
FEC_RXD[0:3]	FEC Receive Data	10/100	—	4	I	21-1/21-8
FEC_RX_DV	FEC Receive Data Valid	10/100	—	1	I	21-1/21-8
FEC_RX_ER	FEC Receiver Error	10/100	—	1	I	21-1/21-8
FEC_RX_CLK	FEC Receive Clock	10/100	—	1	I	21-1/21-8
LAD[0:31]	LBC Address/Data	LBC	—	32	I/O	13-1/13-5
LDP[0:3]	LBC Data Parity	LBC	—	4	I/O	13-1/13-5
LA27	LBC Burst Address	LBC	cfg_cpu_boot	1	O	13-1/13-5
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	13-1/13-5
$\overline{\text{LCS}}[0:2]$	LBC Chip Select 0–2	LBC	cfg_rom_loc[0:2]	3	O	13-1/13-5
$\overline{\text{LCS}}[3:4]$	LBC Chip Select 3–4	LBC	—	2	O	13-1/13-5
$\overline{\text{LCS}}5$	LBC Chip Select 5	LBC	$\overline{\text{DMA\_DREQ}}2$	1	O	13-1/13-5
$\overline{\text{LCS}}6$	LBC Chip Select 6	LBC	$\overline{\text{DMA\_DACK}}2$	1	O	13-1/13-5
$\overline{\text{LCS}}7$	LBC Chip Select 7	LBC	$\overline{\text{DMA\_DDONE}}2$	1	O	13-1/13-5
$\overline{\text{LWE}}[0:1]/$ $\text{LSDDQM}[0:1]/$ $\overline{\text{LBS}}[0:1]$	LBC Write Enable/Byte Lane Data Mask/Byte Select 0–1	LBC	—	2	O	13-1/13-5
$\overline{\text{LWE}}[2:3]/$ $\text{LSDDQM}[2:3]/$ $\overline{\text{LBS}}[2:3]$	LBC Write Enable/Byte Lane Data Mask/Byte Select 2–3	LBC	cfg_host_agt0 cfg_host_agt1	2	O	13-1/13-5
LBCTL	LBC Data Buffer Control	LBC	—	1	O	13-1/13-5
LALE	LBC Address Latch Enable	LBC	cfg_core_pll0	1	O	13-1/13-5
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL1/ $\overline{\text{LSDWE}}$	LBC GP Line 1/SDRAM Write Enable	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL2/ $\overline{\text{LOE}}/$ $\overline{\text{LSDRAS}}$	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-5
LGPL3/ $\overline{\text{LSDCAS}}$	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-5
LGPL4/ $\overline{\text{LGTA}}/$ $\text{LUPWAIT}/\text{LPBSE}$	LBC GP Line 4/GPCM Terminate Access/UPM Wait/Parity Byte Select	LBC	—	1	I/O	13-1/13-5
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq1	1	O	13-1/13-5
LCKE	LBC Clock Enable	LBC	—	1	O	13-1/13-5

Table 3-1. MPC8540 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
LCLK[0:2]	LBC Clock	LBC	—	3	O	13-1/13-5
LSYNC_IN, LSYNC_OUT	LBC DLL Synchronization	LBC	—	2	I/O	13-1/13-5
DMA_DREQ[0:1]	DMA Request 0–1	DMA	—	2	I	15-3/15-6
DMA_DREQ2	DMA Request 2	DMA	LCS5	1	I	15-3/15-6
DMA_DREQ3	DMA Request 3	DMA	IRQ9	1	I	15-3/15-6
DMA_DACK[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	15-3/15-6
DMA_DACK2	DMA Acknowledge 2	DMA	LCS6	1	O	15-3/15-6
DMA_DACK3	DMA Acknowledge 3	DMA	IRQ10	1	O	15-3/15-6
DMA_DDONE[0:1]	DMA Done 0–1	DMA	—	2	O	15-3/15-6
DMA_DDONE2	DMA Done 2	DMA	LCS7	1	O	15-3/15-6
DMA_DDONE3	DMA Done 3	DMA	IRQ11	1	O	15-3/15-6
MCP	Machine Check Processor	PIC	—	1	I	10-3/10-8
UDE	Unconditional Debug Event	PIC	—	1	I	10-3/10-8
IRQ[0:8]	External Interrupt 0–8	PIC	—	9	I	10-3/10-8
IRQ[9]	External Interrupt 9	PIC	DMA_DREQ3	1	I	10-3/10-8
IRQ[10]	External Interrupt 10	PIC	DMA_DACK3	1	I	10-3/10-8
IRQ[11]	External Interrupt 11	PIC	DMA_DDONE3	1	I	10-3/10-8
IRQ_OUT	Interrupt Output	PIC	—	1	O	10-3/10-8
UART_SIN[0:1]	DUART Serial Data In	Dual UART	—	2	I	12-2/12-4
UART_SOUT[0:1]	DUART Serial Data Out	Dual UART	—	2	O	12-2/12-4
UART_CTS[0:1]	DUART Clear To Send	Dual UART	—	2	I	12-2/12-4
UART_RTS[0:1]	DUART Ready to Send	Dual UART	—	2	O	12-2/12-4
IIC_SDA	I ² C Serial Data	I ² C	—	1	I/O	11-2/11-4
IIC_SCL	I ² C Serial Clock	I ² C	—	1	I/O	11-2/11-4
HRESET	Hard Reset	System Control	—	1	I	4-1/4-1
HRESET_REQ	Hard Reset Request	System Control	—	1	O	4-1/4-1
SRESET	Soft Reset	System Control	—	1	I	4-1/4-1
CKSTP_IN	Checkstop In	System Control	—	1	I	18-2/18-3
CKSTP_OUT	Checkstop Out	System Control	—	1	O	18-2/18-3
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	20-1/20-5
TRIG_OUT	Watchpoint Trigger Out	Debug	—	1	I/O	20-1/20-5
MSRCID[0:1]	Memory Debug Source Port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-5/9-10

Table 3-1. MPC8540 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
MSRCID[2:4]	Memory Debug Source Port ID 2–4	Debug	—	3	O	20-1/20-5
MDVAL	Memory Debug Data Valid	Debug	—	1	O	20-1/20-5
ASLEEP	Asleep	Power Mgmt	—	1	O	18-2/18-3
SYCLK	System Clock/PCI Clock	Clock	—	1	I	4-2/4-2
RTC	Real Time Clock	Clock	—	1	I	4-2/4-2
CLK_OUT	Clock Out	Clock	—	1	O	18-2/18-3
LSSD_MODE	LSSD Mode	Test	—	1	I	20-1/20-5
L1_TSTCLK	L1 Test Clock	Test	—	1	I	20-1/20-5
L2_TSTCLK	L2 Test Clock	Test	—	1	I	20-1/20-5
TEST_SEL	Test Select	Test	—	1	I	20-1/20-5
TCK	Test Clock	JTAG	—	1	I	20-1/20-5
TDI	Test Data In	JTAG	—	1	I	20-1/20-5
TDO	Test Data Out	JTAG	—	1	O	20-1/20-5
TMS	Test Mode Select	JTAG	—	1	I	20-1/20-5
TRST	Test Reset	JTAG	—	1	I	20-1/20-5

Table 3-2. MPC8540 Alphabetical Signal Reference

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
ASLEEP	Asleep	Power Mgmt	—	1	O	18-2/18-3
CKSTP_IN	Checkstop In	System Control	—	1	I	18-2/18-3
CKSTP_OUT	Checkstop Out	System Control	—	1	O	18-2/18-3
CLK_OUT	Clock Out	Clock	—	1	O	18-2/18-3
DMA_DACK2	DMA Acknowledge 2	DMA	LCS6	1	O	15-3/15-6
DMA_DACK3	DMA Acknowledge 3	DMA	IRQ10	1	O	15-3/15-6
DMA_DACK[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	15-3/15-6
DMA_DDONE2	DMA Done 2	DMA	LCS7	1	O	15-3/15-6
DMA_DDONE3	DMA Done 3	DMA	IRQ11	1	O	15-3/15-6
DMA_DDONE[0:1]	DMA Done 0–1	DMA	—	2	O	15-3/15-6
DMA_DREQ2	DMA Request 2	DMA	LCS5	1	I	15-3/15-6
DMA_DREQ3	DMA Request 3	DMA	IRQ9	1	I	15-3/15-6
DMA_DREQ[0:1]	DMA Request 0–1	DMA	—	2	I	15-3/15-6
EC_GTX_CLK125	Gigabit Reference Clock	Gigabit Clock	—	1	I	14-1/14-10

Table 3-2. MPC8540 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
EC_MDC	Ethernet Management Data Clock	Ethernet Management	cfg_tsec_reduce	1	O	14-1/14-10
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	14-1/14-10
FEC_COL	FEC Collision Detect	10/100	—	1	I	21-1/21-8
FEC_CRS	FEC Carrier Sense	10/100	—	1	I	21-1/21-8
FEC_RXD[0:3]	FEC Receive Data	10/100	—	4	I	21-1/21-8
FEC_RX_CLK	FEC Receive Clock	10/100	—	1	I	21-1/21-8
FEC_RX_DV	FEC Receive Data Valid	10/100	—	1	I	21-1/21-8
FEC_RX_ER	FEC Receiver Error	10/100	—	1	I	21-1/21-8
FEC_TXD[0:3]	FEC Transmit Data	10/100	—	4	O	21-1/21-8
FEC_TX_CLK	FEC Transmit Clock In	10/100	—	1	I	21-1/21-8
FEC_TX_EN	FEC Transmit Enable	10/100	—	1	O	21-1/21-8
FEC_TX_ER	FEC Transmit Error	10/100	—	1	O	21-1/21-8
$\overline{\text{HRESET}}$	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
IIC_SCL	I ² C Serial Clock	I ² C	—	1	I/O	11-2/11-4
IIC_SDA	I ² C Serial Data	I ² C	—	1	I/O	11-2/11-4
IRQ[0:8]	External Interrupt 0–8	PIC	—	9	I	10-3/10-8
IRQ[10]	External Interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-3/10-8
IRQ[11]	External Interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-3/10-8
IRQ[9]	External Interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-3/10-8
$\overline{\text{IRQ\_OUT}}$	Interrupt Output	PIC	—	1	O	10-3/10-8
L1_TSTCLK	L1 Test Clock	Test	—	1	I	20-1/20-5
L2_TSTCLK	L2 Test Clock	Test	—	1	I	20-1/20-5
LA27	LBC Burst Address	LBC	cfg_cpu_boot	1	O	13-1/13-5
LAD[0:31]	LBC Address/Data	LBC	—	32	I/O	13-1/13-5
LALE	LBC Address Latch Enable	LBC	cfg_core_pll0	1	O	13-1/13-5
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	13-1/13-5
LBCTL	LBC Data Buffer Control	LBC	—	1	O	13-1/13-5
LCKE	LBC Clock Enable	LBC	—	1	O	13-1/13-5
LCLK[0:2]	LBC Clock	LBC	—	3	O	13-1/13-5

Table 3-2. MPC8540 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
$\overline{\text{LCS5}}$	LBC Chip Select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	O	13-1/13-5
$\overline{\text{LCS6}}$	LBC Chip Select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	13-1/13-5
$\overline{\text{LCS7}}$	LBC Chip Select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	13-1/13-5
$\overline{\text{LCS}}[0:2]$	LBC Chip Select 0–2	LBC	cfg_rom_loc[0:2]	3	O	13-1/13-5
$\overline{\text{LCS}}[3:4]$	LBC Chip Select 3–4	LBC	—	2	O	13-1/13-5
$\text{LDP}[0:3]$	LBC Data Parity	LBC	—	4	I/O	13-1/13-5
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL1/ $\overline{\text{LSDWE}}$	LBC GP Line 1/SDRAM Write Enable	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL2/ $\overline{\text{LOE}}$ / $\overline{\text{LSDRAS}}$	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-5
LGPL3/ $\overline{\text{LSDCAS}}$	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-5
LGPL4/ $\overline{\text{LGTA}}$ / LUPWAIT/LPBSE	LBC GP Line 4/GPCM Terminate Access/UPM Wait/Parity Byte Select	LBC	—	1	I/O	13-1/13-5
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq1	1	O	13-1/13-5
$\overline{\text{LSSD\_MODE}}$	LSSD Mode	Test	—	1	I	20-1/20-5
LSYNC_IN, LSYNC_OUT	LBC DLL Synchronization	LBC	—	2	I/O	13-1/13-5
$\overline{\text{LWE}}[0:1]/$ LSDDQM[0:1]/ $\overline{\text{LBS}}[0:1]$	LBC Write Enable/Byte Lane Data Mask/Byte Select 0–1	LBC	—	2	O	13-1/13-5
$\overline{\text{LWE}}[2:3]/$ LSDDQM[2:3]/ $\overline{\text{LBS}}[2:3]$	LBC Write Enable/Byte Lane Data Mask/Byte Select 2–3	LBC	cfg_host_agt0 cfg_host_agt1	2	O	13-1/13-5
MA[0:14]	DDR Address	DDR Memory	—	15	O	9-3/9-6
MBA[0:1]	DDR Bank Select	DDR Memory	—	2	O	9-3/9-6
$\overline{\text{MCAS}}$	DDR Column Address Strobe	DDR Memory	—	1	O	9-3/9-6
MCKE[0:1]	DDR Clock Enable	DDR Memory	—	2	O	9-4/9-9
MCK[0:5], MCK[0:5]	DDR Differential Clocks (3 pairs/DIMM)	DDR Memory	—	12	O	9-4/9-9
$\overline{\text{MCP}}$	Machine Check Processor	PIC	—	1	I	10-3/10-8
$\overline{\text{MCS}}[0:3]$	DDR Chip Select (2/DIMM)	DDR Memory	—	4	O	9-3/9-6
MDM8	DDR ECC Data Mask	DDR Memory	—	1	O	9-3/9-6
MDM[0:7]	DDR Data Mask	DDR Memory	—	8	O	9-3/9-6
MDQS8	DDR ECC Data Strobe	DDR Memory	—	1	I/O	9-3/9-6

Table 3-2. MPC8540 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
MDQS[0:7]	DDR Data Strobe	DDR Memory	—	8	I/O	9-3/9-6
MDQ[0:63]	DDR Data	DDR Memory	—	64	I/O	9-3/9-6
MDVAL	Memory Debug Data Valid	Debug	—	1	O	20-1/20-5
MECC[0:7]	DDR Error Correcting Code	DDR Memory	—	8	I/O	9-3/9-6
$\overline{\text{MRAS}}$	DDR Row Address Strobe	DDR Memory	—	1	O	9-3/9-6
MSRCID[0:1]	Memory Debug Source Port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-5/9-10
MSRCID[2:4]	Memory Debug Source Port ID 2–4	Debug	—	3	O	20-1/20-5
MSYNC_IN, MSYNC_OUT	DDR DLL Synchronization In/Out	DDR Memory	—	2	I/O	9-4/9-9
$\overline{\text{MWE}}$	DDR Write Enable	DDR Memory	—	1	O	9-3/9-6
$\overline{\text{PCI\_ACK64}}$	PCI Acknowledge 64	PCI/X	—	1	I/O	16-1/16-7
PCI_AD[63:0]	PCI Address/Data	PCI/X	—	64	I/O	16-1/16-7
PCI_C/ $\overline{\text{BE}}$ [7:0]	PCI Command/Byte Enable	PCI/X	—	8	I/O	16-1/16-7
$\overline{\text{PCI\_DEVSEL}}$	PCI Device Select	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_FRAME}}$	PCI Frame	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_GNT0}}$	PCI Grant 0	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_GNT}}[4:1]$	PCI Grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	16-1/16-7
PCI_IDSEL	PCI Initial Device Select	PCI/X	—	1	I	16-1/16-7
$\overline{\text{PCI\_IRDY}}$	PCI Initiator Ready	PCI/X	—	1	I/O	16-1/16-7
PCI_PAR	PCI Parity	PCI/X	—	1	I/O	16-1/16-7
PCI_PAR64	PCI Parity 64	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_PERR}}$	PCI Parity Error	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_REQ0}}$	PCI Request 0	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_REQ64}}$	PCI Request 64	PCI/X	cfg_pci_width	1	I/O	16-1/16-7
$\overline{\text{PCI\_REQ}}[4:1]$	PCI Request 4–1	PCI/X	—	4	I	16-1/16-7
$\overline{\text{PCI\_SERR}}$	PCI System Error	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_STOP}}$	PCI Stop	PCI/X	—	1	I/O	16-1/16-7
$\overline{\text{PCI\_TRDY}}$	PCI Target Ready	PCI/X	—	1	I/O	16-1/16-7
RIO_RCLK, $\overline{\text{RIO\_RCLK}}$	RapidIO Receive Clocks	RapidIO	—	2	I	17-4/17-8

Table 3-2. MPC8540 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
RIO_RD[0:7], RIO_RD[0:7]	RapidIO Receive Data	RapidIO	—	16	I	17-4/17-8
RIO_RFRAME, RIO_RFRAME	RapidIO Receive Frame	RapidIO	—	2	I	17-4/17-8
RIO_TCLK, RIO_TCLK	RapidIO Transmit Clock	RapidIO	—	2	O	17-4/17-8
RIO_TD[0:7], RIO_TD[0:7]	RapidIO Transmit Data	RapidIO	—	16	O	17-4/17-8
RIO_TFRAME, RIO_TFRAME	RapidIO Transmit Frame	RapidIO	—	2	O	17-4/17-8
RIO_TX_CLK_IN RIO_TX_CLK_IN	RapidIO Transmit Clock In	RapidIO	—	2	I	17-4/17-8
RTC	Real Time Clock	Clock	—	1	I	4-2/4-2
SRESET	Soft Reset	System Control	—	1	I	4-1/4-1
SYSClk	System Clock/PCI Clock	Clock	—	1	I	4-2/4-2
TCK	Test Clock	JTAG	—	1	I	20-1/20-5
TDI	Test Data In	JTAG	—	1	I	20-1/20-5
TDO	Test Data Out	JTAG	—	1	O	20-1/20-5
TEST_SEL	Test Select	Test	—	1	I	20-1/20-5
TMS	Test Mode Select	JTAG	—	1	I	20-1/20-5
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	20-1/20-5
TRIG_OUT	Watchpoint Trigger Out	Debug	—	1	I/O	20-1/20-5
TRST	Test Reset	JTAG	—	1	I	20-1/20-5
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	14-1/14-10
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I	14-1/14-10
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	cfg_tsec1	1	O	14-1/14-10
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	—	8	I	14-1/14-10
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	14-1/14-10
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	14-1/14-10
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	14-1/14-10
TSEC1_TXD0	TSEC1 Transmit Data 0	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD1	TSEC1 Transmit Data 1	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD2	TSEC1 Transmit Data 2	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD3	TSEC1 Transmit Data 3	TSEC1	—	1	O	14-1/14-10
TSEC1_TXD4	TSEC1 Transmit Data 4	TSEC1	deviceID3	1	O	14-1/14-10

Table 3-2. MPC8540 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
TSEC1_TXD5	TSEC1 Transmit Data 5	TSEC1	deviceID2	1	O	14-1/14-10
TSEC1_TXD6	TSEC1 Transmit Data 6	TSEC1	deviceID1	1	O	14-1/14-10
TSEC1_TXD7	TSEC1 Transmit Data 7	TSEC1	deviceID0	1	O	14-1/14-10
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	14-1/14-10
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	14-1/14-10
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	14-1/14-10
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	14-1/14-10
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I	14-1/14-10
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	cfg_tsec2	1	O	14-1/14-10
TSEC2_RXD[0:7]	TSEC2 Receive Data	TSEC2	—	8	I	14-1/14-10
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	14-1/14-10
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	14-1/14-10
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	14-1/14-10
TSEC2_TXD0	TSEC2 Transmit Data 0	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD1	TSEC2 Transmit Data 1	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD2	TSEC2 Transmit Data 2	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD3]	TSEC2 Transmit Data 3	TSEC2	—	1	O	14-1/14-10
TSEC2_TXD4	TSEC2 Transmit Data 4	TSEC2	deviceID7	1	O	14-1/14-10
TSEC2_TXD5]	TSEC2 Transmit Data 5	TSEC2	deviceID6	1	O	14-1/14-10
TSEC2_TXD6	TSEC2 Transmit Data 6	TSEC2	deviceID5	1	O	14-1/14-10
TSEC2_TXD7	TSEC2 Transmit Data 7	TSEC2	deviceID4	1	O	14-1/14-10
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	14-1/14-10
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	14-1/14-10
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	—	1	O	14-1/14-10
UART_CTS[0:1]	DUART Clear To Send	Dual UART	—	2	I	12-2/12-4
UART_RTS[0:1]	DUART Ready to Send	Dual UART	—	2	O	12-2/12-4
UART_SIN[0:1]	DUART Serial Data In	Dual UART	—	2	I	12-2/12-4
UART_SOUT[0:1]	DUART Serial Data Out	Dual UART	—	2	O	12-2/12-4
UDE	Unconditional Debug Event	PIC	—	1	I	10-3/10-8

3.3 Configuration Signals Sampled at Reset

There are a number of signals that serve alternate functions as configuration input signals during system reset. They are summarized in Table 3-3. The detailed interpretation of their voltage levels during reset are described in Chapter 4, “Reset, Clocking, and Initialization.”

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of $\overline{\text{HRESET}}$. For more information about the timing requirements of these configuration signals relative to the negation of $\overline{\text{HRESET}}$, refer to the *MPC8540 Hardware Specification*.

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 *MPC8540 Hardware Specification*.

Note that the multiplexing of various signals on the MPC8540 is controlled by the PMUXCR register described in Chapter 18, “Global Utilities.”

Table 3-3. MPC8540 Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default	Table/Page
PCI	PCI_REQ64	cfg_pci_width	1	16-1/16-7
	PCI_GNT4	cfg_pci_mode	1	16-1/16-7
	PCI_GNT3	cfg_pci_debug	1	16-1/16-7
	PCI_GNT2	cfg_pci_arbiter	1	16-1/16-7
	PCI_GNT1	cfg_pci_impd	1	16-1/16-7
Ethernet Management	EC_MDC	cfg_tsec_reduce	1	14-1/14-10
TSEC1	TSEC1_TXD7	deviceID0	1	
	TSEC1_TXD6	deviceID1	1	
	TSEC1_TXD5	deviceID2	1	
	TSEC1_TXD4	deviceID3	1	
	TSEC1_GTX_CLK	cfg_tsec1	1	
TSEC2	TSEC2_TXD7	deviceID4	1	
	TSEC2_TXD6	deviceID5	1	
	TSEC2_TXD5	deviceID6	1	
	TSEC2_TXD4	deviceID7	1	
	TSEC2_GTX_CLK	cfg_tsec2	1	

Table 3-3. MPC8540 Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default	Table/Page
LBC	LA27	cfg_cpu_boot	1	13-1/13-5
	LA[28:31]	cfg_sys_pll[0:3]	must be driven	13-1/13-5
	$\overline{\text{LCS}}[0:2]$	cfg_rom_loc[0:2]	111	13-1/13-5
	LWE[2:3]/LBS[2:3]	cfg_host_agt0 cfg_host_agt1	11	13-1/13-5
	LALE	cfg_core_pll0	must be driven	13-1/13-5
	LGPL0/LSDA10	cfg_rio_clk0	1	13-1/13-5
	LGPL1/LSDWE	cfg_rio_clk1	1	13-1/13-5
	LGPL2/LOE/LSDRAS	cfg_core_pll1	must be driven	13-1/13-5
	LGPL3/LSDCAS	cfg_boot_seq0	1	13-1/13-5
	LGPL5/LSDAMUX	cfg_boot_seq1	1	13-1/13-5
Debug	MSRCID0	cfg_mem_debug	1	20-1/20-5
	MSRCID1	cfg_ddr_debug	1	20-1/20-5

3.4 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{HRESET}}$ is asserted), the MPC8540 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 MPC8540 ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state. Table 3-4 shows the states of the output-only signals (that is, the signals that are not multiplexed with other inputs, or are not used as reset configuration signals during system reset).

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset	Alt. Signal
DDR Memory	MDM[0:7]		
DDR Memory	MDM8		
DDR Memory	MBA[0:1]		
DDR Memory	MA[0:14]		
DDR Memory	$\overline{\text{MWE}}$		
DDR Memory	$\overline{\text{MRAS}}$		
DDR Memory	$\overline{\text{MCAS}}$		
DDR Memory	$\overline{\text{MCS}}[0:3]$		

Table 3-4. Output Signal States During System Reset (continued)

Interface	Signal	State During Reset	Alt. Signal
DDR Memory	MCKE[0:1]		
DDR Memory	MCK[0:5], MCK̄[0:5]		—
RapidIO	RIO_TCLK, RIO_TCLK		
RapidIO	RIO_TD[0:7], RIO_T̄D[0:7]		
RapidIO	RIO_TFRAME, RIO_T̄FRAME		
TSEC1	TSEC1_TXD3		
TSEC1	TSEC1_TXD2		
TSEC1	TSEC1_TXD1		
TSEC1	TSEC1_TXD0		
TSEC1	TSEC1_TX_EN		
TSEC1	TSEC1_TX_ER		
TSEC2	TSEC2_TXD3		
TSEC2	TSEC2_TXD2		
TSEC2	TSEC2_TXD1		
TSEC2	TSEC2_TXD[0]		
TSEC2	TSEC2_TX_EN		
TSEC2	TSEC2_TX_ER		
10/100	FEC_TXD[0:3]		
10/100	FEC_TX_EN		
10/100	FEC_TX_ER		
LBC	LCLK[0:2]		
LBC	LCS[3:4]		
LBC	LWE[0:1]/LBS[0:1]		
LBC	LBCTL		
LBC	LAL		
LBC	LGPL0/LSDA10		
LBC	LGPL1/LSDWE		
LBC	LGPL2/LOE/LSDRAS		
LBC	LGPL3/LSDCAS		
DMA	DMA_DACK[0:1]		—
DMA	DMA_DACK2		LCS6
DMA	DMA_DACK3		IRQ10
DMA	DMA_DDONE[0:1]		—
DMA	DMA_DDONE2		LCS7

Table 3-4. Output Signal States During System Reset (continued)

Interface	Signal	State During Reset	Alt. Signal
DMA	DMA_DDONE3		IRQ11
PIC	IRQ_OUT		
Dual UART	UART_SOUT[0:1]		
Dual UART	UART_RTS[0:1]		
System Control	HRESET_REQ		
System Control	CKSTP_OUT		
Debug	MSRCID[2:4]		—
Debug	MDVAL		—
Power Mgmt	ASLEEP		
Clock	CLK_OUT		—

Chapter 4

Reset, Clocking, and Initialization

4.1 Introduction

4.1.1 Overview

This chapter describes the reset, clocking, and initialization of the MPC8540.

4.2 External Signal Description

The following sections describe the reset and clocking signals.

4.2.1 Detailed Signal Descriptions

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

4.2.1.1 System Control Signals

Table 4-1 describes some of the system control signals of the MPC8540. Note that the $\overline{\text{CKSTP\_IN}}$ and $\overline{\text{CKSTP\_OUT}}$ signals are described in Chapter 18, “Global Utilities.”

Table 4-1. System Control Signals—Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{HRESET}}$	I	Hard Reset. The hard reset signal on the MPC8540 ($\overline{\text{HRESET}}$) causes the MPC8540 to abort all current internal and external transactions and set all registers to their default values. $\overline{\text{HRESET}}$ 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 (POR) Configuration,” for more information on the interpretation of the other MPC8540 signals during reset.
		Timing Assertion/Negation—See the <i>MPC8540 Hardware Specification</i> for specific timing information for this signal and the reset configuration signals.

Table 4-1. System Control Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
HRESET_REQ	O	Hard reset request. The hard reset request signal on the MPC8540 (HRESET_REQ) indicates to the board that a condition requiring the assertion of HRESET has been detected.
		State Meaning Asserted—A watchdog timer, a RapidIO command, or a boot sequencer failure has triggered a request for the device to be reset. Negated—Indicates no reset request.
		Timing Assertion/Negation—May occur at any time, synchronous to the CCB clock. Once asserted, HRESET_REQ does not negate until HRESET is asserted.
SRESET	I	The assertion of the soft reset input signal causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts, the assertion of SRESET will cause a checkstop. SRESET does not need to be asserted during a hard reset.
		State Meaning Asserted—Asserting SRESET causes a machine check interrupt (edge sensitive) to the e500 core. SRESET has no affect while HRESET is asserted. The power-on reset 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 2 CCB_clk cycles..

4.2.1.2 Clock Signals

Table 4-2 describes the clock signals of the MPC8540. Note that there is also a CLK_OUT signal in the MPC8540; the signal driven on the CLK_OUT pin is selectable and described in Section 18.3.1.16, “Clock Out Control Register (CLKOCR).”

Table 4-2. Clock Signals—Detailed Signal Descriptions

Signal	I/O	Description
SYSCLK	I	System clock/PCI clock (SYSCLK/PCI_CLK). SYSCLK is the primary clock input to the MPC8540. It is the clock source for the e500 core and all of the devices and interfaces that operate synchronously with the core. The SYSCLK input is multiplied up using a phase 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, include the L2 cache, the DDR DRAM and local bus memory controllers, the PCI interface, as well as other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL in the e500 core as well as the DLL's used to create the DDR DRAM and local bus memory clocks. When the PCI/PCI-X interface is being used, SYSCLK also functions as the PCI_CLK signal. Note that this is true both when the MPC8540 is in agent mode and when it is the host mode; the MPC8540 does not provide a separate PCI_CLK output in host mode.
		Timing Assertion/Negation—See the <i>MPC8540 Hardware Specification</i> for specific timing information for this signal.
RTC	I	Real time clock. The real time clock (RTC) signal is an optional input on the MPC8540. RTC may be used to clock the time base of the e500 core. The RTC input should have a frequency that is no more than half the frequency of the SYSCLK input.
		Timing Assertion/Negation—See the <i>MPC8540 Hardware Specification</i> for specific timing information for this signal.

4.3 Memory Map/Register Definition

There are no unique registers in the reset and clocking blocks of the MPC8540.

4.4 Functional Description

This section describes the various ways to reset the MPC8540 device, the power-on reset configurations, and the clocking on the MPC8540.

4.4.1 Reset Operations

There are two reset input signals on the MPC8540 that cause a hard and soft reset operation, respectively.

4.4.1.1 Hard Reset

The MPC8540 can be completely reset by the assertion of the $\overline{\text{HRESET}}$ input. The assertion of this signal by external logic is the equivalent of a power-on reset.

Refer to the *MPC8540 Hardware Specification* for the timing requirements for $\overline{\text{HRESET}}$ assertion and negation.

The MPC8540 also has a hard reset request output signal ($\overline{\text{HRESET\_REQ}}$) that indicates to external logic that a hard reset is being requested by either software, hardware, or an external agent.

4.4.1.2 Soft Reset

Assertion of $\overline{\text{SRESET}}$ causes a machine check interrupt to the e500 core. When this occurs, the soft reset flag is recorded in the machine check summary register (MCPSUMR) in the global utilities block so that software can identify the machine check as a soft reset condition. See the *e500 User's Manual* for more information on the machine check interrupt, and Section 18.3.1.13, "Machine Check Summary Register (MCPSUMR)." for more information on the setting of the soft reset flag. Note that if $\overline{\text{SRESET}}$ is asserted before the e500 core is configured to handle a machine check interrupt, a checkstop will occur.

4.4.2 Power-On Reset Sequence

The power-on reset (POR) sequence for the MPC8540 is as follows:

1. Power is applied to meet the specifications in the *MPC8540 Hardware Specification*.
2. User asserts $\overline{\text{HRESET}}$ and $\overline{\text{TRST}}$ causing all registers to be initialized to their default states and most I/O drivers to be tri-stated (some clock, clock enabled, and system control signals are active).

3. User applies a stable SYSCLK signal and stable PLL configuration inputs, and the device PLL begins locking to SYSCLK.
4. User negates $\overline{\text{HRESET}}$ after its required hold time and after POR configuration inputs have been valid for at least 4 SYSCLK cycles.

NOTE: $\overline{\text{TRST}}$ and $\overline{\text{SRESET}}$ Signals

If the JTAG signals are not used, then $\overline{\text{TRST}}$ may be tied inactive; however it is recommended that $\overline{\text{TRST}}$ not remain asserted after the negation of $\overline{\text{HRESET}}$. $\overline{\text{TRST}}$ may be connected directly to $\overline{\text{HRESET}}$.

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 will be paused until $\overline{\text{SRESET}}$ is negated. .

5. I/O drivers are enabled.
6. The PCI/PCI-X interface can assert $\overline{\text{DEVSEL}}$ in response to configuration cycles.
7. The e500 PLL configuration inputs are applied, allowing the e500 PLL to begin locking to the device clock (the CCB clock).
8. The CCB clock is cycled for approximately 50 μ s to lock the e500 PLL .
9. The internal hard reset to the e500 core is negated and soft resets are negated to the DLL's and other remaining I/O blocks. The DLL's begin to lock .
10. When DLL locking is completed, the boot sequencer is released, causing it to load configuration data from serial ROMs, if enabled, as described in Section 4.4.3.6, "Boot Sequencer Configuration."
11. When the boot sequencer completes, the RapidIO interface begins training, the PCI interface is released to accept external requests, and the boot vector fetch by the e500 core is allowed to proceed unless processor booting is further held off by POR configuration inputs as described in Section 4.4.3.5, "CPU Boot Configuration." The MPC8540 is now in its "Ready" state.
12. Ready condition is asserted on the external TRIG_OUT signal, synchronized to a rising edge of SYSCLK. Similarly, the ASLEEP signal negates, synchronous with the same rising edge of SYSCLK, to indicate the device has reached its Ready state.

Figure 4-1 below shows a timing diagram of the power-on reset sequence.

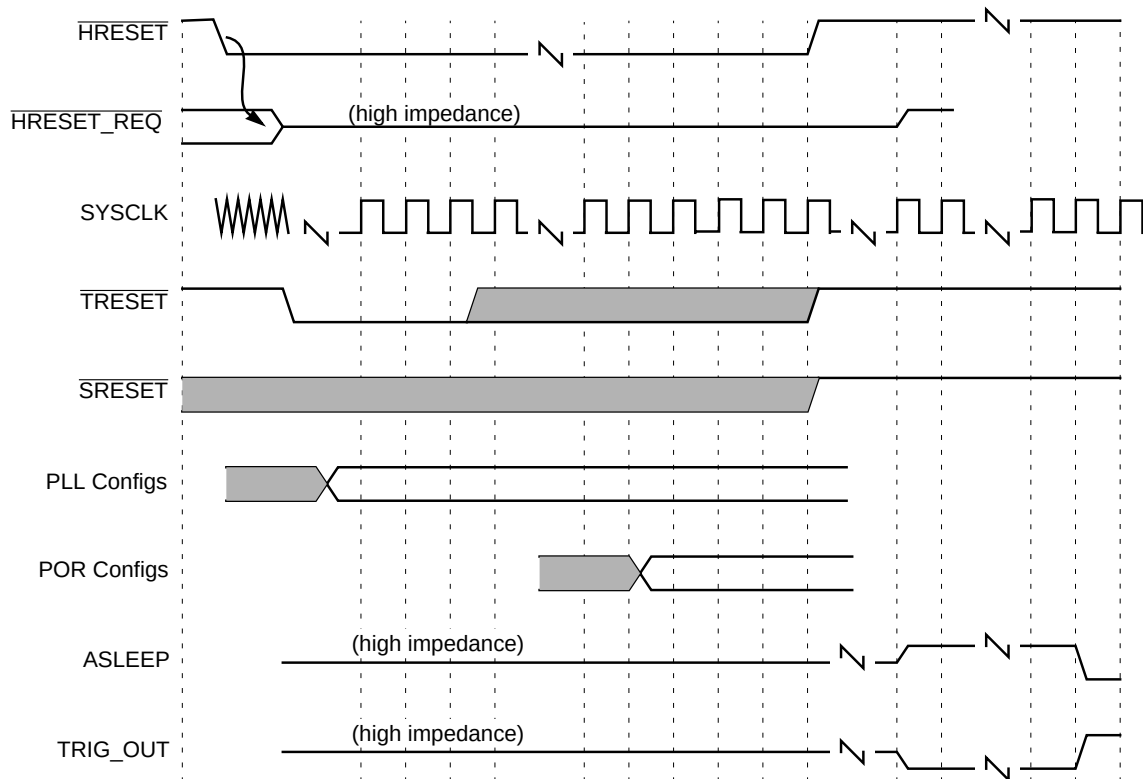


Figure 4-1. Power-on Reset Sequence

4.4.3 Power-On Reset (POR) Configuration

Various device functions are initialized by sampling certain pins during the assertion of $\overline{\text{HRESET}}$. The values of all these pins 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 pins must be in the high-impedance state.

Most POR configuration pins have internal pullup resistors so that if the desired setting is high, there is no need for a pullup resistor on the board. Other POR configuration pins do not use pullups and therefore must be pulled high or low. Refer to the *MPC8540 Hardware Specification* for proper resistor values to be used for pulling POR configuration signals high or low.

The functions and modes configured by power-on reset configuration pins are listed below.

NOTE

In the tables below, the binary value 0b0 represents a signal pulled down to GND and a value of 0b1 represents a pin pulled up to VDD, regardless of the sense of the functional signal name on the pin.

4.4.3.1 System PLL Ratio

The system PLL inputs establish the clock ratio between the PCI_CLK/SYSCLK input and the platform clock used by the MPC8540. The platform clock is the clock that drives the L2 cache, the DDR SDRAM data rate, and the e500 core complex bus (CCB), and is also called the CCB clock.

There is no default value for this PLL ratio; these signals must be pulled to the desired values.

Table 4-3. CCB Clock PLL Ratio

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LA[28:31] No Default	cfg_sys_pll[0:3]	0000	16:1 ratio CCB clock:SYSCLK
		0001	Reserved
		0010	2:1 ratio CCB clock:SYSCLK
		0011	3:1 ratio CCB clock:SYSCLK
		0100	4:1 ratio CCB clock:SYSCLK
		0101	5:1 ratio CCB clock:SYSCLK
		0110	6:1 ratio CCB clock:SYSCLK
		0111	Reserved
		1000	8:1 ratio CCB clock:SYSCLK
		1001	9:1 ratio CCB clock:SYSCLK
		1010	10:1 ratio CCB clock:SYSCLK
		1011	Reserved
		1100	12:1 ratio CCB clock:SYSCLK
		1101	Reserved
		1110	Reserved
		1111	Reserved

4.4.3.2 e500 Core PLL Ratio

These inputs establish the clock ratio between the e500 core complex bus (CCB) and the e500 core clock. There is no default value for this PLL ratio; these signals must be pulled to the desired values.

Table 4-4. e500 Core PLL Ratio

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LALE, LGPL2 No Default	cfg_core_pll[0:1]	00	2:1 e500 core:CCB
		01	5:2 e500 core:CCB
		10	3:1 e500 core:CCB
		11	7:2 e500 core:CCB

4.4.3.3 Boot ROM Location

These inputs establish the 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 4-5. Boot ROM Location

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
$\overline{\text{LCS}}[0:2]$ Default (111)	cfg_rom_loc[0:2]	000	PCI/PCI-X
		001	DDR SDRAM
		010	reserved
		011	RapidIO
		100	reserved
		101	Local bus GPCM—8-bit ROM
		110	Local bus GPCM—16-bit ROM
		111	Local Bus GPCM—32-bit ROM (default)

4.4.3.4 Host/Agent Configuration

These inputs configure the MPC8540 to act as a host or as an agent of a master on another interface. In host mode, the MPC8540 is immediately enabled to master transactions to the RapidIO and PCI interfaces. If the MPC8540 is an agent on the RapidIO or the PCI interfaces, then the MPC8540 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 MPC8540's interfaces appropriately. See details in the PCI and RapidIO programming models described in Chapter 16, "PCI Bus Interface," and Chapter 17, "RapidIO Interface," respectively.

Table 4-6. Host/Agent Configuration

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LWE[2:3] Default (11)	cfg_host_agt[0:1]	00	MPC8540 acts as an agent of both a PCI/PCI-X and a RapidIO device.
		01	MPC8540 acts as an agent of a RapidIO host.
		10	MPC8540 acts as an agent of a PCI/PCI-X host.
		11	MPC8540 acts as the host processor (default).

NOTE

If the MPC8540 is an agent on a particular interface, and the CPU is not in holdoff mode (as described in Section 4.4.3.5, “CPU Boot Configuration”) then the boot ROM should not be located on the interface where the external host exists, because the MPC8540 is not enabled to master reads onto that interface.

4.4.3.5 CPU Boot Configuration

This input specifies the boot configuration mode. If LA27 is sampled low at reset, the e500 core is prevented from fetching boot code until configuration by an external master is complete. The external master frees the CPU to boot by setting EEBPCR[CPU_EN] in the ECM CCB port configuration register (EEBPCR). See Section 8.3.1.2, “ECM CCB Port Configuration Register (EEBPCR),” for more information.

Table 4-7. CPU Boot Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
LA27 Default (1)	cfg_cpu_boot	0	CPU boot holdoff mode. The e500 core is prevented from booting until configured by an external master.
		1	The e500 core is allowed to boot without waiting for configuration by an external master

4.4.3.6 Boot Sequencer Configuration

The boot sequencer configuration options allow the boot sequencer to load configuration data from the serial ROM located on the I²C port before the host tries to configure the MPC8540. These options also specify normal or extended I²C addressing modes. See Section 11.4.10, “Boot Sequencer Mode,” for more information on the boot sequencer.

Table 4-8. Boot Sequencer Configuration

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LGPL3, LGPL5	cfg_boot_seq[0:1]	00	Reserved
Default (11)		01	Normal I ² C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I ² C interface. A valid ROM must be present.
		10	Extended I ² C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I ² C interface. A valid ROM must be present.
		11	Boot sequencer is disabled. No I ² C ROM is accessed.

NOTE

The processor core will be 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 signal described in Section 4.4.3.5, “CPU Boot Configuration.”

4.4.3.7 TSEC Width

This input selects standard versus reduced width for both of the three-speed Ethernet controller interfaces.

Table 4-9. TSEC Width Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
EC_MDC	cfg_tsec_reduce	0	Ethernet interfaces operate in reduced mode, either RTBI or RGMII, using only 4 transmit data signals and 4 receive data signals.
Default (1)		1	Ethernet interfaces operate in their standard TBI or GMII modes using 8 transmit data signals and 8 receive data signals.

NOTE

While the width of both interfaces is controlled by this one configuration input, the protocol (TBI or GMII) used by each is separately controlled with other configuration inputs described in Section 4.4.3.8, “TSEC1 Protocol,” and Section 4.4.3.9, “TSEC2 Protocol.”

4.4.3.8 TSEC1 Protocol

This input selects the protocol (GMII or TBI) used by the TSEC1 controller.

Table 4-10. TSEC1 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
TSEC1_GTX_CLK Default (1)	cfg_tsec1	0	The TSEC1 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width")
		1	The TSEC1 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width")

4.4.3.9 TSEC2 Protocol

This input selects the protocol (GMII or TBI) used by the TSEC2 controller.

Table 4-11. TSEC2 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
TSEC2_GTX_CLK Default (1)	cfg_tsec2	0	The TSEC2 controller operates using the GMII protocol (or RGMII if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width")
		1	The TSEC2 controller operates using the TBI protocol (or RTBI if configured in reduced mode as described in Section 4.4.3.7, "TSEC Width")

4.4.3.10 RapidIO Transmit Clock Source

These inputs specify the source for the RapidIO transmit clock.

Table 4-12. RapidIO Transmit Clock Source

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LGPL0, LGPL1 Default (11)	cfg_rio_clk[0:1]	00	Reserved
		01	The RapidIO receive clock is the source of the transmit clock
		10	The RapidIO transmit clock input (RIO_TX_CLK_IN) is the source of the transmit clock
		11	The CCB clock is the source of the transmit clock (default)

4.4.3.11 RapidIO Device ID

These inputs specify the device ID of the MPC8540 as used by RapidIO hosts. The default value is 0xFF.

Table 4-13. RapidIO Device ID

Functional Signals	Value (binary)	Meaning
TSEC1_TXD7	deviceID0	Device ID used for RapidIO hosts
TSEC1_TXD6	deviceID1	Device ID used for RapidIO hosts
TSEC1_TXD5	deviceID2	Device ID used for RapidIO hosts
TSEC1_TXD4	deviceID3	Device ID used for RapidIO hosts
TSEC2_TXD7	deviceID4	Device ID used for RapidIO hosts
TSEC2_TXD6	deviceID5	Device ID used for RapidIO hosts
TSEC2_TXD5	deviceID6	Device ID used for RapidIO hosts
TSEC2_TXD4	deviceID7	Device ID used for RapidIO hosts

4.4.3.12 PCI Width Configuration

This input configures the PCI/PCI-X interface to be in 32-bit or 64-bit extended mode of operation.

Table 4-14. PCI-32 Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
PCI_REQ64	cfg_pci_width	0	The PCI/PCI-X interface operates as a 64-bit interface.
Default (1)		1	The PCI/PCI-X interface operates as a 32-bit interface.

4.4.3.13 PCI I/O Impedance

This input selects the impedance of the PCI I/O drivers.

Table 4-15. PCI I/O Impedance

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
PCI_GNT1	cfg_pci_impd	0	25 ohm I/O drivers are used on the PCI interface.
Default (1)		1	42 ohm I/O drivers are used on the PCI interface.

4.4.3.14 PCI Arbiter Configuration

This input configures the on-chip PCI/PCI-X arbiter.

Table 4-16. PCI Arbiter Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
PCI_GNT2 Default (1)	cfg_pci_arbiter	0	The on-chip PCI/PCI-X arbiter is disabled. External arbitration is required.
		1	The on-chip PCI/PCI-X arbiter is enabled.

4.4.3.15 PCI Debug Configuration

This input configures PCI/PCI-X debug mode. In this mode, source ID information is driven onto the highest order address bits PCI_AD[63:59] during the bus command phase (PCI) or attribute phase (PCI-X).

Table 4-17. PCI Debug Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
PCI_GNT3 Default (1)	cfg_pci_debug	0	PCI debug is enabled. Source ID information is driven onto the highest order address bits, PCI_AD[63:59], during the bus command phase (PCI) or attribute phase (PCI-X).
		1	PCI operates in normal mode.

4.4.3.16 PCI-X Configuration

This input configures PCI or PCI-X mode on the PCI/PCI-X port. Note that this input does not support the three states of the PCIXCAP input defined in the PCI-X specification. It is sampled as either a 1 or a 0 during reset only.

Table 4-18. PCI/PCI-X Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
PCI_GNT4 Default (1)	cfg_pci_mode	0	PCI-X mode.
		1	PCI mode.

4.4.3.17 Memory Debug Configuration

This input configures the DDR and LBC memory controller debug outputs that can be driven onto the MSRCID and MDVAL debug signals.

Table 4-19. Memory Debug Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
MSRCID0 Default (1)	cfg_mem_debug	0	Debug information from the local bus controller (LBC) is driven on the MSRCID and MDVAL signals
		1	Debug information from the DDR DRAM controller is driven on the MSRCID and MDVAL signals.

4.4.3.18 DDR Debug Configuration

This input configures a DDR memory controller debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto ECC pins. ECC checking and generation are disabled. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8540 in this mode.

Table 4-20. DDR Debug Configuration

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
MSRCID1 Default (1)	cfg_ddr_debug	0	Debug information is driven on the ECC pins instead of normal ECC I/O. Boards must have ECC signals from memory devices disconnected.
		1	Debug information is not driven on ECC pins. ECC pins function in their normal mode.

4.4.3.19 PCI/PCI-X Output Hold Configuration

These inputs configure the output hold times for the PCI or PCI-X output drivers. The meanings are different for PCI and PCI-X because the required hold times are different in the two specifications. In either mode, the default value will meet the hold times required by the respective specification. Hold times are adjusted by adding or subtracting buffer delays to the intrinsic delay of the output driver. Refer to the *MPC8540 Hardware Specification* for specific timing information. Table 4-21 shows the hold time configurations for PCI mode, and Table 4-22 shows the hold time configurations for PCI-X mode.

Table 4-21. PCI Output Hold Configuration (cfg_pci_mode = 1)

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
LWE[0:1] Default (11)	cfg_pci_hold[0:1]	11	2 added buffer delays--required to meet 2ns hold time requirement
		10	3 added buffer delays (default+1)
		01	0 added buffer delays (default+2 mod 4)
		00	1 added buffer delays (default+3 mod 4)

Table 4-22. PCI-X Output Hold Configuration (cfg_pci_mode = 0)

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
$\overline{\text{LWE}}[0:1]$	cfg_pci_hold[0:1]	11	0 added buffer delays--meets 0.7ns hold time requirement
Default (11)		10	1 added buffer delays (default + 1)
		01	2 added buffer delays (default + 2)
		00	2 added buffer delays (default + 3)

4.4.3.20 Local Bus Output Hold Configuration

These inputs configure the output hold times for the local bus interface output drivers. The default values are designed to meet the specified AC timing requirements for the Local Bus. Hold times are adjusted by adding buffer delays to the intrinsic delay of the output driver. Refer to the *MPC8540 Hardware Specification* for specific timing information. Table 4-23 shows the hold time configurations for the local bus output drivers. Note that despite the POR configuration inputs, there will always be a minimum of 1 buffer of output hold delay between the LALE and LAD[0:31] signals.

Table 4-23. Local Bus Output Hold Configuration

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
$\overline{\text{LCS}}[3:4]$	cfg_lbiu_hold[0:1]	11	1 added buffer delay (0 added buffer delays for LALE)
Default (11)		10	2 added buffer delays (default+1) (1 added buffer delay for LALE)
		01	3 added buffer delays (default+2) (1 added buffer delay for LALE)
		00	reserved

4.4.4 Clocking

The following paragraphs describe the clocking within the MPC8540 device.

4.4.4.1 System Clock/PCI Clock

The MPC8540 takes a single input clock, SYSCLK, as its primary clock source for the e500 core and all of the devices and interfaces that operate synchronously with the core. The SYSCLK input is multiplied up using a phase 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, include the L2 cache, the DDR DRAM and local bus memory controllers, the PCI interface, as well as other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL in the e500 core as well as the DLL's used to create the DDR DRAM and local bus memory clocks.

When the PCI/PCI-X interface is being used, SYSCLK also functions as the PCI_CLK signal. Note that this is true both when the MPC8540 is in agent mode and when it is the host mode; the MPC8540 does not provide a separate PCI_CLK output in host mode.

Figure 4-2 below shows a block diagram of this clocking structure.

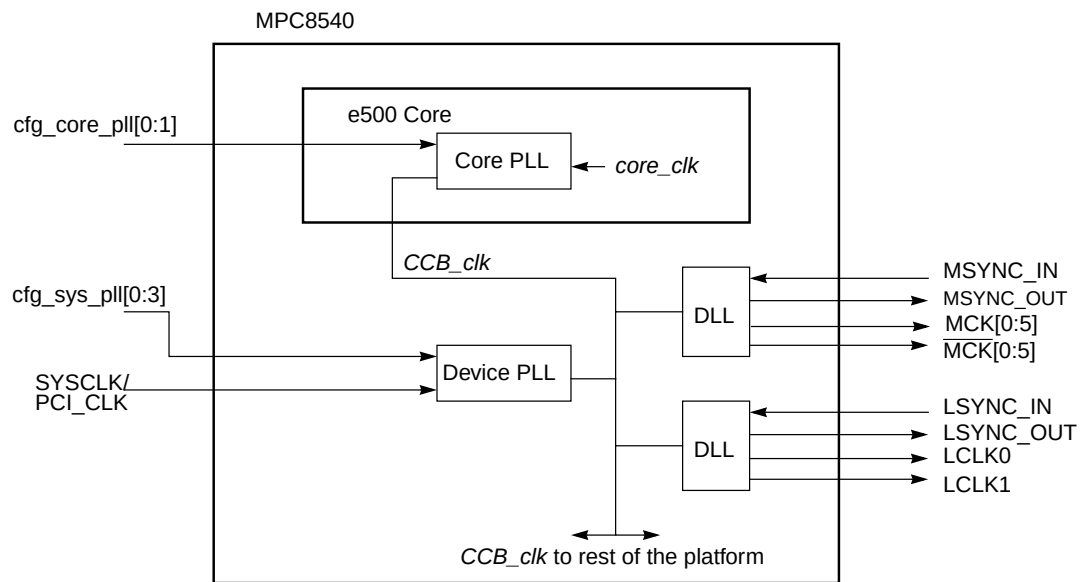


Figure 4-2. Clock Subsystem Block Diagram

4.4.4.2 RapidIO Clocks

The RapidIO transmit clocks (RIO_TCLK and $\overline{\text{RIO\_TCLK}}$) can be selected from one of three sources. If the desired RapidIO clock is being sourced by the device to which the MPC8540 RapidIO port is connected, then the transmit clock can be derived from the receive clock (RIO_RCLK and $\overline{\text{RIO\_RCLK}}$).

If the MPC8540 is to be the initial master of the RapidIO clock, then it can derive the RapidIO transmit clock from either its internal platform (CCB) clock or from a special differential clock input dedicated for this purpose, RIO_TX_CLK_IN and $\overline{\text{RIO\_TX\_CLK\_IN}}$. These inputs use LVDS signaling like other RapidIO inputs.

Figure 4-3 shows these clocking options in more detail.

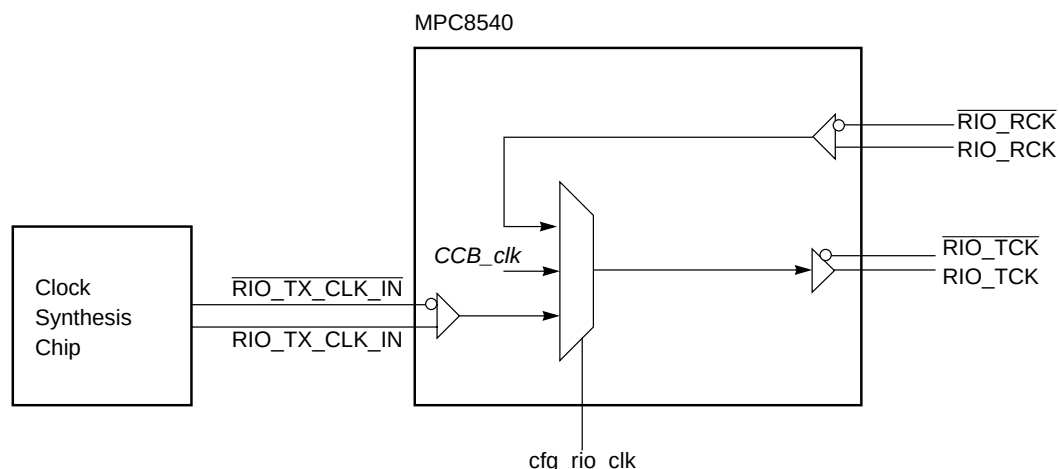


Figure 4-3. RapidIO Transmit Clock Options

4.4.4.3 Ethernet Clocks

The Ethernet blocks operate asynchronously with respect to the rest of the device. The Ethernet blocks use receive and transmit clocks supplied by their respective PHY chips, plus a 125 MHz clock input for gigabit protocols. Data transfers are synchronized to the CCB clock internally.

4.4.4.4 Real Time Clock

The real time clock input, RTC, can optionally be used to clock the e500 core time base unit. The RTC input should have a frequency that is no more than half the frequency of the SYSCLK input.

4.5 Initialization/Applications Information

4.5.1 Local Configuration Control

Table 4-24 shows the memory map for the configuration, control, and status registers.

Table 4-24. Local Configuration Control Register Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	Control configuration status registers base address register (CCSRBAR)	R/W	0x000F_F700	4.5.1.1.2/4-18
0x0_0008	Alternate configuration base address register (ALTCBAR)	R/W	0x0000_0000	4.5.1.2.1/4-19
0x0_0010	Alternate configuration size register (ALTCSR)	R/W	0x0000_0000	4.5.1.2.2/4-20
0x0_0020	Boot page translation register (BPTR)	R/W	0x0000_0000	4.5.1.3.1/4-21

4.5.1.1 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 registers (CCSRBAR). The default base address for the configuration, control and status registers is 0xFF70_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 MPC8540 to all functional units that need to be able to identify or create configuration transactions.

4.5.1.1.1 Updating CCSRBAR

Updates to CCSRBAR that relocate the entire 1-Mbyte region of configuration, control, and status registers, requires special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen. To 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 or RapidIO is configuring the device, it should set CCSRBAR to the desired final location before the e500 core is released to boot.
 - If the e500 core is initializing the device, it should set CCSRBAR to the desired final location before enabling other I/O devices to access the device.
- When the 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 (e.g. boot ROM). Follow this load with an **isync** instruction.
 - Read the contents of CCSRBAR from its new location, followed by another **isync** instruction.

4.5.1.1.2 Configuration Control and Status Register Base Address Register (CCSRBAR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	15		
R	0	0	0	0	0	0	0	0	0	0	0	0	BASE_ADDR			
W																
Reset	0000_0000_0000_1111															
	16							23	24	25	26	27	28	29	30	31
R	BASE_ADDR							0	0	0	0	0	0	0	0	
W																
Reset	1111_0111_0000_0000															
Offset	0x000															

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

Table 4-25. CCSRBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0.
12–23	BASE_ADDR	Identifies the 12 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

4.5.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALT CAR 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 ALT CBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 32-bit address that is mapped to the target specified in ALT CAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 11.4.10, “Boot Sequencer Mode,” for more information on the boot sequencer.

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 the ALTCBAR register.

4.5.1.2.1 Alternate Configuration Base Address Register (ALTCBAR)

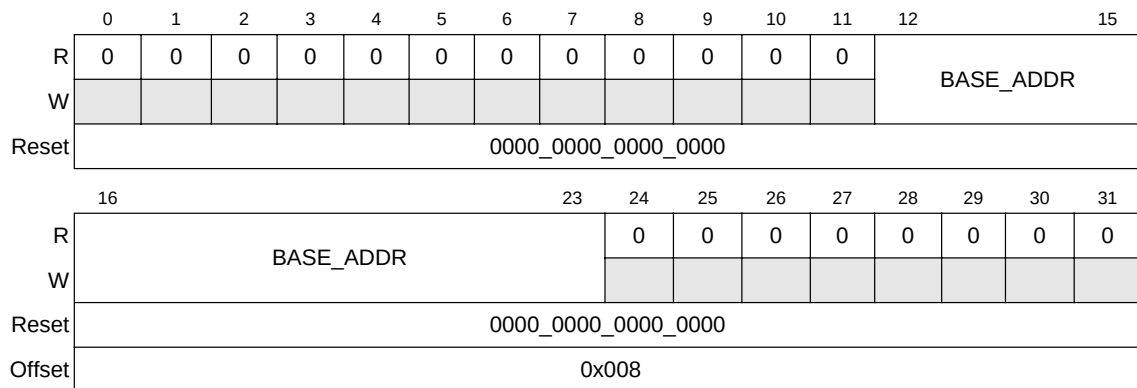


Figure 4-5. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-26. ALTCBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–23	BASE_ADDR	Identifies the 12 most significant address bits of an alternate window used for configuration accesses.
24–31	—	Write reserved, read = 0

4.5.1.2.2 Alternate Configuration Attribute Register (ALTCAR)

	0	1	2	3	4	5	6	7	8	11	12	13	14	15		
R	EN	0	0	0	0	0	0	0	TRGT_ID			0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x010															

Figure 4-6. Alternate Configuration Attribute Register (ALTCCR)

Table 4-27. ALTCAR Bit Settings

Bits	Name	Description
0	EN	Enable for a second configuration window. Like CCSRBAR, it has a set size of 1 Mbyte. 0 Second configuration window is disabled. 1 Second configuration window is enabled.
1–7	—	Write reserved, read = 0
8–11	TRGT_IF	Identifies the device ID to target when a transaction hits in the 1-Mbyte address range defined by the second configuration window. 0000 PCI/PCI-X interface 0001 Reserved 0010 Reserved 0011 Reserved 0100 Local bus controller 0101–0111Reserved 1000 Configuration, control and status registers 1001–1011Reserved 1100 RapidIO 1101 Reserved 1110 Reserved 1111 Local memory —DDR SDRAM and on-chip SRAM
12–31	—	Write reserved, read = 0

4.5.1.3 Boot Page Translation

When the e500 core comes out of reset, its MMU has one 4-Kbyte page defined at 0xFFFF_Fnnn. The core begins fetching boot code with a burst read from address 0xFFFF_FFF0. For systems in which the boot code resides at a different address, the MPC8540 provides boot page translation capability. Boot page translation is controlled by the boot page translation register (BPTR).

Translation of the boot page can be enabled by the boot sequencer or by an external host when the the MPC8540 is configured to be in boot holdoff mode. If translation is performed to a page outside of the default boot ROM address range defined in the MPC8540, 0xFF80_0000 to 0xFFFF_FFFF, then the external host or boot sequencer must 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 boot page translation register only defines the address translation, not the target interface. See Section 11.4.10, “Boot Sequencer Mode,” for more information on the boot sequencer.

4.5.1.3.1 Boot Page Translation Register (BPTR)

	0	1	2	3	4	5	6	7	8	9	10	11	12		15
R	EN	0	0	0	0	0	0	0	0	0	0	0	BOOT_PAGE		
W															
Reset	0000_0000_0000_0000														
	16														31
R	BOOT_PAGE														
W															
Reset	0000_0000_0000_0000														
Offset	0x020														

Figure 4-7. Boot Page Translation Register (BPTR)

Table 4-28 describes BPTR bit settings.

Table 4-28. BPTR Bit Settings

Bits	Name	Description
0	EN	0 Boot page is not translated 1 Boot page is translated as defined in the BPTR[BOOT_PAGE] parameter.
1–11	—	Write reserved, read = 0
12–31	BOOT_PAGE	Translation for boot page. If enabled, the high order 20 bits of accesses to 0xFFFF_Fnnn are replaced with this value.

4.5.2 Boot Sequencer

The boot sequencer is a DMA engine that accesses a serial ROM on the I²C interface and writes data to CCSR memory or the memory space pointed to by the alternate configuration base address register (ALTCBAR). See Section 4.5.1.2, “Accessing Alternate Configuration Space.” The boot sequencer is enabled by reset configuration pins as described in Section 4.4.3.6, “Boot Sequencer Configuration.” If the boot sequencer is

Initialization/Applications Information

enabled, the e500 core is held in reset until the boot sequencer has completed its operation. For more details, see Section 11.4.10, “Boot Sequencer Mode,” in the I²C chapter.

Part II

e500 Core Complex and L2 Cache

This part describes the many features of the MPC8540 core processor at an overview level and it describes the interaction between the core complex and the L2 cache. It contains the following chapters:

- Chapter 5, “e500 Core Complex Overview,” provides an overview of the e500 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, “e500 Register Summary,” provides a listing of the e500 registers in reference form.
- Chapter 7, “L2 Look-Aside Cache/SRAM,” describes the L2 cache of the MPC8540. 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 via the core complex bus (CCB).

Chapter 5

e500 Core Complex Overview

This chapter provides an overview of the e500 microprocessor core. It includes the following:

- An overview of the Book E version of the PowerPC 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)
- Signal descriptions
- A summary of the Book E architecture compatibility and migration from the original version of the PowerPC architecture as it was defined by Apple, IBM, and Motorola (referred to as the AIM version of the PowerPC architecture).

5.1 Overview

The e500 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 PowerPC architecture. The e500 implements the 32-bit portion of the Book E architecture using the lower words in the 64-bit general-purpose registers (GPRs).

Book E allows processors to provide auxiliary processing units (APUs), which are extensions to the architecture that can perform computational or system management functions. The most significant of these on the e500 is the signal processing engine APU (SPE APU), which includes a suite of vector instructions that use the upper and lower halves of the GPRs as a single two-element operand. Most APUs implemented on the e500 are defined by the Motorola Book E implementation standards (EIS).

The e500 provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and a 32-bit floating-point data type. For detailed information regarding the Book E instruction

Overview

set, see Chapter 2, “Instruction Model.” The SPE APU on the e500 also supports signed and unsigned fractional data types of 32 bits.

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.

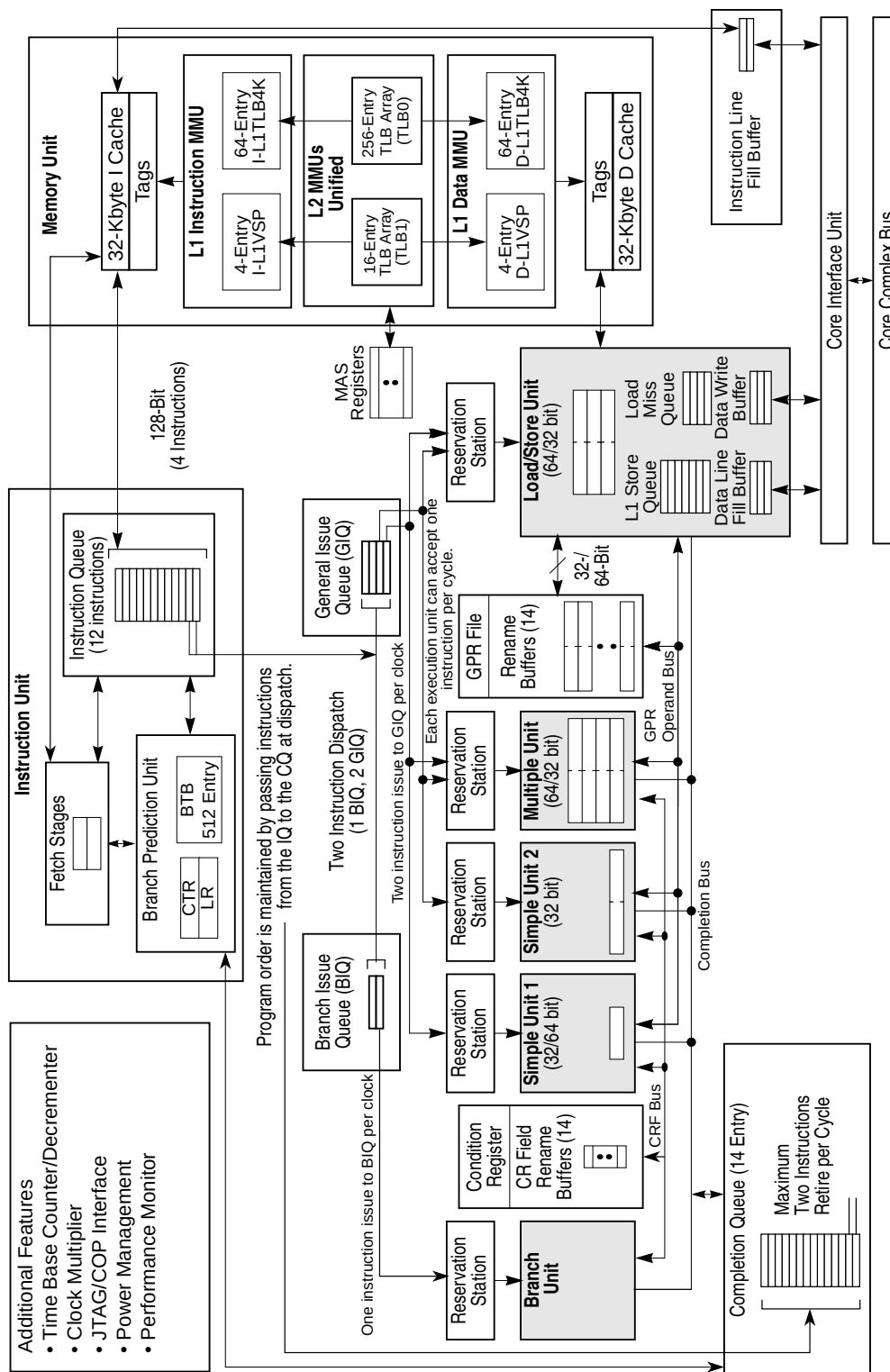


Figure 5-1. e500 Core Complex Block Diagram

The 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 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 APU vector instructions operate on the upper word of the GPRs; the upper 32-bits of the GPRs are not affected by 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 MU, LSU, and SU1 support 64-bit SIMD/signal processing 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 one 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, four-way set-associative, instruction and data TLB arrays that support 4-Kbyte pages. All of these arrays are maintained completely by the hardware with a 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 and a 256-entry, two-way set-associative unified TLB for 4-Kbyte page size support. These 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 Features

Key features of the e500 are summarized as follows:

- Implements Book E 32-bit architecture
- Additional auxiliary processing units.

The following APUs are defined by the Motorola Book E implementation standards (EIS).

- Integer select. This APU consists of the Integer Select instruction, **isel**, which is a conditional register move that helps eliminate conditional branches, decreases latency, and reduces the code footprint.
- 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**). For more information, see Section 5.11, “Performance Monitoring.”
- Cache block lock and unlock. This APU allows instructions and data to be locked into their respective caches on a cache line 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 APU 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 5.7, “Interrupts and Exception Handling.”
- Signal processing engine APU (SPE APU). 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 SPE loads and stores use 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.

To support the SPE, the e500 register set is modified in the following ways:

- GPRs are widened to 64 bits to support SPE vector operations. Note that the upper 32 bits are affected only by SPE instructions.
- A 64-bit accumulator (ACC) has been added

Features

- The signal processing and embedded floating-point status and control register (SPEFSCR) provides interrupt control and status, as well as various condition bits associated with the operations performed by the SPE APU.

These registers are shown in Figure 5-6. SPE instructions are grouped as follows:

- Single-cycle integer add and subtract with the same latencies for SPE APU operations as for the 32-bit equivalent
- Single-cycle logical operations
- Single-cycle shift and rotate
- Four-cycle integer pipelined multiplies
- 4-, 11-, 19-, and 35-cycle integer divides
- If **rA** or **rB** is zero, a floating-point divide takes 4 cycles. All other cases take 29 cycles.
- 4-cycle SIMD pipelined multiply-accumulate (MAC)
- 64-bit accumulator for no-stall MAC operations
- Single-precision floating-point single-instruction, multiple-data (SIMD) operations.
- Single-precision scalar floating-point single-instructions. These instructions are listed in Table 5-2.

The branch target buffer (BTB) locking is specific to the e500. The BTB locking APU gives the user the ability to lock, unlock, and invalidate BTB entries. Note that the BTB locking APU is an e500 cor feature; it is not part of the EIS.

- Cache structure—Separate 32-Kbyte, 32-byte line, 8-way set-associative 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 Book E memory coherency modes
 - Supports EIS-defined cache-locking instructions, listed in Table 5-4.
- 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, four-way set-associative branch target buffer (BTB), supported by the e500 BTB instructions listed in Table 5-1.
 - 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, context-synchronizing instructions
- Issue queues
 - Two-entry branch instruction issue queue (BIQ)
 - Four-entry general instruction issue queue (BIQ)
- 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)
 - Single-cycle add and subtract
 - Single-cycle shift and rotate
 - Single-cycle logical operations
 - Support for 64-bit signal processing engine APU instructions in SU1
- Multiple-cycle unit (MU). The MU is shown in Figure 5-2.

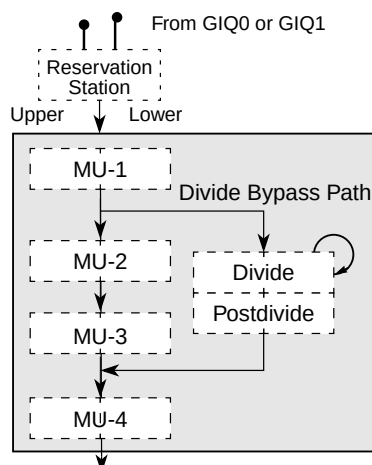


Figure 5-2. Four-Stage MU Pipeline, Showing the Divide Bypass

- The MU has the following features:
 - Four-cycle latency for integer and SPE multiplication (including SPE vector and scalar, integer, fractional, and floating-point multiply instructions).
 - Variable-latency divide: 4, 11, 19, and 35 cycles for all Book E and SPE integer divide instructions. If **rA** or **rB** is zero, a floating-point divide takes 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.
 - Four-cycle floating-point multiply
 - Four-cycle floating-point add and subtract
- Load/store unit (LSU) is shown in Figure 5-3. The LSU has the following features:
 - 3-cycle load latency
 - Fully pipelined
 - Four-entry load miss queue allows up to four load misses before stalling
 - Load hits can continue to be serviced when the load miss queue is full
 - Seven-entry L1 store queue allows full pipelining of stores
 - The three-entry data line fill buffer is used for loads and cacheable stores. Stores are allocated here so loads can access data from the store immediately.
 - There are three entries in the data write buffer: one dedicated for snoop pushes, one dedicated for cast outs, and one that can be used for either.

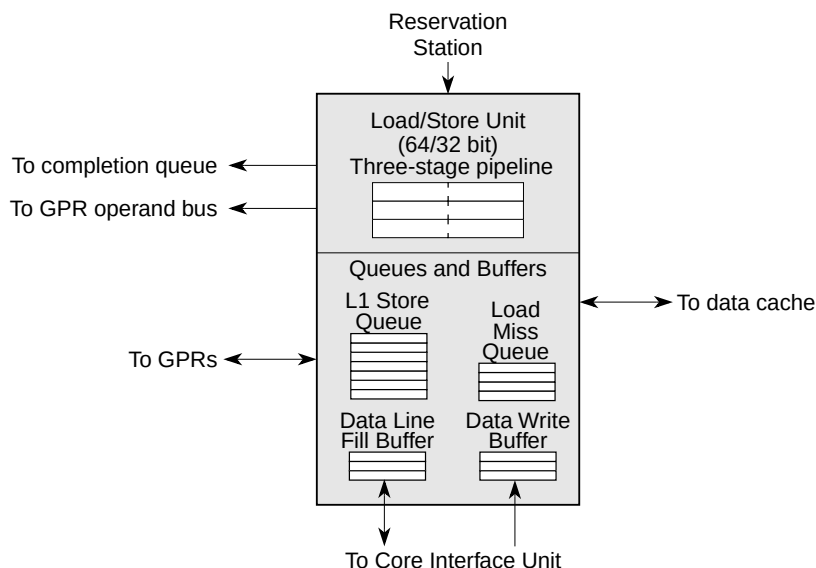


Figure 5-3. Three Stage Load/Store Unit

- Cache coherency
 - Supports both three-state MEI cache coherency and four-state cache coherency: modified-exclusive, exclusive, shared, and invalid (MESI)
 - Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)
 - 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 PowerPC processors (referred to as the 60x bus)
 - Two general-purpose read data, one write data bus
 - 128-bit data plus parity/tags (each data bus)
 - Support for out-of-order reads, in-order writes
 - Little to no data bus arbitration logic required for native systems
 - Easily adaptable to 60x-like environments
 - Support for one-level pipelining of addresses with address-retry responses
- Extended exception handling
 - Supports Book E interrupt model
 - Less than 10-cycle interrupt latency
 - Interrupt vector prefix register (IVPR)
 - Vector offset registers (IVORs) 0–15 as defined in Book E, plus e500-defined IVORs 32–35

Features

- Exception syndrome register (ESR)
 - Book E-defined preempting critical interrupt, including critical interrupt status registers (CSRR0 and CSRR1) and an **rfci** instruction
- e500-specific interrupts not defined in Book E architecture
 - SPE APU unavailable exception
 - Vector floating-point data exception
 - Vector floating-point round exception
 - Performance monitor
- Memory management unit (MMU)
 - 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
 - 256-entry, 2-way set-associative TLB for 4-Kbyte pages
 - Software reload for TLBs
 - Virtual memory support for as much as 4 Gbytes (2^{32}) of virtual memory
 - Real memory support for as much as 4 Gbytes (2^{32}) of physical memory
 - Support for big-endian and true little-endian memory on a per-page basis
- Power management
 - Low power design
 - 1.5-volt design
 - Internal clock multipliers ranging from 1x to 8x from bus clock
 - Power-saving modes: standby and shutdown
 - Dynamic power management of execution units, caches, and MMUs
- Testability
 - LSSD scan design
 - JTAG interface
 - ESP support
 - ABIST for arrays
 - LBIST

- Nexus debug support (not implemented in the MPC8540)
- Reliability and serviceability
 - Internal code parity
 - Parity checking on e500 local bus

5.3 Instruction Set

The e500 implements the following instructions:

- The Book E instruction set for 32-bit implementations. This is composed primarily of the user-level instructions defined by the PowerPC user instruction set architecture (UISA). The e500 does not include the Book E floating-point instructions or the Load String Word Indexed instruction (**lswx**).
- The e500 supports the following implementation-specific instructions:
 - Integer select APU. This APU consists of the Integer Select instruction, **isel**, which incorporates an if...then else statement that selects between two source registers by comparison to a CR bit. This instruction eliminates conditional branches, decreases band latency, and reduces the code footprint.
 - Branch target buffer (BTB) locking APU instructions. The core complex provides a 512-entry BTB for efficient processing of branch instructions. The BTB is a branch target address cache (BTAC), organized as 128 rows with four-way set associativity, that holds the address and target instruction of the 512 most-recently taken branches. Table 5-1 lists the BTB instructions.

Table 5-1. BTB Locking APU Instructions

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

- SPE APU vector instructions. New vector instructions are defined that view the 64-bit GPRs as being composed of a vector of two 32-bit elements (some of the instructions also read or write 16-bit elements). These new instructions can also perform scalar operations by ignoring the results of the upper 32-bit half of the register file. Some scalar instructions are defined for DSP that produce a 64-bit scalar result. These instructions include vector floating-point instructions that operate on two 32-bit single-precision operands that reside in the upper and lower halves of the GPRs. For complete lists of SPE APU vector instructions, see Section 10.3.1.2, “SPE APU Instruction Model.”
- The SPE APU supports a subset of single-precision scalar floating-point instructions that use the lower 32-bits of the GPRs for single-precision

floating-point calculations. This subset is referred to as the SPFP FPU. These instructions are listed in Table 5-2.

Table 5-2. SPE APU Scalar Floating-Point Instructions

Instruction	Mnemonic	Syntax
Floating-Point Absolute Value	efsabs	rD,rA
Floating-Point Add	efsadd	rD,rA,rB
Convert Floating-Point from Signed Fraction	efscfsf	rD,rB
Convert Floating-Point from Signed Integer	efscfsi	rD,rB
Convert Floating-Point from Unsigned Fraction	efscfuf	rD,rB
Convert Floating-Point from Unsigned Integer	efscfui	rD,rB
Floating-Point Compare Equal	efscmpeq	crfD,rA,rB
Floating-Point Compare Greater Than	efscmpgt	crfD,rA,rB
Floating-Point Compare Less Than	efscmplt	crfD,rA,rB
Convert Floating-Point to Signed Fraction	efscfsf	rD,rB
Convert Floating-Point to Signed Integer	efscfsi	rD,rB
Convert Floating-Point to Signed Integer with Round toward Zero	efscfsiz	rD,rB
Convert Floating-Point to Unsigned Fraction	efscfuf	rD,rB
Convert Floating-Point to Unsigned Integer	efscfui	rD,rB
Convert Floating-Point to Unsigned Integer with Round toward Zero	efscfui	rD,rB
Floating-Point Divide	efsdiv	rD,rA,rB
Floating-Point Multiply	efsmul	rD,rA,rB
Floating-Point Negative Absolute Value	efsnabs	rD,rA
Floating-Point Negate	efsneg	rD,rA
Floating-Point Subtract	efssub	rD,rA,rB
Floating-Point Test Equal	efststeq	crfD,rA,rB
Floating-Point Test Greater Than	efststgt	crfD,rA,rB
Floating-Point Test Less Than	efststlt	crfD,rA,rB

— Performance monitor APU—Table 5-3 lists performance monitor APU instructions.

Table 5-3. Performance Monitor APU Instructions

Name	Mnemonic	Syntax
Move from Performance Monitor Register	mfpmr	rD,PMRN
Move to Performance Monitor Register	mtpmr	PMRN,rS

- Cache block lock and unlock APU. The cache block lock and unlock APU consists of the instructions described in Table 5-4.

Table 5-4. Cache Block Lock and Unlock APU Instructions

Name	Mnemonic	Syntax
Data Cache Block Lock Clear	dcblc	CT, rA, rB
Data Cache Block Touch and Lock Set	dcbtls	CT, rA, rB
Data Cache Block Touch for Store and Lock Set	dcbtstls	CT, rA, rB
Instruction Cache Block Lock Clear	icblc	CT, rA, rB
Instruction Cache Block Touch and Lock Set	icbtls	CT, rA, rB

5.4 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 one instruction to pass through all of the pipeline stages, once a pipeline is full, instructions complete with a throughput that is far 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 (SU1, and SU2). The MU and SU1 arithmetic execution units also execute 64-bit SPE vector instruction, 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.

5.4.1 Branch Detection and Prediction

To improve branch performance, the e500 provides an implementation-specific dynamic branch prediction using a branch target buffer (BTB) to resolve branch instructions and improve the accuracy of branch predictions. Each of the 512 entries in the four-way set associative address cache of branch target addresses includes a 2-bit branch history table, whose value is incremented or decremented depending on whether the branch was taken.

These bits can take 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, to accurately predict those branches, to speculatively execute speculative instructions, and to quickly recover and revise incorrect predictions.

In the e500, all branch instructions are assigned positions in the completion queue at dispatch and 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 had 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 update in the BTB entry, and nonspeculative fetching begins from the correct path.

5.4.2 e500 Execution Pipeline

The seven stages of the e500 execution pipeline—fetch, fetch2/predecode, decode/dispatch, issue, execute, complete, and write back—are highlighted in grey in Figure 5-4.

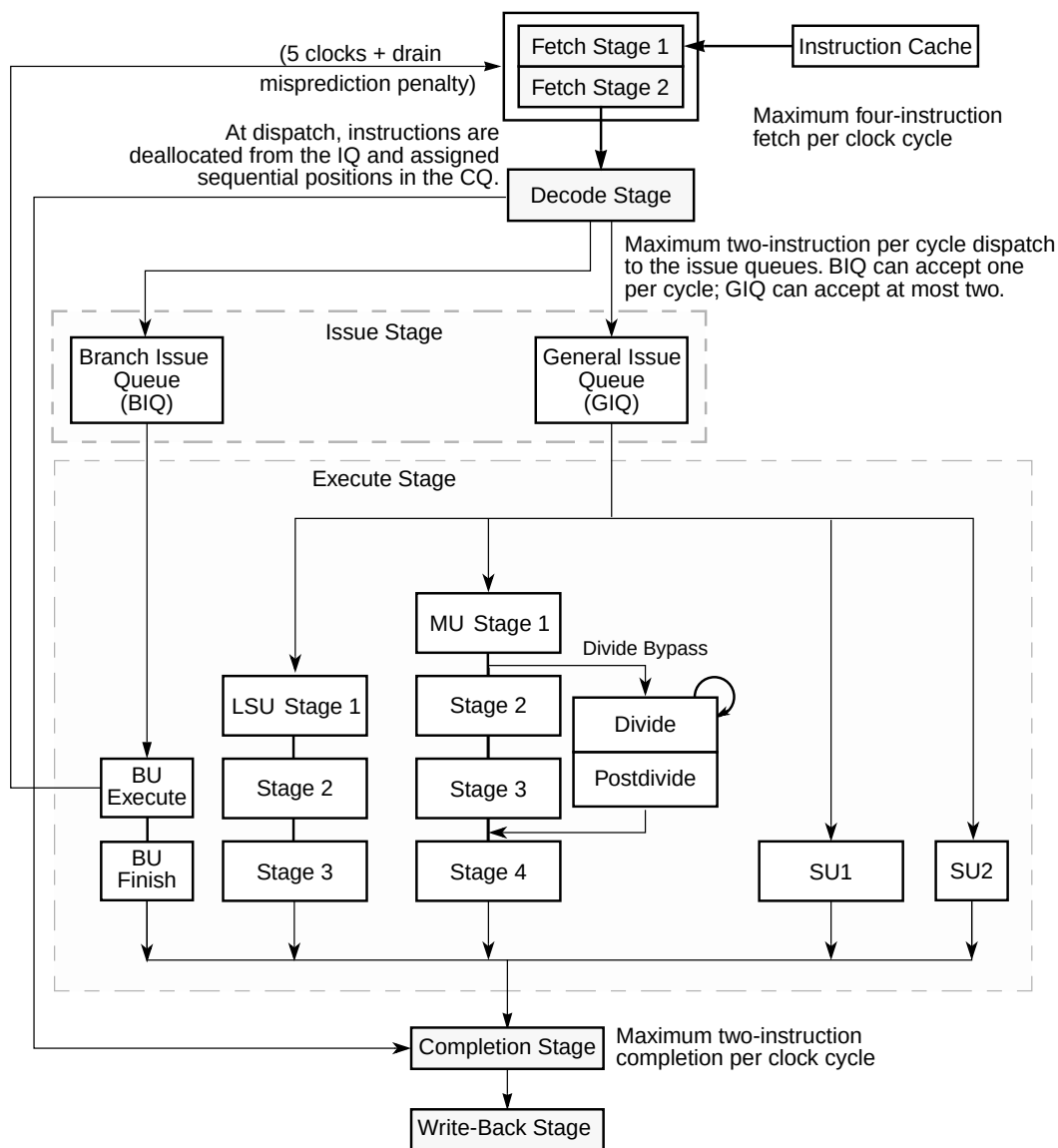


Figure 5-4. Instruction Pipeline Flow

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 (**isync**, **rfi**, **sc**, **nops**, 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. Instruction dispatch requires the following:
 - 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 SPE APU loads and stores) are dispatched to the GIQ, shown in Figure 5-5.

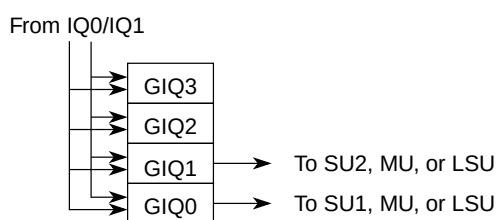


Figure 5-5. GPR Issue Queue (GIQ)

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 5-4.

- 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. Section 4.7.4, “Completion Unit Resource Requirements” describes completion dependencies.

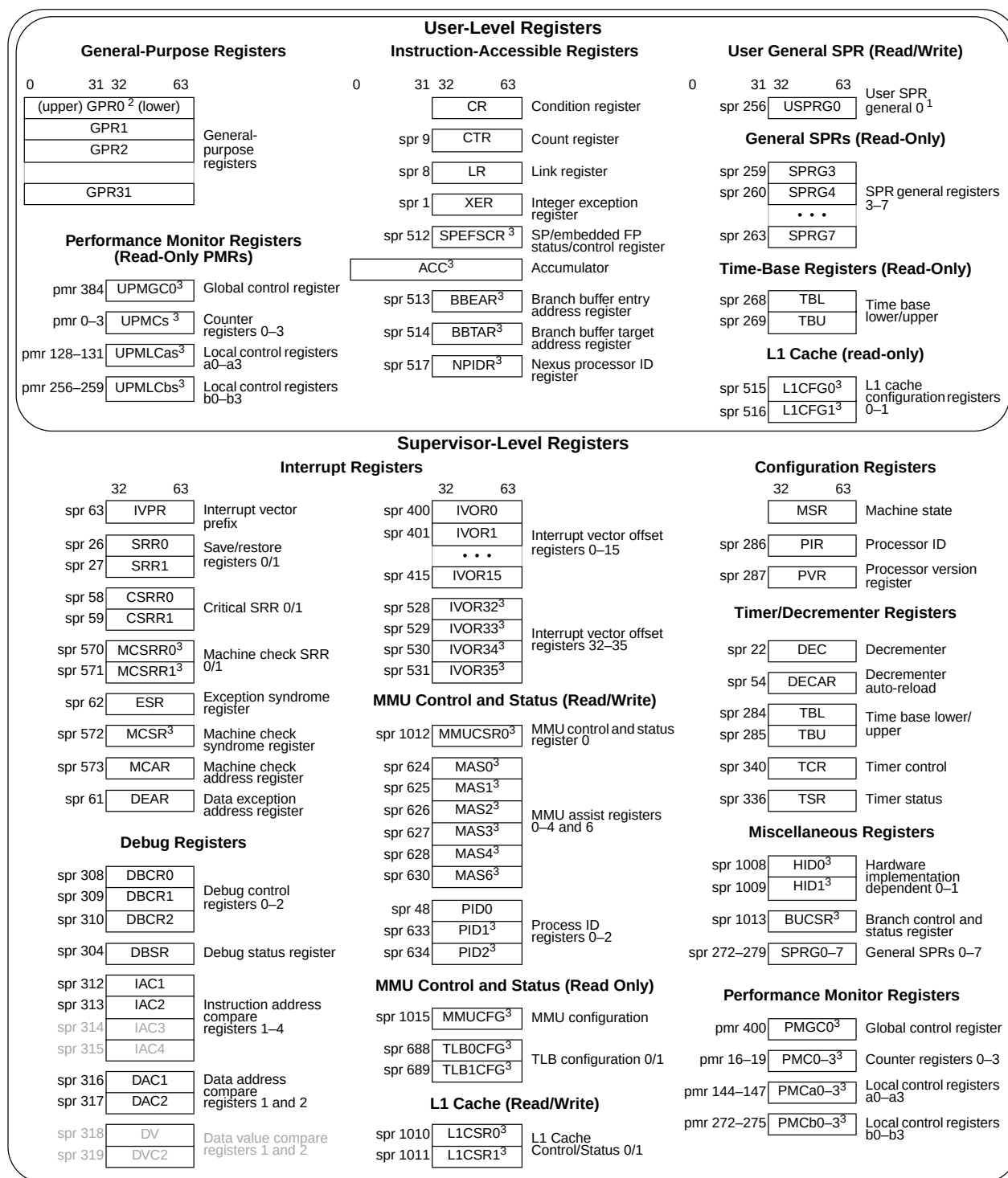
The write-back stage occurs in the clock cycle after the instruction is retired.

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 5-4, the LSU, MU, and SU1 replicate logic to support the 64-bit vector 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.

5.5 Programming Model

The following section describes the e500 core registers defined in Book E, the Motorola Book E implementation standards (EIS), and registers that are specific to the e500. Table 5-6 shows the e500 register set.



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 e500-specific and are not part of the Book E architecture.

Figure 5-6. e500 Core Programming Model

5.6 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.

Data caches support three-state MEI and four-state MESI memory coherency protocols with 3-bit status and 1-bit coherency valid fields. 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.

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 5.9.3, “Cache Control Instructions.”

5.7 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.

5.7.1 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 type, to verify the specific cause of the exception and take appropriate action.

The core complex provides the interrupts described in Section 5.7.5, “Interrupt Registers,” and Table 5-7.

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

1. Machine check—highest priority

2. Critical interrupt—medium priority
3. Noncritical interrupt—lowest priority

5.7.2 Interrupt Classes

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

- Asynchronous interrupts are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported in the 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 are further divided into precise and imprecise types.
 - 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 allow determination of which of the two instructions has been addressed in the appropriate save/restore register.
 - Synchronous imprecise interrupts are those that may indicate the address of the exception 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.

5.7.3 Interrupt Types

There are three types of interrupt, described as follows:

- 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 OEA portion the Power PC 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 non-critical interrupts can be masked by the external interrupt enable bit, MSR[EE].

- **Critical interrupts**—Critical interrupts can be taken during a non-critical 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]. Book E defines the critical input, watchdog timer, and machine check interrupts as critical interrupts, but the e500 defines a third set of resources for the machine check interrupt, as described below.
- **Machine check interrupt**—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].

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 (see Section 5.10, “Interrupt Ordering and Masking”).

5.7.4 Upper Bound on Interrupt Latencies

The core complex interrupt latency is defined here 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). The interrupt latency of the core complex is determinate unless a guarded load or a cache-inhibited **stwcx.** instruction is being executed. As a guarded load or a cache-inhibited **stwcx.** instruction is being executed, the interrupt latency of the core complex is indeterminate. The minimum interrupt latency for the core complex is 3 core clocks and the maximum is 8. These clocks are described in Table 5-5.

Table 5-5. Interrupt Latency

Action	Guarded Window	Reaching a Recoverable State.	Interrupt Processing	Redirect
Clocks	4/0	2/1	1	1

The interrupt latency defined in Table 5-5 does not include the 2 bus clock cycles required to synchronize the interrupt signal from the pad.

When ever an interrupt is taken, all the instructions in the instruction queue are thrown away except when 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 load/store unit is deallocated.

5.7.5 Interrupt Registers

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

Table 5-6. Interrupt Registers

Register	Description
Non-Critical Interrupt Registers	
SRR0	Save/restore registers—Stores the address of the instruction causing the exception or the address of the instruction that will execute after the rfi instruction.
SRR1	Save/restore register 1—SRR1 saves machine state on non-critical interrupts and restores machine state after an rfi instruction is executed.
Critical Interrupt Registers	
CSRR0	Critical save/restore register 0—on critical interrupts, CSRR0 stores either the address of the instruction causing the exception or the address of the instruction that will execute after the rfci instruction.
CSRR1	Critical save/restore register 1—CSRR1 saves 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—MCSRR0 is used to store the address of the instruction that will execute after an rfmci instruction is executed.
MCSRR1	Machine check save/restore register 1—MCSRR1 stores machine state on machine check interrupts and restores machine state (if recoverable) after an rfmci instruction is executed.
MCAR	Machine check address register —MCAR 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—MCSR saves machine state information on machine check interrupts and restores machine state after an rfmci instruction is executed.
ESR	Exception syndrome register—ESR 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 APU 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 APU.

Table 5-6. Interrupt Registers (continued)

Register	Description
Other Interrupt Registers	
DEAR	Data exception address register—DEAR contains 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	Interrupt vector prefix register—IVPR[32–47] contains the high-order 16 bits of the address of the exception processing routines defined in the IVOR registers.
IVORs	Interrupt vector offset registers—The IVORs contain the low-order offset of the address of the exception processing routines defined in the IVOR registers. See Table 5-7.

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. For more information, see Chapter 5, “Interrupts and Exceptions.”

Table 5-7. Interrupt Vector Registers and Exception Conditions

Register	Interrupt
Book E–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 (not supported on the e500)
IVOR8	System call interrupt offset
IVOR9	Auxiliary processor unavailable interrupt offset (not supported on the e500)
IVOR10	Decrementer interrupt offset
IVOR11	Fixed-interval timer interrupt offset
IVOR12	Watchdog timer interrupt offset
IVOR13	Data TLB error interrupt offset
IVOR14	Instruction TLB error interrupt offset
IVOR15	Debug interrupt offset
e500-Specific IVORs	

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

Register	Interrupt
IVOR32	SPE APU unavailable interrupt offset
IVOR33	SPE floating-point data exception interrupt offset
IVOR34	SPE floating-point round exception interrupt offset
IVOR35	Performance monitor

5.8 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 Book E. 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 coherence required (M)
- Guarded (G)
- Endianess (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, as shown in Figure 5-7.

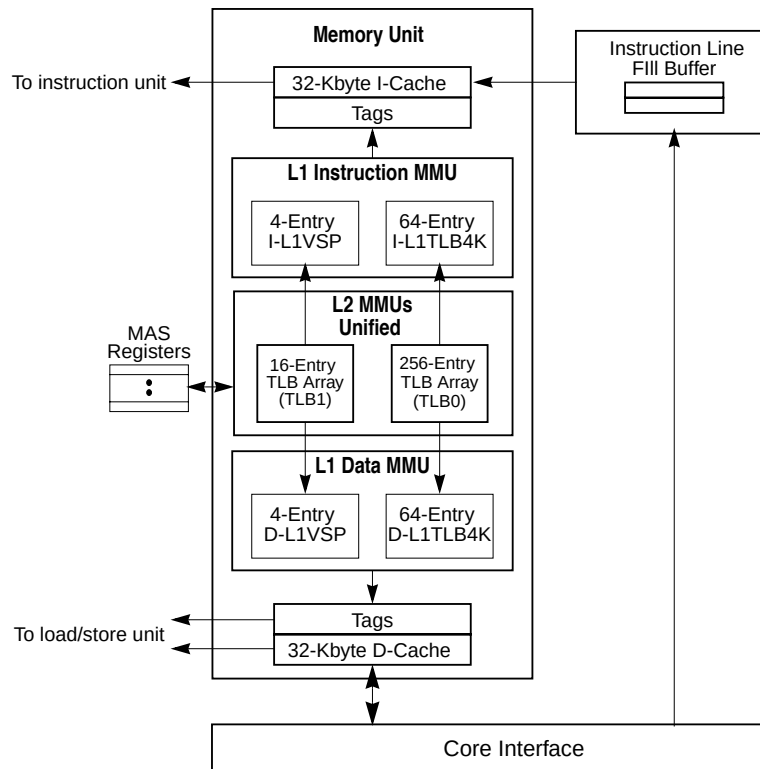


Figure 5-7. MMU Structure

The 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 page sizes
- 256-entry, 2-way set-associative TLB array (TLB0) that supports only 4-Kbyte pages
- 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

5.8.1 Address Translation

The core complex fetch and load/store units generate 32-bit effective addresses. The MMU translates these addresses to real 32-bit addresses (which are used for memory bus accesses) using an interim 41-bit virtual address. The translation flow is shown in Figure 5-8.

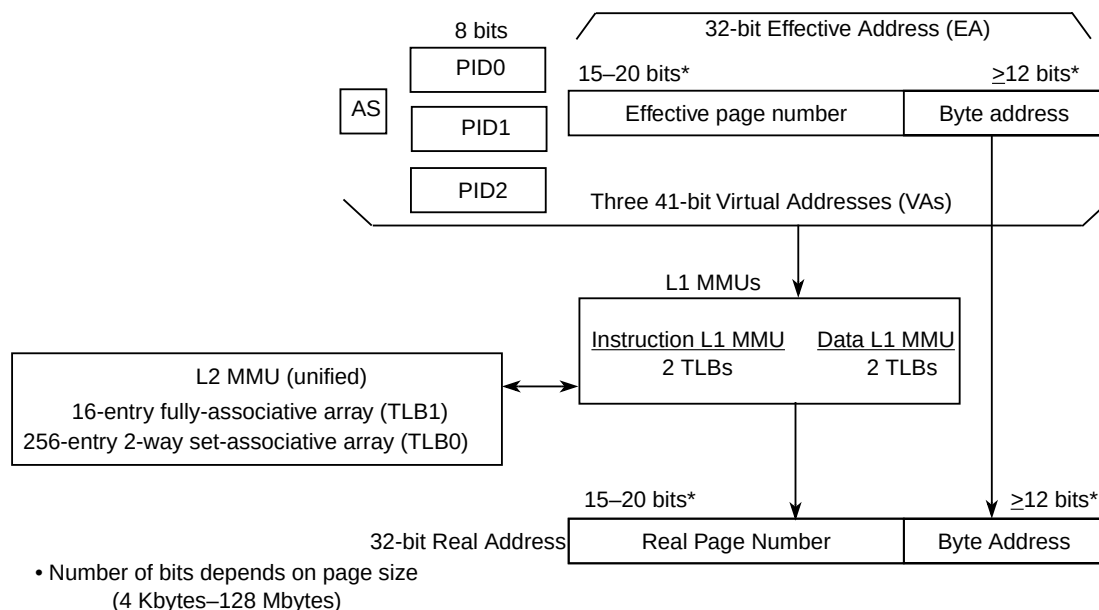


Figure 5-8. Effective-to-Real Address Translation Flow

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.

5.8.2 MMU Assist Registers (MAS1–MAS4 and MAS6)

Book E defines SPR numbers for the MMU assist registers, which 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. Book E leaves MAS register bit definitions up to the implementations. To ensure consistency among Motorola Book E processors, certain aspects of the implementation are defined by the Motorola Book E standard, whereas more specific details are left to individual implementations. 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:

- 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. 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 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.

5.8.3 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.

5.8.4 TLB Coherency

The core complex provides the ability to invalidate a TLB entry as defined in the Book E 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] = 1. The core complex also snoops TLB invalidate transactions on the CCB from other bus masters.

5.9 Memory Coherency

The core complex supports both three- and 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 three-state MEI coherency protocol is suited to

single-processor systems that include DMA and is also applicable to multiple-processor systems with low to moderate real-time data sharing. The four-state MESI protocol supports efficient large-scale real-time data sharing between multiple caching bus masters. The core complex can be configured to default to three- or four-state operation. Additionally, the three- or four-state operation may be selected on a page-by-page basis for system optimization.

5.9.1 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. It is the programmer's responsibility to preserve reservations across context switches and to protect reservations in multiprocessor implementations.

5.9.2 Memory Access Ordering

To optimize performance the core complex supports weakly ordered references to memory. Thus the e500 manages the order and synchronization of instructions to insure 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; **msync** and **mbar** are also broadcast on the CCB to provide the appropriate control in the case of multiprocessor or shared memory systems.

5.9.3 Cache Control Instructions

The core complex supports Book E 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**)

5.9.4 Programmable Page Characteristics

Cache and memory attributes are programmable on a per-page basis. In addition to the write-through, caching-inhibited, memory coherency enforce, and guarded characteristic

defined by the WIMG bits, Book E defines an endianness bit that allows selection of big- or little-endian byte ordering on a per-page basis.

5.10 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.

5.11 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.

5.11.1 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.

5.11.2 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**.

The exception generated on overflow can be masked by clearing MSR[EE].

5.11.3 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 UPMCLa0–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 8 multiplier values in the range of 1 to 128. Any counter may be configured to increment only when an event occurs more than [threshold x multiplier] times. The contents of these registers are reflected to UPMCLb0–UPMLCb3, which can be read from user mode with **mfpmr**.

5.12 Legacy Support of PowerPC Architecture

This section provides an overview of the architectural differences and compatibilities of the e500 core compared with the AIM PowerPC architecture. 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.

In general, the e500 core supports the user-level architecture from the existing AIM architecture. The following subsections are intended to highlight the main differences. For specific implementation details refer to the relevant chapter.

5.12.1 Instruction Set Compatibility

The following sections generally describe the user and supervisor instruction sets.

5.12.1.1 User Instruction Set

The e500 core executes legacy user-mode binaries and object files with the following exceptions.

- The e500 supports only single-precision floating-point operations, with different encoding than the AIM definition of the PowerPC 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; however, there are new instructions specific to the e500 SPE APU.
- String instructions are not implemented on the e500; therefore, trap emulation must be provided to ensure backward compatibility.

5.12.1.2 Supervisor Instruction Set

The supervisor mode instruction set defined by the AIM version of 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.

5.12.2 Memory Subsystem

Both Book E and the AIM version of the PowerPC architecture provide separate instruction and data memory resources. The e500 provides additional cache control features, including cache locking.

5.12.3 Exception Handling

Exception handling is generally the same as that defined in the AIM version of the PowerPC architecture for the e500, with the following differences:

- Book E defines a new critical interrupt, providing an extra level of interrupt nesting. The critical interrupt includes external critical and watchdog timer time-out inputs.
- The e500 implements the machine check exception differently from the Book E and from the AIM definition. It defines the Return from Machine Check Interrupt instruction, **rfmci**, and two machine check save/restore registers, MCSRR0 and MCSRR1.
- Book E processors can use IVPR and IVORs to set exception vectors individually, but they can be set to the address offsets defined in the OEA to provide compatibility.

- Unlike the AIM version of the PowerPC architecture, Book E does not define a reset vector; execution begins at a fixed virtual address, 0xFFFF_FFFC.
- Some Book E and e500-specific SPRs are different from those defined in the AIM version of the PowerPC architecture, particularly those related to the MMU functions. Much of this information has been moved to a new exception syndrome register (ESR).
- Timer services are generally compatible, although Book E defines a new decremter auto reload feature and the fixed-interval timer critical 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 5.7, “Interrupts and Exception Handling.”

5.12.4 Memory Management

The e500 core implements a straightforward virtual address space that complies with the Book E MMU definition, eliminating segment registers and block address translation resources. Book E 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.

5.12.5 Reset

Book E-compliant cores do not share a common reset vector with the AIM version of the PowerPC architecture. Due to the additional features and capabilities of the e500 core, the initialization process is different. Unlike the AIM version of the PowerPC core, once execution begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

5.12.6 Other Differences

Unlike the AIM version of the PowerPC architecture, where little-endian mode is controlled on a system basis, Book E allows control of endian mode on a memory page basis. Additionally, the e500 core effects true little-endian mode by byte swapping.

Chapter 6

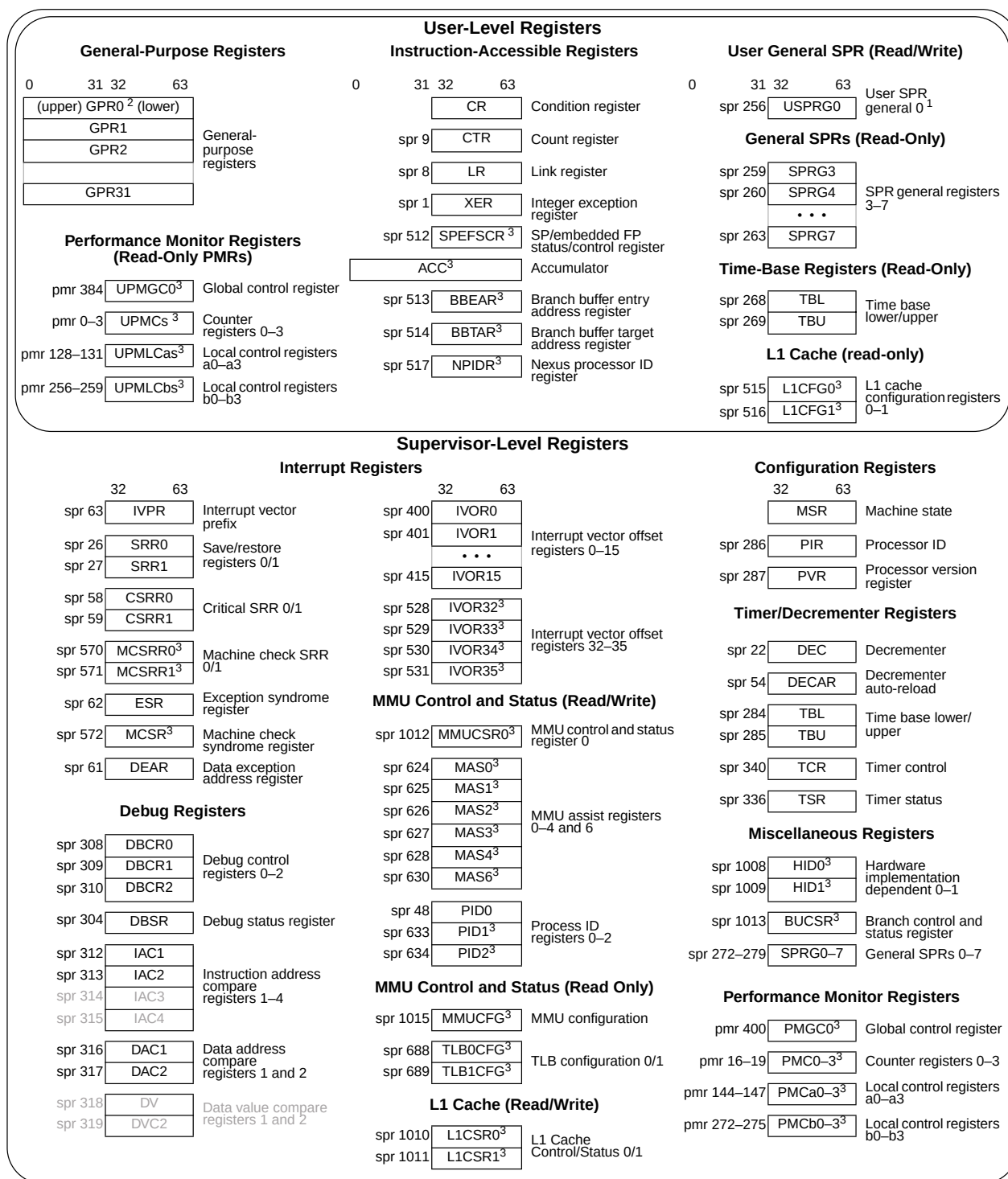
e500 Register Summary

The e500 core complex is a 32-bit implementation of the Book E PowerPC architecture, which includes a recognition that different processor implementations may require extensions or deviations from the architectural descriptions. This chapter provides additional information about the Book E architecture as it relates specifically to the e500 core complex registers. Specific details on how these e500 registers are accessed are provided in the *e500 User's Manual*.

Note that this chapter shows some Book E features that are not implemented in the e500. These resources are shown in gray.

6.1 e500 Register Model

Figure 6-1 shows the e500 core programming model.



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 e500-specific and are not part of the Book E architecture.

Figure 6-1. e500 Core Programming Model

6.2 Register Model for 32-Bit Book E Implementations

- Book E-defined registers that are accessed as part of instruction execution. These include the following:
 - The following registers are used for integer operations:
 - General-purpose registers (GPRs)—Book E defines a set of 32 GPRs used to hold source and destination operands for load, store, arithmetic, and computational instructions, and to read and write to other registers.
 - 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 Integer 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 APU). 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.
- Book E-defined special-purpose registers (SPRs) that are accessed explicitly using the **mtspr** and **mfspir** instructions. These registers are listed in Table 6-1 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- e500-defined SPRs that are accessed explicitly using the **mtspir** and **mfspir** instructions. These registers are listed in Table 6-2 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Section 6.4, “Registers for Branch Operations.” This section includes descriptions of 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)”
- Section 6.9, “Branch Target Buffer (BTB) Registers”
- Section 6.10, “Hardware Implementation-Dependent Registers”
- Section 6.11, “L1 Cache Configuration Registers”
- Section 6.12, “MMU Registers”

- Section 6.13, “Debug Registers”
- Section 6.14, “Signal Processing Engine (SPE) APU Registers”
- Section 6.16, “Instruction Address Resume Register (IARR)”

6.2.1 Special-Purpose Registers (SPRs)

Table 6-1. Book E Special-Purpose Registers (by SPR Abbreviation)

SPR Abbreviation	Name	Defined SPRN		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
CSRR0	Critical save/restore register 0	58	00001 11010	Read/Write	Yes	6.7.1.1/6-16
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-16
CTR	Count register	9	00000 01001	Read/Write	No	6.4.3/6-10
DAC1	Data address compare 1	316	01001 11100	Read/Write	Yes	6.13.4/6-46
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0	308	01001 10100	Read/Write	Yes	6.13.1/6-39
DBCR1	Debug control register 1	309	01001 10101	Read/Write	Yes	
DBCR2	Debug control register 2	310	01001 10110	Read/Write	Yes	
DBSR	Debug status register	304	01001 10000	Read/Clear ¹	Yes	6.13.2/6-45
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.6.5/6-16
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	6.13.3/6-46
IAC2	Instruction address compare 2	313	01001 11001			

Table 6-1. Book E Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPRN		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
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			
IVOR7	Floating-point unavailable interrupt offset (not supported on the e500)	407	01100 10111			
IVOR8	System call interrupt offset	408	01100 11000			
IVOR9	Auxiliary processor unavailable interrupt offset (not supported on the e500)	409	01100 11001			
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	6.7.1.6/6-17
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-10
PID	Process ID register ²	48	00001 10000	Read/Write	Yes	6.12.1/6-31
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. Book E Special-Purpose Registers (by SPR Abbreviation) (continued)

SPR Abbreviation	Name	Defined SPRN		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 ³	
		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	
		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	6.7.1.1/6-16
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-16
TBL	Time base lower	268	01000 01100	Read-only	No	6.6.3/6-15
		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	6.6.1/6-13
TSR	Timer status register	336	01010 10000	Read/Clear ⁴	Yes	6.6.2/6-14
USPRG0	User SPR general 0 ⁵	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 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**.

² Implementations may support more than one PID. The e500 implements the Book E–defined PID as PID0.

³ User-mode read access to SPRG3 is implementation-dependent.

⁴ 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**.

⁵ USPRG0 is a separate physical register from SPRG0.

Table 6-2. Implementation-Specific SPRs (by SPR Abbreviation)

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/Page
BBEAR	Branch buffer entry address register	513	Read/Write	No	6.9.1/6-23
BBTAR	Branch buffer target address register	514	Read/Write	No	6.9.2/6-23
BUCSR	Branch unit control and status register	1013	Read/Write	Yes	6.9.3/6-24
HID0	Hardware implementation dependent reg 0	1008	Read/Write	Yes	6.10.1/6-24
HID1	Hardware implementation dependent reg 1	1009	Read/Write	Yes	6.10.1/6-24
IVOR32	SPE APU unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-18
IVOR33	SPE floating-point data exception interrupt offset	529	Read/Write	Yes	
IVOR34	SPE 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	6.11.3/6-29
L1CFG1	L1 cache configuration register 1	516	Read-only	No	6.11.4/6-30
L1CSR0	L1 cache control and status register 0	1010	Read/Write	Yes	6.11.1/6-27
L1CSR1	L1 cache control and status register 1	1011	Read/Write	Yes	6.11.2/6-28
MAS0	MMU assist register 0	624	Read/Write	Yes	6.12.5.1/6-34
MAS1	MMU assist register 1	625	Read/Write	Yes	6.12.5.2/6-35
MAS2	MMU assist register 2	626	Read/Write	Yes	6.12.5.3/6-36
MAS3	MMU assist register 3	627	Read/Write	Yes	6.12.5.4/6-37
MAS4	MMU assist register 4	628	Read/Write	Yes	6.12.5.5/6-38
MAS6	MMU assist register 6	630	Read/Write	Yes	6.12.5.6/6-39
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.3/6-21
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	1012	Read/Write	Yes	6.12.2/6-31
PID0	Process ID register 0. Book E defines only this PID register and refers to as PID rather than PID0.	48	Read/Write	Yes	6.12.1/6-31
PID1	Process ID register 1	633	Read/Write	Yes	
PID2	Process ID register 2	634	Read/Write	Yes	
SPEFSCR	Signal processing and embedded floating-point status and control register	512	Read/Write	No	6.14.1/6-47
TLB0CFG	TLB configuration register 0	688	Read-only	Yes	6.12.4/6-33
TLB1CFG	TLB configuration register 1	689	Read-only	Yes	6.12.4.2/6-33

6.3 Registers for Integer Operations

6.3.1 General Purpose Registers (GPRs)

Book E implementations provide 32 GPRs (GPR0–GPR31) for 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 and can be used to contain address and integer data, although all instructions except SPE APU vector instructions use and return 32-bit values in GPR bits 32–63.

6.3.2 Integer Exception Register (XER)

	32	33	34	35																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												

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 (mullw , divw , divwu) 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

Branch operations in the device use the following registers:

6.4.1 Condition Register (CR)

	32	35 36	39 40	43 44	47 48	51 52	55 56	59 60	63
Field	CR0	CR1	CR2	CR3	CR4	CR5	CR6	CR7	
Reset	All zeros								
R/W	R/W								

Figure 6-3. Condition Register (CR)

Table 6-4. BI Operand Settings for CR Fields

CRn Bits	CR Bits	BI	Description
CR0[0]	32	00000	Negative (LT)—Set when the result is negative. For SPE APU vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CR0[1]	33	00001	Positive (GT)—Set when the result is positive (and not zero). For SPE APU vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.
CR0[2]	34	00010	Zero (EQ)—Set when the result is zero. For SPE APU vector compare and vector test instructions: Set to the OR of the result of the compare of the high and low elements.
CR0[3]	35	00011	Summary overflow (SO). Copy of XER[SO] at the instruction's completion. For SPE APU 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	Copy of FPSCR[FX] at the instruction's completion. Negative (LT) For SPE APU vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CR1[1]	37	00101	Copy of FPSCR[FEX] at the instruction's completion. Positive (GT) For SPE APU vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.
CR1[2]	38	00110	Copy of FPSCR[VX] at the instruction's completion. Zero (EQ) For SPE APU 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	Copy of FPSCR[OX] at the instruction's completion. Summary overflow (SO) For SPE APU vector compare and vector test instructions: Set to the AND of the result of the compare of the high and low elements. s.
CRn[0]	40 44 48 52 56 60	01000 01100 10000 10100 11000 11100	Less than or floating-point less than (LT, FL). For integer compare instructions: rA < SIMM or rB (signed comparison) or rA < UIMM or rB (unsigned comparison). For floating-point compare instructions: frA < frB. For SPE APU vector compare and vector test instructions: Set if the high-order element of rA is equal to the high-order element of rB; cleared otherwise.
CRn[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than or floating-point greater than (GT, FG). For integer compare instructions: rA > SIMM or rB (signed comparison) or rA > UIMM or rB (unsigned comparison). For floating-point compare instructions: frA > frB. For SPE APU vector compare and vector test instructions: Set if the low-order element of rA is equal to the low-order element of rB; cleared otherwise.

Table 6-4. BI Operand Settings for CR Fields (continued)

CRn Bits	CR Bits	BI	Description
CRn[2]	42	01010	Equal or floating-point equal (EQ, FE).
	46	01110	For integer compare instructions: rA = SIMM, UIMM, or rB.
	50	10010	For floating-point compare instructions: frA = frB.
	54	10110	For SPE APU vector compare and vector test instructions:
	58	11010	Set to the OR of the result of the compare of the high and low elements.
	62	11110	
CRn[3]	43	01011	Summary overflow or floating-point unordered (SO, FU).
	47	01111	For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction.
	51	10011	
	55	10111	For floating-point compare instructions, one or both of frA and frB is a NaN.
	59	11011	For SPE APU vector compare and vector test instructions:
	63	11111	Set to the AND of the result of the compare of the high and low elements.

6.4.2 Link Register (LR)

	32	63
Field	Link address	
Reset	All zeros	
R/W	R/W	
SPR	SPR 8	

Figure 6-4. Link Register (LR)

6.4.3 Count Register (CTR)

	32	63
Field	Count value	
Reset	All zeros	
R/W	R/W	
SPR	SPR 9	

Figure 6-5. Count Register (CTR)

6.5 Processor Control Registers

6.5.1 Machine State Register (MSR)

	32	36	37	38	39	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	—					WE	CE	—	EE	PR	FP	ME	FE0	UBLE	DE	FE1	—	—	IS	DS	—	PMM	—	—	
(e500)	—	UCLE	SPE	—	—	WE	CE	—	EE	PR	—	ME	—	UBLE	DE	—			IS	DS	—	PMM	—	—	
Reset	All zeros																								
R/W	R/W																								

Figure 6-6. Machine State Register (MSR)

Table 6-5. 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 0 If software attempts to execute an SPE APU instruction, the SPE APU unavailable exception is taken. 1 Software can execute any of the SPE APU instructions.
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 signalled 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	—	Preserved for PowerPC ILE
48	EE	External enable 0 External Input, decremter, and fixed-interval timer interrupts are disabled. 1 External input, decremter, and fixed-interval timer 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.

Table 6-5. MSR Field Descriptions (continued)

Bits	Name	Description
50	FP	Floating-point available. Book E defines the operation of FP as follows: 0 The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. Reserved and permanently cleared on the e500, indicating that it does not implement a Book E floating-point unit (FPU.) Setting it has no effect. 1 The processor can execute floating-point instructions.
51	ME	Machine check enable. 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
52	FE0	Floating-point exception mode 0. On the e500, reserved and permanently cleared, indicating the e500 does not implement a Book E floating-point unit. Setting it has no effect.
53	UBLE	Allocated for implementation-dependent use. In the e500, it is the user BTB lock enable bit. 0 Execution of the BTB lock instructions for user mode is disabled; privileged instruction exception taken instead. 1 Execution of the BTB lock instructions for user mode is enabled.
54	DE	Debug interrupt enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled if DBCR0[IDM] = 1. See the description of the DBSR[UDE] in Section 6.13.2, "Debug Status Register (DBSR)."
55	FE1	Floating-point exception mode 1. On the e500, reserved and permanently cleared, indicating the e500 does not implement a Book E floating-point unit. Setting it has no effect.
56	—	Reserved, should be cleared. ¹
57	—	Preserved for PowerPC IP
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 the 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 PowerPC RI and LE, respectively.

¹ An MSR bit that is reserved may be altered by return from interrupt instructions.

6.5.2 Processor ID Register (PIR)

	32	47	48	63
Field	Version			Revision
Reset	All zeros			0000_0000_0000_(core_mid[0:3])
R/W	Read only			
SPR	SPR 286			

Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

	32	47	48	63
Field	Version			Revision
Reset	0x8020_0010			
R/W	Read only			
SPR	SPR 287			

Figure 6-8. Processor Version Register (PVR)

Table 6-6. 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.
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.

6.6 Timer Registers

6.6.1 Timer Control Register (TCR)

	32	33	34	35	36	37	38	39	40	41	42	43	46	47	50	51	63
Field	WP	WRC	WIE	DIE	FP	FIE	ARE	—	WPEXT	FPEXT	—						
Reset	All zeros																
R/W	R/W																
SPR	SPR 340																

Figure 6-9. Timer Control Register (TCR)

Table 6-7. 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[0] WPEXT[0–3],WP[0–1] == 0b11_1111 selects TBL[63]
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 will occur. 01 Force processor checkstop on second timeout of watchdog timer 10 Assert processor reset output (p_resetout_b) on second timeout of watchdog timer 11 Reserved
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, 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[0] FPEXT[0–3],FP[0–1] == 0b11_1111 selects TBL[63]
40	FIE	Fixed interval interrupt enable 0 Fixed interval interrupts disabled 1 Fixed interval interrupts enabled
41	ARE	Auto-reload enable. Controls whether the value in DECAR is reloaded into the DEC when the DEC value reaches 0000_0001. 0 Auto-reload disabled 1 Auto-reload enabled
42	—	Reserved, should be cleared.
43–46	WPEXT	Watchdog timer period extension (see above description for WP).
47–50	FPEXT	Fixed-interval timer period extension (see above description for FP)
50–63	—	Reserved, should be cleared.

6.6.2 Timer Status Register (TSR)

	32	33	34	35	36	37	38	63
Field	ENW	WIS	WRS	DIS	FIS	—		
Reset	All zeros							
R/W	Set by hardware. Read by using mfsprr and cleared by using mtsprr by writing ones to any TSR bit positions to be cleared and zeros in all other bit positions. The write-data to the TSR is not direct data, but a mask. A 1 causes the bit to be cleared, and a 0 has no effect.							
SPR	SPR 336							

Figure 6-10. Timer Status Register (TSR)

Table 6-8. TSR Field Descriptions

Bits	Name	Description
32	ENW	Enable next watchdog time. 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] by executing an mtspr , setting WIS and any other bits to be cleared and a 0 in allother bits. The write-data to the TSR is not direct data, but is a mask. A 1 causes the bit to be cleared, and a 0 has no effect.
33	WIS	Watchdog timer interrupt status. 0 A watchdog timer event has not occurred. 1 A watchdog timer event has 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. Defined at reset (value = 00). Set to TCR[WRC] when a reset is caused by the watchdog timer.
36	DIS	Decrementer interrupt status. 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. 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

	32	63 32	63
Field	TBU	TBL	
Reset	All zeros	All zeros	
R/W	User read/Supervisor write	User read/Supervisor write	
SPR	269 Read/285 Write	268 Read/284 Write	

Figure 6-11. Timer Base Upper/Lower Registers (TBU/TBL)

6.6.4 Decrementer Register

Field	32	63	Decrementer value
Reset			All zeros
R/W			R/W
SPR			SPR 22

Figure 6-12. Decrementer Register (DEC)

6.6.5 Decrementer Auto-Reload Register (DECAR)

Field	32	63	Decrementer auto-reload value
Reset			All zeros
R/W			Write only
SPR			SPR 54

Figure 6-13. Decrementer Auto-Reload Register (DECAR)

6.7 Interrupt Registers

6.7.1 Interrupt Registers Defined by Book E

6.7.1.1 Save/Restore Register 0 (SRR0)

Field	32	63	Next instruction address
Reset			All zeros
R/W			R/W
SPR			SPR 26

Figure 6-14. Save/Restore Register 0 (SRR0)

6.7.1.2 Save/Restore Register 1 (SRR1)

Field	32	63	MSR state information
Reset			All zeros
R/W			R/W
SPR			SPR 27

Figure 6-15. Save/Restore Register 1 (SRR1)

6.7.1.3 Critical Save/Restore Register 0 (CSRR0)

	32	63
Field	Next instruction address	
Reset	All zeros	
R/W	R/W	
SPR	SPR 58	

Figure 6-16. Critical Save/Restore Register 0 (CSRR0)

6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

	32	63
Field	MSR state information	
Reset	All zeros	
R/W	R/W	
SPR	SPR 59	

Figure 6-17. Critical Save/Restore Register 1 (CSRR1)

6.7.1.5 Data Exception Address Register (DEAR)

	32	63
Field	Exception address	
Reset	All zeros	
R/W	R/W	
SPR	SPR 61	

Figure 6-18. Data Exception Address Register (DEAR)

6.7.1.6 Interrupt Vector Prefix Register (IVPR)

	32	47	48	63
Field	Interrupt vector offset		—	
Reset	All zeros			
R/W	R/W			
SPR	SPR 63			

Figure 6-19. Interrupt Vector Prefix Register (IVPR)

6.7.1.7 Interrupt Vector Offset Registers (IVORs)

	32	47	48	59	60	63
Field	Interrupt vector offset					
(e500)	—		Interrupt vector offset		—	
Reset	All zeros					
R/W	R/W					
SPR	See Table 6-9					

Figure 6-20. Interrupt Vector Offset Registers (IVOR)

Table 6-9. 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
IVOR5	405	Alignment
IVOR6	406	Program
IVOR7	407	Floating-point unavailable (not supported on the e500)
IVOR8	408	System call
IVOR9	409	Auxiliary processor unavailable (not supported on the e500)
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	(e500-specific) SPE APU unavailable
IVOR33	529	(e500-specific) SPE floating-point data exception
IVOR34	530	(e500-specific) SPE floating-point round exception
IVOR35	531	(e500-specific) Performance monitor
IVOR36–IVOR63	—	Allocated for implementation-dependent use

6.7.1.8 Exception Syndrome Register (ESR)

	32	35	36	37	38	39	40	41	42	43	44	45	46	47	48	55	56	57	63
Field	—	PIL	PPR	PTR	FP	ST	—	DLK0	DLK1	AP	PUO	BO	PIE	—					
(e500)	—	PIL	PPR	PTR	—	ST	—	DLK	ILK	—	—	BO	—	—	—	SPE	—	—	—
Reset	All zeros																		
R/W	R/W																		
SPR	SPR 62																		

Figure 6-21. Exception Syndrome Register (ESR)

Table 6-10. Exception Syndrome Register (ESR) Definition

Bits	Name	Syndrome	Interrupt Types
32–35	—	Allocated	—
36	PIL	Illegal instruction exception	Program
37	PPR	Privileged instruction exception	Program
38	PTR	Trap exception	Program
39	FP	Floating-point operations (Not supported on the e500). Reserved and permanently cleared because the e500 does not implement a Book E FPU. Setting it has no effect.	Alignment, data storage, data TLB, program
40	ST	Store operation	Alignment, data storage, data TLB error
41	—	Reserved, should be cleared.	—
42	DLK0	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	DLK1	Cache locking. Settings are implementation-dependent.	Data storage
(e500)	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	AP	Auxiliary processor operation (not supported on the e500)	Alignment, data storage, data TLB, program
45	PUO	Unimplemented operation exception (not supported on the e500. Unimplemented instructions are handled as illegal instructions.)	Program
46	BO	Byte-ordering exception	Data storage, instruction storage
47	PIE	Imprecise exception (not supported on the e500)	Program
48–55	—	Reserved, should be cleared.	—
56	—	Reserved, should be cleared.	—

Table 6-10. Exception Syndrome Register (ESR) Definition (continued)

Bits	Name	Syndrome	Interrupt Types
(e500)	SPE	SPE exception bit (e500-specific). 0 Default 1 Any exception caused by an SPE instruction occurred.	
57–63	—	Allocated for implementation-dependent use	—

6.7.2 e500-Specific Interrupt Registers

6.7.2.1 Machine Check Save/Restore Register 0 (MCSRR0)

	32	63
Field	Next instruction address	
Reset	All zeros	
R/W	R/W	
SPR	SPR 570	

Figure 6-22. Machine Check Save/Restore Register 0 (MCSRR0)

6.7.2.2 Machine Check Save/Restore Register 1 (MCSRR1)

	32	63
Field	MSR state information	
Reset	All zeros	
R/W	R/W	
SPR	SPR 571	

Figure 6-23. Machine Check Save/Restore Register 1 (MCSRR1)

6.7.2.3 Machine Check Syndrome Register (MCSR)

	32	33	34	35	36	37	38	39
Field	MCP	ICPERR	DCP_PERR	DCPERR	—	—	—	—
Reset	All zeros							
R/W	R/W							
	40	41	42	43	44	45	46	47
Field	—	—	—	—	—	—	—	—
Reset	All zeros							
R/W	R/W							
	48	49	50	51	52	53	54	55
Field	ABIST_IC	ABIST_DC	ABIST_ITLB	ABIST_DTLB	ABIST_L2TLB	ABIST_TLB1	ABIST_BTBT	—
Reset	All zeros							
R/W	R/W							
	56	57	58	59	60	61	62	63
Field	BUS_IAERR	BUS_RAERR	BUS_WAERR	BUS_IBERR	BUS_RBERR	BUS_WBERR	BUS_IPERR	BUS_RPERR
Reset	All zeros							
R/W	R/W							
SPR	SPR 572							

Figure 6-24. Machine Check Syndrome Register (MCSR)

Table 6-11. Machine Check Syndrome Register (MCSR)

Bit	Name	Description	Recoverable
32	MCP	Machine check input pin	Maybe
33	ICPERR	Instruction cache parity error	Precise
34	DCP_PERR	Data cache push parity error	Maybe
35	DCPERR	Data cache parity error	Precise (cache-inhibited stwcx. instructions and guarded loads that initiate this error are deallocated before a machine check is taken. MCSRR0 holds the address of the next instruction, not the instruction that initiated the error.)
43–46	—	Reserved, should be cleared.	—
47	GL_CI	Set when a guarded load or cache-inhibited stwcx. provokes a parity error in the cache (ICPERR or DCPERR). In this case, the address saved in MCSRR0 is the address after the instruction that caused the error.	
48	ABIST_IC	Abist instruction cache error	NA

Table 6-11. Machine Check Syndrome Register (MCSR) (continued)

Bit	Name	Description	Recoverable
49	ABIST_DC	Abist data cache error	NA
50	ABIST_ITLB	Abist instruction TLB error	NA
51	ABIST_DTLB	Abist data TLB error	NA
52	ABIST_L2TLB	Abist L2 TLB error	NA
53	ABIST_TLB1	Abist TLB1 error	NA
54	ABIST_BTBT	Abist BTB error	NA
56	BUS_IAERR	Bus. Instruction address error	Maybe
57	BUS_RAERR	Bus. Read address error	Maybe
58	BUS_WAERR	Bus. Write address error	Maybe
59	BUS_IBERR	Bus. Instruction data bus error	Maybe
60	BUS_RBERR	Bus. Read data bus error	Maybe
61	BUS_WBERR	Bus. Write bus error	Maybe
62	BUS_IPERR	Bus—instruction parity error	Maybe
63	BUS_RPERR	Bus—read parity error	Maybe

6.8 Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

32	63
Field	Software-determined information
Reset	All zeros
SPR	SPRG0 272 Read/Write Supervisor
R/W	SPRG1 273 Read/Write Supervisor
	SPRG2 274 Read/Write Supervisor
	SPRG3 259 Read-only User (implementation-dependent)/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

Figure 6-25. Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

6.9 Branch Target Buffer (BTB) Registers

6.9.1 Branch Buffer Entry Address Register (BBEAR)

Field	32 Branch buffer entry address	61 62 63 IAB
Reset	All zeros	
R/W	R/W	
SPR	SPR 513	

Figure 6-26. Branch Buffer Entry Address Register (BBEAR)

Table 6-12. BBEAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer entry address	Branch buffer entry address bits 0–30
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 Chapter 2, “Instruction Model.”

6.9.2 Branch Buffer Target Address Register (BBTAR)

Field	32 Branch buffer target address	61 62 63 IAB2 BDIRPR
Reset	All zeros	
R/W	R/W	
SPR	SPR 514	

Figure 6-27. Branch Buffer Target Address Register (BBTAR)

Table 6-13. BBTAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer target address	Branch buffer target address bits 0–30
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 in the bblels instruction in Chapter 2, “Instruction Model.”
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)

	32		53	54	55	56	57	58		62	63
Field	—			BBFI	BBLO	BBUL	BBLFR	—		BPEN	
Reset	All zeros										
R/W	R/W										
SPR	SPR 1013										

Figure 6-28. Branch Unit Control and Status Register (BUCSR)

Table 6-14. 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	BBLFR	Branch buffer lock bits flash reset. Clearing and then setting BBLFR flash clears the lock bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBLFR 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 (HID0)

	32	33		39	40	41	42	43		48	49		50	51		62	63
Field	EMCP	—		DOZE	NAP	SLEEP	—			TBEN	SEL_TBCLK	—					NOPTI
Reset	All zeros																
R/W	R/W																
SPR	SPR 1008																

Figure 6-29. HID0 Register

Table 6-15. HID0 Field Descriptions

Bits	Name	Description
32	EMCP	Enable machine check pin, $\overline{MCP}$. Used to mask out further machine check exceptions caused by assertion of MCP. 0 $\overline{MCP}$ is disabled. 1 $\overline{MCP}$ is enabled. If MSE[ME] = 0, asserting $\overline{MCP}$ causes checkstop. If MSR[ME] = 1, asserting $\overline{MCP}$ 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	SLEEP	Configure for sleep power management mode. If MSR[WE] is set, this bit controls SLEEP mode. 0 Core not in sleep mode 1 Core in sleep mode
43–48	—	Reserved, should be cleared.
49	TBEN	Time base enable 0 Time base disabled 1 Time base enabled
50	SEL_TBCLK	Select time base clock 0 Time base is based on processor clock 1 Time base is based on TBCLK input
51–62	—	Reserved, should be cleared.
63	NOPTI	No-op the data and instruction cache touch instructions. 0 The dcbt , dcbtst , and icbt instructions are enabled. 1 The dcbt , dcbtst , and icbt instructions are treated as no-ops. Note that dcbtlc and dcbtls are not treated as no-ops.

6.10.2 Hardware Implementation-Dependent Register 1 (HID1)

	32	33	34	39	40	46	47	48	49	50	51	52	53	54	55	56	57	59	60	63
Field	PLL_MODE	PLL_CFG	—	NEXEN	R1DPE	R2DPE	ASTME	ABE	—	MPXTT	MSHARS	SSHAR	ATS	—	MID					
Reset	All zeros																			
R/W	R/W																			
SPR	SPR 1009																			

Figure 6-30. HID1 Register

Table 6-16. HID1 Field Descriptions

Bits	Name	Description
32–33	PLL_MODE	Reflected directly from the PLL_MODE input pins (read-only)
34–39	PLL_CFG	Reflected directly from the PLL_CFG input pins (read-only)
40–46	—	Reserved, should be cleared.
47	NEXEN	Nexus bus enable. Note that not all devices that implement the e500 core will support Nexus. 0 Nexus bus disabled 1 Nexus bus enabled
48	R1DPE	R1 data bus parity enable 0 R1 data bus parity checking disabled 1 R1 data bus parity checking enabled
49	R2DPE	R2 data bus parity enable 0 R2 data bus parity checking disabled 1 R2 data bus parity checking enabled
50	ASTME	Address bus streaming mode enable 0 Address bus streaming mode disabled 1 Address bus streaming mode enabled
51	ABE	Address broadcast enable 0 Address broadcasting disabled 1 Address broadcasting enabled
52	—	Reserved, should be cleared.
53	MPXTT	MPX re-map transfer type 0 TTx codes are not remapped 1 Certain TTx codes are remapped for MPX bus compatibility
54	MSHARS	MMU shared select 0 HID1[SSHAR] selects cache coherency model 1 MMU determines 3-or 4-state cache coherency model on a page-by-page basis.
55	SSHAR	Default cache coherency model 0 3-state cache coherency model used 1 4-state cache coherency model used
56	ATS	Atomic status (read-only) Indicates state of atomic status bit in bus unit.
57–59	—	Reserved, should be cleared.
60–63	MID	Reflected directly from the MID input pins (read-only)

6.11 L1 Cache Configuration Registers

6.11.1 L1 Cache Control and Status Register 0 (L1CSR0)

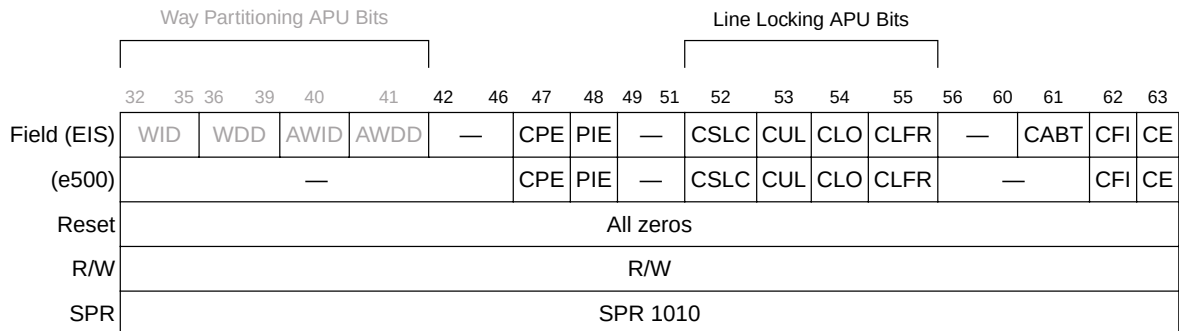


Figure 6-31. L1 Cache Control and Status Register 0 (L1CSR0)

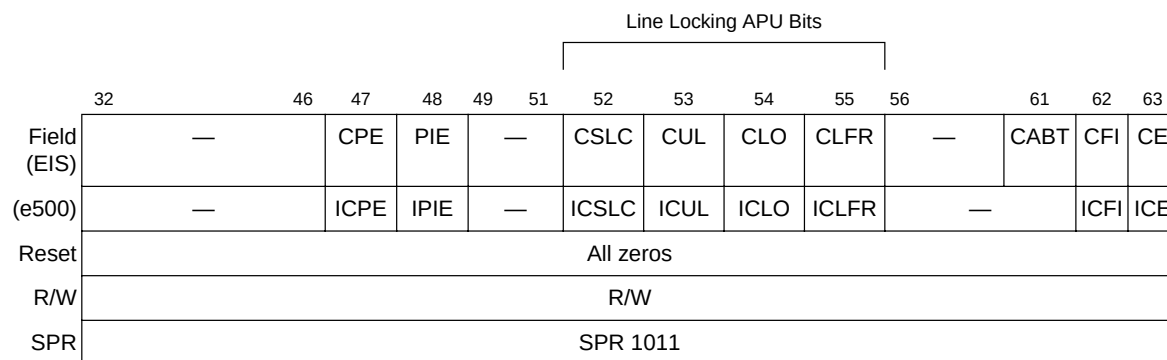
Table 6-17. L1CSR0 Field Descriptions

Bits	Name	Description
32–35	WID	Way instruction disable. (bit 32 = way 0, bit 33 = way 1, ... bit 35 = way 3)
36–39	WDD	Way data disable (bit 36 = way 0, bit 37 = way 1, ... bit 39 = way 3)
40	AWID	Additional ways instruction disable 0 Available for instruction miss use 1 Unavailable for instruction miss use
41	AWDD	Additional ways data disable 0 Available for data miss use 1 Unavailable for data miss use
42–46	—	Reserved, should be cleared.
47	CPE	Cache parity enable 0 Parity checking of the cache disabled 1 Parity checking of the cache enabled
48	PIE	Parity injection enabled 0 Parity injection into cache disabled 1 Parity injection into cache enabled
49–51	—	Reserved, should be cleared.
52	CSLC	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 an dcbi snoop that invalidated a locked line. 1 The cache has encountered an dcbi snoop that invalidated a locked line.
53	CUL	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
54	CLO	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

Table 6-17. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
55	CLFR	Cache lock bits flash reset. When written to a 1, hardware initiates a cache lock bits flash clear operation. Once the operation is complete, this bit is cleared. 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.
56–60	—	Reserved, should be cleared.
61	CABT	Cache operation aborted. 0 No cache operation completed improperly 1 Cache operation did not complete properly
62	CFI	Cache flash invalidate. Cache invalidate 0 No cache invalidate. Writing a 0 to CINV 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	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)

**Figure 6-32. L1 Cache Control and Status Register 1 (L1CSR1)****Table 6-18. 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	IPIE	Instruction parity injection enabled 0 Parity injection into instruction cache disabled 1 Parity injection into instruction cache enabled

Table 6-18. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
49–51	—	Reserved, should be cleared.
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. Written to 0 and then 1 to flash clear the lock bit of all entries in the instruction cache; clearing occurs independently from the value of the enable bit (ICE).
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).
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)

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	49	50	52	53	55	56	63
Field	CARCH	CWPA	CFAHA	CFISWA	—	CBSIZE	CREPL	CLA	CPA	CNWAY				CSIZE							
(e500)	CARCH	—					CBSIZE	CREPL	CLA	CPA	—		CNWAY	—		CSIZE					
Reset	00	00_000					00	01	1	1	000_00		01_1	000		0000_1000					
R/W	Read only																				
SPR	SPR 515																				

Figure 6-33. L1 Cache Configuration Register 0 (L1CFG0)

Table 6-19. L1CFG0 Field Descriptions

Bits	Name	Description
32–33	CARCH	Cache architecture 00 Harvard 01 Unified
34	CWPA	Cache way partitioning available 0 Unavailable 1 Available

Table 6-19. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
35	CFAHA	Cache flush all by hardware available 0 Unavailable 1 Available
36	CFISWA	Cache flush/inval by set and way available 0 Unavailable 1 Available
37–38	—	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
43	CLA	Cache locking APU available 0 Unavailable 1 Available
44	CPA	Cache parity available 0 Unavailable 1 Available
45–52	CNWAY	Cache number of ways. 111Indicates 8 ways.
53–63	CSIZE	Cache size. 0x20: indicates 32 Kbytes (8 -> size in 4-Kbyte blocks)

6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

	32	38	39	40	41	42	43	44	45	52	53	63
Field	—	CBSIZE	CREPL	CLA	CPA	CNWAY				CSIZE		
(e500)	—	CBSIZE	CREPL	CLA	CPA	CNWAY				CSIZE		
Reset	0000_000	0_0	01	1	1	000_0011_1				000_0010_0000		
R/W	Read only											
SPR	SPR 516											

Figure 6-34. L1 Cache Configuration Register 1 (L1CFG1)**Table 6-20. 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. 01Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available. 1 indicates available.
44	ICPA	Instruction cache parity available. 1 indicates available.

Table 6-20. L1CFG1 Field Descriptions (continued)

Bits	Name	Description
45–52	ICNWAY	Instruction cache number of ways. 011 Indicates eight ways.
53–63	ICSIZE	Instruction cache size. 0x8 indicates 32 Kbytes (8 -> size in 4-Kbyte blocks)

6.12 MMU Registers

6.12.1 Process ID Registers (PID0–PID2)

	32		55	56	63
Field (Book E)	Process ID				
(e500)	—			Process ID	
Reset	All zeros				
R/W	R/W				
SPR	SPR 48(PID0 ¹);SPR 633 (PID1 ²);SPR 634 (PID2 ²)				

¹ Book E defines PID0 and refers to it as PID.

² e500-specific registers

Figure 6-35. Process ID Register (PID0–PID2)

6.12.2 MMU Control and Status Register 0 (MMUCSR0)

	32		58	59	60	61	62	63
Field	—			IL1MMU_FI	DL1MMU_FI	L2TLB0_FI	L2TLB1_FI	—
Reset	All zeros							
R/W	R/W							
SPR	SPR 1012							

Figure 6-36. MMU Control and Status Register 0 (MMUCSR0)**Table 6-21. MMUCSR0 Field Descriptions**

Bits	Name	Description
32–58	—	Reserved, should be cleared.
59	IL1MMU_FI	Instruction L1 MMU (TLB) flash invalidate (write to 1 to invalidate)
60	DL1MMU_FI	Data L1 MMU (TLB) flash invalidate (write to 1 to invalidate)
61	L2TLB0_FI	TLB0 flash invalidate (write to 1 to invalidate)

Table 6-21. MMUCSR0 Field Descriptions (continued)

Bits	Name	Description
62	L2TLB1_FI	TLB1 flash invalidate (write to 1 to invalidate)0No 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 is causes an undefined operation.
63	—	Reserved, should be cleared.

6.12.3 MMU Configuration Register (MMUCFG)

	32	48	49	52	53	57	58	59	60	61	62	63
Field	—			NPIDS	PIDSIZE		—		NTLBS	MAVN		
Reset	0000_0000_0000_0000_0			001_1	001_11		00		01	00		
R/W	Read only											
SPR	SPR 1015											

Figure 6-37. MMU Configuration Register 1 (MMUCFG)**Table 6-22. 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 (TLBnCFG)

6.12.4.1 TLB0 Configuration Register (TLB0CFG)

	32	39	40	43	44	47	48	49	50	51	52	63
Field	ASSOC			MINSIZE		MAXSIZE		IPROT	AVAIL	—	NENTRY	
Reset	0000_0010			0001		0001		0	0	00	0001_0000_0000	
R/W	Read only											
SPR	SPR 688											

Figure 6-38. TLB Configuration Register 0 (TLB0CFG)

Table 6-23. 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 4K
44–47	MAXSIZE	Maximum page size of TLB0 0x1: indicates maximum page size is 4k
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))
50–51	—	Reserved, should be cleared.
52–63	NENTRY	Number of entries in TLB0 0x100: TLB0 contains 256 entries

6.12.4.2 TLB Configuration Register 1 (TLB1CFG)

	32	39	40	43	44	47	48	49	50	51	52	63
Field	ASSOC			MINSIZE		MAXSIZE		IPROT	AVAIL	—	NENTRY	
Reset	0001_0000			0001		1001		1	1	00	0000_0001_0000	
R/W	Read only											
SPR	SPR 689											

Figure 6-39. TLB Configuration Register 1 (TLB1CFG)

Table 6-24. 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 0x9: indicates maximum page size is 256 Mbyte
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 (MAS0–MAS4, MAS6)

6.12.5.1 MAS Register 0 (MAS0)

	32	33	34	35	36	43	44	47	48	51	52	62	63
Field (EIS)	—		TLBSEL			ESEL				—		NV	
(e500)	—		TLBSEL		—		ESEL			—			NV
Reset	All zeros												
R/W	R/W												
SPR	SPR 624												

Figure 6-40. MAS Register 0 (MAS0) Format**Table 6-25. MAS0—MMU Read/Write and Replacement Control**

Bit	Name	Comments or Function when Set
32–33	—	Reserved, should be cleared.
34–35	TLBSEL	Selects TLB for access. 00 TLB0 01 TLB1 10 TLB2; reserved (must not be used) on e500 11 TLB3; reserved (must not be used) on e500
36–47	ESEL	Entry select. Number of entry in selected array to be used for tlbwe . ESEL 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.(ESEL allows definition of an array that can have up to 4,096 selections.)

Table 6-25. MAS0—MMU Read/Write and Replacement Control (continued)

Bit	Name	Comments or Function when Set
44–47	ESEL	For the e500, ESEL serves as the way select for the corresponding TLB as follows: When TLBSEL = 00 (TLB0 selected), only bit 47 is used (and bits 44–46 should be cleared). This bit selects between way 0 and way 1 of TLB0. EA bits 45–51 are used to index into the TLB to further select the entry for the operation. When TLBSEL = 01 (TLB1 selected), all four bits are used to select one of 16 entries in the array.
48–51	—	Reserved, should be cleared.
52–63	NV	Next victim. Defined as a 12-bit field by the Motorola standard. Can be used to identify the next victim to be targeted for a TLB miss replacement operation for those TLBs that support the NV field. If the TLB selected by MAS0[TLBSEL] does not support the NV field, then this field is undefined. The specific meaning of this field is implementation-dependent.
63	NV	For the e500 implementation, this bit is the next victim bit value to be written to TLB0[NV] on execution of tlbwe . NV is also updated on TLB error exceptions (misses), tlbsx hit and miss cases, and on execution of tlbre . NV is updated with 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)

	32	33	34	35	36	39	40	47	48	50	51	52	55	56	63
Field (EIS)	V	IPROT	—				TID	—		TS		TSIZE			—
(e500)	V	IPROT		—			TID	—		TS		TSIZE			—
Reset	All zeros														
R/W	R/W														
SPR	SPR 625														

Figure 6-41. MAS Register 1 (MAS1) Format**Table 6-26. 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.
34–35	—	Reserved, should be cleared.
36–47	TID	Translation identity. 12-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the effective address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.

Table 6-26. MAS1 Field Descriptions—Descriptor Context and Configuration Control (continued)

Bits	Name	Descriptions
40–47	TID	For the e500, 8-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the three effective 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. This 4-bit field defines the page size of the TLB entry. For TLB arrays that contain fixed-size TLB entries, this field is ignored. For variable page size TLB arrays, the page size is 4^{TSIZE} Kbytes. Note that the Motorola Book E standard supports all 16 page sizes defined in Book E. For the e500, only 9 pages sizes are supported. They range from 4-Kbytes to 256-Mbytes, increasing by powers of 4.
56–63	—	Reserved, should be cleared.

6.12.5.3 MAS Register 2 (MAS2)

	32	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	EPN				—		—	X0	X1	W	I	M	G	E
(e500)	EPN				—	SHAREN	—	X0	X1	W	I	M	G	E
Reset	All zeros													
R/W	R/W													
SPR	SPR 626													

Figure 6-42. MAS Register 2 (MAS2) Format**Table 6-27. 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 zero.
52–54	—	Reserved for implementation-specific use.
54	SHAREN	For e500 implementation, enables cache fills to use shared cache state. 0 Cache fills do not use the shared cache state for this page. 1 Cache fills use the shared cache state for this page.
55–56	—	Reserved, should be cleared.
57	X0	Implementation-dependent page attribute
58	X1	Implementation-dependent page attribute
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.

Table 6-27. MAS2 Field Descriptions—EPN and Page Attributes (continued)

Bits	Name	Description
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. A read or write to a caching-inhibited page affects only the memory element specified by the operation.
61	M	Memory coherence required 0 Memory coherence is not required. 1 Memory coherence 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 coherence 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 are performed without speculation (that is, they are known to be required).
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. Section 11.1.1.5, "Endianness," describes the Book E definition of endianness. 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)

	32																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							

Figure 6-43. MAS Register 3 (MAS3) Format**Table 6-28. 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 zero.
52–53	—	Reserved, should be cleared.
54–57	U0–U3	User bits. These bits are associated with a TLB entry and can be used by system software. For example, these bits may be used to hold information useful to a page scanning algorithm or be used to 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)

	32	33	34	35	36	43	44	45	46	47	48	50	51	52	55	56	63					
Field	—	TLBSELD			—	TIDSELD			—			TSIZED		—	X0D	X1D	WD	ID	MD	GD	ED	
(e500)	—	TLBSELD			—			TIDSELD		—	SHAREND		TSIZED		—	X0D	X1D	WD	ID	MD	GD	ED
Reset	All zeros																					
R/W	R/W																					
SPR	SPR 628																					

Figure 6-44. MAS Register 4 (MAS4) Format

Table 6-29. MAS4 Field Descriptions—Hardware Replacement Assist Configuration

Bits	Name	Description
32–33	—	Reserved, should be cleared.
34–35	TLBSELD	TLBSEL default value. 2-bit field that specifies the default value to be loaded in MAS0[TLBSEL] on a TLB miss exception.
36–43	—	Reserved, should be cleared.
44–47	TIDSELD	TID default selection value. 4-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 bits 44–45 as reserved and bits 46–47 as follows: 00 = PID0 01 = PID1 10 = PID2 11 = TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
51	SHAREND	On the e500 implementation this fields is defined as the default SHAREN value.
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)

	32			62	63
Field	—	SPID0	—	SPID1	SAS
(e500)	—	SPID0	—		SAS
Reset	All zeros				SAS
R/W	R/W				
SPR	SPR 630				

Figure 6-45. MAS Register 6 (MAS6)

Table 6-30. MAS 6 Field Descriptions

Bits	Name	Description
32–35 62	—	Reserved, should be cleared.
36–39	—	For the e500 implementation, reserved, should be cleared.
36–47	SPID0	Search PID0. Specifies the value of PID0 used when searching the TLB during execution of tlbsx .
40–47	SPID0	For the e500 implementation, this field contains the 8-bit search PID0 value. Specifies the value of PID0 used when searching the TLB during execution of tlbsx .
48–50	—	Reserved, should be cleared.
51–62	SPID1	Search PID1. Specifies the value of PID1 used when searching the TLB during execution of tlbsx .
63	SAS	Address space (AS) value for searches. Specifies the value of AS used when searching the TLB during execution of tlbsx .

6.13 Debug Registers

6.13.1 Debug Control Registers

6.13.1.1 Debug Control Register 0 (DBCR0)

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49		62	63
Field	—	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC2	RET		—					FT
(e500)	—	IDM	RST	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—	—	DAC1	DAC2	RET		—					FT
Reset	All zeros																				
R/W	R/W																				
SPR	SPR 308																				

Figure 6-46. Debug Control Register 0 (DBCR0)

Table 6-31. DBCR0 Field Descriptions

Bits	Name	Description
32	—	Allocated for implementation-dependent use. The e500 does not implement this bit. See the user's manual.
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 via MSR[DE]. Otherwise, redundant debug interrupts are taken for the same debug event.
34–35	RST	Reset. Book E defines this field such that 00 is always no action and all other settings are implementation specific. The e500 implements these bits as follows: 0x Default (No action) 1x Causes a hard reset if MSR[DE] and DBCR0[IDM] are set. Always cleared on subsequent cycle.
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 Note: Critical interrupts do not cause an IRPT debug event if MSR[DE]=0.
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	IAC3	Not implemented on the E500. Defined by Book E as instruction address compare 3 debug event enable. 0 IAC3 debug events cannot occur 1 IAC3 debug events can occur
43	IAC4	Not implemented on the E500; defined by Book E as instruction address compare 4 debug event enable. 0 IAC4 debug events cannot occur 1 IAC4 debug events can occur
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

Table 6-31. DBCR0 Field Descriptions (continued)

Bits	Name	Description
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
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

	32	33	34	35	36	37	38	39	40	41	42	47	48	49	50	51	52	53	54	55	56	57	58	63
Field	IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	—	IAC3US	IAC3ER	IAC4US	IAC4ER	IAC34M	—												
(e500)	IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	—																		
Reset	All zeros																							
R/W	R/W																							
SPR	SPR 309																							

Figure 6-47. Debug Control Register 1 (DBCR1)**Table 6-32. 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 IAC1 debug events are based on real addresses. 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

Table 6-32. DBCR1 Field Descriptions (continued)

Bits	Name	Description
38–39	IAC2ER	Instruction address compare 2 effective/real mode 00 IAC2 debug events are based on effective addresses 01 IAC2 debug events are based on real addresses. 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
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 can 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 can 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.
42–47	—	Reserved, should be cleared.
48–49	IAC3US	Instruction address compare 3 user/supervisor mode 00 IAC3 debug events can occur 01 Reserved 10 IAC3 debug events can occur only if MSR[PR]=0 11 IAC3 debug events can occur only if MSR[PR]=1
50–51	IAC3ER	Instruction address compare 3 effective/real mode 00 IAC3 debug events are based on effective addresses 01 IAC3 debug events are based on real addresses 10 IAC3 debug events are based on effective addresses and can occur only if MSR[IS]=0 11 IAC3 debug events are based on effective addresses and can occur only if MSR[IS]=1
52–53	IAC4US	Instruction address compare 4 user/supervisor mode 00 IAC4 debug events can occur 01 Reserved 10 IAC4 debug events can occur only if MSR[PR]=0 11 IAC4 debug events can occur only if MSR[PR]=1
54–55	IAC4ER	Instruction address compare 4 effective/real mode 00 IAC4 debug events are based on effective addresses 01 IAC4 debug events are based on real addresses 10 IAC4 debug events are based on effective addresses and can occur only if MSR[IS]=0 11 IAC4 debug events are based on effective addresses and can occur only if MSR[IS]=1

Table 6-32. DBCR1 Field Descriptions (continued)

Bits	Name	Description
56–57	IAC34M	Instruction address compare 3/4 mode 00 Exact address compare. IAC3 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 and IAC4 debug events can occur only if the address of the data storage access, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. If IAC3USπIAC4US or IAC3ERπIAC4ER, results are boundedly undefined. 10 Inclusive address range compare. IAC3 and IAC4 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. If IAC3USπIAC4US or IAC3ERπIAC4ER, results are boundedly undefined. 11 Exclusive address range compare. IAC3 and IAC4 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. If IAC3USπIAC4US or IAC3ERπIAC4ER, results are boundedly undefined.
58–63	—	Reserved, should be cleared.

6.13.1.3 Debug Control Register 2 (DBCR2)

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	55	56	63	
Field	DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	—	DVC1M	DVC2M	DVC1BE	DVC2BE											
(e500)	DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	—															
Reset	All zeros																				
R/W	R/W																				
SPR	SPR 310																				

Figure 6-48. Debug Control Register 2 (DBCR2)**Table 6-33. 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 DAC1 debug events are based on real addresses. 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 6-33. 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 DAC2 debug events are based on real addresses. 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.
42–43	—	Reserved, should be cleared.
44–45	DVC1M	Data value compare 1 mode 00 DAC1 debug events can occur 01 DAC1 debug events can occur only when all bytes specified in DBCR2[DVC1BE] in the data value of the data storage access match their corresponding bytes in DVC1 10 DAC1 debug events can occur only when at least one of the bytes specified in DBCR2[DVC1BE] in the data value of the data storage access matches its corresponding byte in DVC1 11 DAC1 debug events can occur only when all bytes specified in DBCR2[DVC1BE] within at least one of the half words of the data value of the data storage access matches their corresponding bytes in DVC1
46–47	DVC2M	Data value compare 2 mode 00 DAC2 debug events can occur 01 DAC2 debug events can occur only when all bytes specified in DBCR2[DVC2BE] in the data value of the data storage access match their corresponding bytes in DVC2. 10 DAC2 debug events can occur only when at least one of the bytes specified in DBCR2[DVC2BE] in the data value of the data storage access matches its corresponding byte in DVC2. 11 DAC2 debug events can occur only when all bytes specified in DBCR2[DVC2BE] within at least one of the half words of the data value of the data storage access matches their corresponding bytes in DVC2.
48–55	DVC1BE	Data value compare 1 byte enables. Specifies which bytes in the aligned data value being read or written by the storage access are compared to the corresponding bytes in DVC1.
56–63	DVC2BE	Data value compare 2 byte enables. Specifies which bytes in the aligned data value being read or written by the storage access are compared to the corresponding bytes in DVC2.

6.13.2 Debug Status Register (DBSR)

	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	63
Field	IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1R	DAC1W	DAC2R	DAC2W	RET	—		
(e500)	IDE	UDE	MRR	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	—	—	DAC1R	DAC1W	DAC2R	DAC2W	RET	—		
Reset	00		Undefined	0000_0000_0000															
R/W	R/W																		
SPR	SPR 304																		

Figure 6-49. Debug Status Register (DBSR)

Table 6-34. 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.									
33	UDE	Unconditional debug event. Set if an unconditional debug event occurred. If the $\overline{UDE}$ signal (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE] DBCR0[IDM] <table> <tr> <td>X</td><td>0</td><td>No action.</td></tr> <tr> <td>0</td><td>1</td><td>DBSR[UDE] is set.</td></tr> <tr> <td>1</td><td>1</td><td>DBSR[UDE] is set and a debug interrupt is taken.</td></tr> </table>	X	0	No action.	0	1	DBSR[UDE] is set.	1	1	DBSR[UDE] is set and a debug interrupt is taken.
X	0	No action.									
0	1	DBSR[UDE] is set.									
1	1	DBSR[UDE] is set and a debug interrupt is taken.									
34–35	MRR	Most recent reset. Set of three values when a reset occurs. Undefined at power-up. 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. See the User's Manual for the implementation for further details									
36	ICMP	Instruction complete debug event. Set if an instruction completion debug event occurred and DBCR0[ICMP] = 1.									
37	BRT	Branch taken debug event. Set if a branch taken debug event occurred (DBCR0[BRT]=1).									
38	IRPT	Interrupt taken debug event. Set if an interrupt taken debug event occurred (DBCR0[IRPT]=1).									
39	TRAP	Trap instruction debug event. Set if a trap Instruction debug event occurred (DBCR0[TRAP]=1).									
40	IAC1	Instruction address compare 1 debug event. Set if an IAC1 debug event occurred (DBCR0[IAC1]=1).									
41	IAC2	Instruction address compare 2 debug event. Set if an IAC2 debug event occurred (DBCR0[IAC2]=1).									
42	IAC3	Instruction address compare 3 debug event. Set if an IAC3 debug event occurred (DBCR0[IAC3]=1).									
43	IAC4	Instruction address compare 4 debug event. Set if an IAC4 debug event occurred (DBCR0[IAC4]=1).									
44	DAC1R	Data address compare 1 read debug event. Set if a read-type DAC1 debug event occurred (DBCR0[DAC1]=10 or 11).									
45	DAC1W	Data address compare 1 write debug event. Set if a write-type DAC1 debug event occurred (DBCR0[DAC1]=01 or 11).									
46	DAC2R	Data address compare 2 read debug event. Set if a read-type DAC2 debug event occurred (DBCR0[DAC2]=10 or 11).									
47	DAC2W	Data address compare 2 write debug event. Set if a write-type DAC2 debug event occurred (DBCR0[DAC2] =01 or 11).									

Table 6-34. DBSR Field Descriptions (continued)

Bits	Name	Description
48	RET	Return debug event. Set if a return debug event occurred (DBCR0[RET]=1).
49–63	—	Reserved, should be cleared.

6.13.3 Instruction Address Compare Registers (IAC1–IAC2)

	32		61 62 63
Field	Instruction address		—
Reset	All zeros		
R/W	R/W		
SPR	SPR 312 (IAC1); SPR 313 (IAC2); SPR 314 (IAC3); SPR 315 (IAC4)		

Figure 6-50. Instruction Address Compare Registers (IAC1–IAC2)

6.13.4 Data Address Compare Registers (DAC1–DAC2)

	32		63
Field	Data address		
Reset	All zeros		
R/W	R/W		
SPR	SPR 316 (DAC1); SPR 317 (DAC2)		

Figure 6-51. Data Address Compare Registers (DAC1–DAC2)

6.14 Signal Processing Engine (SPE) APU Registers

6.14.1 Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

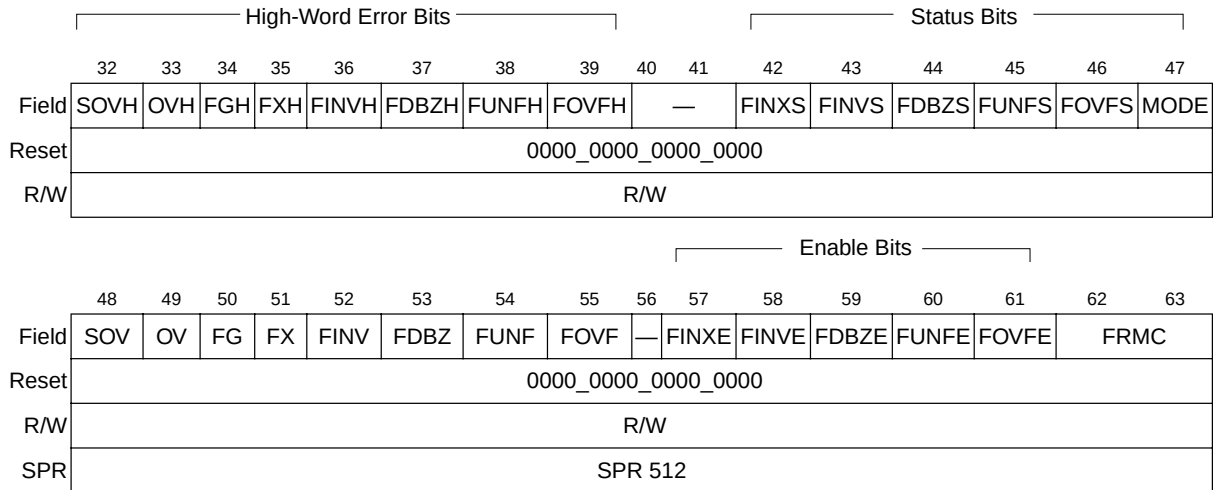


Figure 6-52. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Table 6-35. 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.
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. $FINXS = FINXS \mid FGH \mid FXH \mid FG \mid FX$

Table 6-35. SPEFSCR Field Descriptions

Bits	Name	Function
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
51	FX	Embedded floating-point sticky bit
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 if 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 enable
59	FDBZE	Embedded floating-point divide by zero enable
60	FUNFE	Embedded floating-point underflow enable
61	FOVFE	Embedded floating-point overflow enable
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

6.14.2 Accumulator (ACC)

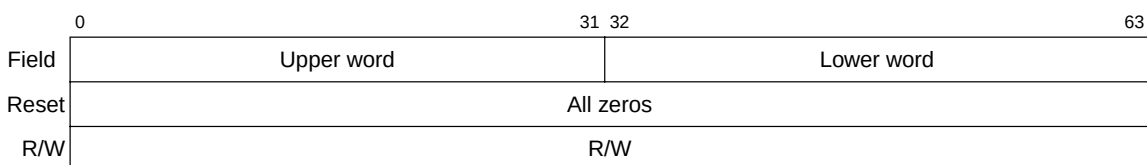
**Figure 6-53. Accumulator (ACC)**

Table 6-36. Accumulator (ACC)

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

Table 6-37. Performance Monitor Registers–Supervisor Level

PMR Number	pmr[0–4]	pmr[5–9]	Register Name	Symbol
16	00000	10000	Counter 0	PMC0
17	00000	10001	Counter 1	PMC1
18	00000	10010	Counter 2	PMC2
19	00000	10011	Counter 3	PMC3
144	00100	10000	Local control a0	PMLCa0
145	00100	10001	Local control a1	PMLCa1
146	00100	10010	Local control a2	PMLCa2
147	00100	10011	Local control a3	PMLCa3
272	01000	10000	Local control b0	PMLCb0
273	01000	10001	Local control b1	PMLCb1
274	01000	10010	Local control b2	PMLCb2
275	01000	10011	Local control b3	PMLCb3
400	01100	10000	Global control 0	PMGC0

Table 6-38. Performance Monitor Registers–User Level (Read-Only)

PMR Number	pmr[0–4]	pmr[5–9]	Register Name	Symbol
0	00000	00000	Counter 0	UPMC0
1	00000	00001	Counter 1	UPMC1
2	00000	00010	Counter 2	UPMC2
3	00000	00011	Counter 3	UPMC3
128	00100	00000	Local control a0	UPMLCa0
129	00100	00001	Local control a1	UPMLCa1
130	00100	00010	Local control a2	UPMLCa2
131	00100	00011	Local control a3	UPMLCa3
256	01000	00000	Local control b0	UPMLCb0

Table 6-38. Performance Monitor Registers–User Level (Read-Only) (continued)

PMR Number	pmr[0–4]	pmr[5–9]	Register Name	Symbol
257	01000	00001	Local control b1	UPMLCb1
258	01000	00010	Local control b2	UPMLCb2
259	01000	00011	Local control b3	UPMLCb3
384	01100	00000	Global control 0	UPMGC0

6.15.1 Global Control Register 0 (PMGC0)

	32	33	34	35	63
Field	FAC	PMIE	FCECE	—	
Reset	All zeros				
R/W	PMGC0: Supervisor R/W; UPMGC0: User read-only				
PMR	PMGC0: PMR400; UPMGC0: PMR384				

**Figure 6-54. Performance Monitor Global Control Register 0 (PMGC0)
User Performance Monitor Global Control Register 0 (UPMGC0)**

Table 6-39. 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 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.
35–63	—	Reserved, should be cleared.

6.15.2 User Global Control Register 0 (UPMGC0)

6.15.3 Local Control A Registers (PMLCa0–PMLCa3)

	32	33	34	35	36	37	38	40	41		47	48		63
Field	FC	FCS	FCU	FCM1	FCM0	CE	—			EVENT				
Reset	All zeros													
R/W	PMLCa0–PMLCa3: Supervisor R/W UPMLCa0–UPMLCa3: User read-only													
PMR	PMLCa0: PMR144, PMLCa1: PMR145, PMLCa2: PMR146, PMLCa3: PMR147 UPMLCa0: PMR128, UPMLCa1: PMR129, UPMLCa2: PMR130, UPMLCa3: PMR131													

**Figure 6-55. Local Control A Registers (PMLCa0–PMLCa3)
User Local Control A Registers (UPMLCa0–UPMLCa3)**

Table 6-40. 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.
37	CE	Condition enable 0 PMCx overflow conditions cannot occur (PMCx cannot cause interrupts, cannot freeze counters) 1 Overflow conditions occur when the most-significant-bit of PMCx is equal to 1. It is recommended that CE be cleared when counter PMCx is selected for chaining.
38–40	—	Reserved, should be cleared.
41–47	EVENT	Event selector. Up to 128 events selectable.
48–63	—	Reserved, should be cleared.

6.15.4 User Local Control A Registers (UPMLCa0–UPMLCa3)

6.15.5 Local Control B Registers (PMLCb0–PMLCb3)

	32		52	53		55	56	57	58		63
Field	—				THRESHMUL	—		THRESHOLD			
Reset	All zeros										
R/W	PMLCb0–PMLCb3: Supervisor R/W UPMLCb0–UPMLCb3: User read-only										
PMR	PMLCb0: PMR272, PMLCb1: PMR273, PMLCb2: PMR274, PMLCb3: PMR275 UPMLCb0: PMR256, UPMLCb1: PMR257, UPMLCb2: PMR258, UPMLCb3: PMR259										

**Figure 6-56. Local Control B Registers (PMLCb0–PMLCb3)
User Local Control B Registers (UPMLCb0–UPMLCb3)**

Table 6-41. 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 (PMLCbn[THRESHOLD] * 1) 001 Threshold field is multiplied by 2 (PMLCbn[THRESHOLD] * 2) 010 Threshold field is multiplied by 4 (PMLCbn[THRESHOLD] * 4) 011 Threshold field is multiplied by 8 (PMLCbn[THRESHOLD] * 8) 100 Threshold field is multiplied by 16 (PMLCbn[THRESHOLD] * 16) 101 Threshold field is multiplied by 32 (PMLCbn[THRESHOLD] * 32) 110 Threshold field is multiplied by 64 (PMLCbn[THRESHOLD] * 64) 111 Threshold field is multiplied by 128 (PMLCbn[THRESHOLD] * 128)
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the THRESHOLD value are counted. Events to which a threshold value applies are implementation-dependent as are the dimension (for example duration in cycles) and the granularity with which the threshold value is interpreted. By varying the threshold 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 value each time.

6.15.6 User Local Control B Registers (UPMLCb0–UPMLCb3)

6.15.7 Performance Monitor Counter Registers (PMC0–PMC3)

	32	33	63
Field	OV	Counter value	
Reset	All zeros		
R/W	PMC0–PMC3: Supervisor R/W UPMC0–UPMC3: User read-only		
PMR	PMC0: PMR16, PMC1: PMR17, PMC2,: PMR18, PMC3: PMR19 UPMC0: PMR0, UPMC1: PMR1, UPMC2,: PMR2, UPMC3: PMR3		

**Figure 6-57. Performance Monitor Counter Registers (PMC0–PMC3)
User Performance Monitor Counter Registers (UPMC0–UPMC3)**

Table 6-42. 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 MPC8540 features an integrated 256-Kbyte L2 cache organized as 1024, eight-way sets of 32-byte cache lines based on 32-bit physical addresses, as shown in Figure 7-1. The SRAM can also be configured as one or more blocks of externally accessible memory-mapped SRAM in addition to or instead of cache. Configuration and control of the L2/SRAM is accomplished with memory-mapped registers.

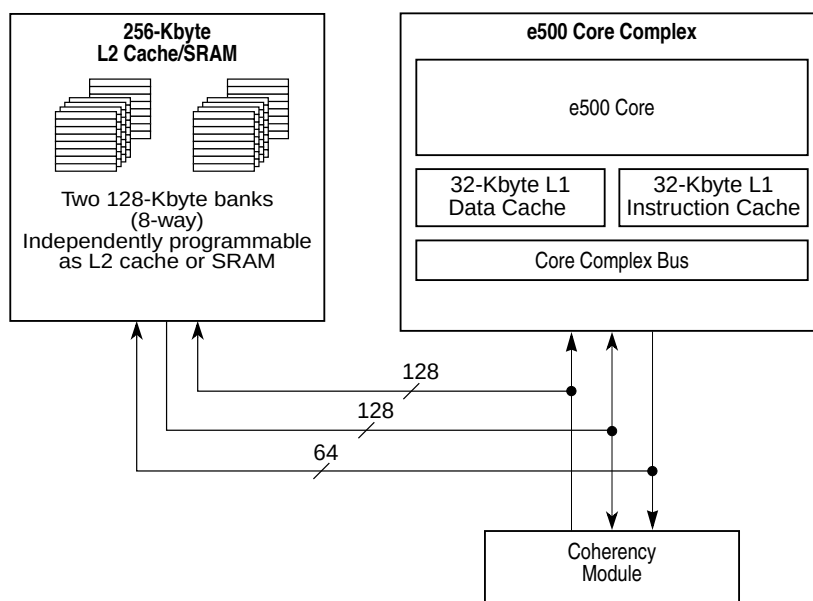


Figure 7-1. L2 Cache/SRAM Configuration

The L2 cache can operate in the following modes:

- Full cache mode (256-Kbyte cache).

Cache Organization

- Full memory-mapped SRAM mode (256-Kbyte SRAM memory-mapped)
- Half SRAM and half cache mode (128-Kbyte cache and 128-Kbyte memory-mapped SRAM)

These modes are described in Section 7.2, “Cache Organization.”

7.1.1 L2 Cache/SRAM Features

The L2 cache has the following characteristics:

- Two input data buses (64 and 128 bits wide) and one output data bus (128 bits wide)
- Eight-way set associativity
- 256-Kbyte array organized as 1024, eight-way sets of 32-byte cache lines
- Tag arrays contain 17 bits of tag and one bit of tag parity for each line to support 256 Kbytes (1024 sets) cache size, or 18 bits of tag to support 128 Kbytes (512 sets) cache size.
- Pseudo-LRU (7-bit replacement algorithm)
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag parity (one bit covering all tag bits)
- Lines can be locked in the cache (through core-initiated instructions or system writes) and guaranteed not to be evicted.
- Locks for the instruction and data caches can be set and cleared separately
- Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges
- Locks for the entire cache may be set and cleared by accesses to memory mapped control registers
- I/O devices can force writes to memory to be allocated into the L2 cache through programmed memory ranges
- 128 Kbytes or 256 Kbytes of array can be designated as externally-accessible memory-mapped SRAM instead of cache

7.2 Cache Organization

When the 256-Kbyte array is used as a cache, it has two banks each containing 512 sets of eight cache blocks (8 ways) as shown in Figure 7-2. Each block consists of 32 bytes of data and an address tag.

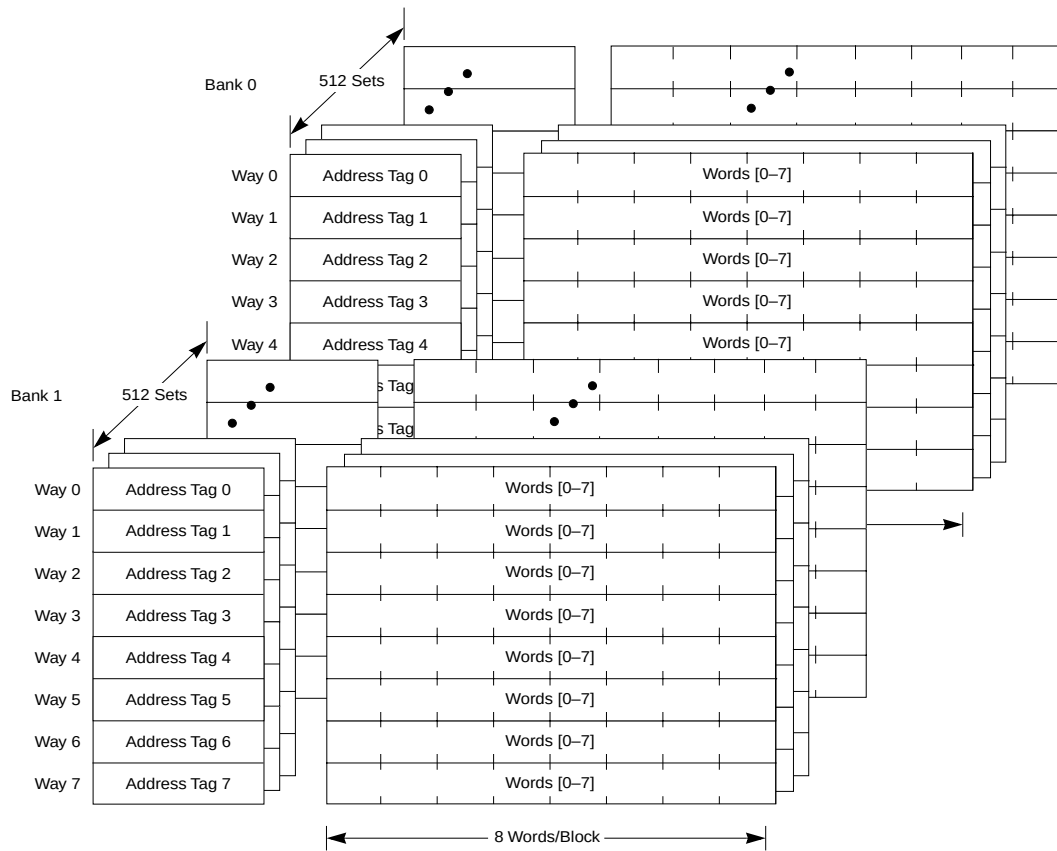


Figure 7-2. Cache Organization

Figure 7-3 shows how physical address bits are used to access the L2 in full cache mode (256-Kbyte cache). Physical address bits 17–26 identify the bank and set of the tag and data. Physical address bits 0–16 are compared against the tags of all 8 ways of the set. A match of a valid tag selects a 32-byte block of data within the set. Physical address bits 27–31 are used to identify the byte or bytes of data within the block. If the L2 is programmed as one block of 256-Kbyte SRAM, physical address bits 14–16 are used as the way select, without a tag lookup.

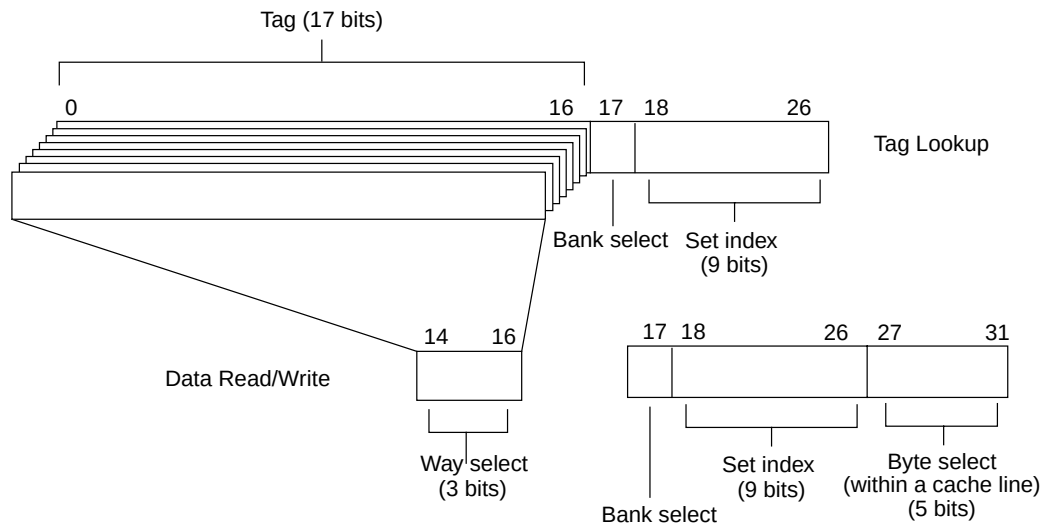


Figure 7-3. 256-Kbyte L2 Cache Address Configuration—Full Cache Mode

In half SRAM, half cache mode (128-Kbyte cache), there is no bank select, as shown in Figure 7-4, so the tag is 18 bits for cache accesses. The cache resides in bank 1, and the SRAM in bank 0. For SRAM accesses, physical address bits 15–17 are used as the way select. If the L2 is programmed as two blocks of 128-Kbyte SRAM, bank 0 is block 0 (defined by L2SRBAR0, see Table 7-5) and bank 1 is block 1.

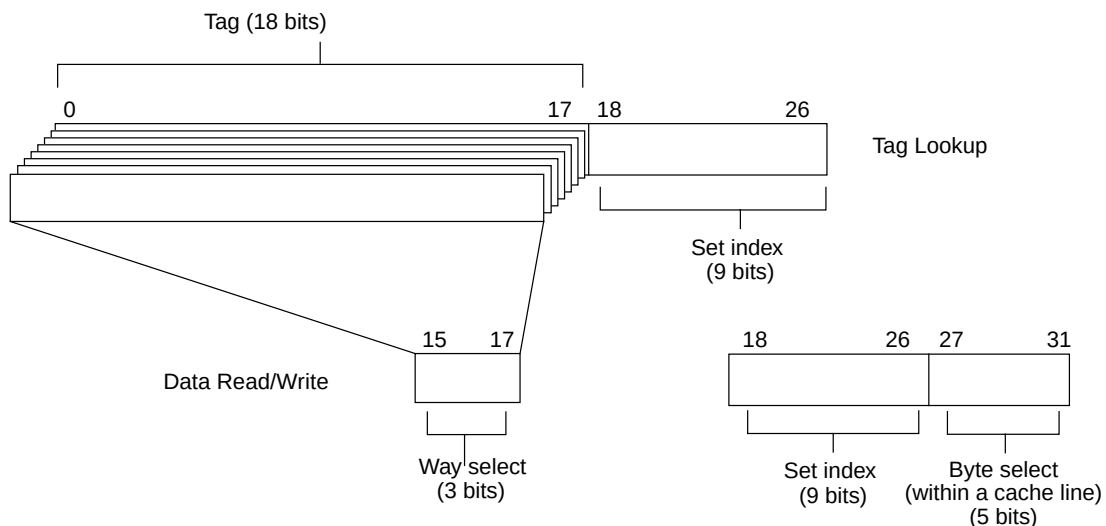


Figure 7-4. 128-Kbyte L2 Cache Address Configuration—Half SRAM, Half Cache Mode

The e500 core connects to the L2 and the system interface through the high-speed core complex bus (CCB). Both the e500 core and the L2 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.

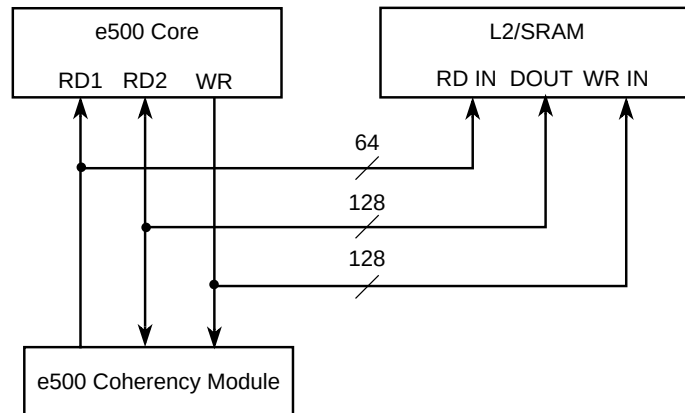


Figure 7-5. Data Bus Connection of CCB

Figure 7-6 shows address connections of the e500 core and L2/SRAM.

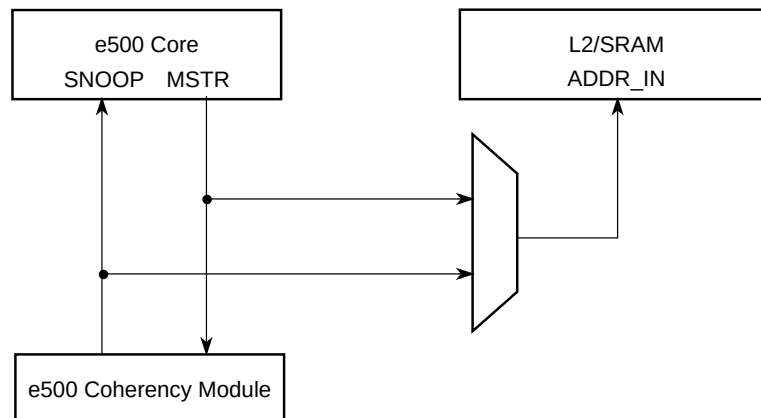


Figure 7-6. Address Bus Connection of CCB

7.3 Memory Map/Register Definition

Table 7-1 contains the memory map for the L2/SRAM registers.

Table 7-1. L2/SRAM Memory-Mapped Registers

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x2_0000	L2 control register (L2CTL)	R/W	0x2000_0000	7.3.1.1/7-6
0x2_0010	L2 cache external write address register 0 (L2CEWAR0)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0018	L2 cache external write control register 0 (L2CEWCR0)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0020	L2 cache external write address register 1 (L2CEWAR1)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0028	L2 cache external write control register 1 (L2CEWCR1)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0030	L2 cache external write address register 2 (L2CEWAR2)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0038	L2 cache external write control register 2 (L2CEWCR2)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0040	L2 cache external write address register 3 (L2CEWAR3)	R/W	0x0000_0000	7.3.1.2/7-9
0x2_0048	L2 cache external write control register 3 (L2CEWCR3)	R/W	0x0000_0000	7.3.1.3/7-10
0x2_0100	L2 memory-mapped SRAM base address register 0 (L2SRBAR0)	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0108	L2 memory-mapped SRAM base address register 1 (L2SRBAR1)	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0E00	L2 error injection mask high register (L2ERRINJHI)	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E04	L2 error injection mask low register (L2ERRINJLO)	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E08	L2 error injection tag/ECC control register (L2ERRINJCTL)	R/W	0x0000_0000	7.3.1.5.1/7-12
0x2_0E20	L2 error data high capture register (L2CAPTDATAHI)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E24	L2 error data low capture register (L2CAPTDATALO)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E28	L2 error ECC capture register (L2CAPTECC)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E40	L2 error detect register (L2ERRDET)	Special	0x0000_0000	7.3.1.5.2/7-14
0x2_0E44	L2 error disable register (L2ERRDIS)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E48	L2 error interrupt enable register (L2ERRINTEN)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E4C	L2 error attributes capture register (L2ERRATTR)	R/W	0x0000_0000	7.3.1.5.2/7-14
0x2_0E50	L2 error address capture register (L2ERRADDR)	R	0x0000_0000	7.3.1.5.2/7-14
0x2_0E58	L2 error control register (L2ERRCTL)	R/W	0x0000_0000	7.3.1.5.2/7-14

7.3.1 L2/SRAM Register Descriptions

The following sections describe the 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**

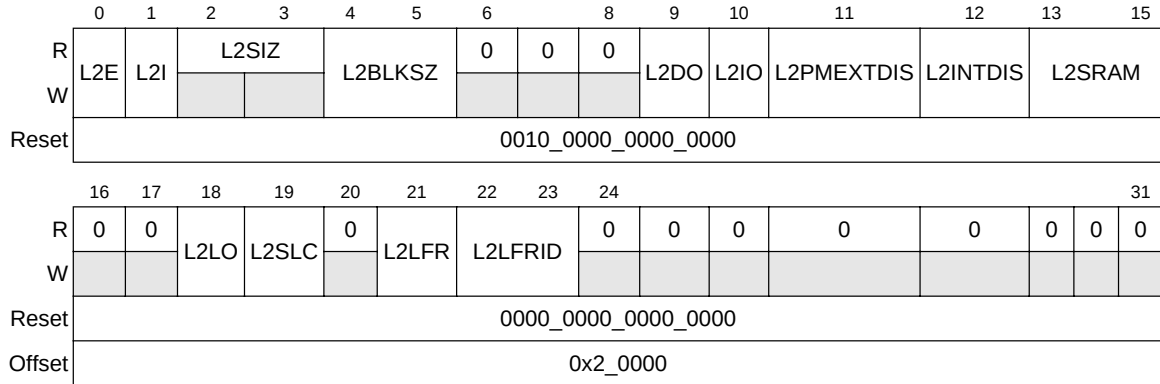
**Figure 7-7. L2 Control Register (L2CTL)**

Table 7-2 describes L2CTL fields.

Table 7-2. 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.
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. L2I may be set while the L2 cache is enabled (L2E = 1). Accesses 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 L2 SRAM (to be configured as cache or memory-mapped SRAM). 00 Reserved 01 Reserved 10 256 Kbyte 11 Reserved
4–5	L2BLKSZ	L2 cache/memory-mapped SRAM block size. Determines the L2 cache/memory-mapped SRAM block size. 00 Reserved 01 128 Kbyte 10 256 Kbyte 11 Reserved To change these bits, the L2 must be disabled (L2CTL[L2E] = 0). L2BLKSZ must be ≤ L2SIZ when L2E = 1. The L2CTL[L2SRAM] description for information on configuring the total SRAM as cache or memory-mapped SRAM

Table 7-2. L2CTL Field Descriptions (continued)

Bits	Name	Description
6–8	—	Reserved
9	L2DO	L2 data-only. Reserved in full memory-mapped SRAM mode. L2 cache lines are allocated for data cache transactions only. L2DO may be changed while the L2 is enabled or disabled. 0 The L2 cache allocates entries for instruction accesses that miss. 1 The L2 cache does not allocate entries for instruction accesses that miss in the L2. Instruction accesses that hit in the L2, data accesses, and accesses from the system are unaffected. If both L2DO and L2IO are set, no new lines are allocated in the L2 cache for core accesses, effectively locking the entire cache.
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 allocates entries for data accesses that miss. 1 The L2 cache does not allocate entries for data accesses that miss in the L2. Data accesses that hit, instruction accesses, and system accesses are unaffected. If L2DO and L2IO are set, no new L2 cache lines are allocated for core accesses, effectively locking the entire cache.
11	L2PMEXTDIS	Memory-mapped SRAM external access disable. Disables external access to memory-mapped SRAM. May be changed while the L2 is enabled or disabled. 0 External access to memory-mapped SRAM is enabled 1 External access to memory-mapped SRAM is disabled
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 1 Cache intervention is disabled
13–15	L2SRAM	L2 block assignment. Determines the L2 cache/memory-mapped SRAM block assignment. <u>L2SIZ = L2BLKSIZ (1 block):</u> 000 Block 0 = Cache 001 Block 0 = SRAM0 01x Reserved 1xx Reserved <u>L2SIZ = L2BLKSIZ * 2 (2 blocks):</u> 000 Block 0 unused; Block 1 = cache 001 Block 0 = SRAM0; Block 1 unused 010 Block 0 = SRAM0; Block 1 = cache 011 Block 0 = SRAM0; Block 1 = SRAM1 1xx Reserved The L2 must be disabled (L2CTL[L2E] = 0) in order to change these bits.
16–17	—	Reserved
18	L2LO	L2 cache lock overflow. Reserved in full memory-mapped SRAM mode. This sticky bit is set if a lock overflow 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. 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

Table 7-2. L2CTL Field Descriptions (continued)

Bits	Name	Description
19	L2SLC	L2 snoop lock clear. This sticky bit is set if a snoop invalidated a locked line in the data cache. 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 The L2 cache did not encounter a snoop that invalidated a locked line. L2SLC is cleared only by software. 1 The L2 cache encountered a snoop that invalidated a locked line.
20	—	Reserved
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 lock bits in the L2 cache are not cleared or the clear operation has completed. 1 A reset operation is issued that clears the lock bits of each L2 cache line. Depending on the value of L2LFRID, 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 (provided that 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
24–31	—	Reserved

7.3.1.2 L2 Cache External Write Address Registers (L2CEWAR_n)

The MPC8540 supports allocating and/or locking of lines in the L2 from external agents, such as PCI. This functionality is called cache external write. The L2 cache external write address registers 0–3 (L2CEWAR_n) work in conjunction with the L2 cache external write control registers 0–3 (L2CEWCR_n) to control the cache external write functionality.

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). L2CEWAR₀ and L2CEWCR₀ define one cache external write region; L2CEWAR₁ and L2CEWCR₁ define a second region, and so forth. The address register must be naturally aligned to the window size in the corresponding control register. Section 7.8.2, “L2 State Transitions,” lists cache state transitions due to cache external write transactions. L2CEWAR_n registers contain identical fields, as shown in Figure 7-8.

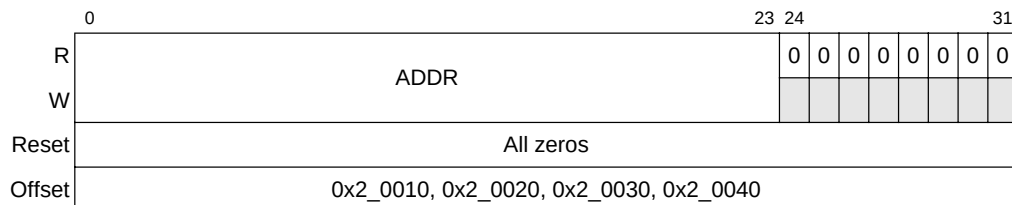
Figure 7-8. Cache External Write Address Registers (L2CEWAR_n)

Table 6-4 describes L2CEWAR fields.

Table 7-3. L2CEWAR_n Field Descriptions

Bits	Name	Description
0–23	ADDR	L2 cache external write base address
24–31	—	Reserved

7.3.1.3 L2 Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

The L2 cache external write address registers 0–3 (L2CEWAR_n), shown in Figure 7-9, are paired with the L2 cache external write control registers 0–3 (L2CEWCR_n), to control the cache external write functionality.

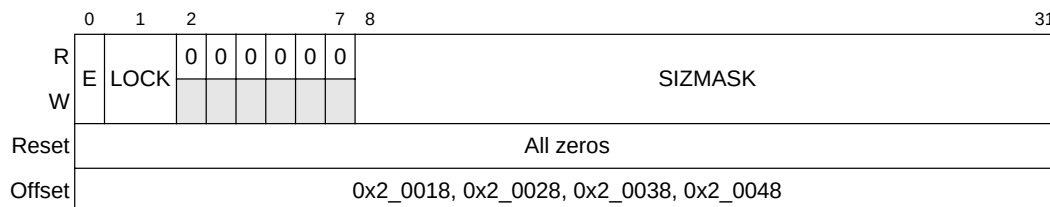


Figure 7-9. Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

The L2CEWCR registers contain identical fields, which are described in Table 7-4.

Table 7-4. L2CEWCR_n Field Descriptions

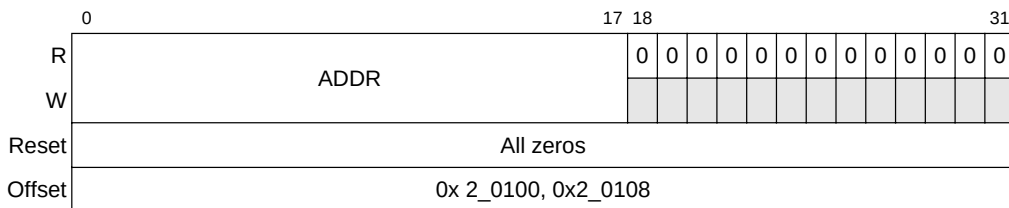
Bits	Name	Description
0	E	External write enable. An external write matching the address window defined by L2CEWAR _x /L2CEWCR _x is allocated or updated in the L2 cache. 0 External writes for L2CEWAR _x /L2CEWCR _x register pair are disabled. 1 External writes are enabled for L2CEWAR _x /L2CEWCR _x register pair.
1	LOCK	Lock lines in the targeted cache. An external write matching the address window defined by L2CEWAR _x /L2CEWCR _x is locked in the L2 cache when it is allocated or updated. 0 Allocated or existing lines in the targeted cache are not locked. 1 Lines in the targeted cache are allocated as locked. Existing lines are locked.

Table 7-4. L2CEWCR_n Field Descriptions (continued)

Bits	Name	Description
2–7	—	Reserved
8–31	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. 1111 1111 1111 1111 1111 1111 256 bytes 1111 1111 1111 1111 1111 1110 512 bytes 1111 1111 1111 1111 1111 1100 1 Kbyte 1111 1111 1111 1111 1111 1000 2 Kbytes 1111 1111 1111 1111 1111 0000 4 Kbytes 1111 1111 1111 1111 1110 0000 8 Kbytes 1111 1111 1111 1111 1100 0000 16 Kbytes 1111 1111 1111 1111 1000 0000 32 Kbytes 1111 1111 1111 1111 0000 0000 64 Kbytes 1111 1111 1111 1110 0000 0000 128 Kbytes 1111 1111 1111 1100 0000 0000 256 Kbytes 1111 1111 1111 1000 0000 0000 512 Kbytes 1111 1111 1111 0000 0000 0000 1 Mbytes 1111 1111 1110 0000 0000 0000 2 Mbytes 1111 1111 1100 0000 0000 0000 4 Mbytes 1111 1111 1000 0000 0000 0000 8 Mbytes 1111 1111 0000 0000 0000 0000 16 Mbytes 1111 1110 0000 0000 0000 0000 32 Mbytes 1111 1100 0000 0000 0000 0000 64 Mbytes 1111 1000 0000 0000 0000 0000 128 Mbytes 1111 0000 0000 0000 0000 0000 256 Mbytes 1110 0000 0000 0000 0000 0000 512 Mbytes 1100 0000 0000 0000 0000 0000 1 Gbytes 1000 0000 0000 0000 0000 0000 2 Gbytes 0000 0000 0000 0000 0000 0000 4 Gbytes Any other value is illegal and produces boundedly undefined results.

7.3.1.4 L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR_n)

The L2 memory-mapped SRAM base address registers (L2SRBAR_n), shown in Figure 7-10, control the memory-mapped SRAM mode functionality. Specified addresses must be aligned to the value in L2CTL[L2BLKSZ]. If L2CTL[L2SRAM] specifies one memory-mapped SRAM block, its base address must be written to L2SRBAR0; if it specifies two memory-mapped SRAM blocks, L2SRBAR0 and L2SRBAR1 are used.

**Figure 7-10. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR_n)**

L2SRBAR bits are described in Table 7-5.

Table 7-5. L2SRBAR0–L2SRBAR1 Field Descriptions

Bits	Name	Description
0–17	ADDR	L2 memory-mapped SRAM base address
18–31	—	Reserved

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 MPC8540 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 only the first error has the associated information captured. 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, 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.

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.

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 using 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
dcbt1s_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, per 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 7-11 shows the L2 error injection mask high register (L2ERRINJHI).

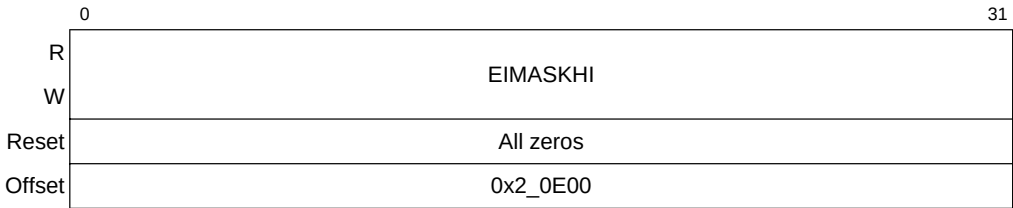


Figure 7-11. L2 Error Injection Mask High Register (L2ERRINJHI)

Table 7-6 describes L2ERRINJHI[EIMASKHI].

Table 7-6. L2ERRINJHI Field Description

Bits	Name	Description
0–31	EIMASKHI	Error injection mask for the high word of the data path. 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 7-12 shows the L2 error injection mask low register (L2ERRINJLO).

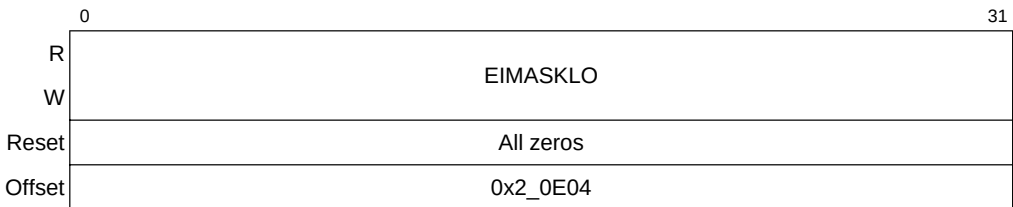


Figure 7-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

Table 7-7 describes L2ERRINJLO[EIMASKLO].

Table 7-7. L2ERRINJLO Field Description

Bits	Name	Description
0–31	EIMASKLO	Error injection mask for the low word of the data path. 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.

Table 7-13 shows the L2 error injection mask control register (L2ERRINJCTL).

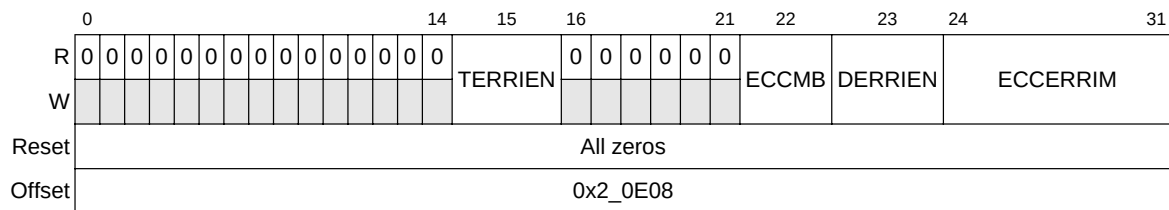


Figure 7-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

Table 6-4 describes L2ERRINJCTL fields.

Table 7-8. 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 All subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data error injection masks and ECC error injection mask and/or data path byte mirrored onto ECC as specified by ECC mirror byte enable. 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 the 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 7-14 shows the L2 error capture data high register (L2CAPTDATAHI).

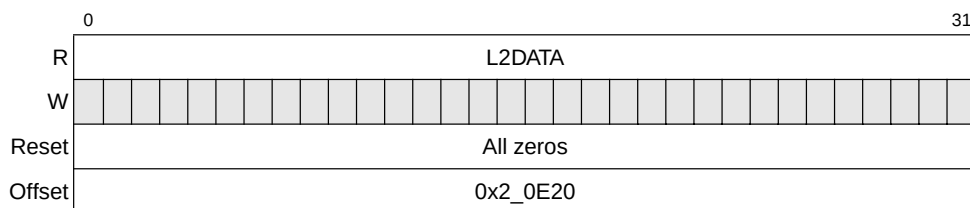


Figure 7-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

Table 7-9 describes L2CAPTDATAHI[L2DATA].

Table 7-9. L2CAPTDATAHI Field Description

Bits	Name	Description
0–31	L2DATA	L2 data high word

Figure 7-15 shows the L2 error capture data low register (L2CAPTDATALO).

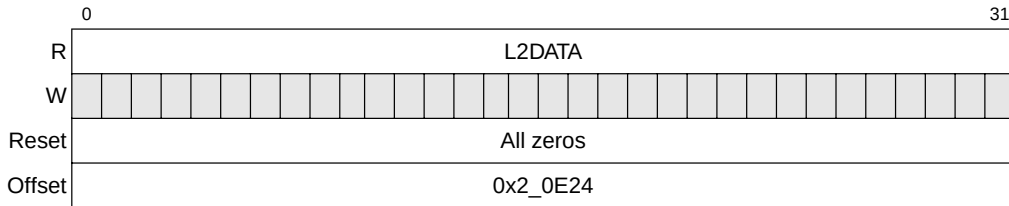


Figure 7-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

Table 7-6 describes L2CAPTDATALO[L2DATA].

Table 7-10. L2CAPTDATALO Field Description

Bits	Name	Description
0–31	L2DATA	L2 data low word

Figure 7-16 shows the L2 error syndrome register (L2CAPTECC).

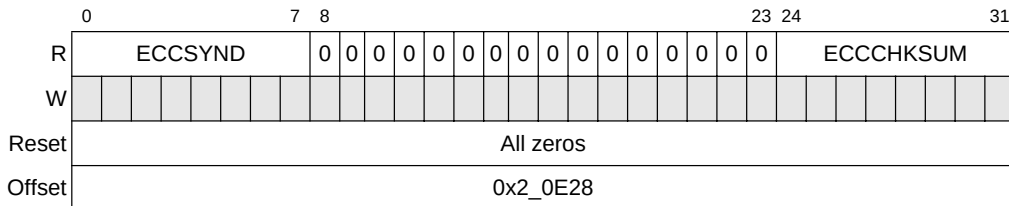


Figure 7-16. L2 Error Syndrome Register (L2CAPTECC)

Table 7-11 describes L2CAPTECC fields.

Table 7-11. L2CAPTECC Field Descriptions

Bits	Name	Description
0–7	ECCSYND	The calculated ECC syndrome of the failing doubleword
8–23	—	Reserved
24–31	ECCCHKSUM	The datapath ECC of the failing doubleword

Figure 7-17 shows the L2 error detect register (L2ERRDET).

	0	1									10	11		12		13		14		15
R	MULL2ERR	0	0	0	0	0	0	0	0	0	0	0		0		0		0		0
W																				
Reset	0000_0000_0000_0000																			

	16	17										26	27		28		29		30		31
R	0	0	0	0	0	0	0	0	0	0	0	0	TPARERR	MBECCERR	SBECCERR		0		L2CFGDIS		
W																					
Reset	0000_0000_0000_0000																				
Offset	0x2_0E40																				

Figure 7-17. L2 Error Detect Register (L2ERRDET)

Table 7-12 describes L2ERRDET fields.

Table 7-12. L2ERRDET Field Descriptions

Bits	Name	Description
0	MULL2ERR	Multiple L2 errors (Bit reset, write 1 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 clear) 0 Tag parity error was not detected 1 Tag parity error was detected
28	MBECCERR	Multiple-bit ECC error (Bit reset, write 1 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 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 clear) 0 L2 configuration errors were not detected 1 L2 illegal configuration error detected

Figure 7-18 shows the L2 error disable register (L2ERRDIS).

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16										26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	TPARDIS	MBECCDIS0	SBECCDIS	0	L2CFGDIS
W																
Reset	0000_0000_0000_0000															
Offset	0x2_0E44															

Figure 7-18. L2 Error Disable Register (L2ERRDIS)

Table 7-13 describes L2ERRDIS fields.

Table 7-13. 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 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 7-19 shows the L2 error interrupt enable register (L2ERRINTEN). When any of these error conditions exist and it is enabled, the L2 signals an interrupt to the e500 core through the internal *int* signal.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16											26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	TPARINTEN	MBECCINTEN	SBECCINTEN	0	L2CFGINTEN
W																	
Reset	0000_0000_0000_0000																
Offset	0x2_0E48																

Figure 7-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

Table 7-14 describes L2ERRINTEN fields.

Table 7-14. 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 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 7-20 shows the L2 error attributes capture register (L2ERRATTR).

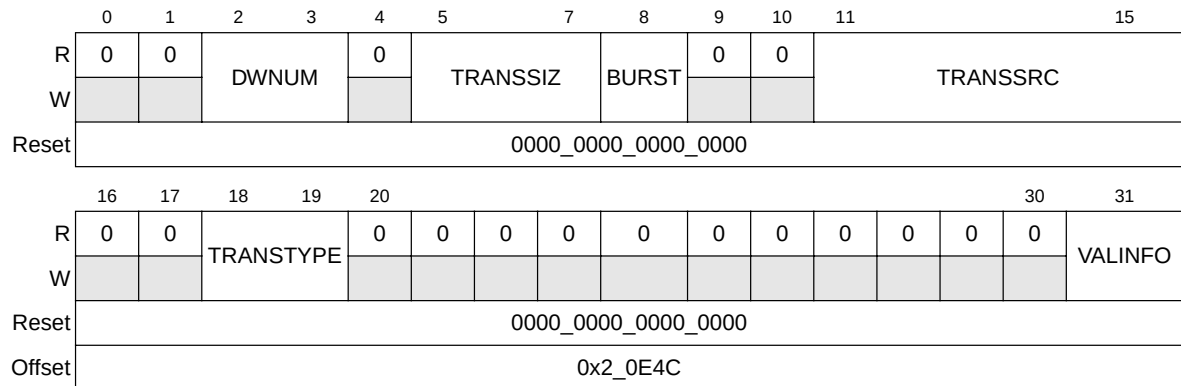
**Figure 7-20. L2 Error Attributes Capture Register (L2ERRATTR)**

Table 7-15 describes L2ERRATTR fields.

Table 7-15. 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 000 8 bytes (single-beat) or reserved (burst) 001 1 byte (single-beat) or 16 bytes (burst) 010 2 bytes (single-beat) or 32 bytes (burst) 011 3 bytes (single beat) or reserved (burst) 100 4 bytes (single-beat) or reserved (burst) 101 5 bytes (single-beat) or reserved (burst) 110 6 bytes (single-beat) or reserved (burst) 111 7 bytes (single-beat) or reserved (burst)
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 (inst) 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

Table 7-15. L2ERRATTR Field Descriptions (continued)

Bits	Name	Description
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 7-21 shows the L2 error address error capture register (L2ERRADDR).

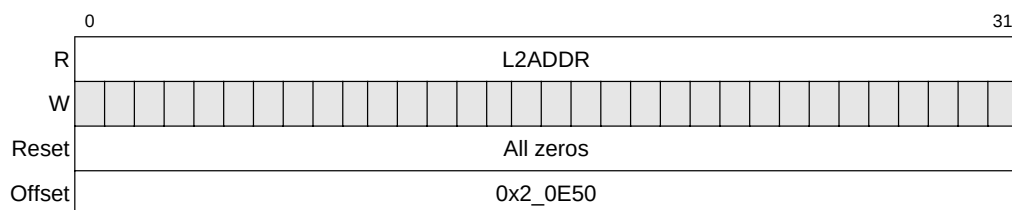
**Figure 7-21. L2 Error Address Error Capture Register (L2ERRADDR)**

Table 7-16 describes L2ERRADDR[L2ADDR].

Table 7-16. L2ERRADDR Field Description

Bits	Name	Description
0–31	L2ADDR	L2 address corresponding to detected error

Figure 7-22 shows the L2 error control register (L2ERRCTL).

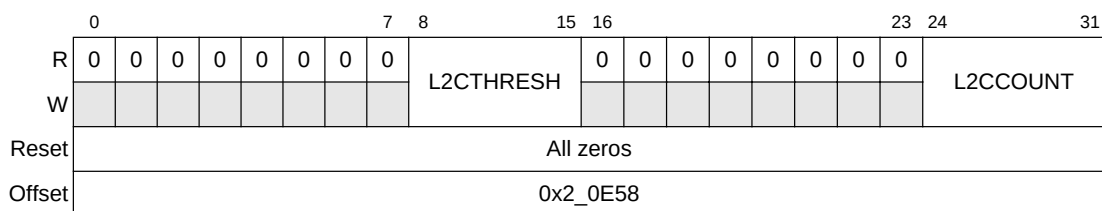
**Figure 7-22. L2 Error Control Register (L2ERRCTL)**

Table 7-17 describes L2ERRCTL fields.

Table 7-17. 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.

Table 7-17. L2ERRCTL Field Descriptions (continued)

Bits	Name	Description
16–23	—	Reserved
24–31	L2CCOUNT	L2 count. Counts ECC single-bit errors that have been detected. If L2CCOUNT equals the ECC single-bit error trigger threshold, an error is reported if single-bit error reporting is enabled. Software can write to this field.

7.4 L2 Cache Timing

Table 7-18 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 frequency of the L2 cache. 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 7-18. Fastest Read Timing—Hit in L2

Core Clks	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	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
CCB addr bus load 1		BG	TS																							
CCB addr bus load 2																										
CCB data bus load 1																										
CCB data bus load 2																										

7.5 SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term snoop transaction refers to transactions initiated by the MPC8540 system logic or I/O traffic, as opposed to e500 core-initiated transactions.

7.5.1 L2 Cache Coherency Rules

- The L2 is non-inclusive of the L1—lines that are valid in the L1 may be either valid or invalid in the L2.
- The L2 cache does not hold modified data. Data is in one of 4 states—invalid, exclusive, exclusive locked, or stale.
- The L2 allocates entries for castout data (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 cache does not respond to cache-inhibited read transactions.
- e500 core initiated cache-inhibited non-burst store transactions invalidate the line when they hit on a valid L2 line. If the line is locked the line 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 castout 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 on a stale line in the L2 cache 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 valid locked state for that line. 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 on a valid L2 line:
 - Global snoop FLUSH
 - **dcbf** instruction
 - **dcbi** instruction
- Non-global snoop transactions cause no change in the state of a line in the L2 cache.

- 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 I/O controllers via 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 a line in the L2 cache from peripheral devices or I/O controllers via write transactions with addresses that hit in a programmed memory range which has the lock attribute set.
- The L2 cache supports burst writes which allocate and/or lock a line in the L2 cache from peripheral devices or I/O controllers via 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.

7.5.2 Memory-Mapped SRAM Coherency Rules

- External (non core initiated) accesses to memory-mapped SRAM must be marked coherency required. Any external access 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.

7.6 L2 Cache Locking

7.6.1 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. If lines must be invalidated to satisfy coherency requirements, they are invalidated and 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 $L2CTL[L2DO]$ has 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.

7.6.2 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.

7.6.3 Locking Selected Lines

Individual lines are locked in the L2 when the L2 receives one of the following burst transactions:

- **icbtls** (CT = 1)—instruction cache block touch lock set instruction.
- **dcbtls** (CT = 1)—data cache block touch lock set instruction.
- **dcbtstls** (CT = 1)—data cache block touch for store 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. When the L2 cache is specified, data is not actually placed in the L1, only the L2. If the L1 cache is specified, 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 modified in the L1, the modified data is allocated into the L2 (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 already locked, the requested line is not allocated in the L2, and the L2 cache lock overflow bit (L2CTL[L2LO]) is set.

If lines must be invalidated to satisfy coherency requirements, they are invalidated and cannot be reallocated while they remain locked.

7.6.4 Clearing Locks On Selected Lines

Clearing of individual locks in the L2 is done by performing a lock clear (**icblc** or **dcblc**, CT = 1) instruction. This operation 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.

7.6.5 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 of locks is the only method by which data locks can be cleared without clearing instruction locks, or instruction locks can be cleared without clearing data locks. All instructions and snoop transactions that clear locks clear both data and instruction locks.

7.6.6 Locks With Stale Data

If data is locked in the L2 and the e500 core performs a cacheable copyback store or **dcbtst** instruction that misses in the L1, the L2 invalidates the line. In this case, although the L2 clears the valid bit for the data, the lock remains, and the line cannot be victimized. If the e500 core casts out the modified data or pushes it in response to a non-flush snoop, the L2 updates the data and sets the valid bit again, thus maintaining the lock and keeping the data in the cache hierarchy.

7.7 L2 Replacement Policy

Line replacement is performed using a binary decision tree pseudo least-recently-used (PLRU) algorithm. There is a valid bit (V0–V7) for each line in the L2. There are seven PLRU bits (P0–P6) for each set in the L2 cache to determine the replacement victim (the line that needs to be cast out). PLRU bits are updated every time a new line is allocated or replaced and every time a line is modified or invalidated. In addition, 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.

Figure 7-23 shows the binary decision tree used to generate the victim line. In the figure, the eight ways of the L2 cache are labeled W0–W7 and the seven pseudo-LRU (PLRU) bits are labeled P0–P6.

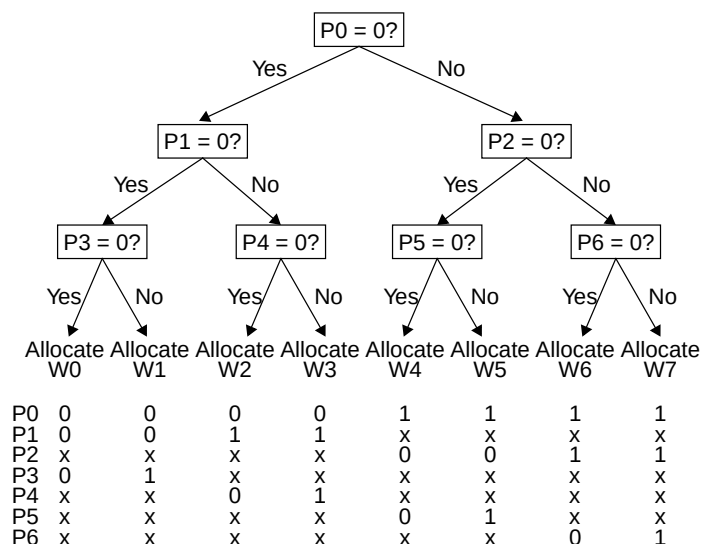


Figure 7-23. L2 Cache Line Replacement Algorithm

7.7.1 PLRU Bit Update Considerations

PLRU bits are updated when a way is modified, either by a write, an invalidate, or an allocation of a new line. PLRU bit updates depend on which cache way was most recently accessed as summarized in Table 7-19.

Table 7-19. PLRU Bit Update Algorithm

Most Recently Accessed Way	PLRU Bits Updated	Most Recently Accessed Way	PLRU Bits Updated
0	P0 is set P1 is set P3 is set	4	P0 is cleared P2 is set P5 is set
1	P0 is set P1 is set P3 is cleared	5	P0 is cleared P2 is set P5 is cleared
2	P0 is set P1 is cleared P4 is set	6	P0 is cleared P2 is cleared P6 is set
3	P0 is set P1 is cleared P4 is cleared	7	P0 is cleared P2 is cleared P6 is cleared

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.

7.7.2 Allocation of Lines

L2 cache lines are locked through the lock bits in the status array. There are two lock bits for each way of each set (1024sets by 8 ways). The lock bits are set or cleared through special L2 controller commands.

Lock bits are used at allocate time to steer the PLRU algorithm away from selecting locked victims. In the following discussion, the 8 lock bits for a particular set are called L0–L7.

Where Lock Way i : $L_i = D_i \mid I_i$, $i=0..7$ (D_i = data lock, I_i = instruction lock)

An effective value of each PLRU bit is calculated as follows:

$$\begin{aligned}
 P0\_eff &= f(P0, L0, L1, L2, L3, L4, L5, L6, L7) = (L0 \& L1 \& L2 \& L3) \mid (P0 \& \sim(L4 \& L5 \& L6 \& L7)) \\
 P1\_eff &= f(P1, L0, L1, L2, L3) = (L0 \& L1) \mid (P1 \& \sim(L2 \& L3)) \\
 P2\_eff &= f(P2, L4, L5, L6, L7) = (L4 \& L5) \mid (P2 \& \sim(L6 \& L7)) \\
 P3\_eff &= f(P3, L0, L1) = L0 \mid (P3 \& \sim L1) \\
 P4\_eff &= f(P4, L2, L3) = L2 \mid (P4 \& \sim L3) \\
 P5\_eff &= f(P5, L4, L5) = L4 \mid (P5 \& \sim L5) \\
 P6\_eff &= f(P6, L6, L7) = L6 \mid (P6 \& \sim L7)
 \end{aligned}$$

These effective PLRU bits are used to select a victim as indicated in Table 7-20.

Table 7-20. PLRU-Based Victim Selection Mechanism

Way Selected	PLRU State (Binary)	Reduced Logic Equation
W0	00x0xxx	$\sim P0 \& \sim P1 \& \sim P3$
W1	00x1xxx	$\sim P0 \& \sim P1 \& P3$
W2	01xx0xx	$\sim P0 \& P1 \& \sim P4$
W3	01xx1xx	$\sim P0 \& P1 \& P4$
W4	1x0xx0x	$P0 \& \sim P2 \& \sim P5$
W5	1x0xx1x	$P0 \& \sim P2 \& P5$
W6	1x1xxx0	$P0 \& P2 \& \sim P6$
W7	1x1xxx1	$P0 \& P2 \& P6$

7.8 L2 Cache Operation

7.8.1 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)

7.8.2 L2 State Transitions

Table 7-51 lists the L2 cache state transitions for all e500 core-initiated transactions that change the L2 cache state. The remaining core-initiated transactions (caused by execution of the **msync**, **mbar**, **tlbivax**, and **tlbsync** instructions by the core), do not change the L2 cache state. The table does not list initial L1 states for those transactions that hit in the L1 (iL1 or dL1) and are not sent to the L2.

In the table, L2 hit indicates that the L2 provides data (reads) or captures data (writes) 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 7-21. State Transitions Due to Core-Initiated Transactions

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable instruction fetch icbtl_L1	iL1 I	I/T	No	same	I/V	Donly = 1. L2 touch instructions not allocated in L1
		I	No	E	I/V	Donly = 0
icbt_L2	dL1 I,S,E	E/EL	Yes	same	I/V	
		T	No	EL	I/V	Donly = 0. Restore locked line in L2 with valid data from bus
icbtl_L2	dL1 I,S,E	I/T	No	same	I	Donly = 1
		E	Yes	I	I	Donly = 1
		EL	Yes	T	I	Donly = 1
		I	No	EL	I	Donly = 0
		E	Yes	EL	I	Donly = 0
		EL	Yes	same	I	Donly = 0
		T	No	EL	I	Donly = 0. Restore locked line in L2 with valid data from bus

Table 7-21. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
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) dcbtIs_L1 (4-state)	dL1 I	I/T	No	same	E	IOnly = 1
		E	Yes	I	E	IOnly = 1
		EL	Yes	T	E	IOnly = 1
		I	No	E	E	IOnly = 0
		E/EL	Yes	same	E	IOnly = 0
		T	No	EL	EL	IOnly = 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,S	I/T	No	same	M	L2 allocates when a line is cast out of L1.
		E	Yes	I	M	
		EL	Yes	T	M	
Writeback stwcx	dL1 I,S	I/T	No	same	M	
		E	Yes	I	M	
		EL	Yes	T	M	
Cacheable load (3-state) Cacheable lwarx (3-state) dcbt_L1 (3-state) dcbtIs_L1 (3-state)	dL1 I,S	I/S	No	I	E/I	IOnly = 1
		T	No	T	E/I	IOnly = 1
		E	Yes	I	E/I	IOnly = 1
		EL	Yes	T	E/I	IOnly = 1
		I	No	E	E/I	IOnly = 0
dcbt_L2 dcbtst_L2	dL1 I,S,E	E/EL	Yes	same	E/I	IOnly = 0
		T	No	EL	E/I	IOnly = 0. Restore locked line with valid data from bus
dcbtst_L1 dcbtstIs_L1	dL1 I,S	I/T	No	same	E	
		E	Yes	I	E	
		EL	Yes	T	E	

Table 7-21. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbtIs_L2 dcbtstIs_L2	dL1 I,S,E	I/S	No	I	I	IOnly = 1
		T	No	T	I	IOnly = 1
		E	Yes	I	I	IOnly = 1
		EL	Yes	T	I	IOnly = 1
		I	No	EL	I	IOnly = 0
		E/EL	Yes	EL	I	IOnly = 0
		T	No	EL	I	IOnly = 0. Restore locked line with valid data from bus
Writethrough store	dL1 I,S,E,M	I/T	No	I	same	
		E/EL	Yes	same	same	Read-modify-write
Cache-inhibited store	N/A	I/E	No	I	N/A	Invalidate line
		EL/T	No	T	N/A	Invalidate data, keep lock
Cache-inhibited stwcx	N/A	I/E	No	I	N/A	Invalidate line
		EL/T	No	T	N/A	Invalidate data, keep lock
dcbIc_L2 icbIc_L2	dL1 I,S,E,M	I/E	No	same	same	
		EL	No	E	same	
		T	No	I	same	
Victim Castout dcbt_L2 icbt_L2 dcbtst_L2	dL1 M	I/T	No	same	I	IOnly = 1. If software is sharing cache lines between instructions and data and wishes to capture instruction lines in L2 with IOnly = 1, it must perform dcbst to flush the line out of the dL1 before fetching it into L2.
		I	No	E	I	IOnly = 0
		E/EL	Yes	same	I	
		T	Yes	EL	I	IOnly = 0
dcbtIs_L2 icbtIs_L2 dcbtstIs_L2		I	No	EL	I	An icbtIs_L2 that hits modified in L1 cannot be distinguished from dcbtIs_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set ilocks in L2, it must perform dcbst to flush the line out of the dL1 before locking it in L2.
		E/EL/T	Yes	EL	I	
Snoop push	dL1 M	I/E	No	I	I/S/E	
		EL/T	No	T	I/S/E	Invalidate data, keep lock

Table 7-21. State Transitions Due to Core-Initiated Transactions (continued)

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbf dcbst	dL1 M	I/E/EL	No	I	I	
dcbz dcba	dL1 I,S	I/E	No	I	M	
		EL	No	T	M	
dcbi	dL1 I,S,E,M	I/ E/EL/T	No	I	I	
dcbf dcbst	dL1 I,S,E	I/ E/EL/T	No	I	I	
icbi	iL1 I,V	I/ E/EL/T	No	I	I	

Table 7-22 lists the L2 cache state transitions for all system (non-core) initiated transactions that change the L2. The transaction types and attributes listed follow MPX bus nomenclature, with the addition of write allocate (burst write with L2 cache allocation). Table 7-22 does not take into account changes caused by L1 snoop pushes triggered by snoops, which are listed in Table 7-51.

Table 7-22. 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	0	1	0	I/E/EL/T	EL	Allocate and lock regardless of cache external write (CEW) window
	1	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. Set lock if CEW lock attribute set.
				EL	same	
				E	E/EL	
				T	EL	
	x	0	0	I/E	I	Invalidate line
				EL/T	T	Invalidate data, keep lock

Table 7-22. State Transitions Due to System-Initiated Transactions (continued)

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
< 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
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/S/SL/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	

Part III

Memory and I/O Interfaces

This part defines the memory and I/O interfaces of the MPC8540 and it describes how these blocks interact with one another and with other blocks on the device. It contains the following chapters:

- Chapter 8, “e500 Coherency Module” defines the e500 coherency module and how it facilitates communication between the e500 core complex, the L2 cache and the other blocks that comprise the coherent memory domain of the MPC8540.

The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8540.

- Chapter 9, “DDR Memory Controller” describes the DDR SDRAM memory controller of the MPC8540. This fully programmable controller supports 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 DLL software override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, “Programmable Interrupt Controller” describes the embedded programmable interrupt controller (PIC) of the MPC8540. This controller is an OpenPIC-compliant interrupt controller that provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them and delivering them to the CPU for servicing.
- Chapter 11, “I2C Interface” describes the inter-IC (IIC or I²C) bus controller of the MPC8540. 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 MPC8540 powers up in boot sequencer mode which allows the I²C controller to initialize configuration registers.

- Chapter 12, “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 13, “Local Bus Controller” describes the local bus controller of the MPC8540. The main component of the local bus controller (LBC) 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 high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.
- Chapter 14, “Three-Speed Ethernet Controller” describes the two triple-speed Ethernet controllers on the MPC8540. These controllers provide 10/100/1000 Mbps Ethernet support with a complete set of media-independent interface options including GMII, RGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8540 memory coherency module.
- Chapter 15, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8540. The DMA controller transfers blocks of data, independent of the e500 core or external hosts. Data movement occurs among RapidIO and the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 16, “PCI Bus Interface,” describes the PCI/PCI-X controller of the MPC8540.
- Chapter 17, “RapidIO Interface,” describes the RapidIO controller of the MPC8540.

Chapter 8

e500 Coherency Module

8.1 Introduction

Figure 8-1 shows a high level overview of the e500 coherency module (ECM).

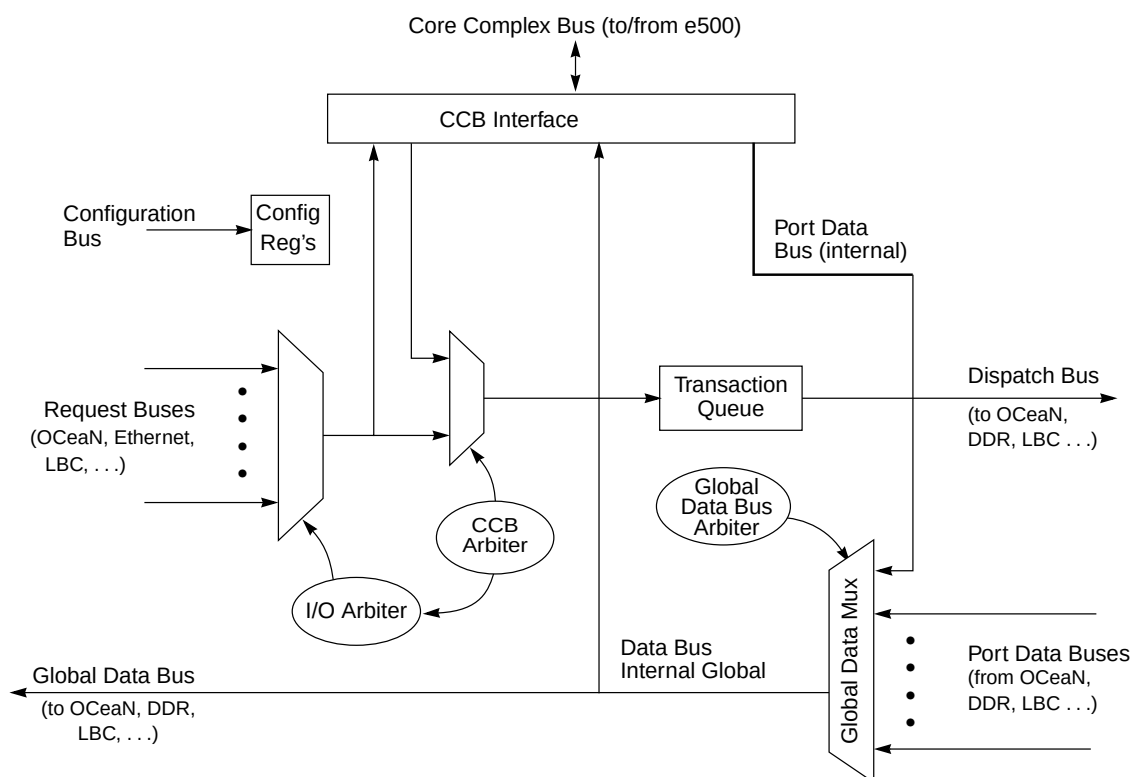


Figure 8-1. e500 Coherency Module Block Diagram

8.2 Overview

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. 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 take appropriate action to maintain coherency across cacheable memory.

8.2.1 Features

The ECM includes these distinctive features:

- Support for e500 core and an L2/SRAM on the CCB and includes a CCB arbiter.
 - Sources a 64-bit data bus for returning read data from the ECM to the e500 core 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. It also sinks one 128-bit write data bus from the e500 core.
- The ECM supports four connection points for I/O initiating (mastering into the device) interfaces. One of those connection points services OCeaN initiators, one services gigabit Ethernet channel 1, one services gigabit Ethernet channel 2, and the last services other I/O initiators. The ECM supports five connection points for I/O targets. The DDR memory controller, local bus, OCeaN targets, programmable interrupt controller, 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 for data moving to and from initiators and targets.
- Ensures proper ordering of I/O-initiated transactions.
- Speculative read bus for low-latency dispatch of reads to the DDR controller.
- Low-latency path for returning read data from DDR to the e500.
- Error registers to trap those transactions that do not map. These errors can be programmed to generate interrupts to the e500 core.

8.3 Memory Map/Register Definition

Table 8-1 shows a summary of the ECM's memory map. If a 4-byte address space between offset 0x000–0xFFFF is not defined here, it is reserved.

Table 8-1. ECM Memory Map

Address (offset)	Register	Access	Reset	Section/Page
Registers				
0x0_1000	ECM CCB address configuration register (EEBACR)	R/W	0x0000_0003	8.3.1.1/8-3
0x0_1010	ECM CCB port configuration register (EEBPCR)	R/W	0x0*00_0000	8.3.1.2/8-4
0x0_1E00	ECM error detect register (EEDR)	Special	0x0000_0000	8.3.1.3/8-5
0x0_1E08	ECM error enable register (EEER)	R/W	0x0000_0000	8.3.1.4/8-5

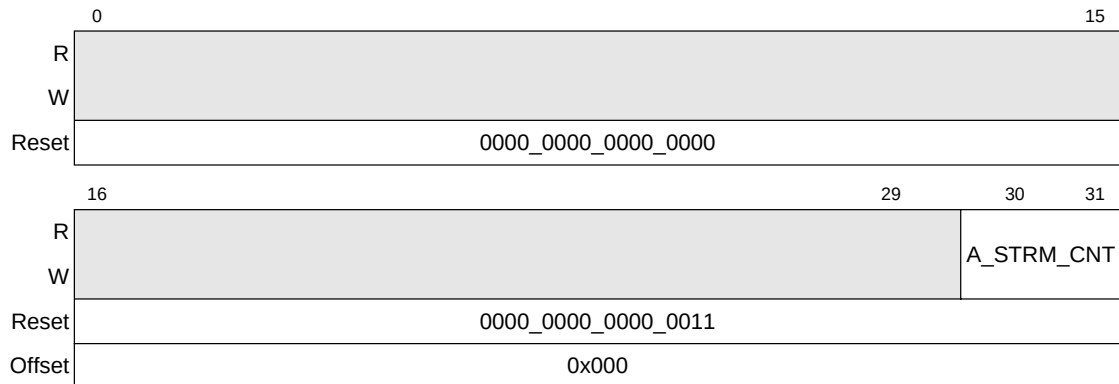
Table 8-1. ECM Memory Map (continued)

0x0_1E0C	ECM error attributes capture register (EEATR)	R	0x0000_0000	8.3.1.5/8-6
0x0_1E10	ECM error address capture register (EEADR)	R	0x0000_0000	8.3.1.6/8-8

8.3.1 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.

8.3.1.1 ECM CCB Address Configuration Register (EEBACR)

**Figure 8-2. ECM CCB Address Configuration Register (EEBACR)****Table 8-2. EEBACR Bit Settings**

Bits	Name	Description
0–29	—	Reserved
30–31	A_STRM_CNT	Identifies the maximum number of transactions that can be streamed together from a particular master once an initial transaction is issued on the CCB. 00 Reserved 01 One transaction can be streamed together with initial transaction 10 Two transactions can be streamed together with initial transaction 11 Three transactions can be streamed together with initial transaction

8.3.1.2 ECM CCB Port Configuration Register (EEBPCR)

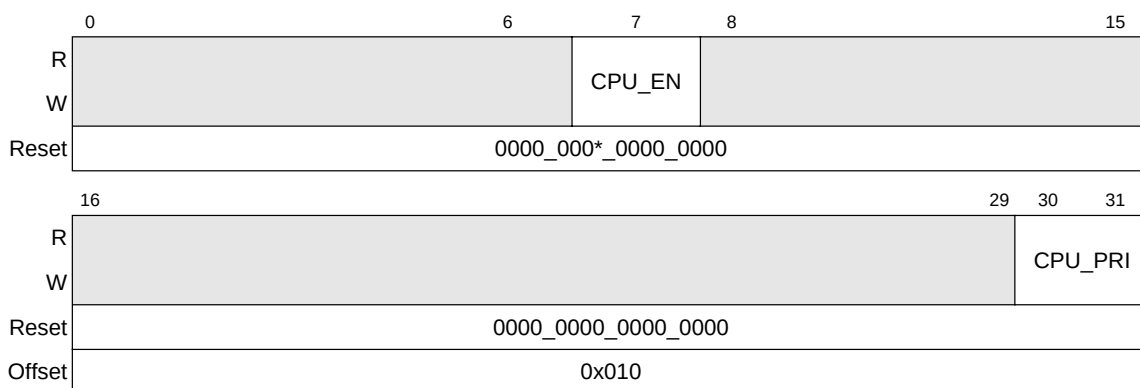


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

Table 8-3. EEBPCR Bit Settings

Bits	Name	Description
0–6	—	Reserved
7	CPU_EN	CPU port enable. Identifies whether the e500 core (CPU) port can receive bus grants on the CCB. Note that the CPU Boot Configuration Power-on-reset pincfg_port0_en determines the default value. If the pin samples a logic 1 during reset, the e500 core port powers up enabled, otherwise it powers up disabled. 0 Port is disabled and does not receive bus grants. 1 Port is enabled and receives bus grants in response to bus requests.
8–29	—	Reserved
30–31	CPU_PRI	Identifies the priority level of the e500 core (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

8.3.1.3 ECM Error Detect Register (EEDR)

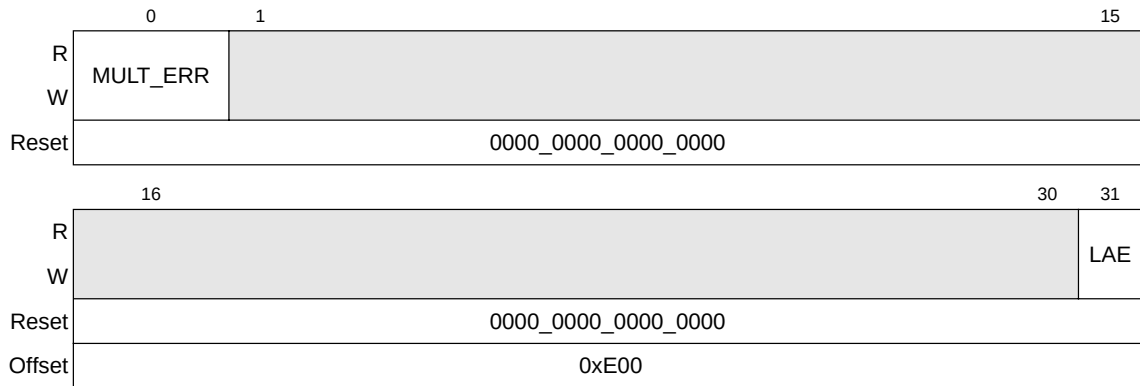


Figure 8-4. ECM Error Detect Register (EEDR)

Table 8-4. EEDR Bit Settings

Bits	Name	Description
0	MULT_ERR	Cleared (if it is set) by writing a 1 to it. Other software writes have no effect on the register. 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. Cleared (if it is set) by writing a 1 to it. Other software writes have no effect. There are three cases that can generate LAE's: - if the transaction does not map to any target. - if the source ID and target ID identify the transaction as targeting a port on OCN when the transaction was actually initiated by a port on OCN. This “loopback” behavior can result from programming errors where the inbound ATMU window targets are not consistent with the targets configured in the local access windows for a given address range. For this “loopback” type of LAE, the dispatch (to OCN target in this case) is not screened off. This means that the LAE error is reported but the transaction is still sent down to its OCN target. This is different from the other two conditions that generate LAE's, where the dispatch to the target (if it maps at all) is screened off, read responses are injected by the ECM (with corrupt attribute), and write data is simply dropped. when inbound read atomic set, clear, inc, or dec transactions from RapidIO target an interface other than DDR. The only target that supports these read atomic set, clear, inc, and dec transactions is DDR. It is recommended that the interrupt controller be configured to route the ECM interrupt (the only one it can generate is this LAE) to the external interrupt pin. The third type of LAE (described above) may cause an internal hang condition, so an interrupt service routine handled internally by the core may never be able to read this register to find out about the LAE. Routing the ECM interrupt externally ensures that the user finds out about the LAE error condition. 0 Local access error not found. 1 Local access error found.

8.3.1.4 ECM Error Enable Register (EEER)

This register enables the reporting of error conditions to the e500 core through the internal *int* interrupt signal.

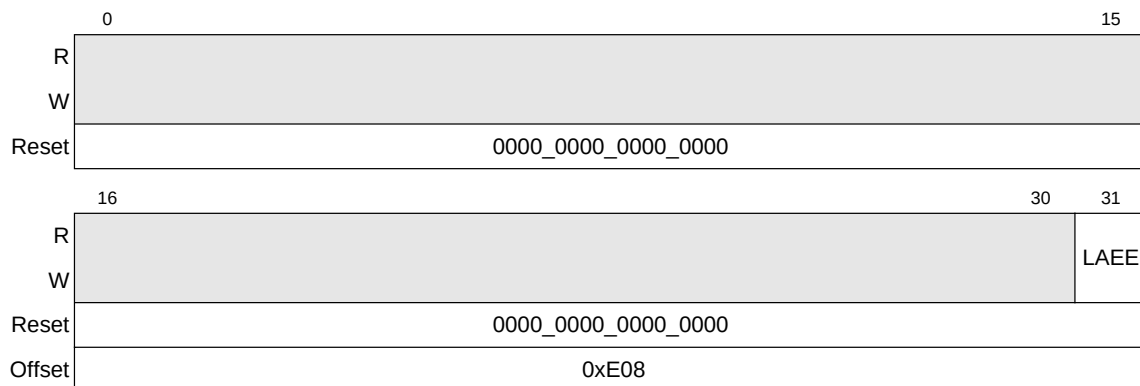


Figure 8-5. ECM Error Enable Register (EEER)

Table 8-5. EEER Bit Settings

Bits	Name	Description
0–30	—	Reserved
31	LAEE	0 Disable reporting local access errors as interrupts. 1 Enable reporting local access errors as interrupts.

8.3.1.5 ECM Error Attributes Capture Register (EEATR)

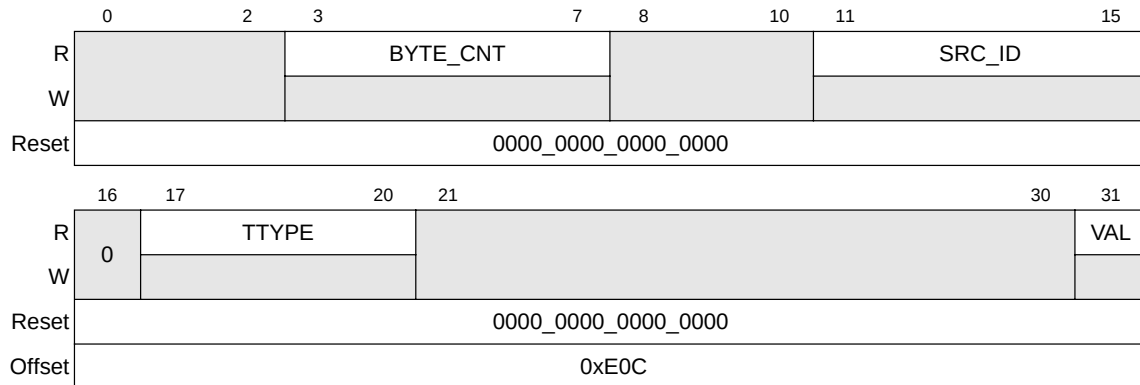


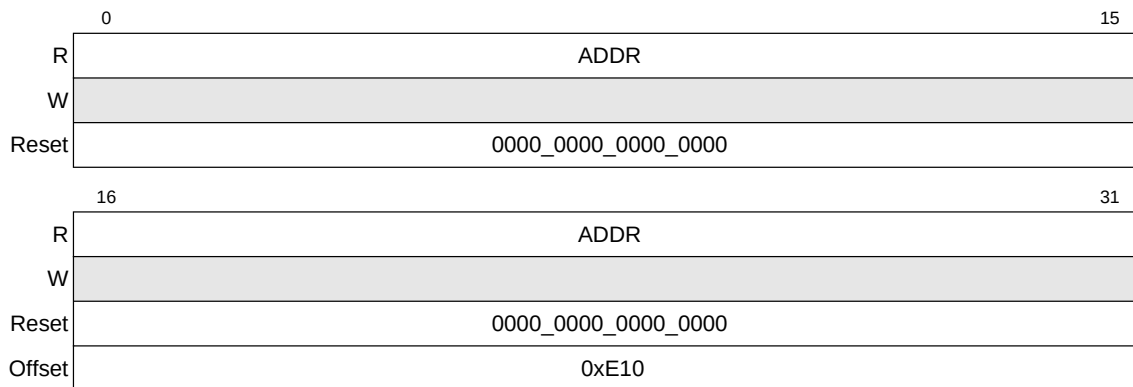
Figure 8-6. ECM Error Attributes Capture Register (EEATR)

Table 8-6. EEATR Bit Settings

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 00011 3 bytes 00100 4 bytes 01000 8 bytes 10000 16 bytes
8–10	—	Reserved
11–15	SRC_ID	Source ID. Identifies the source device mastering the transaction. 00000 PCI/PCI-X interface 00001 Reserved : : 01001 Reserved 01010 Boot Seq. 01011 Reserved 01100 RapidIO 01101 Reserved : : 01111 Reserved 10000 Processor 1 (instruction) 10001 Processor 1 (data) 10010 Reserved 10011 Reserved 10100 Reserved 10101 DMA 10110 Reserved 10111 SAP 11000 Ethernet 1 11001 Ethernet 2 11010 Ethernet 3 11011 Reserved 11100 Rapid IO Message Units 11101 RapidIO Doorbell Units 11110 RapidIO Port-write Units 11111 Reserved
16	—	Reserved

Table 8-6. EEATR Bit Settings (continued)

Bits	Name	Description
17–20	TTYE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction 0101 Reserved 011x Reserved 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
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.

8.3.1.6 ECM Error Address Capture Register (EEADR)**Figure 8-7. ECM Error Address Capture Register (EEADR)****Table 8-7. EEADR Bit Settings**

Bits	Name	Description
0–31	ADDR	Address. Specifies the 32-bit address of the transaction. Qualified by EEATR[VAL].

8.4 Functional Description

8.4.1 I/O Arbiter

Figure 8-1 shows the I/O arbiter block that manages I/O-initiated address tenure requests coming in on the request buses. Four request buses compete for access to the ECM, and this block looks at the longest waiting request bus port in combination with priority levels to determine the winning request bus. The transaction from the winning request bus competes with e500 core requests for the CCB and entry into the transaction queue.

8.4.2 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 core and the winning request bus and consequently controls when these new transactions are allowed to enter the transaction queue. The arbiter operates in an LRU mode.

8.4.3 Transaction Queue

The ECM's transaction queue performs three basic functions: 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 masters other than the e500 core (all I/O masters) are strictly ordered and are dispatched to their target interfaces in the same order they are submitted. For those transactions accessing address space marked as snoopable, or space that may be cached by the e500 core, the ECM enforces coherency, snooping those transactions on the CCB, and taking castouts from the e500 core as is necessary.

8.4.4 Global Data Multiplexor

Figure 8-1 shows how the global data multiplexor (mux) takes data bus connections and multiplexes them onto one global data bus. The global data mux allows initiators of write transactions to route data to their targets, and also allows read targets to return data to the initiators.

8.4.5 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-bits wide) and read (128 bits 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).

8.5 Initialization/Application Information

If the e500 core is used to initialize the MPC8540, the CPU Boot Configuration Power-on-reset pin should be pulled high to initially set EEBPCR[CPU_EN]. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on power-up reset initialization.

If any device other than the e500 core, such as the boot sequencer or PCI, is used to initialize the MPC8540, the CPU Boot Configuration Power-on-reset pin should be pulled low to initially clear EEBPCR[CPU_EN]. This prevents the e500 core 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[CPU_EN] to re-enable the e500 core.

EEBPCR[CPU_PRI] identifies the priority level associated with all e500 core initiated transactions. This field defaults to 0b00 such that all e500 core (CPU) initiated transactions have the lowest priority. This priority level affects whether the e500 core requests can interrupt the streaming of address tenures initiated by the ECM (on behalf of I/O masters). The higher the priority level given to the e500, the lower the average latency needed to obtain bus grants from the ECM.

EEBACR[A_STRM_CNT] defaults to 0b11 which causes the ECM to attempt to stream as many as four transactions initiated from the same CCB master. Increasing the value of this field increases the maximum number of transactions that may be streamed together from one CCB master. However, streaming more transactions together from one CCB master may increase latency for lower priority transactions from another CCB master, as only those transactions with a priority greater than the current transaction on the CCB can interrupt streaming

Chapter 9

DDR Memory Controller

This fully programmable DDR SDRAM controller supports most JEDEC standard x8 or x16 DDR memories available today, including both buffered and unbuffered DIMMs. (However, mixing unbuffered and registered DIMMs in the same system is not supported.) 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 DLL software override, crawl mode, and ECC error injection support rapid system debug.

NOTE

Throughout this document, the word ‘bank’ refers to a physical bank that is specified by a chip select, and the term ‘logical bank’ refers to a bank within a chip that is specified by a bank address.

Figure 9-2 shows a high-level block diagram of the DDR memory controller with its associated interfaces. Section 9.4, “Functional Description,” contains more detailed figures of the controller.

Features

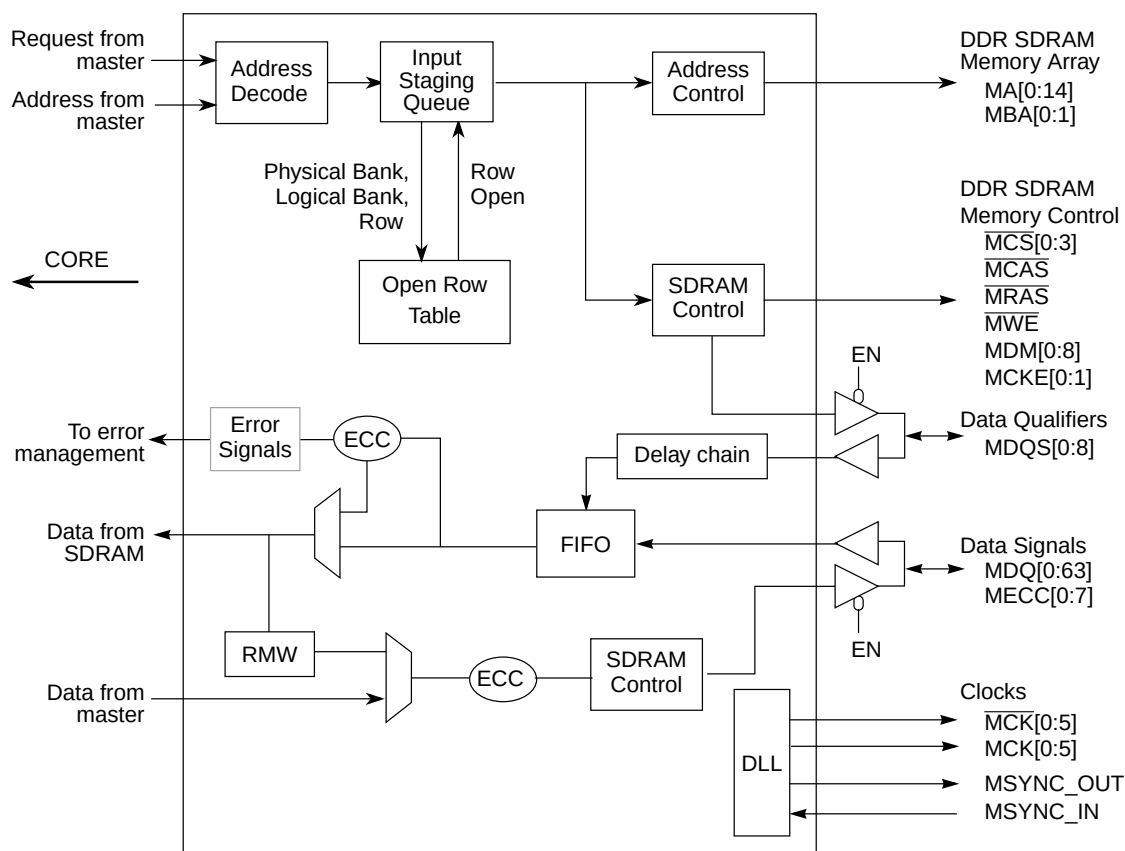


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.1 Features

The DDR memory controller includes these distinctive features:

- Support for DDR SDRAM
- 64/72-bit SDRAM data bus
- Programmable settings for meeting all SDRAM timing parameters
- Many different SDRAM configurations supported
 - Support for as many as four physical banks (chip selects), each bank independently addressable
 - Support for 64-Mbit to 1-Gbit devices with x8/x16 data ports (no direct x4 support)
 - Support for unbuffered and registered DIMMs
- Support for data mask signals and read-modify-write for sub-double word writes
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)

- Two-entry input request queue
- Open page management (dedicated entry for each sub-bank)
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management mode
- Support for error injection

9.1.1 Modes of Operation

The DDR memory controller supports the following modes of operation:

- 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. In addition, auto precharge mode may be enabled for separate chip selects.

9.2 External Signal Description

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.

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{AS}$ (address strobe). 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 NMI (nonmaskable interrupt), are referred to as asserted when they are high and negated when they are low.

9.2.1 Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 9-1 lists the external signals of the DDR memory controller, showing how the signals are grouped. Refer to the *MPC8540 Hardware Specification* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

Table 9-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
MDQ[0:63]	Data bus	All zeros	64	I/O
MDQS[0:8]	Data strobes	All zeros	9	I/O
MECC[0:7]	Error checking and correcting	All zeros	8	I/O
$\overline{\text{MCAS}}$	Column address strobe	All ones	1	O
MA[0:14]	Address bus	All zeros	14	O
MBA[0:1]	Logical bank address	All zeros	2	O
$\overline{\text{MCS}}[0:3]$	Chip select	All ones	4	O
$\overline{\text{MWE}}$	Write enable	All ones	1	O
$\overline{\text{MRAS}}$	Row address strobe	All ones	1	O
MDM[0:8]	Data mask	All zeros	9	O
MCK[0:5]	DRAM clock outputs	All zeros	6	O
$\overline{\text{MCK}}[0:5]$	DRAM clock outputs (complement)	All ones	6	O
MCKE[0:1]	Clock enable	All zeros	1	O
MSYNC_IN	DRAM DLL synchronization input	All zeros	1	I
MSYNC_OUT	DRAM DLL synchronization output	All ones	1	O
MDVAL	Data valid debug signal output	All zeros	1	O
MSRCID[0:4]	Source port debug signal	All zeros	5	O

Table 9-2 shows memory address signal mappings.

Table 9-2. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
		SDRAM 168-Pin DIMM
msb	MA14	
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10(AP)
	MA9	A9
	MA8	A8
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4
	MA3	A3
	MA2	A2
	MA1	A1
	MA0	A0
	MBA1	BA1
lsb	MBA0	BA0

9.2.2 Detailed Signal Descriptions

The following subsections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

9.2.2.1 Memory Interface Signals

Table 9-3 contains the detailed descriptions of the memory interface signals of the DDR controller.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:63]	I/O	Data bus. The data bus (MDQ[0:63]) consists of 64 signals that are both input and output signals on the DDR memory controller.	
	O	As outputs for the bi-directional 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—Indicates that there is no READ or WRITE command in progress, as data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bi-directional 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 DIMM drives data during a READ transaction. High-impedance—Indicates that there is no READ or WRITE command in progress, as data is not being driven by the memory controller or the DRAM.
MDQS[0:8]	I/O	Data strobes. The nine bidirectional data strobes serve as inputs with read data and as outputs with write data and they comprise the bi-directional data strobes.	
	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/received and driven low when negative capture data is transmitted/received. In addition, this is centered inside of the data eye for writes and it is driven coincident with the data eye for reads. This is treated as a clock and data is valid when this signal toggles. See Table 9-26 for byte lane assignments.
		Timing	Assertion/Negation—If a WRITE command is registered at clock edge n , data strobes at the DRAM assert centered inside of the data eye on clock edge $n + 1$.
	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 the latching of data.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted/received and driven low when negative capture data is transmitted/received. In addition, this is centered inside of the data eye for writes and it is driven coincident with the data eye for reads. This is treated as a clock and data is valid when this signal toggles. See Table 9-26 for byte lane assignments.
		Timing	Assertion/Negation—If a READ command is registered at clock edge n , and the latency is programmed in <code>TIMING_CFG_1</code> to be m clocks, data strobes at the DRAM assert coincident with the data on clock edge $n + m$.

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
MECC[0:7]	I/O	Error checking and correcting. MECC[0:7] comprise the input and output signals for the DDR memory controller's bidirectional ECC bus.
	O	As outputs, the ECC signals represent the state of ECC driven by the DDR controller on writes.
		State Meaning Asserted/Negated—Represent the state of ECC being driven by the DDR SDRAMs 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—Represent the state of ECC being driven by the DDR SDRAMs on writes.
		Timing Assertion/Negation—Same timing as MDQ High-impedance—Same timing as MDQ
MA[0:14]	O	Address bus. These are the output signals of the memory controller for the address to the DRAM. The MA[0:14] signals carry 15 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA[0] is the least significant bit of the address output from the memory controller.
		State Meaning Asserted/Negated—Represents the address value being driven by the DDR memory controller. Contains different portions of the address depending on the size of memory in use, and the current DRAM command being issued by the memory controller. See Table 9-28 for a complete description of the mapping of these signals in all cases.
		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 $\overline{MCSx}$ is negated). High-impedance—When the memory controller is disabled
MBA[0:1]	O	Logical bank address. The MBA[0:1] signals are outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four 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. MBA[0] 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. See Table 9-28," for a complete description of the mapping of these signals in all cases.
		Timing Assertion/Negation—Same timing as MAX. High-impedance—Same timing as MAX

Table 9-3. Memory Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{MCAS}}$	O	Column address strobe. The DDR controller column address strobe ($\overline{\text{MCAS}}$) is an 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 9-29 for more information on the states required on the $\overline{\text{MCAS}}$ signal for various other SDRAM commands. Negated—The column address is not guaranteed to be valid.
		Timing Assertion/Negation—The timing on $\overline{\text{MCAS}}$ assertion and negation is directed by the values in the TIMING_CFG_1 register described in Section 9.3.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High-impedance— $\overline{\text{MCAS}}$ is always driven unless the memory controller is disabled.
$\overline{\text{MRAS}}$	O	Row address strobe. The DDR controller row address strobe ($\overline{\text{MRAS}}$) is an active-low SDRAM address multiplexing signal. This signal is asserted for activate commands. In addition, this signal is 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 9-29 for more information on the states required on the $\overline{\text{MRAS}}$ signal for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		Timing Assertion/Negation—The timing on $\overline{\text{MRAS}}$ assertion and negation is directed by the values in the TIMING_CFG_1 register described in Section 9.3.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High-impedance— $\overline{\text{MRAS}}$ is always driven unless the memory controller is disabled.
$\overline{\text{MCS}}[0:3]$	O	Chip select. The $\overline{\text{MCS}}[0:3]$ are output signals used as memory chip selects. Each of these signals comprises one of the four chip selects supported by the memory controller.
		State Meaning Asserted—Selects a physical SDRAM bank to perform a memory operation as configured in the chip select configuration registers described in Section 9.3.1.1, "Chip Select Memory Bounds (CSn_BNDS)," and Section 9.3.1.2, "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—These signals are asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_1. High-impedance—These signals are always driven unless the memory controller is disabled.
$\overline{\text{MWE}}$	O	Write enable. The DDR controller write enable ($\overline{\text{MWE}}$) is an output signal 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 9-29 for more information on the states required on the $\overline{\text{MWE}}$ signal for various other SDRAM commands. Negated—Indicates a memory read operation.
		Timing Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. This signal is 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
MDM[0:8]	O	DDR SDRAM data output mask. The nine DDR SDRAM data output mask (MDM[0:8]) signals are outputs on the memory controller that mask unwanted bytes of data transferred during a burst 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 data byte, and MDM7 corresponds to the least significant data byte. MDM8 corresponds to the ECC byte. See Table 9-27 for the byte lane encodings.
		State Meaning Asserted—Prevents writing to DDR SDRAM. These signals are 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—These signals are always driven unless the memory controller is disabled.

9.2.2.2 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], MCK[0:5]	O	DRAM clock outputs and their complements. See Section 9.4.4.1, “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—These signals should be synchronized at the SDRAM with the internal clocks of the MPC8540. This is done with the DLL.
MCKE	O	Clock Enable. This is the output signal of the memory controller to be used as the clock enable to the SDRAM. The clock enable signal (MCKE) can be negated to stop clocking the DDR SDRAM. While this results in system power savings, the user is cautioned to disable SDRAM clocking only when there are no transactions on the interface.
		State Meaning Asserted—Clocking to the SDRAM is enabled. Negated—Clocking to the SDRAM is disabled and the SDRAM should ignore the signal transitions on MCK or $\overline{\text{MCK}}$. MCK/ $\overline{\text{MCK}}$ are don't cares when MCKE is negated.
		Timing Assertion/Negation—Similar timing to MAX. High-impedance—These signals are always driven unless the memory controller is disabled.
MSYNC_IN	I	DRAM DLL synchronization input. This is the input from the feedback loop that the DLL uses to synchronize the SDRAM clocks with the internal clocks of the MPC8540. See Section 9.4.4.1, “Clock Distribution.”
		State Meaning Asserted/Negated—This signal toggles at the same frequency as the applied SDRAM clocks (MCKx), but it will not be in phase with MCKx.
		Timing Assertion/Negation—This is used for locking the DLL.

Table 9-4. Clock Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
MSYNC_OUT	O	DRAM DLL synchronization output. This is the output for the DLL feedback loop. See Section 9.4.4.1, "Clock Distribution."
		State Meaning Asserted/Negated—See MSYNC_IN description

9.2.2.3 Debug Signals

Table 9-5 contains the detailed descriptions of the debug signals of the DDR controller.

Table 9-5. Debug Signals—Detailed Signal Descriptions

Signal	I/O	Description
MSRCID[0:4]	O	Source Port Debug. These are debug signals and are valid when a read or write transaction is issued to the SDRAM. The value of these signals can be decoded to determine which source port the transaction originated from. See Chapter 19, "Debug Features and Watchpoint Facility," for more information.
		State Meaning Asserted/Negated—Represent the source of the command currently being driven by the DDR memory controller.
		Timing Assertion/Negation—Similar timing to MA.
MDVAL	O	Data Valid Debug. This is a debug signal. This signal asserts when read or write data is valid on the pins. See Chapter 19, "Debug Features and Watchpoint Facility" for more information.
		State Meaning Asserted/Negated—Indicates that data is valid on the data bus.
		Timing Assertion/Negation— This is a debug signal that asserts when read or write data is valid. For read data, it has similar timings to MA. For write data, it has similar timings to the output MDQ.

9.3 Memory Map/Register Definition

Table 9-6 shows the register memory map for the DDR memory controller.

Table 9-6. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
0x0_2000	Chip Select 0 Memory Bounds (CS0_BNDS)	R/W	0x0000_0000	9.3.1.1/9-11
0x0_2008	Chip Select 1 Memory Bounds (CS1_BNDS)			
0x0_2010	Chip Select 2 Memory Bounds (CS2_BNDS)			
0x0_2018	Chip Select 3 Memory Bounds (CS3_BNDS)			
0x0_2080	Chip Select 0 Configuration (CS0_CONFIG)	R/W	0x0000_0000	9.3.1.2/9-12
0x0_2084	Chip Select 1 Configuration (CS1_CONFIG)			
0x0_2088	Chip Select 2 Configuration (CS2_CONFIG)			
0x0_208C	Chip Select 3 Configuration (CS3_CONFIG)			

Table 9-6. DDR Memory Controller Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_2108	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)	R/W	0x0000_0000	9.3.1.3/9-13
0x0_210C	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)	R/W	0x0000_0000	9.3.1.4/9-16
0x0_2110	DDR SDRAM Control Configuration (DDR_SDRAM_CFG)	R/W	0x0200_0000	9.3.1.5/9-17
0x0_2118	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)	R/W	0x0000_0000	9.3.1.6/9-18
0x0_2124	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)	R/W	0x0000_0000	9.3.1.7/9-19
0x0_2E00	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	R/W	0x0000_0000	9.3.1.8/9-20
0x0_2E04	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	R/W	0x0000_0000	9.3.1.9/9-21
0x0_2E08	Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT)	R/W	0x0000_0000	9.3.1.10/9-21
0x0_2E20	Memory Data Path Read Capture High (CAPTURE_DATA_HI)	R/W	0x0000_0000	9.3.1.11/9-22
0x0_2E24	Memory Data Path Read Capture Low (CAPTURE_DATA_LO)	R/W	0x0000_0000	9.3.1.12/9-23
0x0_2E28	Memory Data Path Read Capture ECC (CAPTURE_ECC)	R/W	0x0000_0000	9.3.1.13/9-23
0x0_2E40	Memory Error Detect (ERR_DETECT)	Special	0x0000_0000	9.3.1.14/9-24
0x0_2E44	Memory Error Disable (ERR_DISABLE)	R/W	0x0000_0000	9.3.1.15/9-25
0x0_2E48	Memory Error Interrupt Enable (ERR_INT_EN)	R/W	0x0000_0000	9.3.1.16/9-25
0x0_2E4C	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)	R/W	0x0000_0000	9.3.1.17/9-26
0x0_2E50	Memory Error Address Capture (CAPTURE_ADDRESS)	R/W	0x0000_0000	9.3.1.18/9-28
0x0_2E54	Memory Error Extended Address Capture (CAPTURE_EXT_ADDRESS)	R/W	0x0000_0000	9.3.1.19/9-34
0x0_2E58	Single-Bit ECC Memory Error Management (ERR_SBE)	R/W	0x0000_0000	9.3.1.19/9-29

9.3.1 Register Descriptions

This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

9.3.1.1 Chip Select Memory Bounds (CSn_BNDS)

The chip select bounds registers (CSn_BNDS) shown in Figure 9-2 define the starting address of the memory space that corresponds to the individual chip selects.

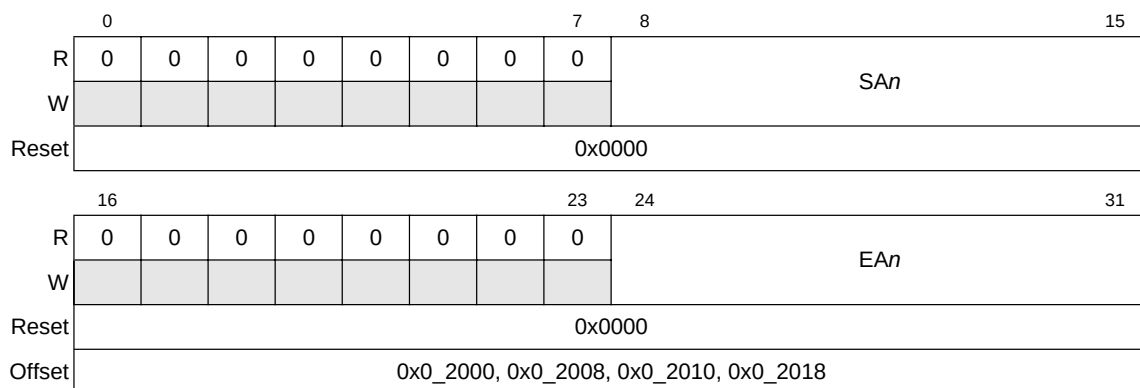
**Figure 9-2. Chip Select Bounds Registers (*CSn_BNDS*)**

Table 9-7 describes the *CSn_BNDS* register fields.

Table 9-7. *CSn_BNDS* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	<i>SAn</i>	Starting address for chip select (bank) <i>n</i> . This value is compared against the most significant 8 bits of the address.
16–23	—	Reserved
24–31	<i>EAn</i>	Ending address for chip select (bank) <i>n</i> . This value is compared against the most significant 8 bits of the address.

9.3.1.2 Chip Select Configuration (*CSn_CONFIG*)

The chip select configuration (*CSn_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.

The chip select configuration registers should be loaded with the correct number of row and column bits for each SDRAM. Because address multiplexing is established by *CSn_CONFIG*[*ROW_BITS_CS_n*] and *CSn_CONFIG*[*COL_BITS_CS_n*] the user should take great care to set these values correctly.

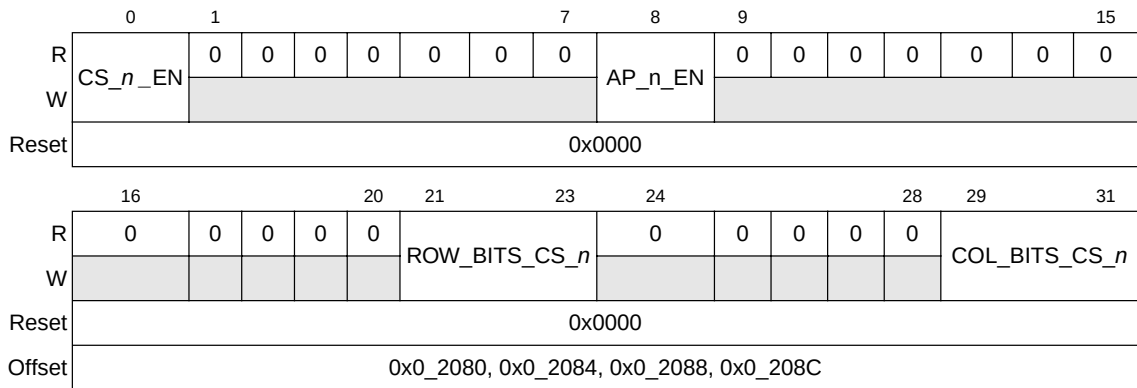
**Figure 9-3. Chip Select Configuration Register (CS_n_CONFIG)**

Table 9-8 describes the CS_n_CONFIG register fields.

Table 9-8. CS_n_CONFIG Field Descriptions

Bits	Name	Description
0	CS _n _EN	Chip select <i>n</i> enable 0 Chip select <i>n</i> is not active 1 Chip select <i>n</i> is active and assumes the state set in CS _n _BNDS.
1–7	—	Reserved
8	AP _n _EN	Chip select <i>n</i> auto precharge enable 0 Chip select <i>n</i> will only be auto precharged if global auto precharge mode is enabled ('BSTOPRE' = 0). 1 Chip select <i>n</i> will always issue an auto precharge for read and write transactions.
9–20	—	Reserved
21–23	ROW_BITS_CS _n	Number of row bits for SDRAM on chip select <i>n</i> 000 12 row bits 001 13 row bits 010 14 row bits 011 - 111 Reserved
24–28	—	Reserved
29–31	COL_BITS_CS _n	Number of column bits for SDRAM on chip select <i>n</i> 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100 - 111 Reserved

9.3.1.3 DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

DDR SDRAM timing configuration register 1, shown in Figure 9-4, sets the number of clock cycles between various SDRAM control commands.

Memory Map/Register Definition

	0	1	3	4	5	7	8	9	11	12	13	15		
R	0	PRETOACT		0	ACTTOPRE		0	ACTTORW		0	CASLAT			
W														
Reset	0x0000													
	16	19	20	21	22	23	24	25	27	28	29	30	31	
R	REFREC			0	0	WRREC		0	ACTTOACT		0	0	WRTORD	
W														
Reset	0x0000													
Offset	0x0_2108													

Figure 9-4. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

Table 9-9 describes the TIMING_CFG_1 fields.

Table 9-9. TIMING_CFG_1 Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	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. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
4	—	Reserved, should be cleared.
5–7	ACTTOPRE	Activate to precharge interval (t_{ras}). Determines the number of clock cycles from an activate command until a precharge command is allowed. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
8	—	Reserved, should be cleared.

Table 9-9. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
9–11	ACTTORW	Activate to read/write interval for SDRAM (t_{rd}). Controls the number of clock cycles from an activate command until a read or write command is allowed. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12	—	Reserved, should be cleared.
13–15	CASLAT	MCAS latency from read command. Specifies read latency. The READ latency is the delay, in clock cycles, between the 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, the data is available nominally coincident with clock edge $n + m$. This value must also be programmed into the DDR SDRAM at initialization (via the DDR SDRAM mode register described in Section 9.3.1.6, "DDR SDRAM Mode Configuration (DDR_SDRAM_MODE).") 000 Reserved 001 1 clock 010 1.5 clocks 011 2 clocks 100 2.5 clocks 101 3 clocks 110 3.5 clocks 111 4 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. Refresh recovery time is equal to eight plus the value entered in this field. 0000 Reserved 0001 9 clocks 0010 10 clocks 0011 11 clocks . . . 1110 22 clocks 1111 23 clocks
20–21	—	Reserved, should be cleared.
22–23	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. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
24	—	Reserved, should be cleared.

Table 9-9. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
25–27	ACTTOACT	Activate to activate interval (t_{rrd}). Specifies the number of clock cycles from an activate command until another activate command is allowed for a different logical bank within the same physical bank (chip select). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 Reserved 110 Reserved 111 Reserved
28–29	—	Reserved, should be cleared.
30–31	WRTORD	Last write data pair to read command issue interval (t_{wtr}). Determines the number of clock cycles between the last write data pair and the subsequent read command to the same physical bank. 00 Reserved 01 1 clock 10 2 clocks 11 Reserved

9.3.1.4 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 9-5, sets the amount of clock delay to data for writes.

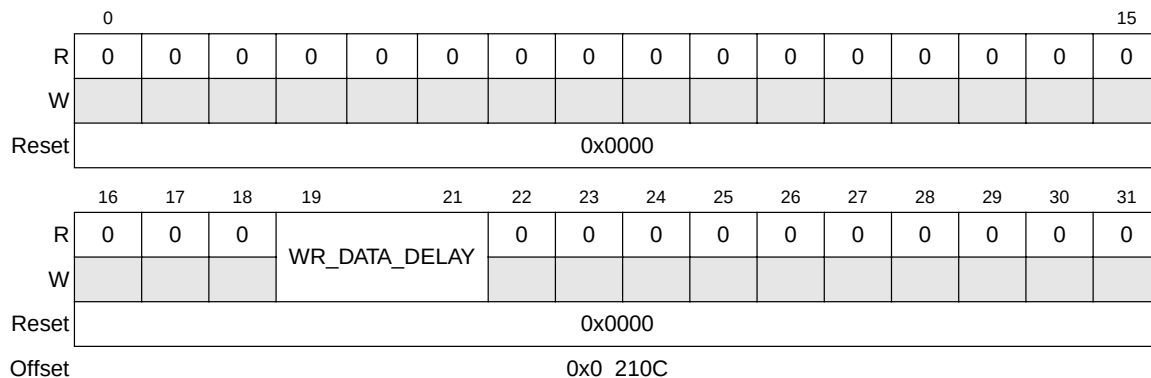
**Figure 9-5. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)**

Table 9-10 describes the TIMING_CFG_2 fields.

Table 9-10. TIMING_CFG_2 Register Field Descriptions

Bits	Name	Description
0–18	—	Reserved
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. 000 0 clock delay 001 2/8 clock delay (recommended) 010 4/8 clock delay 011 6/8 clock delay 100 1 clock delay 101 Reserved . . . 111 Reserved
22–31	—	Reserved

9.3.1.5 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 9-6, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMS, and dynamic power management.

	0	1	2	3	4	5	6	7	8	9	10	11		15		
R	MEM_EN	SREN	ECC_EN	RD_EN	0	0	SDRAM_TYPE	0	0		DYN_PWR	0	0	0	0	0
W																
Reset	0x0200															

	16															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															
Offset	0x0_2110															

Figure 9-6. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

Table 9-11 describes the DDR_SDRAM_CFG fields.

Table 9-11. 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 or soft-stop) 0 SDRAM self refresh is disabled during sleep or soft-stop. Whenever self refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep or soft-stop. 1 SDRAM self refresh is enabled during sleep or soft-stop
2	ECC_EN	ECC enable 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
4–5	—	Reserved
6–7	SDRAM_TYPE	Type of SDRAM device to be used 00 Reserved 01 Reserved 10 DDR SDRAM 11 Reserved
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.
11–31	—	Reserved

9.3.1.6 DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register, shown in Figure 9-7, sets the values loaded into the DDR's mode registers.

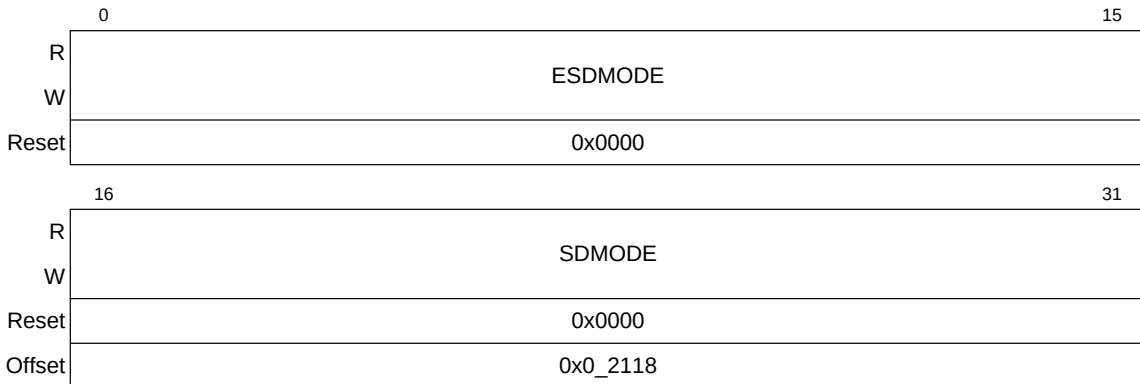


Figure 9-7. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 9-12 describes the DDR_SDRAM_MODE fields.

Table 9-12. 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 of legal values and the meaning of each is specified by the DDR SDRAM manufacturer. Note that when this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the least significant bit of ESDMODE, which, in the big endian convention shown in Figure 9-7 corresponds to ESDMODE[15]. The most significant bit of the SDRAM extended mode register value must be stored at ESDMODE[0].
16–31	SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values and the meaning of each is specified by the DDR SDRAM manufacturer. Note that when this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the least significant bit of SDMODE, which, in the big endian convention shown in Figure 9-7 corresponds to SDMODE[15]. The most significant bit of the SDRAM extended mode register value must be stored at ESDMODE[0]. Because the memory controller forces SDMODE[3:8] to certain values depending upon the state of the initialization sequence (for resetting the SDRAM's DLL) the corresponding bits of this field will be ignored by the memory controller.

9.3.1.7 DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register, shown in Figure 9-8, 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.

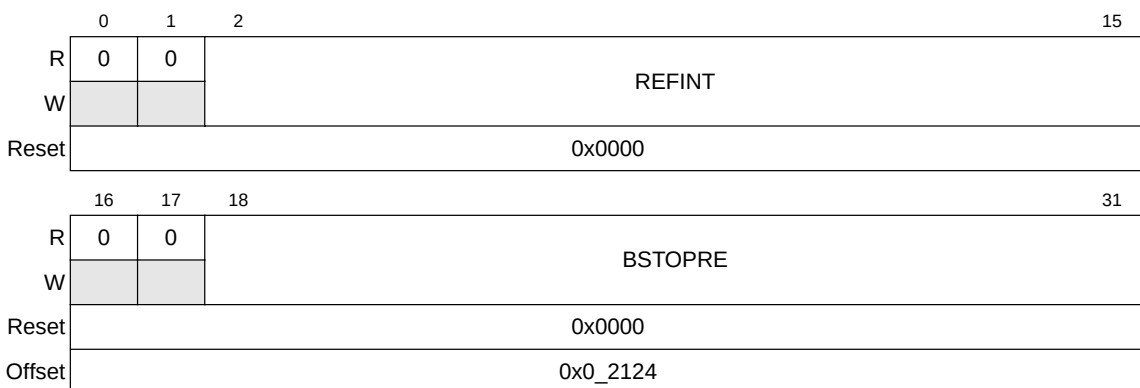
**Figure 9-8. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)**

Table 9-13 describes the DDR_SDRAM_INTERVAL fields.

Table 9-13. DDR_SDRAM_INTERVAL Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–15	REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. One row is 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.
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 cleared, the DDR memory controller uses auto precharge read and write commands rather than operating in page mode. This is referred to as global auto precharge mode.

9.3.1.8 Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)

The memory data path error injection mask high register is shown in Figure 9-9.

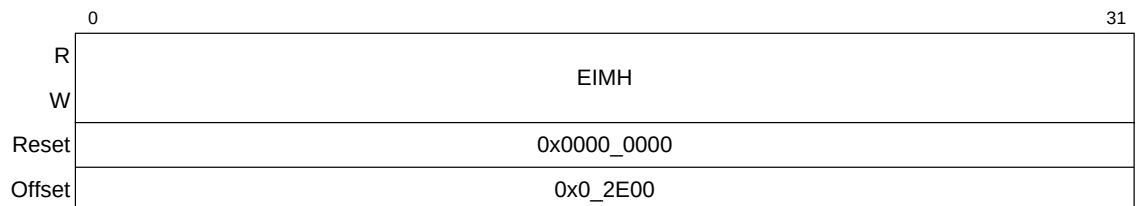
**Figure 9-9. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)**

Table 9-14 describes the DATA_ERR_INJECT_HI fields.

Table 9-14. DATA_ERR_INJECT_HI Field Descriptions

Bits	Name	Description
0–31	EIMH	Error injection mask high data path. Used to test ECC, this field allows the user to force 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.

9.3.1.9 Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)

The memory data path error injection mask low register is shown in Figure 9-10.

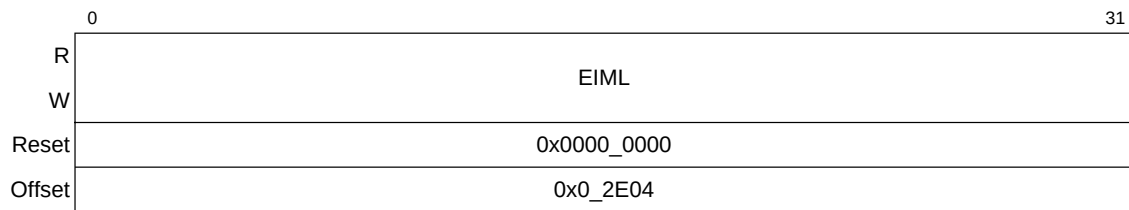
**Figure 9-10. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)**

Table 9-15 describes the DATA_ERR_INJECT_LO fields.

Table 9-15. DATA_ERR_INJECT_LO Field Descriptions

Bits	Name	Description
0–31	EIML	Error injection mask low data path. Used to test ECC, this field allows the user to force 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.

9.3.1.10 Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT)

The memory data path error injection mask ECC register, shown in Figure 9-11, sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte.

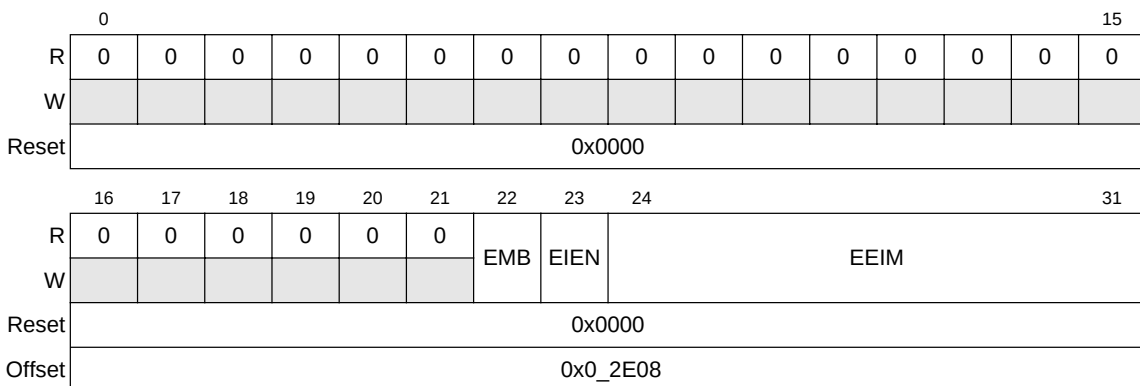


Figure 9-11. Memory Data Path Error Injection Mask ECC Register (ECC_ERR_INJECT)

Table 9-16 describes the ECC_ERR_INJECT fields.

Table 9-16. ECC_ERR_INJECT Field Descriptions

Bits	Name	Description
0–21	—	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.
24–31	EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

9.3.1.11 Memory Data Path Read Capture High (CAPTURE_DATA_HI)

The memory data path read capture high register, shown in Figure 9-12, stores the high word of the read data path during error capture.

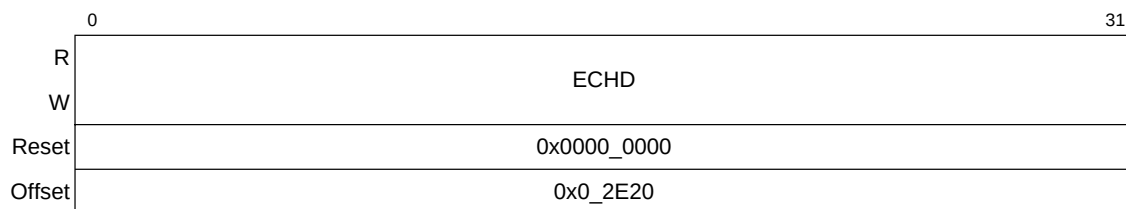


Figure 9-12. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 9-17 describes the CAPTURE_DATA_HI fields.

Table 9-17. 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.

9.3.1.12 Memory Data Path Read Capture Low (CAPTURE_DATA_LO)

The memory data path read capture low register, shown in Figure 9-13, stores the low word of the read data path during error capture.

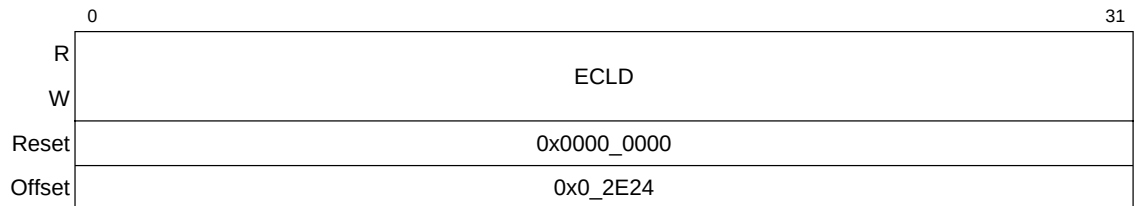


Figure 9-13. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 9-18 describes the CAPTURE_DATA_LO fields.

Table 9-18. 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.

9.3.1.13 Memory Data Path Read Capture ECC (CAPTURE_ECC)

The memory data path read capture ECC register, shown in Figure 9-14, stores the ECC syndrome bits that were on the data bus when an error was detected.



Figure 9-14. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-19 describes the CAPTURE_ECC fields.

Table 9-19. CAPTURE_ECC Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected.

9.3.1.14 Memory Error Detect (ERR_DETECT)

The memory error detect register, shown in Figure 9-15, 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 high value to the bit. System software can determine the type of memory error by examining the contents of this register. If an error has been disabled with ERR_DISABLE, the corresponding error will never be detected and captured in the ERR_DETECT register.

	0	1													15	
R	MME	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0x0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MBE	SBE	0	MSE
W																
Reset	0x0000															
Offset	0x0_2E40															

Figure 9-15. Memory Error Detect Register (ERR_DETECT)

Table 9-20 describes the ERR_DETECT fields.

Table 9-20. 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
1–27	—	Reserved
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 programmed threshold set in ERR_SBE[SBET] 1 The number of single-bit ECC errors detected crossed the programmed 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

9.3.1.15 Memory Error Disable (ERR_DISABLE)

The memory error disable register, shown in Figure 9-16, allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

	0															15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0x0000																

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MBED	SBED	0	MSED
W																
Reset	0x0000															
Offset	0x0_2E44															

Figure 9-16. Memory Error Disable Register (ERR_DISABLE)

Table 9-21 describes the ERR_DISABLE fields.

Table 9-21. ERR_DISABLE Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG:ECC_EN is set and reported if ERR_INT_EN[MBEE] is set 1 Multiple-bit ECC errors are not detected or reported
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled 1 Single-bit ECC errors are disabled
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled 1 Memory select errors are disabled

9.3.1.16 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 9-17, enables ECC single-bit or multiple-bit interrupts or memory select error interrupts. When one of these enabled interrupt conditions occur, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

	0														15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0x0000														

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	MBEE	SBEE	0	MSEE
W																
Reset	0x0000															
Offset	0x0_2E48															

Figure 9-17. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 9-22 describes the ERR_INT_EN fields.

Table 9-22. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable 0 Multiple-bit ECC errors cannot generate interrupts 1 Multiple-bit ECC errors generate interrupts
29	SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts 1 Single-bit ECC errors generate interrupts
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts 1 Memory select errors generate interrupts

9.3.1.17 Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)

The memory error attributes capture register, shown in Figure 9-18, sets attributes for errors including type, size, source, and others.

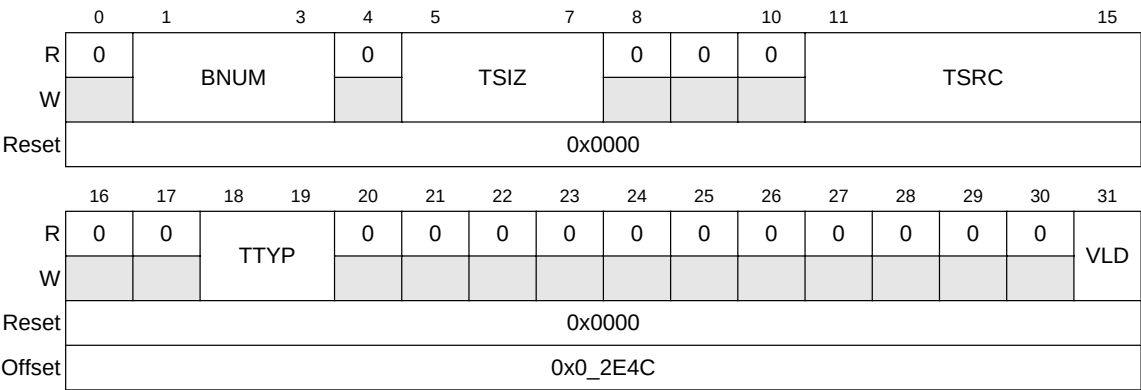


Figure 9-18. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 9-23 describes the CAPTURE_ATTRIBUTES fields.

Table 9-23. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number for the error. Captures the data beat 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.
8–10	—	Reserved

Table 9-23. CAPTURE_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
11–15	TSRC	Transaction source for the error 00000 — PCI 00001 — reserved 00010 — reserved 00011 — reserved 00100 — Local bus 00101 — reserved 00110 — reserved 00111 — reserved 01000 — Configuration space 01001 — reserved 01010 — Boot sequencer 01011 — reserved 01100 — RapidIO 01101 — reserved 01110 — reserved 01111 — reserved 10000 — Processor (inst) 10001 — Processor (data) 10010 — reserved 10011 — reserved 10100 — reserved 10101 — DMA 10110 — reserved 10111 — SAP 11000 — Ethernet 1 11001 — Ethernet 2 11010 — Ethernet 3 11011 — reserved 11100 — RapidIO message units 11101 — RapidIO doorbell units 11110 — RapidIO port—write units 11111 — reserved
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	This bit is set once valid information has been captured in the error capture registers.

9.3.1.18 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 9-19, holds the 32 least significant bits of a transaction when a DDR ECC error is detected.

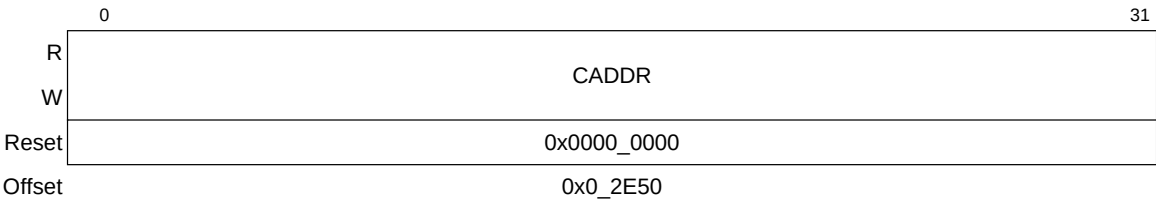


Figure 9-19. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 9-24 describes the CAPTURE_ADDRESS fields.

Table 9-24. CAPTURE_ADDRESS Field Descriptions

Bits	Name	Description
0–31	CADDR	Captured address. Captures the least significant 32-bits of the transaction address when an error is detected.

9.3.1.19 Single-Bit ECC Memory Error Management (ERR_SBE)

The single-bit ECC memory error management register, shown in Figure 9-20, stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. After the counter field has reached the threshold, it wraps back to the reset value (0). If it needs to be cleared again after software has managed the error, software will need to clear the counter.

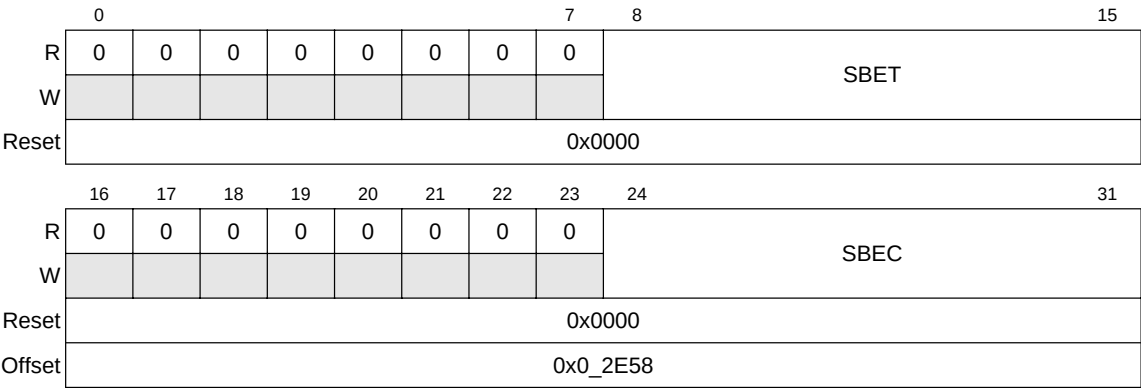


Figure 9-20. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 9-25 describes the ERR_SBE fields.

Table 9-25. ERR_SBE Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SBET	Single-bit error threshold. Establishes the 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 that have been detected and corrected since the last error report. If single-bit error reporting is enabled, then an error is reported and a machine check or critical interrupt is generated when this value equals ERR_SBE[SBET]. This counter value is automatically cleared after the threshold value has been reached.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ACSM	0	0	0	0	0	0	0	DOE	0	0	0	0	0	0	0
W																
Reset	0x0000															
	16															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															
Offset	0xF0C															

9.4 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR SDRAMs (first generation dual data rate). The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

The DDR SDRAM interface supports as many as four physical banks of 64/72 bit-wide memory. A bank size up to 1 Gbyte is supported, providing up to a maximum of 4 Gbytes of DDR main memory. However, 4 Gbytes would span the entire 32-bit address space, and some space must be reserved for boot ROM, configuration registers, and other important addressable locations.

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-bit data bus, detects all double-bit errors within the 64-bit data bus, and detects all errors within a nibble. The controller allows as many as 16 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE]. Pages are replaced using a pseudo-LRU (PLRU) algorithm.

Read and write accesses to the DDR SDRAM are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (2, 4, or 8) in a programmed sequence, (sequential or interleaved). Accesses to closed pages start with the registration of an ACTIVE (activate) 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.

Figure 9-21 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 then loaded into the input staging queue with the decoded information. The lower two entries of the input queue are compared with values in the row open table to determine if the address maps to an open page. If the address from either entry does not map to an open page, an activate command is issued for the entry that did not hit an open page, with the lowest entry having priority over the next lowest. Commands are always issued from the lowest entry in the input queue.

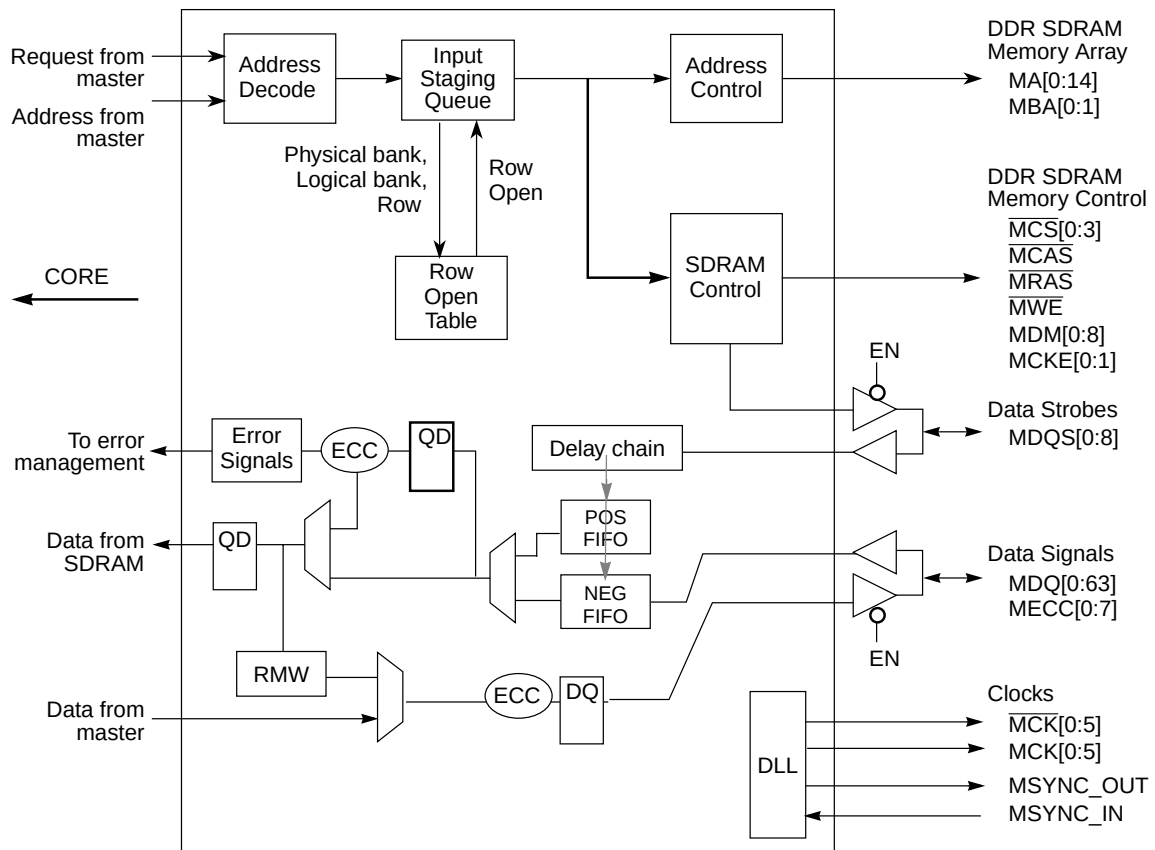


Figure 9-21. DDR Memory Controller Block Diagram

Functional Description

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 is implemented in the controller with delay chains for the data strobe signals during reads and a delay chain on the data multiplexer select during writes.

When ECC is enabled, one 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.

Figure 9-22 shows an example of how the on-chip DLL can be used with the memory controller.

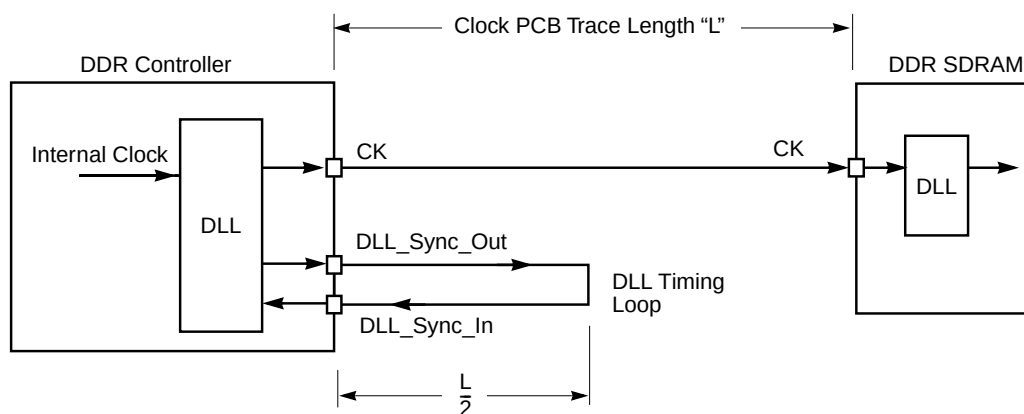


Figure 9-22. Controller DLL Timing Loop

Figure 9-23 shows an example DDR SDRAM configuration with four physical banks and Figure 9-24 shows some typical signal connections.

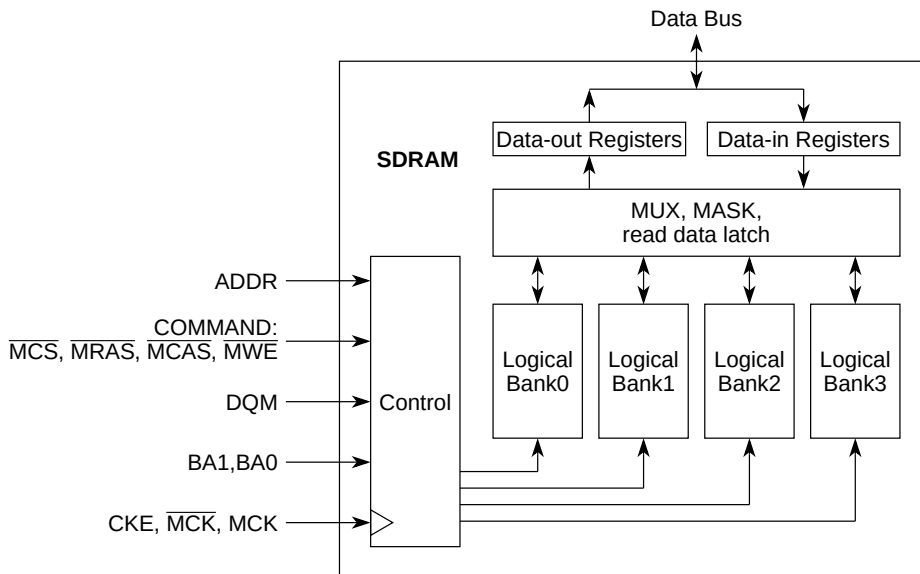


Figure 9-23. Typical Dual Data Rate SDRAM Internal Organization

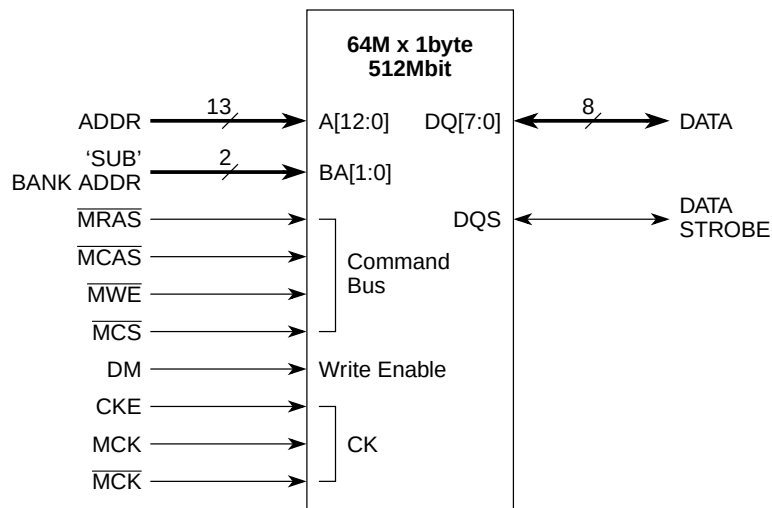
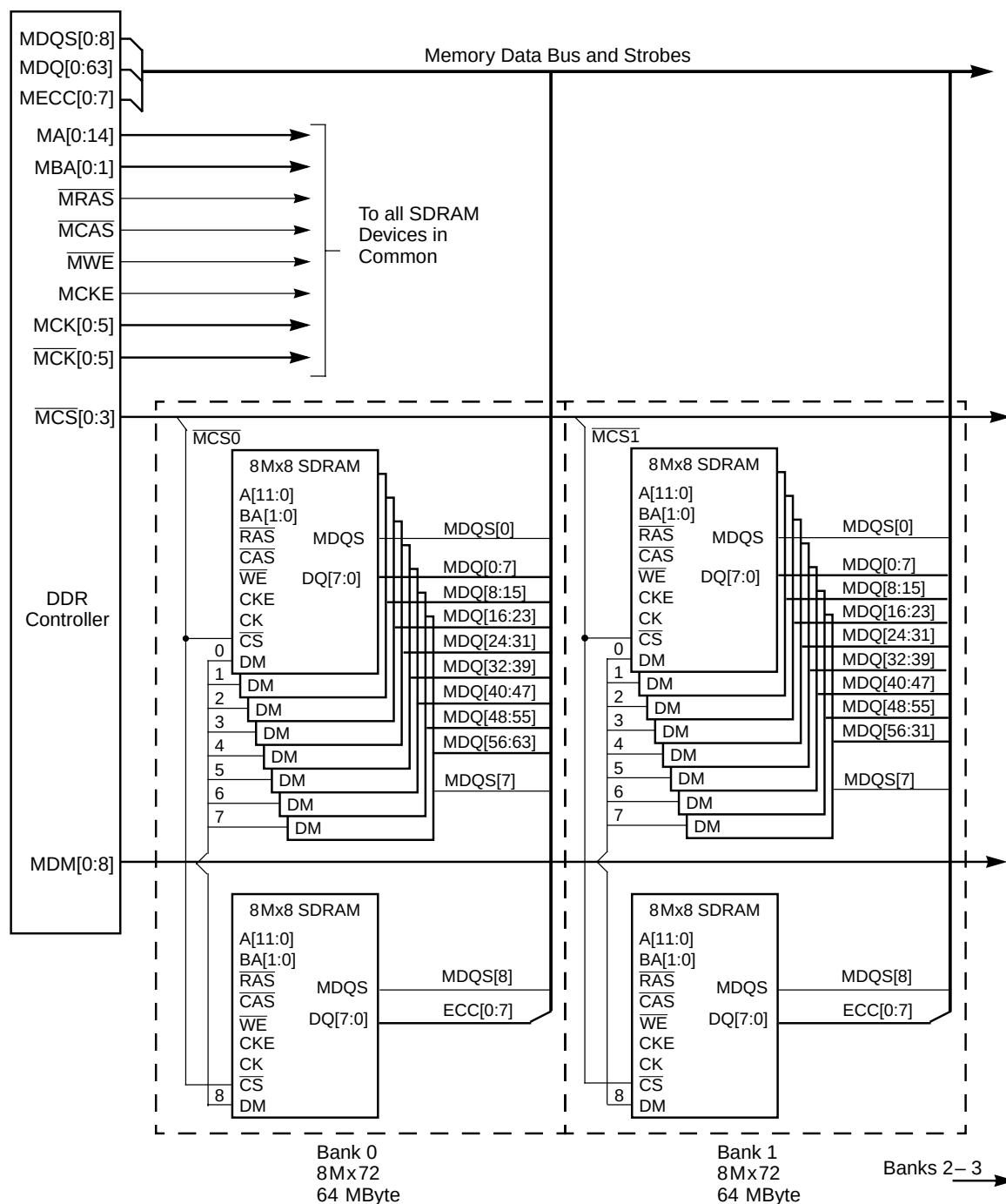


Figure 9-24. Typical DDR SDRAM Interface Signals

Figure 9-25 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M x 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the MPC8540 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 MPC8540 DDR memory controller drives 14 address pins, but in this example the DDR SDRAM devices use only 12 bits.

Functional Description



1. All signals are connected in common (in parallel) except for $\overline{MCS}[0:3]$, $\overline{MCK}[0:8]$, the DQM signals and the data bus signals.
2. Each of the $\overline{MCS}[0:3]$ signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. $\overline{MCK}[0:5]$ may be apportioned among all memory devices. Complementary bus is not shown.

Figure 9-25. Example 256 MByte DDR SDRAM Configuration With ECC

The DDR memory controller handles errors as explained in Section 9.4.12, “Error Management.”

9.4.1 DDR SDRAM Interface Operation

Many different DDR SDRAM configurations are supported by the DDR memory controller. SDRAMs with different sizes can be used in the same system. Fourteen multiplexed address signals and two logical bank select signals provide support for device densities from 64 Mbit to 1 Gbit. 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-bit wide; 72 bits with ECC. The MPC8540 DDR memory controller supports physical bank sizes from 32 Mbytes to 1 Gbytes. The physical banks can be constructed using $\times 8$, or $\times 16$ memory devices. The memory technologies supported are 64 Mbit, 128 Mbit, 256 Mbit, 512 Mbit, and 1 Gbit. 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 sixteen 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 9-26 shows the DDR memory controller’s relationships between data byte lane 0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63].

Table 9-26. 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]

9.4.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes the row and column address bits onto 14 memory address signals and two logical bank select signals, individual physical banks may be implemented with memory devices requiring fewer than 28 address bits. Each physical bank may be individually configured to provide from 12 to 14 row address bits, plus two logical bank-select bits and from eight to eleven column address bits. Table 9-27 describes DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

The largest supported DDR SDRAM is limited to 27 total address bits.

Table 9-27. Supported DDR SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column Bits	64-Bit Bank Size	Four Banks of Memory
64Mb	8Mb x 8	12 x 9	64MB	256 MB
64Mb	4Mb x 16	12 x 8	32MB	128 MB
128Mb	16Mb x 8	12 x 10	128MB	512 MB
128Mb	8Mb x 16	12 x 9	64MB	256 MB
256Mb	32Mb x 8	13 x 10	256MB	1 GB
256Mb	16Mb x 16	13 x 9	128MB	512 MB
512Mb	64Mb x 8	13 x 11	512MB	2 GB
512Mb	32Mb x 16	13 x 10	256MB	1 GB
1Gb	128Mb x 8	14 x 11	1GB	4 GB
1Gb	64Mb x 16	14 x 10	512 MB	2 GB

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 9.4.12, “Error Management.”

If a disabled bank has its starting and ending address defined as overlapping an enabled bank’s address space, there may be system memory corruption in the overlapping address range. Unused memory banks’ starting and ending addresses should be mapped to unused memory space.

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware configures the DDR memory controller to correctly map the size of each bank in memory using the memory-boundary registers. The memory controller uses its bank map to assert the appropriate $\overline{MCSn}$ 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.

9.4.2 DDR SDRAM Address Multiplexing

Table 9-28 shows the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[14:0] use MA[14] as the most-significant bit and MA[0] as the least-significant bit. Also, MA[10] is used as the auto precharge bit for reads and writes, so the column address can never use MA[10].

Table 9-28. DDR SDRAM Address Multiplexing

Row x Col		msb			Address from Core Master																												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	29-31		
14 x 11	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	MCAS																				11	9	8	7	6	5	4	3	2	1	0		
14 x 10	MRAS				13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0		
13 x 11	MRAS				12	11	10	9	8	7	6	5	4	3	2	1	0																
	MBA																		1	0													
	MCAS																				11	9	8	7	6	5	4	3	2	1	0		
13 x 10	MRAS					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0		
13 x 9	MRAS						12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0												
	MCAS																						8	7	6	5	4	3	2	1	0		
12 x 10	MRAS						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		1	0													
	MCAS																					9	8	7	6	5	4	3	2	1	0		
12 x 9	MRAS							11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																			1	0												
	MCAS																						8	7	6	5	4	3	2	1	0		
12 x 8	MRAS								11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																					1	0										
	MCAS																							7	6	5	4	3	2	1	0		

9.4.3 JEDEC Standard DDR SDRAM Interface Commands

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 9-29) 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 will change 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 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.
- Mode register set (for configuration)—Allows setting of DDR SDRAM options. These options are: $\overline{\text{MCAS}}$ latency, 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}, 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. 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.

- 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, all logical banks are in a precharged state.

Table 9-29. 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

9.4.4 SDRAM Interface Timing

The DDR memory controller supports four-beat bursts to SDRAM. For single-beat reads, the DDR memory controller performs a four-beat burst read, but ignores the last three beats. Single-beat writes are performed by masking the last three beats of the four-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 when some of the 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 9-30 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-30. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from an active command to a logical bank until another active command is allowed for any logical bank within that bank. This interval is listed in the AC specifications of the SDRAM.
ACTOPRE	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.
ACTORW	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.
BSTOPRE	The number of clock cycles to maintain a page open after an access. A subsequent access can generate a page hit during this interval. A page hit reloads the BSTOPRE counter. When the interval expires, a precharge is issued to the page as soon as possible.
CASLAT	The READ latency (CASLAT) is the delay, in 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 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.
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. One row is 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.
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.
WRTORD	Last write pair to read command issue for DDR and DDRII. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank.

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_1 and TIMING_CFG_2 registers as described in Section 9.3.1.3, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” and Section 9.3.1.4, “DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)”) 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 9-26 through Figure 9-28 show DDR SDRAM timing for various types of accesses; see Figure 9-26 for a single-beat read operation, Figure 9-27 for a single-beat write operation, and Figure 9-28 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.

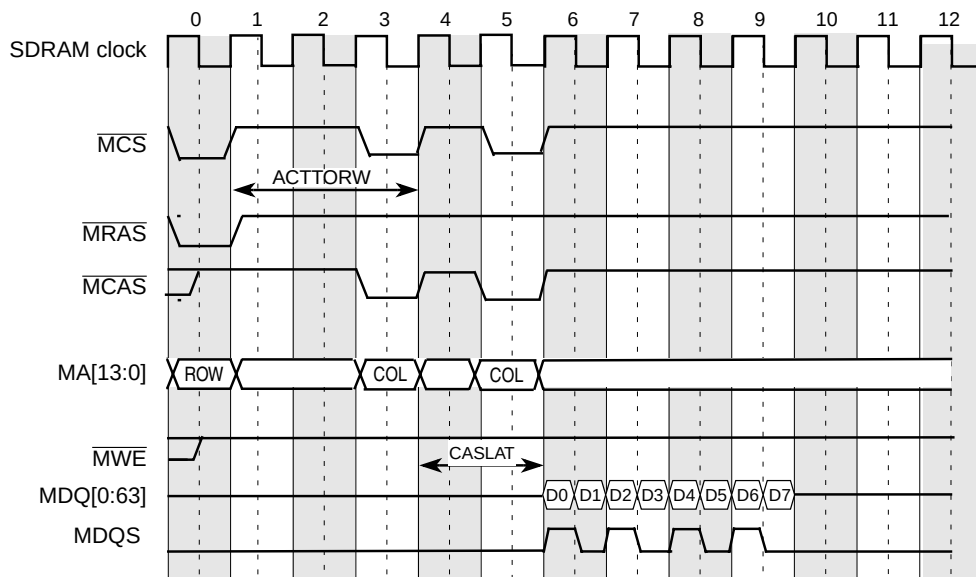


Figure 9-26. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

Functional Description

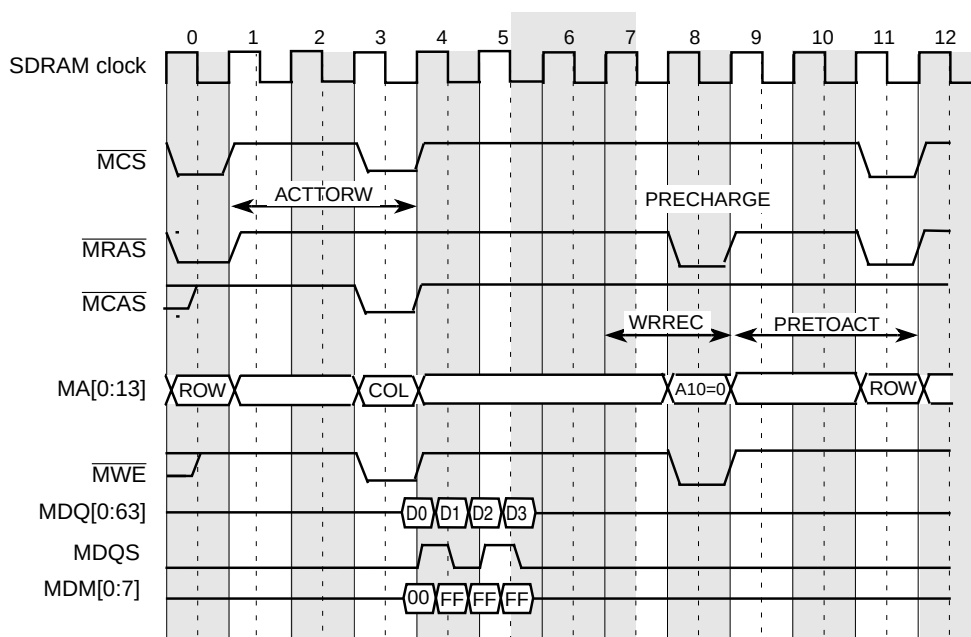


Figure 9-27. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

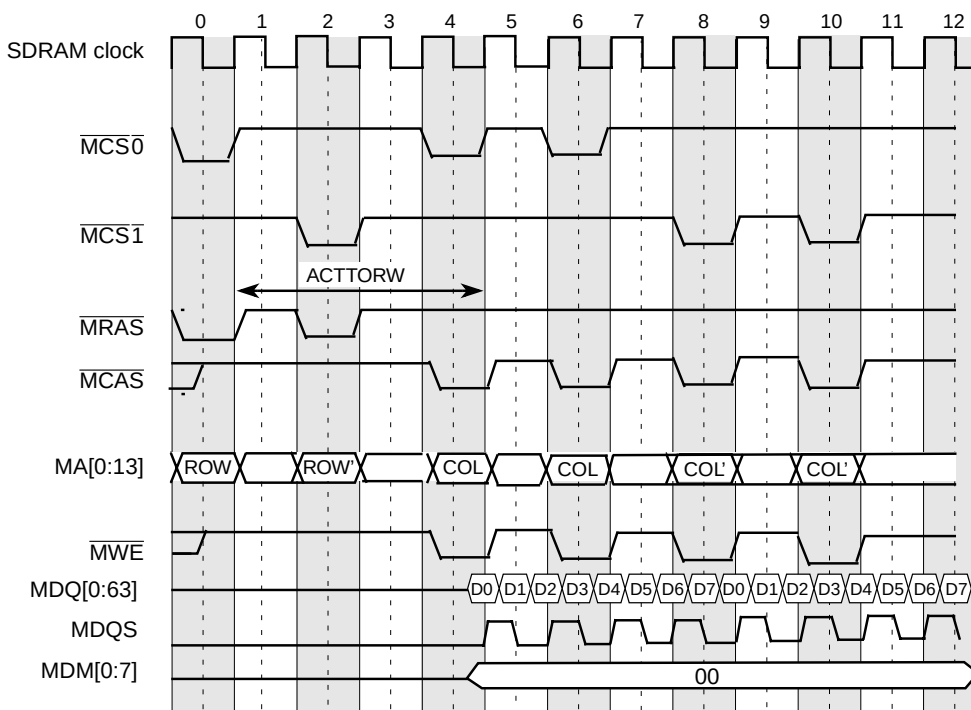


Figure 9-28. DDR SDRAM 8-Beat Burst Write Timing—ACTTORW = 4

9.4.4.1 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.
- The DLL timing loop PCB route should mimic the MCK route as closely as possible, although clock edges can be intentionally advanced or retarded by modifying this loop. With all DDR clocks as nearly identical as possible, the single DLL timing loop can be finely adjusted to advance or retard all of them.
- A 72 bit x 64 MB DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/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 as the DLL timing loop.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

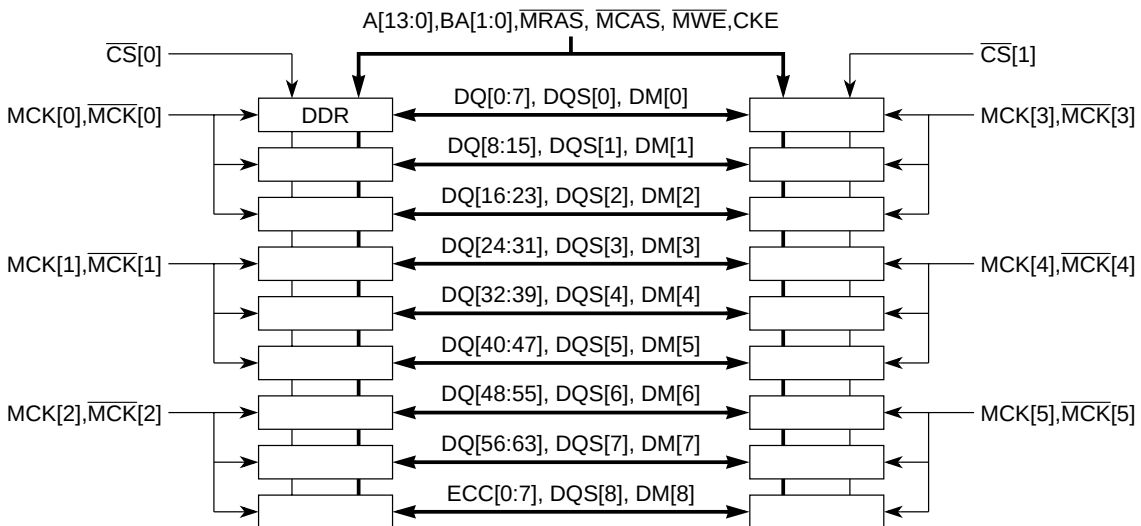


Figure 9-29. DDR SDRAM Clock Distribution Example

9.4.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the extended mode register data `DDR_SDRAM_MODE[ESDMODE]` and the base mode register data `DDR_SDRAM_MODE[SDMODE]` to the SDRAM array by issuing two mode-set commands separated by two SDRAM clock periods. The timing of the mode-set command is shown in Figure 9-30. The first transfer corresponds to the ESDMODE code and the second one corresponds to the SDMODE code. Following commands must wait two SDRAM cycles.

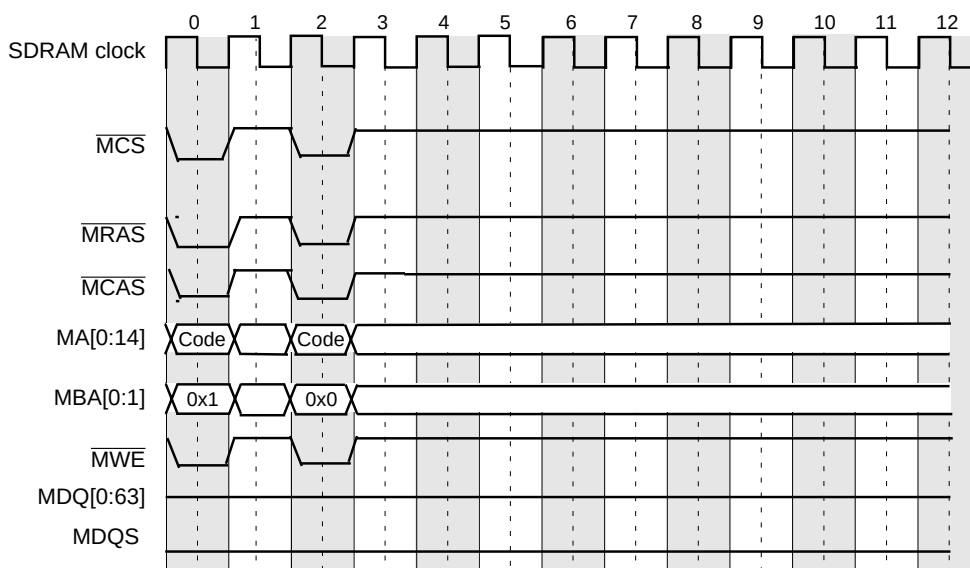


Figure 9-30. DDR SDRAM Mode-Set Command Timing

9.4.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Enabling the registered DIMM mode by 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 one additional SDRAM clock cycle.

Enabling registered DIMM mode has no effect on bus timing for DDR reads. However to compensate for latch delay on the registered DIMMs' control signals, the programmed SDRAM read latency value (`TIMING_CFG_1[CASLAT]`) must be one greater than the value needed for non-registered DIMMs. Figure 9-31 shows the registered DDR SDRAM DIMM single-beat write timing.

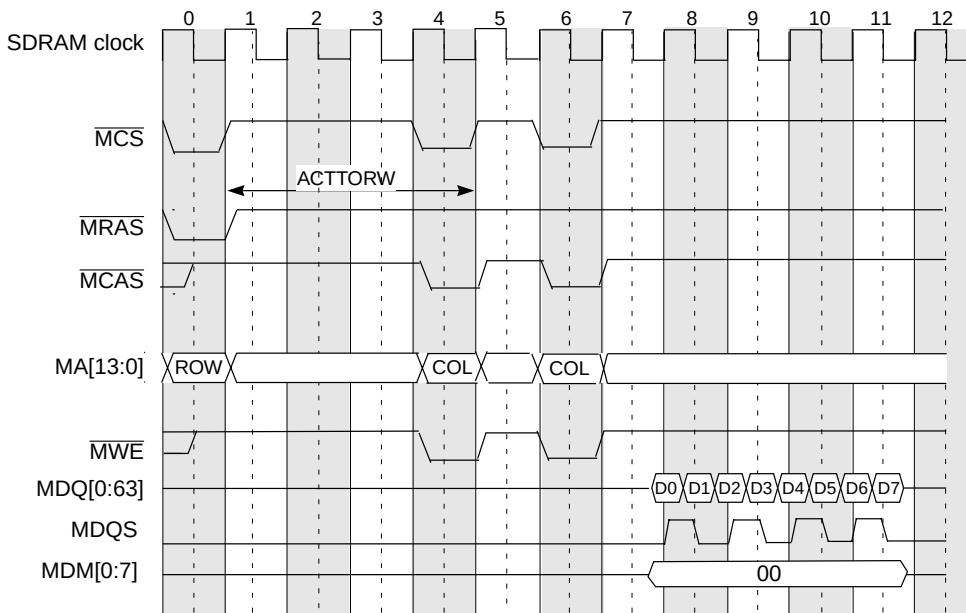


Figure 9-31. Registered DDR SDRAM DIMM Burst Write Timing

9.4.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMMS. TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4th SDRAM clock periods starting with the default value of 1/4th period delay. Figure 9-32 shows the use of the WR_DATA_DELAY parameter.

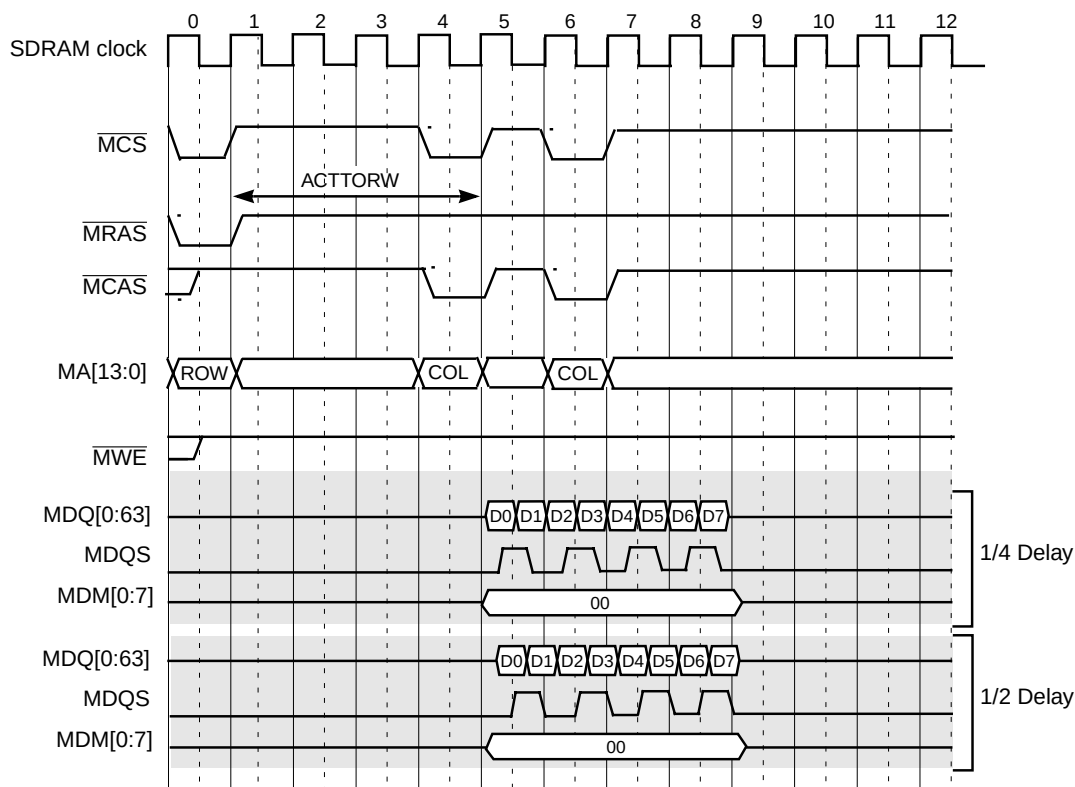


Figure 9-32. Write Timing Adjustments Example

9.4.8 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 value stored in `DDR_SDRAM_INTERVAL[REFINT]`; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state or a soft-stop state. The value in `DDR_SDRAM_INTERVAL[REFINT]` represents the number of memory bus clock cycles between refresh cycles. The value must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the `DDR_SDRAM_INTERVAL[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 `DDR_SDRAM_INTERVAL[REFINT]` be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller performs the following actions:

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 an auto-refresh command to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each of the logical banks in the selected physical bank.

The auto-refresh commands are staggered across the 4 possible banks to reduce the system's instantaneous power requirements. Two sets of auto refresh commands must be issued on consecutive cycles when the memory is fully populated with 2 DIMMs. The initial PRECHARGE-ALL commands are also staggered in two groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. CKE is negated at this time, so it would not be possible to stagger two more refresh commands. For this entire refresh sequence, no cycle optimization occurs for the usual case where fewer than 4 banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC].

9.4.8.1 DDR SDRAM Refresh Timing

The refresh timing for 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 9-33 (TIMING_CFG_1 [REFREC] = 10 in this example).

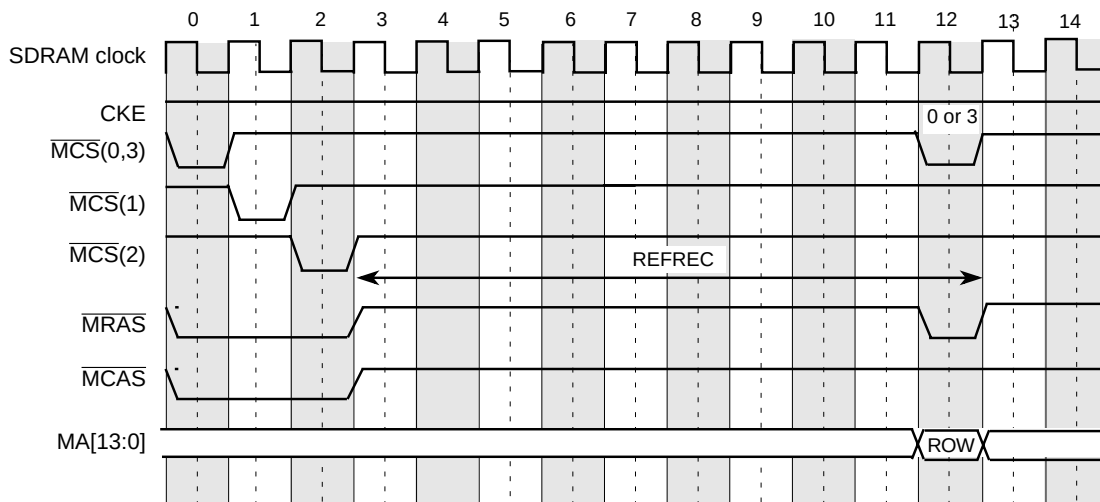


Figure 9-33. DDR SDRAM Bank Staggered Auto Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] at reset. Configuration must be completed before any DDR SDRAM accesses are attempted.

9.4.8.2 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 9-31 summarizes the refresh types available in each power-saving mode.

Table 9-31. 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 allows the user to tightly control the power consumption of the memory system by trading power for performance using DDR_SDRAM_INTERVAL[BSTOPRE]. Powering up the DDR SDRAM when a new memory reference is scheduled causes a one-clock access latency penalty, as shown in Figure 9-34.

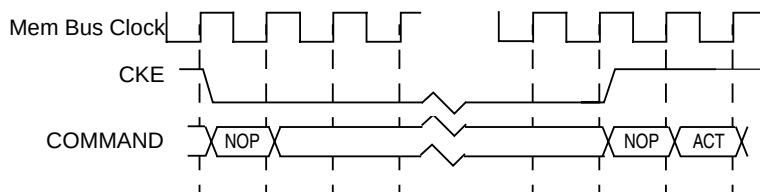


Figure 9-34. DDR SDRAM Power-Down Mode

9.4.8.2.1 Self-Refresh in Sleep Modes

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-35 and Figure 9-36.

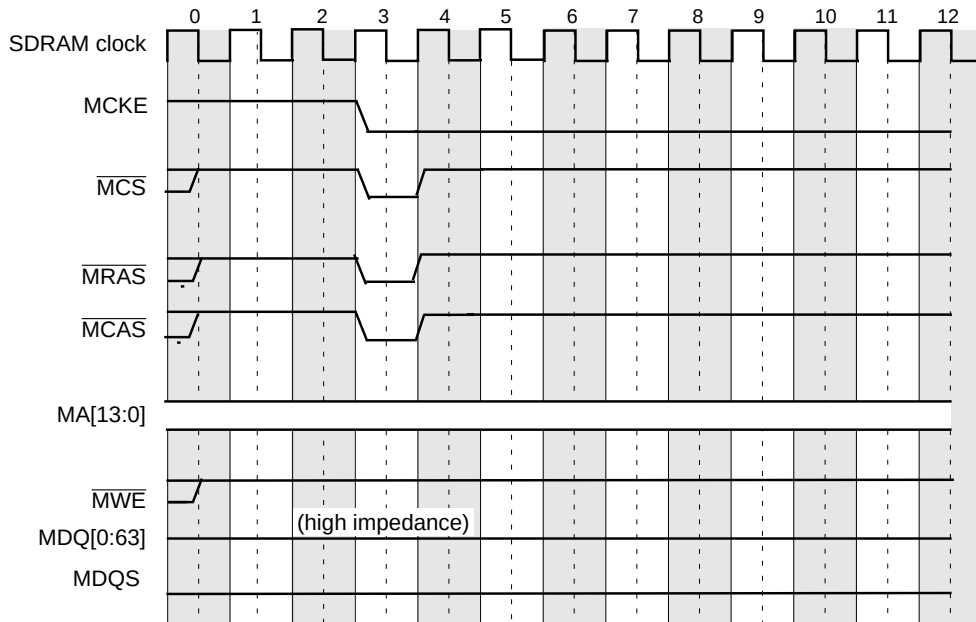


Figure 9-35. DDR SDRAM Self-Refresh Entry Timing

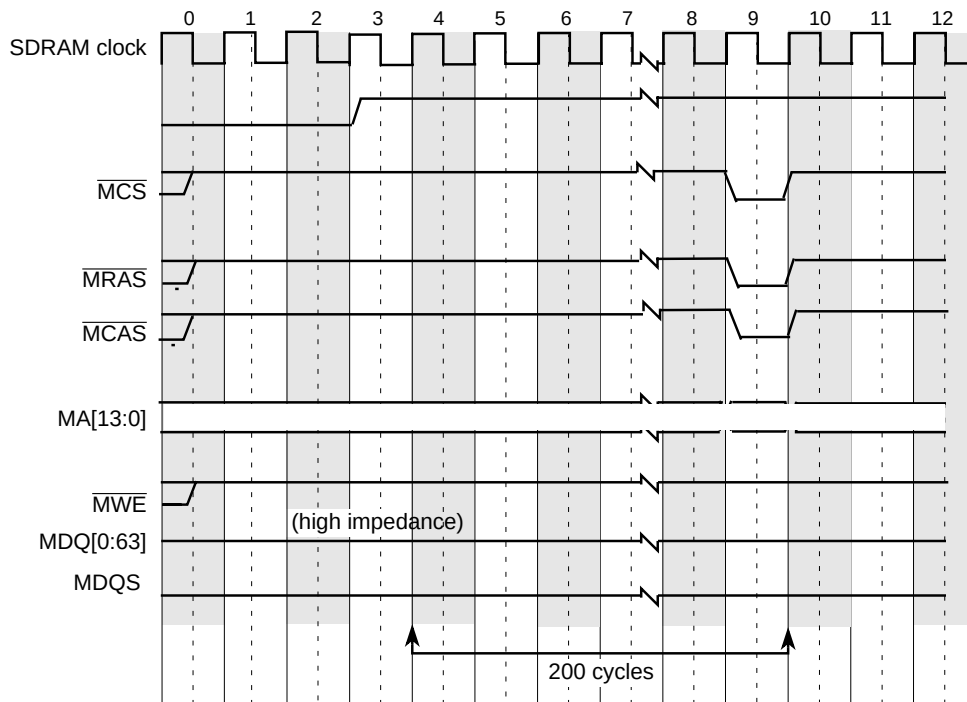


Figure 9-36. DDR SDRAM Self-Refresh Exit Timing

9.4.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four-beat bursts (4-beat = 32 bytes). For transfer sizes other than four beats, the data transfers are still operated as four-beat bursts. If ECC is enabled and byte enables are used, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or byte enables are ignored, the data masks (MDM[0:8]) 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 1 doubleword (8 bytes), then the 2nd, 3rd, and 4th beats of data would not be written to DRAM.

Table 9-32 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 9-32. Memory Controller–Data Beat Ordering

Transfer Size	Starting Double Word Offset	Double Word Sequence ¹ to/from DRAM and queues
1 double word	0	<u>0</u> - 1 - 2 - 3
	1	<u>1</u> - 2 - 3 - 0
	2	<u>2</u> - 3 - 0 - 1
	3	<u>3</u> - 0 - 1 - 2
2 double words	0	<u>0</u> - <u>1</u> - 2 - 3
	1	<u>1</u> - <u>2</u> - 3 - 0
	2	<u>2</u> - <u>3</u> - 0 - 1
3 double words	0	<u>0</u> - <u>1</u> - <u>2</u> - 3
	1	<u>1</u> - <u>2</u> - <u>3</u> - 0
4 double words	0	<u>0</u> - <u>1</u> - <u>2</u> - <u>3</u>
	1	<u>1</u> - <u>2</u> - <u>3</u> - <u>0</u>
	2	<u>2</u> - <u>3</u> - <u>0</u> - <u>1</u>
	3	<u>3</u> - <u>0</u> - <u>1</u> - <u>2</u>

¹ All underlined double-word offsets are valid for the transaction.

9.4.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each sub-bank of DRAM used. In closed page mode, the DDR memory controller uses the SDRAM AUTO PRECHARGE feature which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable AUTO PRECHARGE. AUTO PRECHARGE is non-persistent in

that it is either enabled or disabled for each individual READ or WRITE command. It can however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains open until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable value controlled by `DDR_SDRAM_INTERVAL[BSTOPRE]` is exceeded.
- There is a row collision with another transaction of the same bank and sub-bank that needs to 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]` and `CS_n_CONFIG[AP_n_EN]`

9.4.11 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. Thus, the single-bit error registers may be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes to system memory 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 9-33 and Table 9-34.

Table 9-33. DDR SDRAM ECC Syndrome Encoding

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

Table 9-34. DDR SDRAM ECC Syndrome Encoding (continued)

Syndrome Bit	Data Bit																															
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
0			x				x				x				x					x				x				x			x	
1				x				x				x				x	x				x				x							x
2	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x				x				x						x
3	x				x				x				x						x				x				x			x	x	x
4		x	x	x		x	x	x		x	x	x		x	x	x														x	x	x
5					x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x							
6									x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x
7	x	x					x	x			x	x	x	x							x	x	x	x	x	x	x	x		x	x	x

Table 9-35. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Syndrome Bit	Check Bit							
	0	1	2	3	4	5	6	7
0	x							
1		x						
2			x					
3				x				
4					x			
5						x		

Table 9-35. DDR SDRAM ECC Syndrome Encoding (Check Bits) (continued)

Syndrome Bit	Check Bit							
	0	1	2	3	4	5	6	7
6							x	
7								x

9.4.12 Error Management

The DDR memory controller detects three different kinds of errors: 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 9.3.1.16, “Memory Error Interrupt Enable (ERR_INT_EN),” Section 9.3.1.15, “Memory Error Disable (ERR_DISABLE),” and Section 9.3.1.14, “Memory Error Detect (ERR_DETECT).”

Single-bit errors are counted and reported based on the value of the single-bit ECC Memory error management register (ERR_SBE). 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 machine check or 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 9.3.1.15, “Memory Error Disable (ERR_DISABLE)”). The final error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate a machine check or critical interrupt (if enabled, as described in Section 9.3.1.14, “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 sample points are used for sampling (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 9-36 shows the errors with their descriptions.

Table 9-36. Memory Controller Errors

Category	Error	Description(s)	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors have reached the threshold specified in the ECC single-bit error register.	The error is reported via machine check or critical interrupt if enabled.	The error control register only logs read versus write, not full ttype
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.		

9.5 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 $\overline{MCS}[0:3]$ signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to multiplex appropriately the row and column address bits for each bank. Refer to row-address configuration in Section 9.3.1.2, “Chip Select Configuration (CS_n_CONFIG).” Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (bootcode) must set up the programmable parameters in the memory interface configuration registers (MICRs). See Table 9.3.1 on page 9-11 for more detailed descriptions of the configuration registers. These parameters are shown in Table 9-37.

Table 9-37. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/Page
CS _n _BNDS	Chip select memory bounds.	SA _n EA _n	9.3.1.1/9-11
CS _n _CONFIG	Chip select configuration	CS _n _EN ROW_BITS_CS _n COL_BITS_CS _n	9.3.1.2/9-12
TIMING_CFG_1	DDR SDRAM timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD WR_DATA_DELAY	9.3.1.3/9-13

**Table 9-37. Memory Interface Configuration Register
Initialization Parameters (continued)**

Name	Description	Parameter	Section/Page
DDR_SDRAM_CFG	DDR SDRAM control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR	9.3.1.5/9-17
DDR_SDRAM_MODE	DDR SDRAM mode configuration	ESDMODE SDMODE	9.3.1.6/9-18
DDR_SDRAM_INTERVAL	DDR SDRAM interval configuration	REFINT BSTOPRE	9.3.1.7/9-19

9.5.1 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 μ s must elapse after the DLL has locked before the MEM_EN bit can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. The DDR memory controller then conducts an initialization sequence to prepare the DDR SDRAM array for accesses. The initialization sequence for JEDEC compliant DDR SDRAM is as follows:

1. PRECHARGE ALL
2. MODE REGISTER SET for extended mode register
3. MODE REGISTER SET for mode register
4. PRECHARGE ALL
5. Two AUTO REFRESH commands
6. MODE REGISTER SET for mode register with reset DLL bit deactivated

When these steps are complete, the memory array is ready for access.

Chapter 10

Programmable Interrupt Controller

This chapter describes the programmable interrupt controller (PIC) interrupt protocol, various types of interrupt sources controlled by the PIC unit, and the PIC registers with some programming guidelines.

The MPC8540 PIC unit prioritizes and manages interrupts from the L2 cache, the ECM, the DDR controller, the local bus controller (LBC), the 4-channel DMA controller, the PCI/PCI-X block, the RapidIO interface, the dual three-speed Ethernet controllers (TSEC), the 10/100 Ethernet controller, the DUART, the performance monitor, and the I²C controller. It also manages the interrupts generated by the PIC itself, and off-chip interrupt sources.

The *int* signal is the main interrupt output from the PIC unit to the processor core and causes the external interrupt exception. As an alternative to the *int* signal, interrupts from the PIC can also be designated as critical interrupts and can be programmed to cause a critical interrupt exception in the processor core (by causing the internal *cint* signal to be asserted). Note that the external $\overline{\text{SRESET}}$ signal or the internal *sreset* output of the PIC unit causes the soft reset processor exception. The machine check exception is caused by certain error conditions that assert the internal *mcp* signal generated by the common on-chip processor (COP) unit or by assertion of the external $\overline{\text{MCP}}$ signal.

Note that several other external signals controlled by the COP unit such as $\overline{\text{UDE}}$ and hard reset ($\overline{\text{HRESET}}$), also cause exceptions in the processor core.

10.1 Introduction

A block diagram of the MPC8540 showing the relationship of the various functional blocks and external signals to the PIC unit is shown in Figure 10-1.

The PIC unit receives interrupt signals from three sources: external to the integrated device, internal to the integrated device, and intrinsic to the PIC itself. The PIC selects among all current interrupts the one with the highest priority and forwards it to the internal processor core, or, in pass-through mode, off-chip for external servicing.

The interrupt router of the PIC monitors the outputs of its internal interrupt request register (IRR) and other configuration registers, and when the priority of an interrupt signal it

receives is higher than the priority in the task priority register, it asserts the internal interrupt signal (*int*) to indicate an interrupt request to the processor. This causes the processor to vector to the external interrupt handler, and the interrupt handler code running on the processor then acknowledges the interrupt by explicitly reading the IACK register. The PIC unit interprets this read as an interrupt acknowledge (IACK) cycle. In response to this read, the PIC unit returns the vector associated with the interrupt source to the interrupt handler routine, and the handler can further vector to different branches of interrupt handling accordingly.

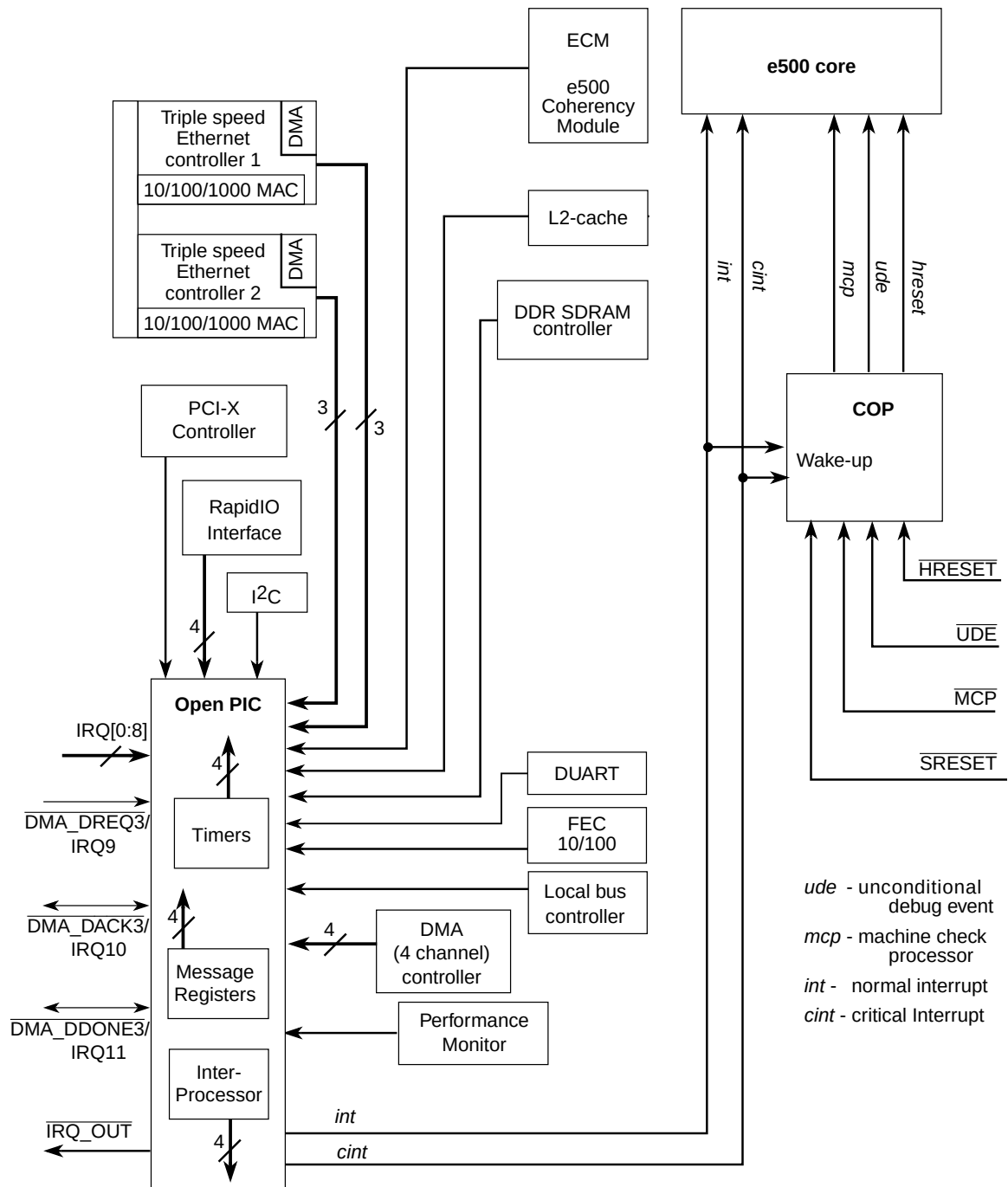


Figure 10-1. MPC8540 Interrupts Block Diagram

There is also an internal in-service register (ISR) that keeps track of all the interrupts that are in-service. An interrupt is considered in-service from the time its vector is read through

an IACK cycle, until the end of interrupt (EOI) register is written, generating what the PIC considers an EOI signal.

The PIC unit can receive 47 separate interrupts from five different sources as follows:

- 12 external—off-chip signals, IRQ[0:11]
- 23 internal—on-chip. Sources are L2, ECM, DDR, LBC, DMA, PCI/PCI-X, RapidIO, TSEC, FEC, DUART, performance monitor, and I²C.
- 4 global timers—from inside the PIC
- 4 inter-processor (IPI)—intended for communication between different processor cores on the same device. Only used for self-interrupt in a single-core device such as the MPC8540.
- 4 message registers—from inside the PIC. Triggered on register write, cleared on read. Used for inter-process communication.

When the PIC receives an internal or external interrupt, its destination register is checked to determine if it should be routed off-chip (to the external $\overline{\text{IRQ\_OUT}}$ signal) or serviced as a normal external interrupt by the processor core (through the *int* signal). As a third alternative, if the incoming interrupt has been configured in xIDR[CI] as a critical interrupt, the PIC completes its processing of the interrupt by asserting the *cint* input to the core, causing it to be serviced as a critical interrupt.

If the incoming interrupt has its xIDR[EP] bit set, the PIC completes its processing of the interrupt by asserting $\overline{\text{IRQ\_OUT}}$. If neither xIDR[CI] or xIDR[EP] are set, the interrupt is latched by the interrupt pending register (IPR) and routed to the processor core through the internal *int* signal.

10.1.1 Overview

The programmable interrupt controller (PIC) is compliant with the OpenPIC architecture. The interrupt controller provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.

10.1.2 Features

- Programming model compliant with the OpenPIC architecture
- Support for 12 external and 23 internal interrupt sources. Serial interrupts are not supported.
- Four interprocessor interrupt channels
- Four 32-bit messaging interrupt channels
- Four global high resolution timers
- Fully-nested interrupt delivery

- Processor initialization control
- Programmable resetting of the PIC unit through the global configuration register
- 16 programmable interrupt priority levels
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller
- In 8259 mode, it generates a local (internal) interrupt output signal, $\overline{\text{IRQ\_OUT}}$
- Recovery from spurious interrupts

10.1.3 Modes of Operation

The PIC unit can be programmed to operate in mixed mode or pass-through mode by setting or clearing GCR[M] as described in Section 10.3.1.2, “Global Configuration Register (GCR).”

10.1.3.1 Mixed Mode (GCR[M] Set)

In mixed mode, the external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in Section 10.3.6.1, “External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11),” through Section 10.3.6.4, “Internal Interrupt Destination Registers (IIDR0–IIDR31).”

10.1.3.2 Pass-Through Mode (GCR[M] Cleared)

The PIC unit 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 unit defaults to the pass-through mode. In this mode, active-high interrupts from external source IRQ0 are passed directly to the e500 core; 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. The PIC unit does not perform a vector fetch from an 8259 interrupt controller.

When pass-through mode is enabled, none of the internally-generated interrupts are forwarded to the e500 core. Instead, the PIC passes the raw interrupts from the internal sources to the $\overline{\text{IRQ\_OUT}}$ signal. Note that interrupts generated by the PIC itself, that is, global timers, interprocessor, and message register interrupts, cannot be used in pass-through mode.

If internal or PIC-generated interrupts must be reported internally to the processor, pass-through mode must be disabled.

10.1.4 Internal Interrupt Sources

Table 10-1 shows the assignments of the 23 internal interrupt sources for the MPC8540. Note that this list does not include the interrupts generated by the PIC unit.

Table 10-1. Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source
0	L2 cache
1	ECM
2	DDR DRAM
3	LBC
4	DMA channel 0
5	DMA channel 1
6	DMA channel 2
7	DMA channel 3
8	PCI/PCI-X
9	RapidIO inbound port write/ error interrupt
10	RapidIO doorbell inbound interrupt
11	RapidIO outbound message interrupt
12	RapidIO Inbound message interrupt
13	TSEC 1 transmit interrupt
14	TSEC 1 receive interrupt
15	reserved
16	reserved
17	reserved
18	TSEC 1 receive/transmit error interrupt
19	TSEC 2 transmit interrupt
20	TSEC 2 receive interrupt
21	reserved
22	reserved
23	reserved
24	TSEC 2 receive/transmit error interrupt
25	Fast Ethernet controller
26	DUART
27	I ² C controller
28	Performance monitor interrupt
29	unused

Table 10-1. Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
30	unused
31	unused

10.2 External Signal Description

10.2.1 Signal Overview

There are twelve distinct external interrupt request input signals (IRQ[0:11]) and one interrupt request output signal (IRQ_OUT). The PIC interface signals are defined in Table 10-2. As shown in the table, three IRQ inputs are multiplexed with the DMA signals for DMA channel 3.

Table 10-2. PIC Interface Signals

Signal Name	I/O	Description
IRQ[0:8]	I	External interrupts
IRQ9/DMA_DREQ3 ¹	I	External interrupt /DMA channel 3 request
IRQ10/DMA_DACK3 ¹	I or O	External interrupt (input)/DMA channel 3 acknowledge (output)
IRQ11/DMA_DDONE3 ¹	I or O	External interrupt (input)/DMA channel 3 done (output)
IRQ_OUT	O	Interrupt request out
MCP	I	Machine check processor
UDE	I	Unconditional debug event

¹ IRQ9–11 are multiplexed with DMA3 signals. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in Section 18.3.1.10, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

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 completely asynchronously.	
		State Meaning	Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the priority is checked by the PIC unit, and the interrupt is conditionally passed to the processor. In pass-through mode, only interrupts detected on IRQ0 are passed directly to the processor core. Negated—There is no incoming interrupt from that source.
		Timing	Assertion—All of these inputs can be asserted completely asynchronously. Negation—Interrupts programmed as level-sensitive must remain asserted until serviced. The timing requirements for edge-sensitive interrupts can be found in the <i>MPC8540Electrical Specifications</i> .
IRQ_OUT	O	Interrupt request out. Active-low, open drain. When the PIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See Section 10.1.3, “Modes of Operation” for more details.	
		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 CLK 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 CLK cycles External interrupt: 4 cycles. Message interrupts: .
$\overline{\text{MCP}}$	I	Machine check processor. Assertion of the external $\overline{\text{MCP}}$ signal causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts, assertion of $\overline{\text{MCP}}$ causes a checkstop condition. Note that there are internal sources for the internal <i>mcp</i> signal that can also cause a machine check interrupt to the processor core.	
		State Meaning	Asserted—The MPC8540 should initiate 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—.
$\overline{\text{UDE}}$	I	Unconditional debug event. The assertion of the external $\overline{\text{UDE}}$ signal causes an unconditional debug event interrupt to the e500 core.	
		State Meaning	Asserted—Indicates that the MPC8540 should initiate an unconditional debug event interrupt to the processor core. Negated—Indicates that unconditional debug event handling is not being requested by $\overline{\text{UDE}}$.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—

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 only be accessed as 32-bit quantities.

The PIC address offset map, shown in Table 10-4, is divided into three areas:

- 0xnn4_0000–0xnn4_FFF0—Global registers
- 0xnn5_0000–0xnn5_FFF0—Interrupt source configuration registers
- 0xnn6_0000–0xnn7_FFF0—Per CPU registers

Table 10-4. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	Interprocessor interrupt 0 (IPI 0) dispatch register (IPIDR0)	W	0x0000_0000	10.3.7.1/10-41
0x4_0050	IPI 1 dispatch register (IPIDR1)			
0x4_0060	IPI 2 dispatch register (IPIDR2)			
0x4_0070	IPI 3 dispatch register (IPIDR3)			
0x4_0080	Current task priority register (CTPR)	R/W	0x0000_000F	10.3.7.2/10-42
0x4_0090	Who am I register (WHOAMI0)	R	0x0000_0000	10.3.7.3/10-43
0x4_00A0	Interrupt acknowledge register (IACK)	R	0x0000_0000	10.3.7.4/10-44
0x4_00B0	End of interrupt register (EOI)	W	0x0000_0000	10.3.7.5/10-45
0x4_00C–0x4_0FF0	Reserved	—	—	—
0x4_1000	Feature reporting register (FRR)	R	0x0037_0002	10.3.1.1/10-15
0x4_1010	Reserved	—	—	—
0x4_1020	Global configuration register (GCR)	R/W	0x0000_0000	10.3.1.2/10-16
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	Vendor identification register (VIR)	R	0x0000_0000	10.3.1.3/10-16
0x4_1090	Processor initialization register (PIR)	R/W	0x0000_0000	10.3.1.4/10-17

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_10A0	IPI 0 vector/priority register (IPIVPR0)	R/W	0x8000_0000	10.3.1.5/10-18
0x4_10B0	IPI 1 vector/priority register (IPIVPR1)			
0x4_10C0	IPI 2 vector/priority register (IPIVPR2)			
0x4_10D0	IPI 3 vector/priority register (IPIVPR3)			
0x4_10E0	Spurious vector register (SVR)	R/W	0x0000_FFFF	10.3.1.6/10-19
0x4_10F0	Timer frequency reporting register (TFRR)	R/W	0x0000_0000	10.3.2.1/10-20
0x4_1100	Global timer 0 current count register (GTCCR0)	R	0x0000_0000	10.3.2.2/10-20
0x4_1110	Global timer 0 base count register (GTBCR0)	R/W	0x8000_0000	10.3.2.3/10-21
0x4_1120	Global timer 0 vector/priority register (GTVPR0)	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1130	Global timer 0 destination register (GTDR0)	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1140	Global timer 1 current count register (GTCCR1)	R	0x0000_0000	10.3.2.2/10-20
0x4_1150	Global Timer 1 Base Count Register (GTBCR1)	R/W	0x8000_0000	10.3.2.3/10-21
0x4_1160	Global timer 1 vector/priority register (GTVPR1)	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1170	Global timer 1 destination register (GTDR1)	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1180	Global timer 2 current count register (GTCCR2)	R	0x0000_0000	10.3.2.2/10-20
0x4_1190	Global timer 2 base count register (GTBCR2)	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11A0	Global timer 2 vector/priority register (GTVPR2)	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11B0	Global timer 2 destination register (GTDR2)	R/W	0x0000_0001	10.3.2.5/10-23
0x4_11C0	Global timer 3 current count register (GTCCR3)	R	0x0000_0000	10.3.2.2/10-20
0x4_11D0	Global timer 3 base count register (GTBCR3)	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11E0	Global timer 3 vector/priority register (GTVPR3)	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11F0	Global timer 3 destination register (GTDR3)	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1200– 0x4_12F0	Reserved	—	—	—
0x4_1300	Timer control register (TCR)	R/W	0x0000_0000	10.3.2.6/10-24
0x4_1310	IRQ_OUT summary register 0 (IRQSR0)	R	0x0000_0000	10.3.3.1/10-26
0x4_1320	IRQ_OUT summary register 1 (IRQSR1)	R	0x0000_0000	10.3.3.2/10-27
0x4_1330	Critical Interrupt summary register 0 (CISR0)	R	0x0000_0000	10.3.3.3/10-27
0x4_1340	Critical Interrupt summary register 1 (CISR1)	R	0x0000_0000	10.3.3.4/10-28
0x4_1350	Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1360	Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-30
0x4_1370	Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1380	Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-30
0x4_1390	Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_13A0	Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-30
0x4_13B0	Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_13C0	Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-30
0x4_13D0– 0x4_13F0	Reserved	—	—	—
0x4_1400	Message register 0 (MSGR0)	R/W	0x0000_0000	10.3.5.1/10-31
0x4_1410	Message register 1 (MSGR1)			
0x4_1420	Message register 2 (MSGR2)			
0x4_1430	Message register 3 (MSGR3)			
0x4_1440– 0x4_14F0	Reserved	—	—	—
0x4_1500	Message enable register (MER)	R/W	0x0000_0000	10.3.5.2/10-31
0x4_1510	Message status register (MSR)	R/W	0x0000_0000	10.3.5.3/10-32
0x4_1520– 0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	External interrupt 0 (IRQ0) vector/priority register (EIVPR0)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0010	External interrupt 0 (IRQ0) destination register (EIDR0)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0020	External interrupt 1 (IRQ1) vector/priority register (EIVPR1)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0030	External interrupt 1 (IRQ1) destination register (EIDR1)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0040	External interrupt 2 (IRQ2) vector/priority register (EIVPR2)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0050	External interrupt 2 (IRQ2) destination register (EIDR2)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0060	External interrupt 3 (IRQ3) vector/priority register (EIVPR3)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0070	External interrupt 3 (IRQ3) destination register (EIDR3)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0080	External interrupt 4 (IRQ4) vector/priority register (EIVPR4)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0090	External interrupt 4 (IRQ4) destination register (EIDR4)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_00A0	External interrupt 5 (IRQ5) vector/priority register (EIVPR5)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_00B0	External interrupt 5 (IRQ5) destination register (EIDR5)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_00C0	External interrupt 6 (IRQ6) vector/priority register (EIVPR6)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_00D0	External interrupt 6 (IRQ6) destination register (EIDR6)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_00E0	External interrupt 7 (IRQ7) vector/priority register (EIVPR7)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_00F0	External interrupt 7 (IRQ7) destination register (EIDR7)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0100	External interrupt 8 (IRQ8) vector/priority register (EIVPR8)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0110	External interrupt 8 (IRQ8) destination register (EIDR8)	R/W	0x0000_0001	10.3.6.2/10-34

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0120	External interrupt 9 (IRQ9) vector/priority register (EIVPR9)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0130	External interrupt 9 (IRQ9) destination register (EIDR9)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0140	External interrupt 10 (IRQ10) vector/priority register (EIVPR10)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0150	External interrupt 10 (IRQ10) destination register (EIDR10)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0160	External interrupt 11 (IRQ11) vector/priority register (EIVPR11)	R/W	0x8000_0000	10.3.6.1/10-33
0x5_0170	External interrupt 11 (IRQ11) destination register (EIDR11)	R/W	0x0000_0001	10.3.6.2/10-34
0x5_0180– 0x5_01F0	Reserved	—	—	—
0x5_0200	Internal interrupt 0 vector/priority register (IIVPR0)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0210	Internal interrupt 0 destination register (IIDR0)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0220	Internal interrupt 1 vector/priority register (IIVPR1)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0230	Internal interrupt 1 destination register (IIDR1)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0240	Internal interrupt 2 vector/priority register (IIVPR2)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0250	Internal interrupt 2 destination register (IIDR2)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0260	Internal interrupt 3 vector/priority register (IIVPR3)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0270	Internal interrupt 3 destination register (IIDR3)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0280	Internal interrupt 4 vector/priority register (IIVPR4)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0290	Internal interrupt 4 destination register (IIDR4)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_02A0	Internal interrupt 5 vector/priority register (IIVPR5)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_02B0	Internal interrupt 5 destination register (IIDR5)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_02C0	Internal interrupt 6 vector/priority register (IIVPR6)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_02D0	Internal interrupt 6 destination register (IIDR6)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_02E0	Internal interrupt 7 vector/priority register (IIVPR7)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_02F0	Internal interrupt 7 destination register (IIDR7)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0300	Internal interrupt 8 vector/priority register (IIVPR8)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0310	Internal interrupt 8 destination register (IIDR8)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0320	Internal interrupt 9 vector/priority register (IIVPR9)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0330	Internal interrupt 9 destination register (IIDR9)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0340	Internal interrupt 10 vector/priority register (IIVPR10)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0350	Internal interrupt 10 destination register (IIDR10)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0360	Internal interrupt 11 vector/priority register (IIVPR11)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0370	Internal interrupt 11 destination register (IIDR11)	R/W	0x0000_0001	10.3.6.4/10-36

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0380	Internal interrupt 12 vector/priority register (IIVPR12)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0390	Internal interrupt 12 destination register (IIDR12)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_03A0	Internal interrupt 13 vector/priority register (IIVPR13)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_03B0	Internal interrupt 13 destination register (IIDR13)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_03C0	Internal interrupt 14 vector/priority register (IIVPR14)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_03D0	Internal interrupt 14 destination register (IIDR14)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_03E0	Internal interrupt 15 vector/priority register (IIVPR15)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_03F0	Internal interrupt 15 destination register (IIDR15)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0400	Internal interrupt 16 vector/priority register (IIVPR16)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0410	Internal interrupt 16 destination register (IIDR16)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0420	Internal interrupt 17 vector/priority register (IIVPR17)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0430	Internal interrupt 17 destination register (IIDR17)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0440	Internal interrupt 18 vector/priority register (IIVPR18)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0450	Internal interrupt 18 destination register (IIDR18)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0460	Internal interrupt 19 vector/priority register (IIVPR19)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0470	Internal interrupt 19 destination register (IIDR19)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0480	Internal interrupt 20 vector/priority register (IIVPR20)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0490	Internal interrupt 20 destination register (IIDR20)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_04A0	Internal interrupt 21 vector/priority register (IIVPR21)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_04B0	Internal interrupt 21 destination register (IIDR21)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_04C0	Internal interrupt 22 vector/priority register (IIVPR22)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_04D0	Internal interrupt 22 destination register (IIDR22)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_04E0	Internal interrupt 23 vector/priority register (IIVPR23)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_04F0	Internal interrupt 23 destination register (IIDR23)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0500	Internal interrupt 24 vector/priority register (IIVPR24)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0510	Internal interrupt 24 destination register (IIDR24)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0520	Internal interrupt 25 vector/priority register (IIVPR25)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0530	Internal interrupt 25 destination register (IIDR25)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0540	Internal interrupt 26 vector/priority register (IIVPR26)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0550	Internal interrupt 26 destination register (IIDR126)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0560	Internal interrupt 27 vector/priority register (IIVPR27)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_0570	Internal interrupt 27 destination register (IIDR27)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0580	Internal interrupt 28 vector/priority register (IIVPR28)	R/W	0x8080_0000	10.3.6.3/10-35

Table 10-4. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0590	Internal interrupt 28 destination register (IIDR28)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_05A0	Internal interrupt 29 vector/priority register (IIVPR29)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_05B0	Internal interrupt 29 destination register (IIDR29)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_05C0	Internal interrupt 30 vector/priority register (IIVPR30)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_05D0	Internal interrupt 30 destination register (IIDR30)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_05E0	Internal interrupt 31 vector/priority register (IIVPR31)	R/W	0x8080_0000	10.3.6.3/10-35
0x5_05F0	Internal interrupt 31 destination register (IIDR31)	R/W	0x0000_0001	10.3.6.4/10-36
0x5_0600– 0x5_15F0	Reserved	—	—	—
0x5_1600	Messaging interrupt 0 (MSG 0) vector/priority register (MIVPR0)	R/W	0x8000_0000	10.3.6.5/10-37
0x5_1610	Messaging interrupt 0 (MSG 0) destination register (MIDR0)	R/W	0x0000_0001	10.3.6.6/10-38
0x5_1620	Messaging interrupt 1 (MSG 1) vector/priority register (MIVPR1)	R/W	0x8000_0000	10.3.6.5/10-37
0x5_1630	Messaging interrupt 1 (MSG 1) destination register (MIDR1)	R/W	0x0000_0001	10.3.6.6/10-38
0x5_1640	Messaging interrupt 2 (MSG 2) vector/priority register (MIVPR2)	R/W	0x8000_0000	10.3.6.5/10-37
0x5_1650	Messaging interrupt 2 (MSG 2) destination register (MIDR2)	R/W	0x0000_0001	10.3.6.6/10-38
0x5_1660	Messaging interrupt 3 (MSG 3) vector/priority register (MIVPR3)	R/W	0x8000_0000	10.3.6.5/10-37
0x5_1670	Messaging interrupt 3 (MSG 3) destination register (MIDR3)	R/W	0x0000_0001	10.3.6.6/10-38
0x5_1680– 0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per CPU Registers				
0x6_0000– 0x6_0030	Reserved	—	—	—
0x6_0040	P0 IPI 0 dispatch register (IPIDR0)	W	all zeros	10.3.7.1/10-41
0x6_0050	P0 IPI 1 dispatch register (IPIDR1)			
0x6_0060	P0 IPI 2 dispatch register (IPIDR2)			
0x6_0070	P0 IPI 3 dispatch register (IPIDR3)			
0x6_0080	P0 current task priority register (CTPR0)	R/W	0x0000_000F	10.3.7.2/10-42
0x6_0090	P0 who am I register (WHOAMI0)	R	all zeros	10.3.7.3/10-43
0x6_00A0	P0 interrupt acknowledge register (IACK0)	R	all zeros	10.3.7.4/10-44
0x6_00B0	P0 end of interrupt register (EOI0)	W	all zeros	10.3.7.5/10-45

10.3.1 Global Registers

Most registers have one address. Some registers are replicated for each processor in a multiprocessor device. In this case, each processor accesses their separate registers using the same address, the address decoding being sensitive to the processor ID. A copy of the per CPU registers is available to each processor core at the same physical address, that is, the private access address space. The private access address space acts like an alias to a processor's own copy of the per CPU registers. As shown in Figure 10-30, the ID of the processor initiating the read/write transaction is used to determine which processor's per CPU registers to access. For more information on the per CPU registers, refer to Section 10.3.7, "Per CPU Registers."

10.3.1.1 Feature Reporting Register (FRR)

The feature reporting register (FRR) shown in Figure 10-2 provides information about the interrupt and processor configurations. This read-only register also informs the programming environment of the controller version.

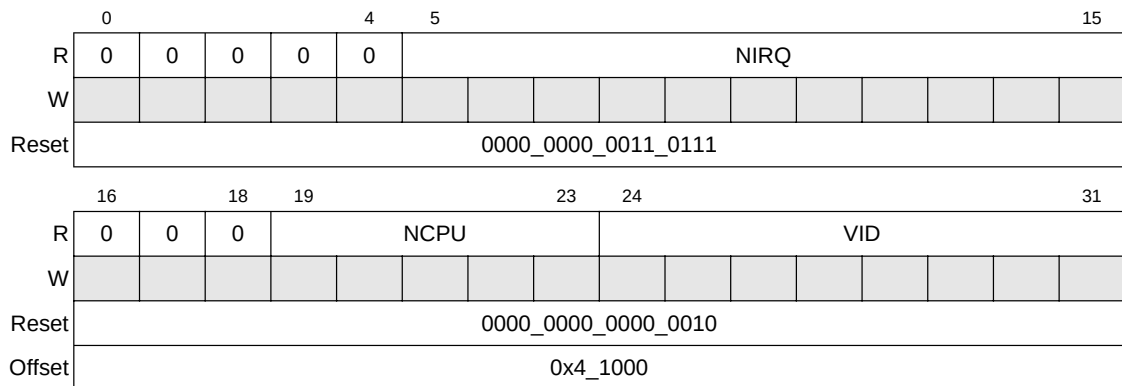


Figure 10-2. Feature Reporting Register (FRR)

Table 10-5 describes the FRR fields.

Table 10-5. FRR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–15	NIRQ	Number of interrupts. Contains the binary value of 1 minus the maximum number of interrupt sources supported. The value is 55 (0x37) because this device supports 56 interrupts: 12 external sources, 32 internal sources (only 23 used), four timer sources, four IPI sources and four messaging sources. A zero in this field corresponds to one source.
16–18	—	Reserved, should be cleared.

Table 10-5. FRR Field Descriptions (continued)

Bits	Name	Description
19–23	NCPU	Number of CPUs. The number of the highest physical CPU supported. The MPC8540 implements one CPU—the processor core, referenced as P0.
24–31	VID	Version ID. Version ID for this interrupt controller. Reports the Open PIC specification revision level supported by this implementation. A value of two corresponds to revision 1.2 which is the revision level currently supported.

10.3.1.2 Global Configuration Register (GCR)

The global configuration register (GCR) shown in Figure 10-3 controls the PIC's operating mode, and allows software to reset the PIC.

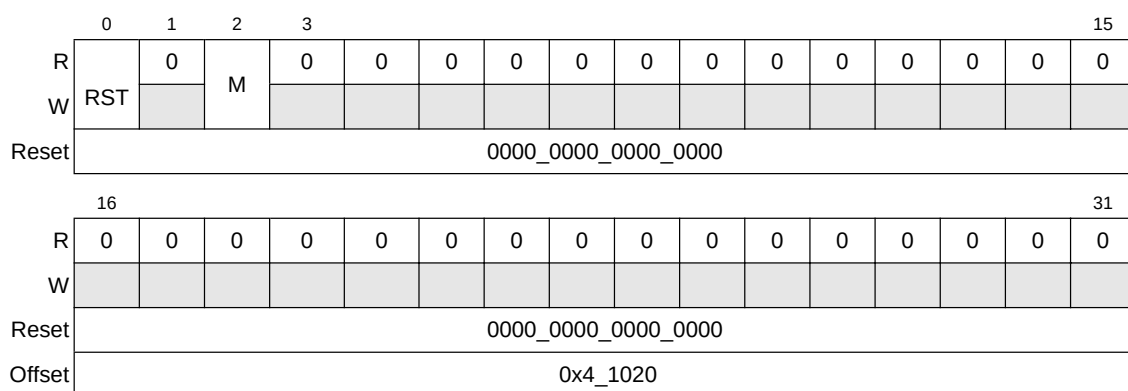
**Figure 10-3. Global Configuration Register (GCR)**

Table 10-6 describes the GCR fields.

Table 10-6. GCR Field Descriptions

Bits	Name	Description
0	RST	Reset. Setting this field forces the PIC to be reset. Cleared automatically when the reset sequence is complete. See Section 10.4.9, "Reset," for more information on resetting the PIC.
1	—	Reserved, should be cleared.
2	M	Mode. PIC operating mode. 0 Pass-through mode. On-chip PIC is disabled and interrupts detected on IRQ0 are passed directly to the processor core. See Section 10.1.3, "Modes of Operation," for more details. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the PIC. See Section 10.1.3, "Modes of Operation," for more details.
3–31	—	Reserved, should be cleared.

10.3.1.3 Vendor Identification Register (VIR)

The vector identification register (VIR) shown in Figure 10-4 has specific read-only information about the vendor and the device revision.

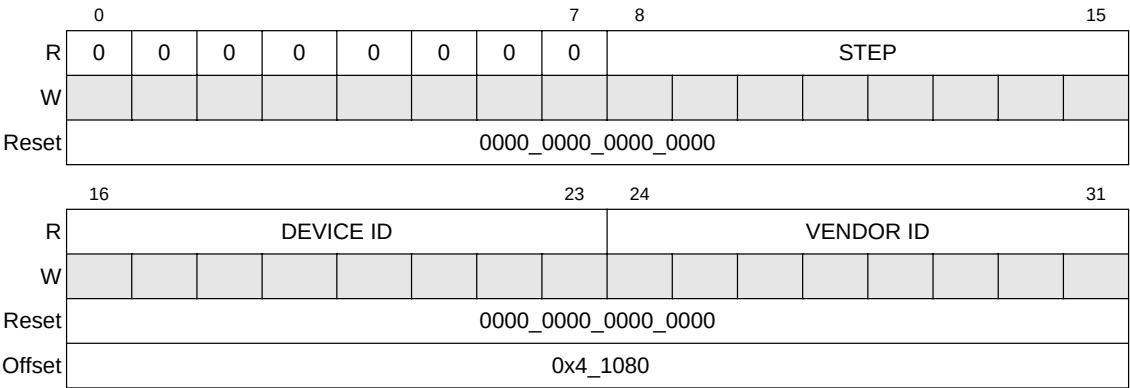


Figure 10-4. Vendor Identification Register (VIR)

Table 10-7 describes the VIR fields.

Table 10-7. 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 when the VENDOR ID value is zero.
16–23	DEVICE ID	Device identification. Vendor-specified identifier for this device. Has no meaning when the VENDOR ID value is zero.
24–31	VENDOR ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic PIC-compliant device.

10.3.1.4 Processor Initialization Register (PIR)

The processor initialization register provides a mechanism through the PIC for the software to reset the processor. The *s_reset* signal to the processor is held active until a zero is written to the processor initialization register field. Thus this field should be cleared by software as soon as possible in the software reset exception handler.

Note that while the architecture was designed to support multiple processing cores, only the fields corresponding to the processors on the device are implemented, as shown in Figure 10-5. In a uniprocessor system such as the MPC8540, only PIR[P0] is implemented.

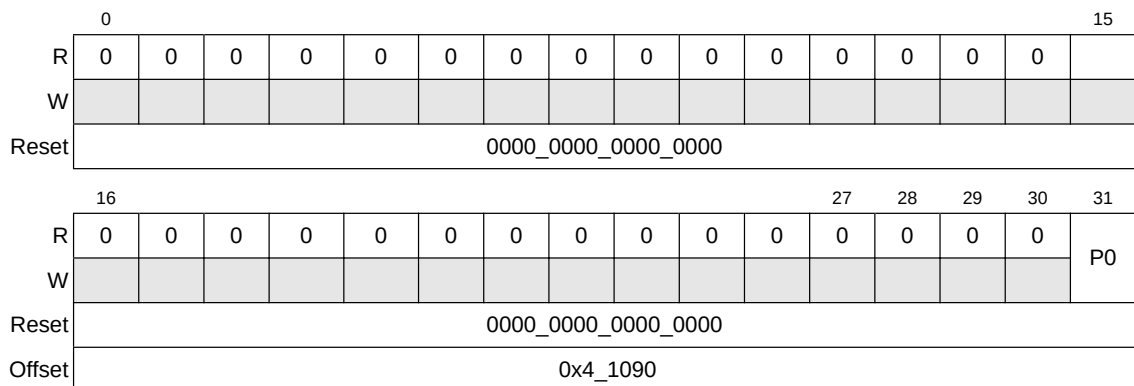
**Figure 10-5. Processor Initialization Register (PIR)**

Table 10-8 describes the PIR fields.

Table 10-8. PIR Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	P0	Processor 0 soft reset. Setting this bit causes the PIC unit to assert the s_reset signal to processor 0 (the e500 core).

10.3.1.5 IPI Vector/Priority Registers (IPIVPR_n)

The interprocessor interrupt (IPI) vector/priority registers contain the interrupt vector and priority fields for the four interprocessor interrupt channels as shown in Figure 10-6. There is one IPI vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while PIVPR_n[A] is set.

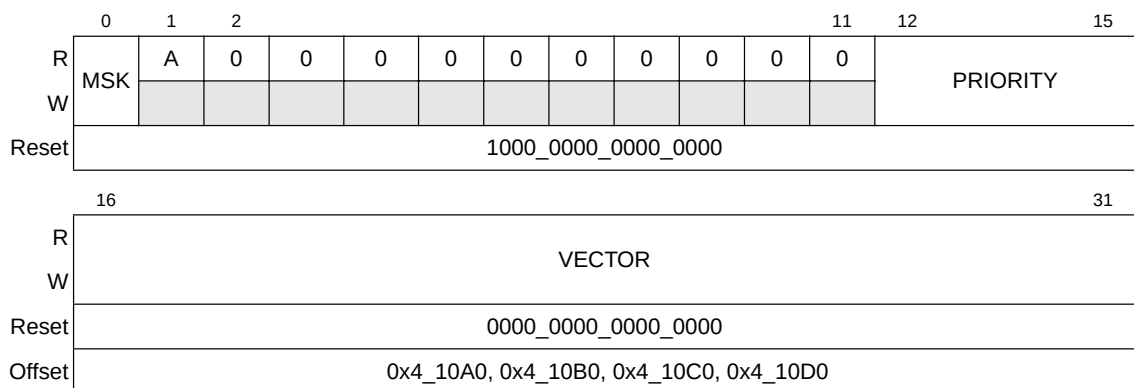
**Figure 10-6. IPI Vector/Priority Register (IPIVPR_n)**

Table 10-9 describes the IPIVPR_n fields.

Table 10-9. IPIVPR_n Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. Always set following reset. 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. Note that this field is read only. The VECTOR and PRIORITY values should not be changed while PIVPR _n [A] is set. 0 No current interrupt activity associated with this source 1 The interrupt bit for this source in the IPR or ISR is set.
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 disables interrupt reporting from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-36.

10.3.1.6 Spurious Vector Register (SVR)

The spurious vector register (SVR) shown in Figure 10-7 contains the 16-bit vector returned to the processor when the IACK register is read for a spurious interrupt.

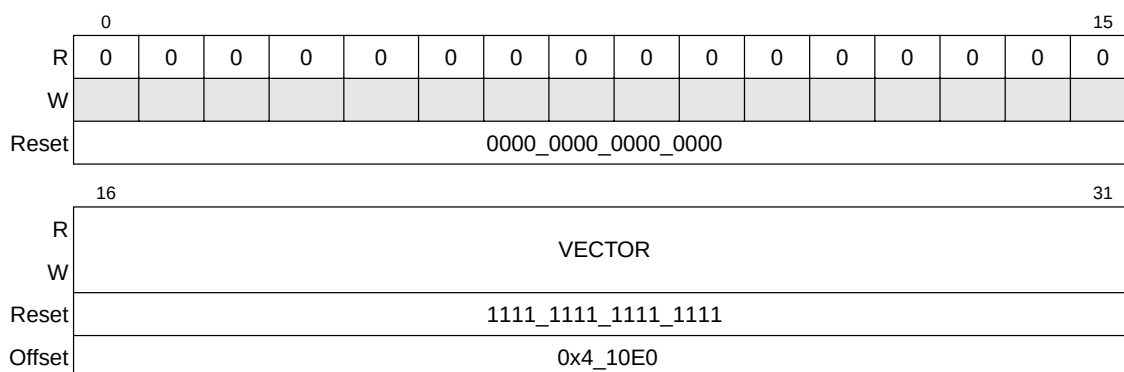
**Figure 10-7. Spurious Vector Register (SVR)**

Table 10-10 describes the SVR fields.

Table 10-10. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Spurious interrupt vector. Value returned when the interrupt acknowledge (IACK) register is read during a spurious vector fetch. See Section 10.4.6, “Spurious Vector Generation,” for more discussion of the events and conditions that may cause a spurious vector fetch.

10.3.2 Global Timer Registers

This section describes the global timer registers. Note that each of the four timers have four individual configuration registers (GTCCR_n, GTBCR_n, GTVPR_n, GTDR_n), but they are only shown once in this section.

10.3.2.1 Timer Frequency Reporting Register (TFRR)

The timer frequency reporting register (TFRR) is written by software to report the clocking frequency of the PIC timers. Note that although this register is read/write, the value of this register is ignored by the PIC unit. The TFRR fields are shown in Figure 10-8.

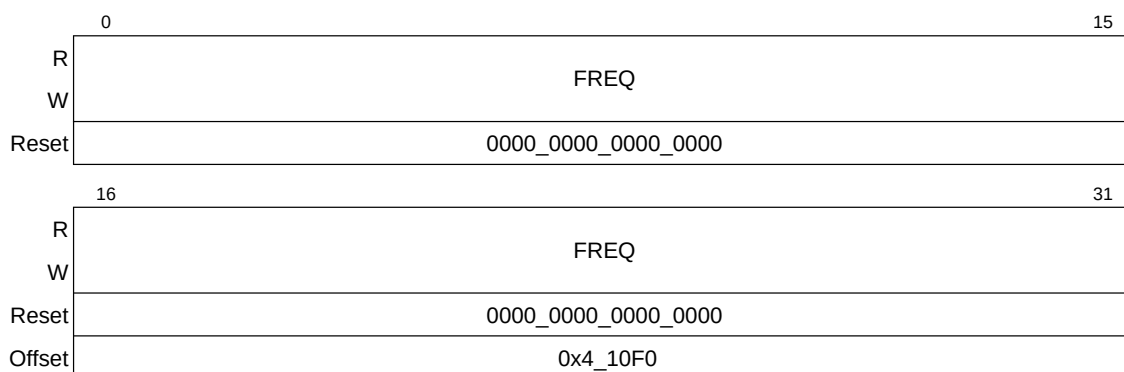


Figure 10-8. Timer Frequency Reporting Register (TFRR)

Table 10-11 describes the TFRR fields.

Table 10-11. TFRR 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, which is identically the system clock, to user software. This register 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 set by TCR[CLKR]. See Section 10.3.2.6, "Timer Control Register (TCR)," for more information.

10.3.2.2 Global Timer Current Count Registers (GTCCR_n)

The global timer current count registers (GTCCRs) contain the current count for each of the four PIC timers. The GTCCR fields are shown in Figure 10-9.

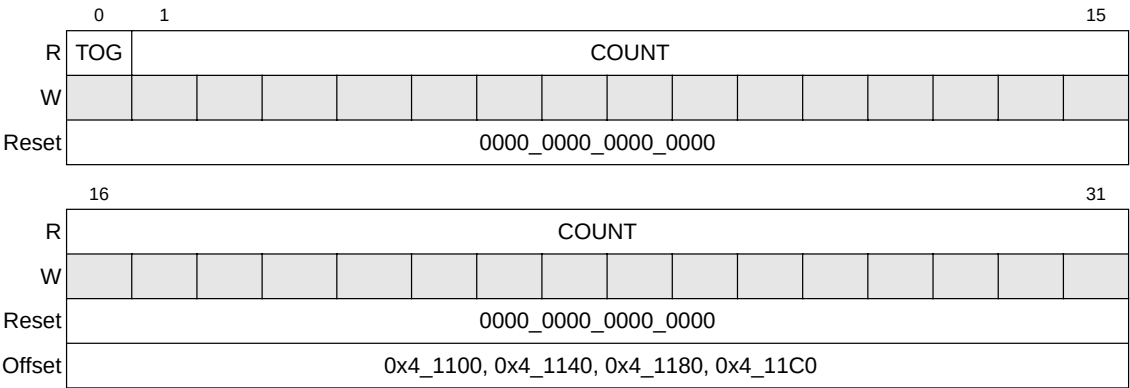


Figure 10-9. Global Timer Current Count Registers (GTCCR_n)

Table 10-12 describes the GTCCRs fields.

Table 10-12. GTCCR_n Field Descriptions

Bits	Name	Description
0	TOG	Toggle. Toggles whenever the current count decrements to zero. Cleared when GTBCR[CI] transitions from 1 to 0.
1–31	COUNT	Current count. Decrementing while GTBCR[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. The reload value is either all ones, or the contents of the base count register, depending on the TCR[ROVR] field. See Section 10.3.2.6, “Timer Control Register (TCR),” for more details.

10.3.2.3 Global Timer Base Count Registers (GTBCR_n)

The global timer base count registers (GTBCRs) contain the base counts for each of the four PIC timers as shown in Figure 10-10. This value is reloaded into the corresponding GTCCR when the current count reaches zero. Note that when zero is written to the base count field, and GTCCR[CI] is cleared, the timer generates an interrupt on every timer cycle.

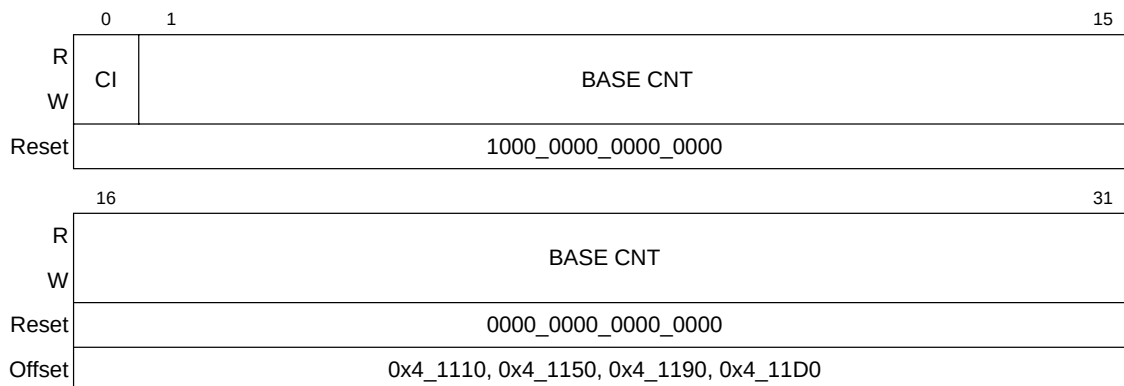
**Figure 10-10. Global Timer Base Count Register (GTBCR_n)**

Table 10-13 describes the GTBCR fields.

Table 10-13. GTBCR_n 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 the CI bit transitions from 1 to 0, this value is copied into the corresponding current count register and the toggle bit is cleared. If GTBCR[CI] is already cleared (counting is in progress), the base count is copied to the current count register at the next zero crossing of the current count.

10.3.2.4 Global Timer Vector/Priority Registers (GTVPR_n)

The global timer vector/priority registers (GTVPRs) contain the interrupt vector and the interrupt priority as shown in Figure 10-11. They also contain the mask and activity fields.

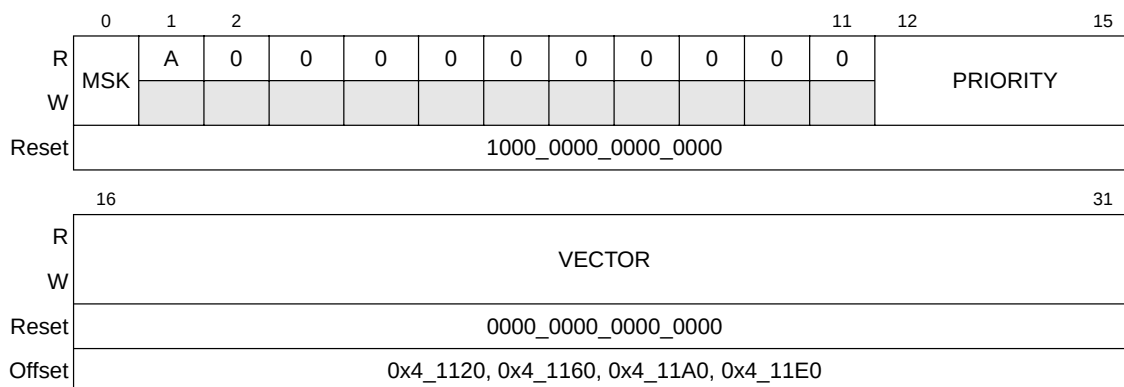
**Figure 10-11. Global Timer Vector/Priority Register (GTVPR_n)**

Table 10-14 describes the GTVPRs fields.

Table 10-14. GTVPR_n Field Descriptions

Bits	Name	Description
0	MSK	Mask. Inhibits interrupts from this source. 0 Interrupt requests are generated when the corresponding IPR bit is set 1 Further interrupts from this source are disabled.
1	A	Activity. This field is read-only. 0 No current interrupt activity associated with this source 1 An interrupt has been requested by the corresponding source or is currently being serviced. The interrupt bit for this source is set in the IPR or ISR. The VECTOR and PRIORITY values should not be changed while the A bit is set.
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 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-36.

10.3.2.5 Global Timer Destination Registers (GTDR)

The global timer destination register controls the destination processor for this timer's interrupt, as shown in Figure 10-12.

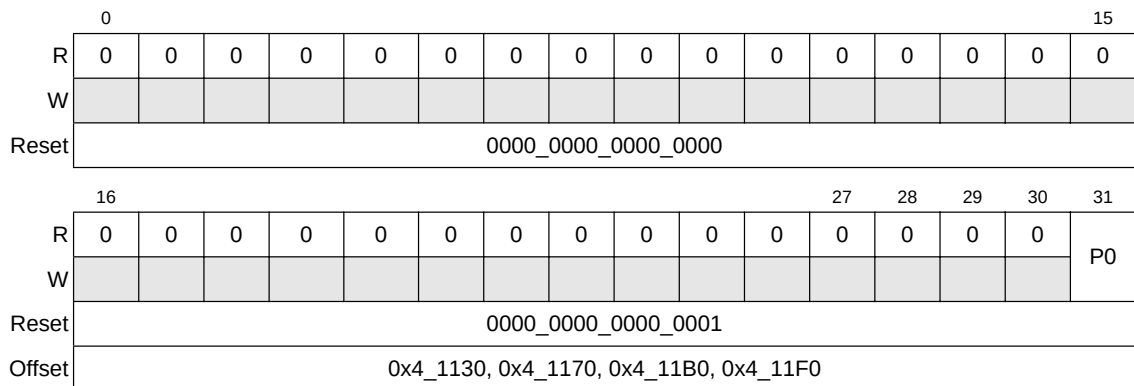
**Figure 10-12. Global Timer Destination Register (GTDR)**

Table 10-15 describes the GTDR fields.

Table 10-15. GTDR Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	P0	Processor 0. 0 No effect 1 Directs the timer interrupt to processor 0

10.3.2.6 Timer Control Register (TCR)

The timer control register (TCR) shown in Figure 10-14 provides various configuration options such as count frequency and roll-over behavior.

The TCR also provides the ability 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.

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field TCR[ROVR]. Setting a timer's roll-over field causes its current count register 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 TCR[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 settings in TCR[ROVR].

For example, timers 0, 1, and 2 can be cascaded to generate one interrupt every hour. As shown in Table 10-16, given a system clock of 333 MHz, letting the timer clock frequency default to 1/8 the system clock, (TCR[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 will generate one output per second. Setting both timers 1 and 2 to 59, and cascading all three timers, will generate one interrupt every hour from timer 2.

Table 10-16. 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.

$$\frac{41.625 \times 10^6 \left(\frac{\text{ticks}}{\text{sec}} \right) \times 60 \left(\frac{\text{sec}}{\text{min}} \right) \times 60 \left(\frac{\text{min}}{\text{hr}} \right)}{(333 \times 10^6) / 8} = 1 \left(\frac{\text{tick}}{\text{hr}} \right)$$

Figure 10-13. Example Calculation for Cascaded Timers

	0					4	5	7	8						15	
R	0	0	0	0	0	ROVR			0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16					21	22	23	24	28					29	31
R	0	0	0	0	0	0	CLKR			0	0	0	0	0	CASC	
W																
Reset	0000_0000_0000_0000															
Offset	0x4_1300															

Figure 10-14. Timer Control Register (TCR)

Table 10-17 describes the TCR fields.

Table 10-17. TCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5–7	ROVR	Roll-over control. 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 base count register. 0 Timer does not roll over. When the count reaches zero, current count register is reloaded with the base count register value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, current count register 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)
8–21	—	Reserved, should be cleared.
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the system clock. The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64

Table 10-17. TCR Field Descriptions (continued)

Bits	Name	Description
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 Summary Registers

The summary registers can be read to determine which interrupt source was directed to the $\overline{\text{IRQ\_OUT}}$ output pin or caused a critical interrupt.

The $\overline{\text{IRQ\_OUT}}$ summary registers shown in Figure 10-15 and Figure 10-16 contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the EP field in its corresponding destination register is set).

The critical interrupt summary registers shown in Figure 10-17 and Figure 10-18 contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to the processor's critical interrupt signal *cint* (that is, if the CI field in its corresponding destination register is set).

10.3.3.1 $\overline{\text{IRQ\_OUT}}$ Summary Register 0 (IRQSR0)

Figure 10-15 shows the IRQSR0 fields.

**Figure 10-15. $\overline{\text{IRQ\_OUT}}$ Summary Register 0**

Table 10-18 describes the IRQSR0 fields.

Table 10-18. IRQSR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–19	MSG	Message interrupts 0–3. 0 Corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the EP field in its corresponding destination register is set)
20–31	EXT	External interrupts 0–11 0 Corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the EP field in its corresponding destination register is set)

10.3.3.2 $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Figure 10-16 shows the IRQSR1 fields.

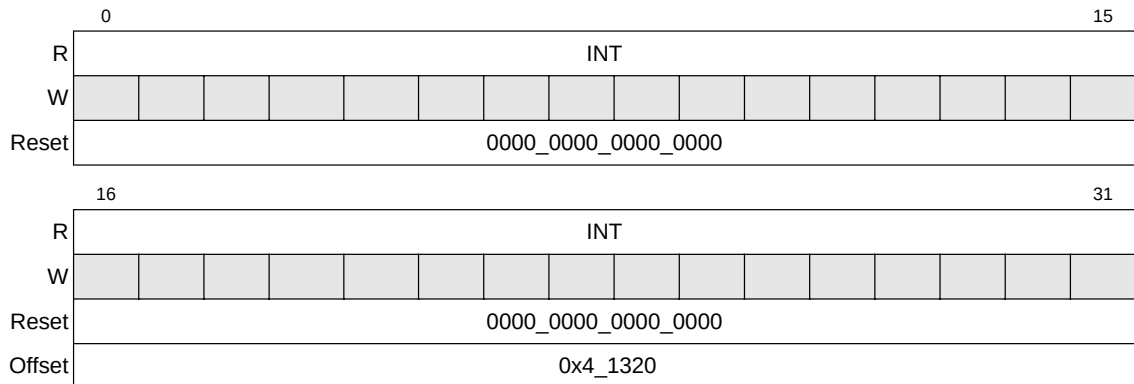


Figure 10-16. $\overline{\text{IRQ\_OUT}}$ Summary Register 1

Table 10-19 describes the IRQSR1 fields.

Table 10-19. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 Corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is directed to the $\overline{\text{IRQ\_OUT}}$ pin (that is, if the EP field in its corresponding destination register is set)

10.3.3.3 Critical Interrupt Summary Register 0 (CISR0)

Figure 10-17 shows the CISR0 fields.

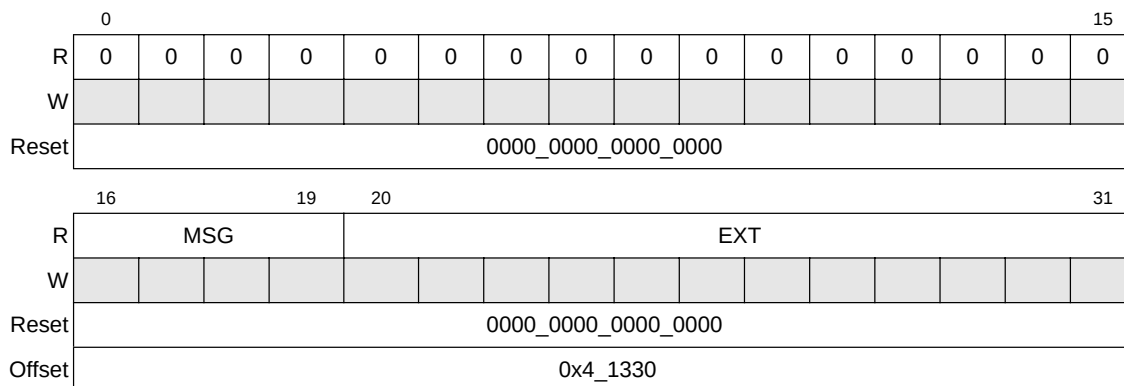


Figure 10-17. Critical Interrupt Summary Register 0

Table 10-20 describes the CISR0 fields.

Table 10-20. CISR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–19	MSG	Message interrupts 0–3 0 Corresponding interrupt is not active or not directed to <i>cint</i> . 1 The corresponding interrupt is active and is directed to the <i>cint</i> (that is, if the CI field in its corresponding destination register is set)
20–31	EXT	External interrupts 0–11 0 Corresponding interrupt is not active or not directed to <i>cint</i> . 1 The corresponding interrupt is active and is directed to the <i>cint</i> (that is, if the CI field in its corresponding destination register is set)

10.3.3.4 Critical Interrupt Summary Register 1 (CISR1)

Figure 10-18 shows the CISR1 fields.

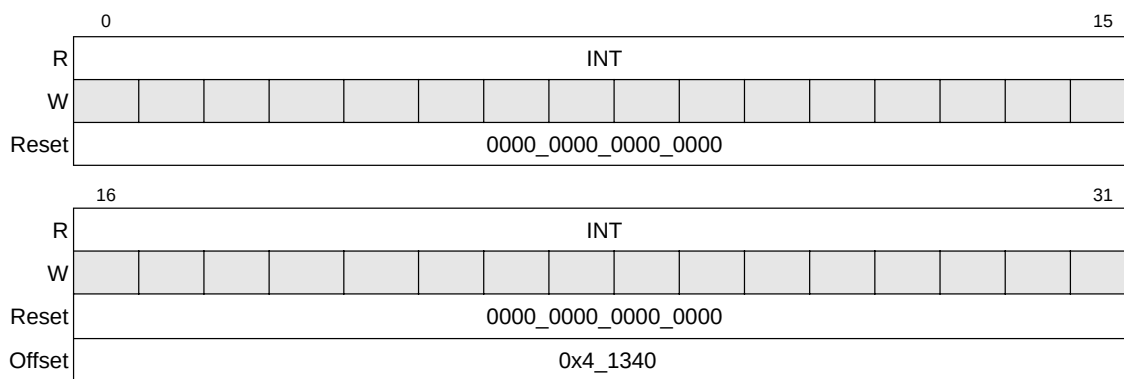


Figure 10-18. Critical Interrupt Summary Register 1

Table 10-21 describes the CISR1 fields.

Table 10-21. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 Corresponding interrupt is not active or not directed to <i>cint</i> . 1 The corresponding interrupt is active and is directed to the <i>cint</i> (that is, if the CI field in its corresponding destination register is set)

10.3.4 Performance Monitor Mask Registers (PMMRs)

The performance monitor mask registers (PMMRs) can be used to generate a performance monitor event for any one of the interrupt sources. The PMMRs consist of four pairs of 32-bit registers. Because only one bit per register pair can be unmasked, these four register pairs can be configured to monitor only four different interrupt sources.

10.3.4.1 Performance Monitor Mask Register (Even) (PMMR0/2/4/6)

Figure 10-19 shows the PMMR0/2/4/6 fields.

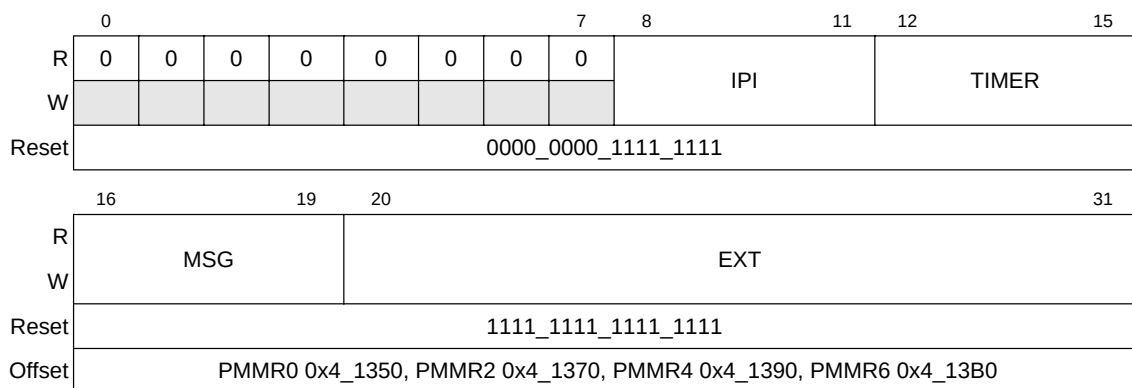


Figure 10-19. Performance Monitor Mask Registers (Even)

Table 10-22 describes the PMMR0/2/4/6 fields.

Table 10-22. PMMR0/2/4/6 Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8–11	IPI	IPI interrupts 0–3 1 Corresponding interrupt does not generate a performance monitor event. 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs.
12–15	TIMER	Timer interrupts 0–3 1 Corresponding interrupt does not generate a performance monitor event. 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs.

Table 10-22. PMMR0/2/4/6 Field Descriptions (continued)

Bits	Name	Description
16–19	MSG	Message interrupts 0–3 1 Corresponding interrupt does not generate a performance monitor event. 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs.
20–31	EXT	External interrupts 0–11 1 Corresponding interrupt does not generate a performance monitor event. 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs.

10.3.4.2 Performance Monitor Mask Registers (Odd) (PMMR1/3/5/7)

Figure 10-20 shows the PMMR1/3/5/7 fields.

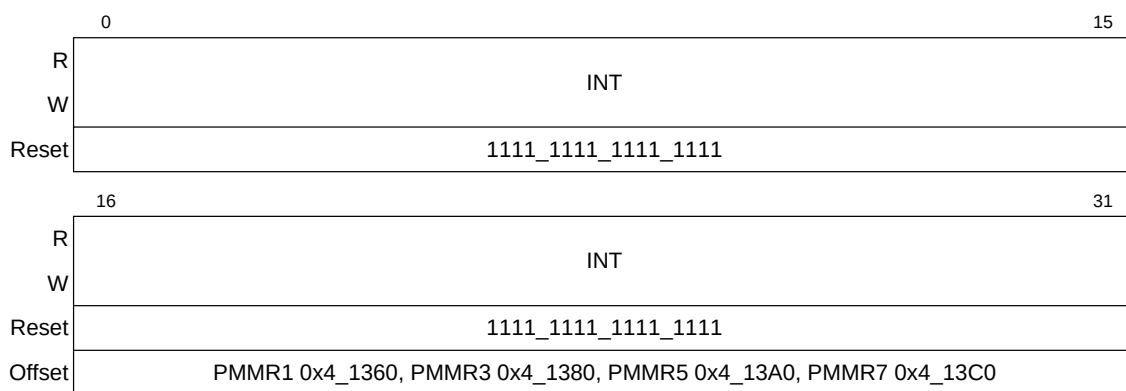
**Figure 10-20. Performance Monitor Mask Registers (Odd)**

Table 10-23 describes the PMMR1/3/5/7 fields.

Table 10-23. PMMR1/3/5/7 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 1 Corresponding interrupt does not generate a performance monitor event. 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs.

10.3.5 Message Registers

Writing to one of the four message registers (MSGR0 to MSGR3) causes a messaging interrupt to the processor. Reading this 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 MSR. The MSR format is shown in Figure 10-23.

10.3.5.1 Message Registers (MSGR0–MSGR3)

The message registers (MSGR0–MSGR3) contain a 32-bit message.

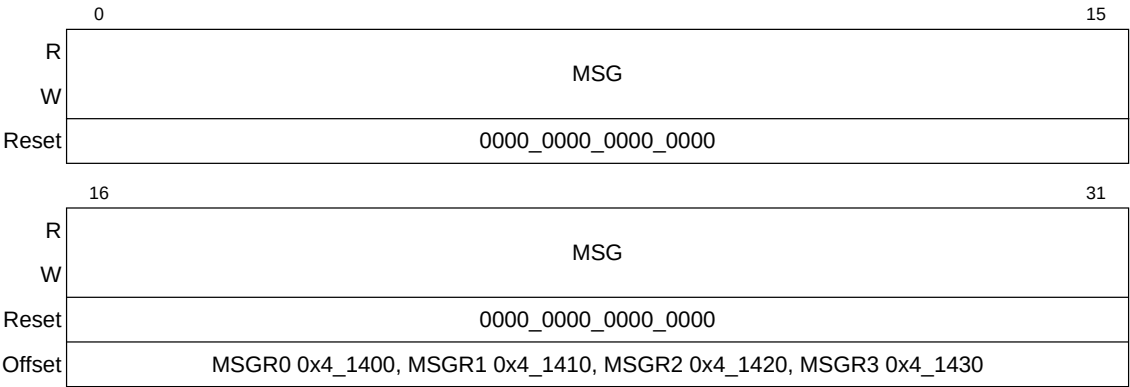


Figure 10-21. Message Registers (MSGRs)

Table 10-24 describes the MSGR fields.

Table 10-24. MSGRn Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data.

10.3.5.2 Message Enable Register (MER)

The message enable register (MER) shown in Figure 10-22 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.

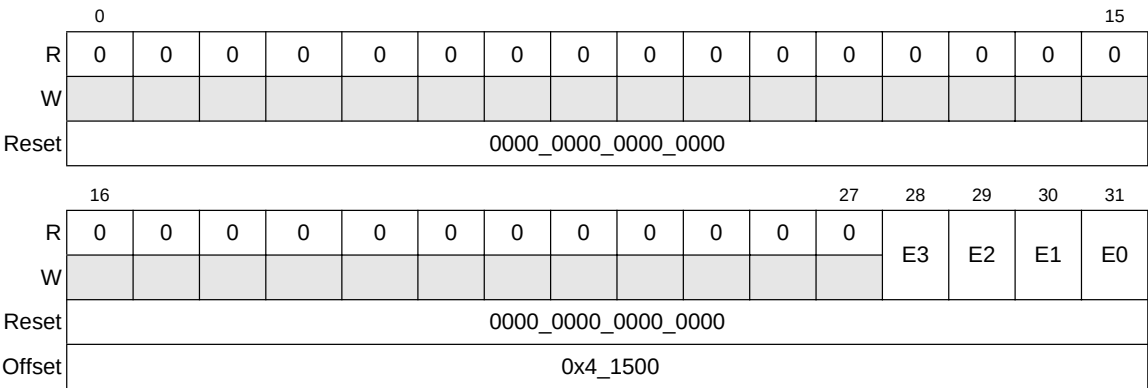


Figure 10-22. Message Enable Register (MER)

Table 10-25 describes the MER fields.

Table 10-25. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28	E3	Enable 3. Used to enable interrupt generation for MSGR3. 0 Interrupt generation for MSGR3 disabled 1 Interrupt generation for MSGR3 enabled
29	E2	Enable 2. Used to enable interrupt generation for MSGR2. 0 Interrupt generation for MSGR2 disabled 1 Interrupt generation for MSGR2 enabled
30	E1	Enable 1. Used to enable interrupt generation for MSGR1. 0 Interrupt generation for MSGR1 disabled 1 Interrupt generation for MSGR1 enabled
31	E0	Enable 0. Used to enable interrupt generation for MSGR0. 0 Interrupt generation for MSGR0 disabled 1 Interrupt generation for MSGR0 enabled

10.3.5.3 Message Status Register (MSR)

The message status register (MSR) shown in Figure 10-23 contains status bits for each message register. The 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.

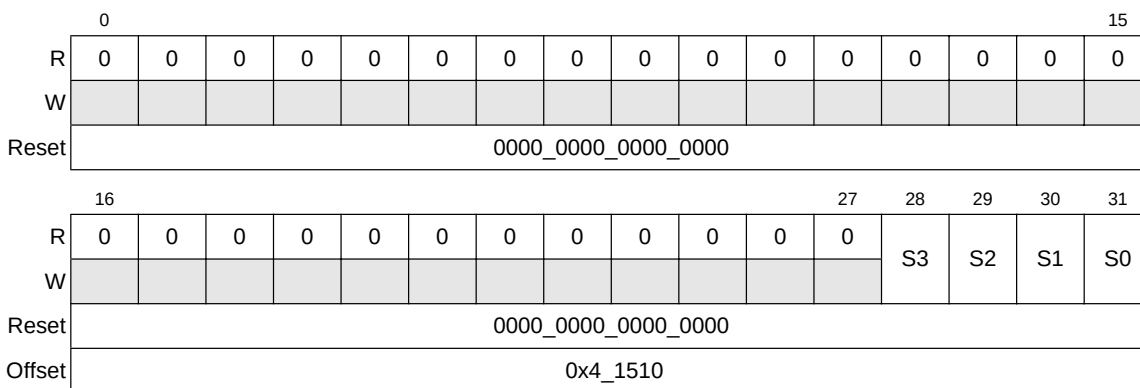
**Figure 10-23. Message Status Register (MSR)**

Table 10-26 describes the MSR fields.

Table 10-26. MSR Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28	S3	Status 3. Reports status of messaging interrupt 3. Writing a 1 clears this field. 0 Messaging interrupt 3 is not active. 1 Messaging interrupt 3 is active.

Table 10-26. MSR Field Descriptions (continued)

Bits	Name	Description
29	S2	Status 2. Reports status of messaging interrupt 2. Writing a 1 clears this field. 0 Messaging interrupt 2 is not active. 1 Messaging interrupt 2 is active.
30	S1	Status 1. Reports status of messaging interrupt 1. Writing a 1 clears this field. 0 Messaging interrupt 1 is not active. 1 Messaging interrupt 1 is active.
31	S0	Status 0. Reports status of messaging interrupt 0. Writing a 1 clears this field. 0 Messaging interrupt 0 is not active. 1 Messaging interrupt 0 is active

10.3.6 Interrupt Source Configuration Registers

The interrupt source configuration registers control the source of each interrupt, specifying parameters such as the interrupting event, signal polarity, and relative priority of each interrupt.

10.3.6.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The external interrupt vector/priority registers (EIVPRs) contain polarity and sense fields for the external interrupts caused by the assertion of any of IRQ[0:11] in addition to all the fields of the GTVPRs. The format of the EIVPRs is shown in Figure 10-24.

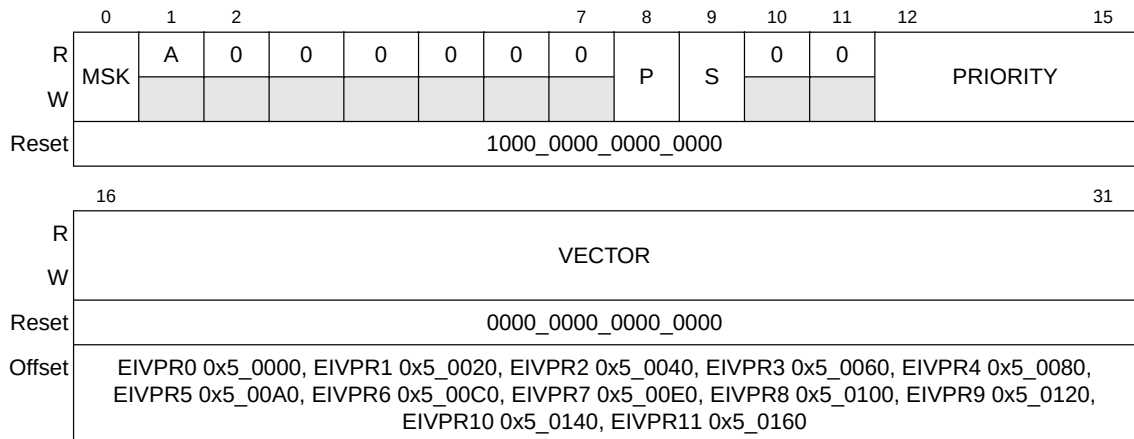
**Figure 10-24. External Interrupt Vector/Priority Registers (EIVPR0 to EIVPR11)**

Table 10-27 describes the EIVPR fields.

Table 10-27. EIVPR_n Field Descriptions

Bits	Name	Description
0	MSK	Mask. Masks interrupts from this source. 0 If the corresponding IPR bit is set an interrupt request is generated 1 Interrupts from this source are disabled.
1	A	Activity. A read-only field indicating that an interrupt has been requested or is in-service. Note: The P (polarity), S (sense), VECTOR and PRIORITY values should not be changed while the corresponding interrupt is active, that is, while EIVPR _n [A] is set. 0 No current interrupt activity associated with this source 1 The interrupt bit for this source is set in the IPR or ISR.
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
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 disables interrupts from this source.
16–31	VECTOR	Vector. Contains the value returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-36.

10.3.6.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The external interrupt destination registers (EIDRs) control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. All external interrupts are directed to the processor 0 interrupt, *int*, by default. The external interrupts can be selectively directed to $\overline{\text{IRQ\_OUT}}$ or to the processor 0 critical interrupt signal, *cint*, instead of *int* by writing to the appropriate fields in the external interrupt destination register.

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

Because external interrupts can be channeled only to processor 0, the P0 bit is permanently set. As shown in Figure 10-36 if either the CI or EP fields are set, the interrupt is not sent to the processor's interrupt input.

The EP or CI fields must be set only for level-sensitive interrupts. Setting these fields for edge-sensitive interrupts does not provide reliable interrupt response. The EIDRs field definition is shown in Figure 10-25.

	0	1	2												15
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
	16													30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W															
Reset	0000_0000_0000_0001														
Offset	EIDR0 0x5_0010, EIDR1 0x5_0030, EIDR2 0x5_0050, EIDR3 0x5_0070, EIDR4 0x5_0090, EIDR5 0x5_00B0, EIDR6 0x5_00D0, EIDR7 0x5_00F0, EIDR8 0x5_0110, EIDR9 0x5_0130, EIDR10 0x5_0150, EIDR11 0x5_0170,														

Figure 10-25. External Interrupt Destination Registers (EIDRs)

Table 10-28 describes the EIDR fields.

Table 10-28. EIDR_n Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows external interrupt to be serviced externally. 0 External interrupt is serviced internally with <i>int</i> signal to the processor core 1 External interrupt is directed to IRQ_OUT for external service
1	CI	Critical interrupt. 0 External interrupt is serviced internally with <i>int</i> signal to the processor core 1 External interrupt is directed to the processor 0 as a critical interrupt with the <i>cint</i> signal
2–30	—	Reserved, should be cleared.
31	P0	Processor 0. Permanently set and read only. Indicates that processor 0 will handle the interrupt. Only meaningful in a multi-core device. Because the MPC8540 is a single core device all interrupts that are serviced internally are always directed to processor 0. 0 Device is either defective or not an MPC8540 1 Interrupt directed to processor 0. Default value.

10.3.6.3 Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR31)

The internal interrupt vector/priority registers (IIVPRs) have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in Table 10-1. These interrupts are all level sensitive.

NOTE

Because all internal interrupts are active-high, clearing the polarity field of any IIVPR_n disables that interrupt. Care should be taken to ensure this field is not inadvertently corrupted when loading (or reloading) the IIVPRs with priority, mask, and vector data.

The format of the IIVPRs is shown in Figure 10-26.

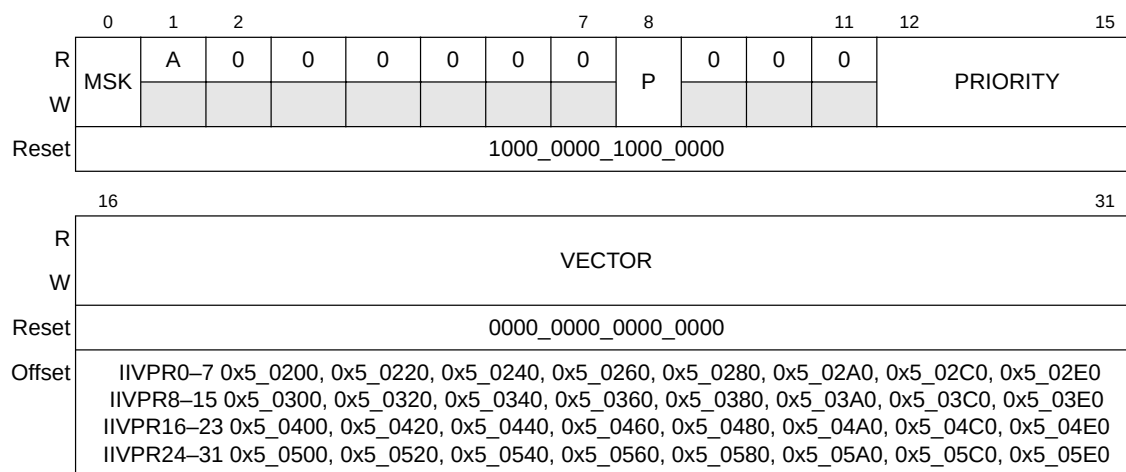


Figure 10-26. Internal Interrupt Vector/Priority Registers (IIVPRs)

Table 10-29 describes the IIVPR fields.

Table 10-29. IIVPR_n Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while IIVPR _n [A] is set. 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 the 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. This is the default value established at reset and should not be changed.
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 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-36.

10.3.6.4 Internal Interrupt Destination Registers (IIDR0–IIDR31)

The internal interrupt destination registers (IIDRs) have the same fields and format as the EIVDRs, except that they apply to the internal interrupt sources listed in Table 10-1.

Internal interrupts can only be directed to processor 0, just like external interrupts. The IIDR format is shown in Figure 10-27.

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

	0	1	2													15
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16														30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																
Reset	0000_0000_0000_0001															
Offset	IIDR0–7 0x5_0210, 0x5_0230, 0x5_0250, 0x5_0270, 0x5_0290, 0x5_02B0, 0x5_02D0, 0x5_02F0 IIDR8–15 0x5_0310, 0x5_0330, 0x5_0350, 0x5_0370, 0x5_0390, 0x5_03B0, 0x5_03D0, 0x5_03F0 IIDR16–23 0x5_0410, 0x5_0430, 0x5_0450, 0x5_0470, 0x5_0490, 0x5_04B0, 0x5_04D0, 0x5_04F0 IIDR24–31 0x5_0510, 0x5_0530, 0x5_0550, 0x5_0570, 0x5_0590, 0x5_05B0, 0x5_05D0, 0x5_05F0															

Figure 10-27. Internal Interrupt Destination Registers (IIDRs)

Table 10-30 describes the IIDR fields.

Table 10-30. IIDR_n Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows internal interrupt to be serviced externally. 0 Internal interrupt is serviced internally with <i>int</i> signal to the processor core 1 Internal interrupt is directed to <i>IRQ_OUT</i> for external service.
1	CI	Critical interrupt. Used to specify which interrupts should be handled as critical interrupts. More discussion of the distinction between normal and critical interrupts can be found at Section 10.4, "Functional Description." 0 Internal interrupt is serviced internally with <i>int</i> signal to the processor core 1 Internal interrupt is directed to the processor 0 as a critical interrupt with the <i>cint</i> signal
2–30	—	Reserved, should be cleared.
31	P0	Processor 0. Indicates that processor 0 will handle the interrupt. This bit is only meaningful in a multi-core device. Because the MCP8540 is a single core device all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0

10.3.6.5 Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3)

The messaging interrupt vector/priority registers (MIVPRs) have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

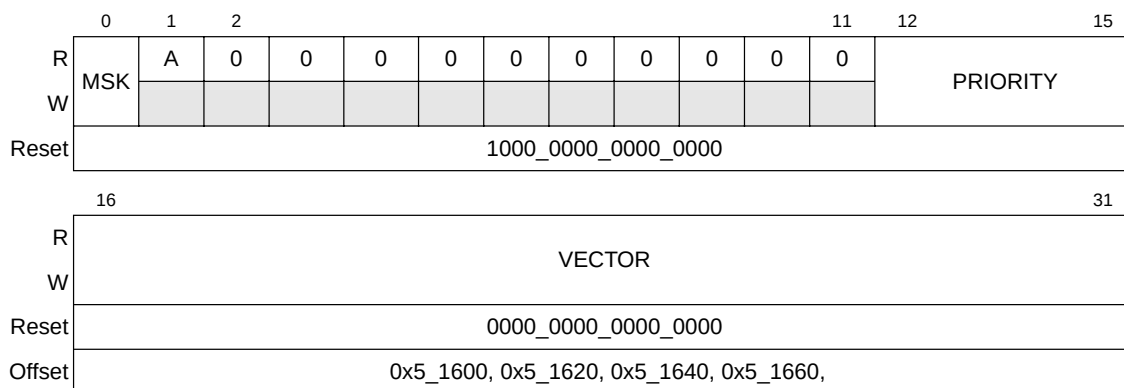
**Figure 10-28. Messaging Interrupt Vector/Priority Registers (MIVPRs)**

Table 10-31 describes the MIVPR fields.

Table 10-31. MIVPR_n Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. 0 An interrupt request is generated when the corresponding IPR field is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in-service. Note this field is read only. The VECTOR and PRIORITY values should not be changed while MIVPR _n [A] is set. 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 disables interrupts from this source.
16–31	VECTOR	Vector. The vector value in this field is returned when the interrupt acknowledge (IACK) register is read and this interrupt resides in the interrupt request register shown in Figure 10-36.

10.3.6.6 Messaging Interrupt Destination Registers (MIDR0–MIDR3)

The messaging interrupt destination registers (MIDRs) control the destination for the messaging interrupts. Shown in Figure 10-29, the MIDR enables the user to direct the interrupt to either the external interrupt output pin ($\overline{\text{IRQ\_OUT}}$), the core's critical interrupt input (*cint*), or to its normal interrupt input (*int*), respectively.

NOTE

The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

	0	1	2												15
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
	16													30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W															
Reset	0000_0000_0000_0001														
Offset	0x5_1610, 0x5_1630, 0x5_1650, 0x5_1670														

Figure 10-29. Messaging Interrupt Destination Registers (MIDRs)

Table 10-32 describes the MIDR fields.

Table 10-32. MIDR_n Field Descriptions

Bits	Name	Description
0	EP	External pin. Allows message interrupt to be serviced externally. 0 Message interrupt is serviced internally with <i>int</i> signal to the processor core. 1 Message interrupt is directed to <i>IRQ_OUT</i> for external service.
1	CI	Critical interrupt. Used to specify which interrupts should be handled as critical interrupts. More discussion of the distinction between normal and critical interrupts can be found at Section 10.4, “Functional Description.” 0 Message interrupt is serviced internally with <i>int</i> signal to the processor core. 1 Message interrupt is directed to the processor 0 as a critical interrupt with the <i>cint</i> signal.
2–30	—	Reserved, should be cleared.
31	P0	Processor 0. Indicates that processor 0 will handle the interrupt. This bit is only meaningful in a multi-core device. Because the MCP8540 is a single core device all interrupts that are serviced internally are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0

10.3.7 Per CPU Registers

The Open PIC programming model supports multiprocessor systems including up to 32 separate processors. As such, the Open PIC interface specification includes provisions for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. In support of this programming model, the MPC8540 PIC incorporates several of these multiprocessor capabilities. Because the value of features such as private address space for per CPU registers and interprocessor interrupts is fully realized only in a multi-core environment, their utility in this single-core device is not intuitive.

The registers listed in Table 10-33 are referred to as per CPU registers, because, in a multi-core device they would be duplicated for each core. The Open PIC 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 that initiates the transaction to determine which processor’s per

CPU registers to access. These addresses, shown in Table 10-33, appear in the memory map at the same offset for every processor and are called the private access space. Because the MPC8540 has only one core, there is only one set of per CPU registers, each register having two addresses. For example, the CTPR is located normally at 0x6_0080, and also at the private access address of 0x4_0080. While this double mapping seems superfluous on a single-core device, the purpose of this feature is to enable user code to execute correctly in an MP environment without needing to know which CPU it is running on. It is included on this device to simplify the porting of such code.

An example of how the different registers are addressed in a four-core device is illustrated in Figure 10-30. 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.

Table 10-33. Per CPU Registers—Private Access Address Offsets

Register Name	Offsets
IPI 0 dispatch register (IPIDR0)	0x4_0040
IPI 1 dispatch register (IPIDR1)	0x4_0050
IPI 2 dispatch register (IPIDR2)	0x4_0060
IPI 3 dispatch register (IPIDR3)	0x4_0070
Current task priority register (CTPR)	0x4_0080
Who am I register (WHOAMI0)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

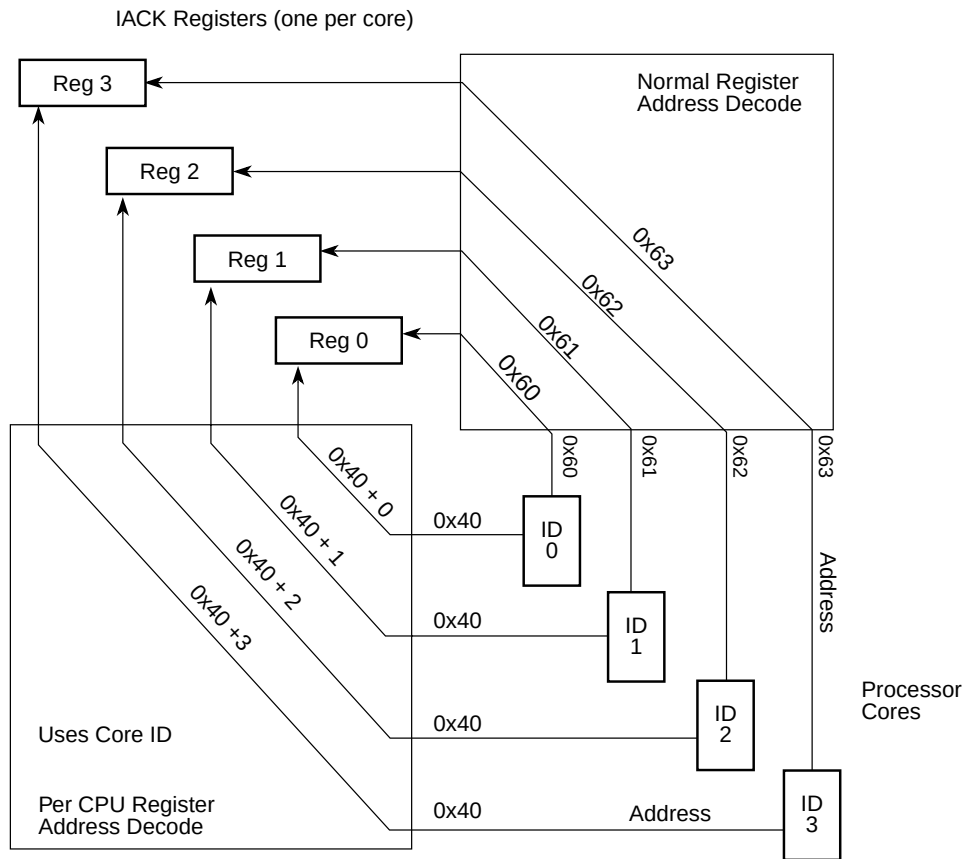


Figure 10-30. Per CPU Register Address Decoding in a Four-Core Device

10.3.7.1 Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)

There are four interprocessor interrupt dispatch registers (IPIDRs), one for each IPI channel, as shown in Figure 10-31. Writing to an IPI dispatch register with a bit set causes a self interrupt. While the concept of interprocessor interrupts apparently makes little sense in a single-core device, this feature can serve as a doorbell type interrupt because external bus masters can write to these registers.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16											27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W															P0	
Reset	0000_0000_0000_0000															
Offset	IPIDR0 0x6_0040, IPIDR1 0x6_0050 IPIDR2 0x6_0060, IPIDR3 0x6_0070															

Figure 10-31. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 10-34 describes the IPIDR fields.

Table 10-34. IPIDR_n Field Descriptions

Bits	Name	Description
0–30	—	Reserved, should be cleared.
31	P0	Processor 0. Determines if processor 0 receives the interrupt. (write only). 0 Processor 0 does not receive the interrupt 1 Directs the interrupt to processor 0

10.3.7.2 Processor Current Task Priority Register (CTPR)

There is one CTPR on this device shown in Figure 10-32. Software should write the priority of the current processor task in the CTPR. The PIC unit uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to the core if the following apply:

- It is not masked
- Its priority is higher than the values in the CTPR and ISR

Priority levels from 0 (lowest) to 15 (highest) are supported. Setting the task priority to 15 masks all interrupts to the processor. Hardware sets the task register to 0x0000_000F during reset or when the Px field of the processor initialization register is set.

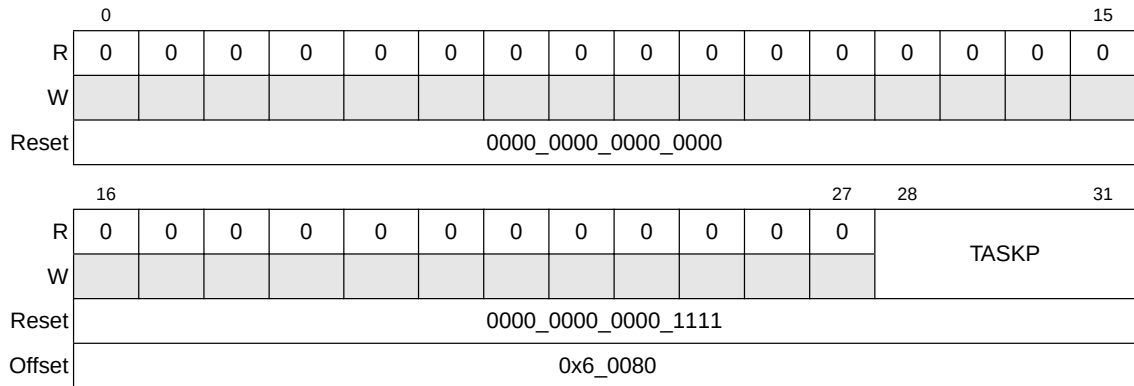
**Figure 10-32. Processor Current Task Priority Register (CTPR)**

Table 10-35 describes the CTPR fields.

Table 10-35. CTPR Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	TASKP	Task priority. This field is set from 0 to 15, where 15 corresponds to the highest priority for processor tasks. When CTPR[TASKP] = 0xF, no interrupts are signaled to the processor.

10.3.7.3 Who Am I Register (WHOAMI)

The processor who am I register (WHOAMI) can be read by the processor 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. The WHOAMI register is shown in Figure 10-33.

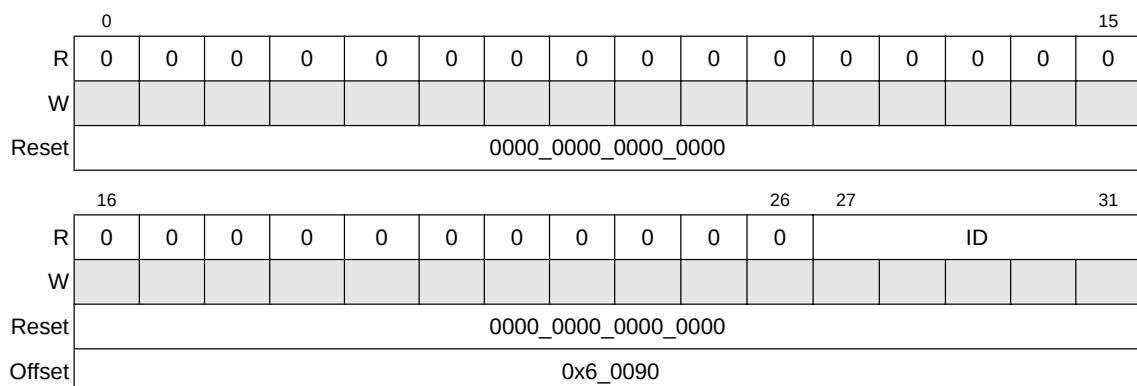
**Figure 10-33. Processor Who Am I Register (WHOAMI)**

Table 10-36 describes the WHOAMI fields.

Table 10-36. WHOAMI Field Descriptions

Bits	Name	Description
0–26	—	Reserved, should be cleared.
27–31	ID	ID. Returns the ID of the processor reading this register. Always returns zero.

10.3.7.4 Processor Interrupt Acknowledge Register (IACK)

In systems based on processors that implement the PowerPC architecture, the interrupt acknowledge function occurs as an explicit read request to a memory-mapped interrupt acknowledge register (IACK) in the PIC unit. Reading the interrupt acknowledge register returns the interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the interrupt pending register is cleared for edge-sensitive interrupts.
- The in-service register is updated.
- The interrupt signal (*int* or *cint*) to the processor is negated.

Reading this register when there is no interrupt pending returns the spurious vector value as described in Section 10.3.1.6, “Spurious Vector Register (SVR).” The IACK register is shown in Figure 10-34.

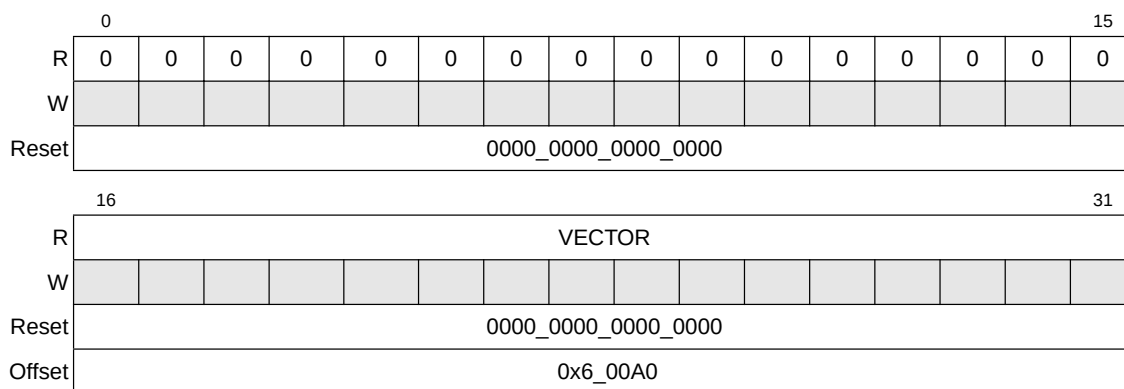
**Figure 10-34. Processor Interrupt Acknowledge Register (IACK)**

Table 10-37 describes the IACK fields.

Table 10-37. IACK Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared.
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only)

10.3.7.5 Processor End of Interrupt Register (EOI)

Writing to the end of interrupt (EOI) register shown in Figure 10-35 signals the end of processing for the highest-priority interrupt currently in service by the processor. The write to the EOI updates the ISR by retiring the highest priority interrupt. Data values written to this register are ignored, and zero is assumed.

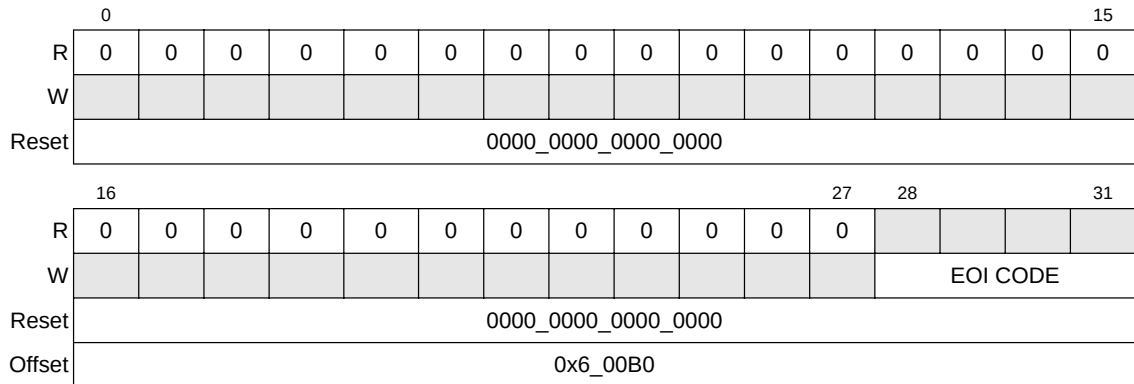


Figure 10-35. End of Interrupt Register (EOI)

Table 10-38 describes the EOI fields.

Table 10-38. EOI Field Descriptions

Bits	Name	Description
0–27	—	Reserved, should be cleared.
28–31	EOI CODE	0000 (write only)

10.4 Functional Description

A block diagram of the PIC unit is shown in Figure 10-36. The PIC unit receives interrupt signals from both external and internal sources. These signals are qualified and latched in the interrupt pending register (IPR). The output of the IPR feeds the interrupt selector (IS). The IS selects the interrupt with the highest priority, and provides this value to the interrupt request register (IRR). The IRR passes the output of the IS, and acts as a pipeline latch.

The interrupt router monitors the outputs of the IRR and other configuration registers. When the priority of the interrupt signal it receives is higher than the value in the task priority register, it asserts the interrupt signal to indicate an interrupt request to the processor. The interrupt handler executing on the processor acknowledges the interrupt by reading the IACK register, and the PIC unit returns the vector associated with the interrupt source to the processor.

Functional Description

An in-service register (ISR) keeps track of all the interrupts that are in-service, which is from the time its vector is received during an interrupt acknowledge (IACK) cycle, until it receives an end of interrupt (EOI) signal.

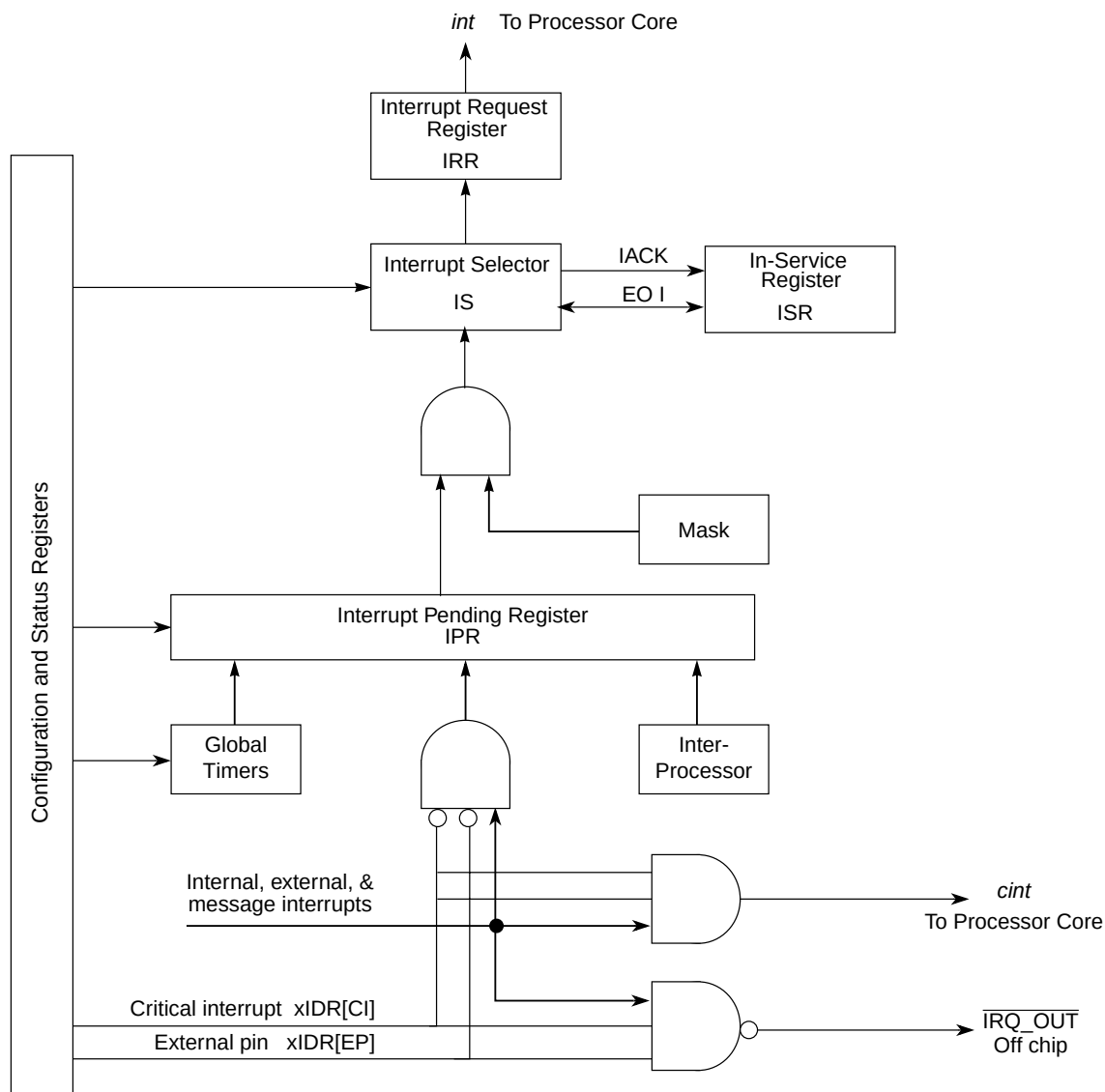


Figure 10-36. PIC Interrupt Processing Block Diagram

10.4.1 Interrupt Source Priority

Each interrupt source is assigned a priority value through its corresponding vector/priority register. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the interrupt source is greater than the current task priority. Therefore setting a source priority to zero inhibits that interrupt.

The PIC unit services simultaneous interrupts occurring with the same priority according to the following set order:

1. MSG0–MSG3
2. IPI0–IPI3
3. timer0–timer3
4. IRQ[0:11]
5. internal0–internal 31.

10.4.2 Processor Current Task Priority

The current task priority register is set by system software to indicate the relative importance of the task running on the processor. The processor does not receive interrupts with a priority level equal to or lower than its current task priority. Therefore setting the current task priority to 15 for a particular processor prevents the delivery of any interrupt to the processor.

10.4.3 Interrupt Acknowledge

The PIC unit notifies the processor core of an interrupt by asserting the *int* signal. When the processor core acknowledges the interrupt request by reading the interrupt acknowledge register (IACK) in the PIC unit, the PIC returns the 16-bit vector associated with the interrupt source to the processor. The interrupt is then considered to be in service, and remains in service until the processor performs a write to the PIC unit end of interrupt register (EOI). Writing to the EOI is referred to as an EOI cycle.

10.4.4 Nesting of Interrupts

If the processor is servicing an interrupt, it can only be interrupted again if the PIC receives an interrupt request from an interrupt source with a greater priority than the one currently being serviced.

Thus, although several interrupts may be simultaneously in service for the processor, the code currently executing is always handling the highest priority of all the interrupts that are being serviced. When the processor performs an EOI cycle, this highest priority interrupt is taken out of service. The next EOI cycle takes the next highest priority interrupt out of service, and so on.

10.4.5 Processor Initialization

The processor can reset itself by writing to the processor initialization register (PIR). This causes the assertion of the *sreset* output signal. When this occurs, the processor current task priority register also gets written to 0x000F to disable the delivery of any interrupts.

10.4.6 Spurious Vector Generation

Under certain circumstances, the PIC has no valid vector to return to the processor 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 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 in spite of the fact that the *int* signal has not been asserted by the PIC.

In all cases, a spurious vector is not returned if there is another pending interrupt that has sufficient priority to interrupt that processor. If such an interrupt is available, the vector for that interrupt source is returned. The EOI register should not be written in response to the spurious vector. Otherwise, a previously accepted interrupt might be cleared unintentionally.

10.4.7 Messaging Interrupts

There are four 32-bit message registers which can be used to send 32-bit messages to the processor. A messaging interrupt is generated by reading a message register if the corresponding enable bit in the message enable/status register is set, and the interrupt is not masked. Reading the message register or writing a 1 to the status bit clears the interrupt.

10.4.8 Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the four internal timers of the PIC unit. The timers can be individually programmed to generate a processor interrupt when they count down to zero and can be used to generate regular periodic interrupts. Each timer has four configuration and control registers. They are as follows:

- Global timer current count register (GTCCR)
- Global timer base count register (GTBCR)
- Global timer vector-priority register (GTVPR)
- Global timer destination register (GTDR)

The timer frequency should be written to the TFRR. (All of the timers run at this frequency.) Refer to Section 10.3.2.1, “Timer Frequency Reporting Register (TFRR),” for a description of this register.

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 option also exists to change the timer frequency by setting the appropriate fields of the TCR. See Section 9.3.2.6, “Timer Control Register (TCR).”

10.4.9 Reset

The PIC unit is reset by setting GCR[RST] which causes the following:

- All pending and in-service interrupts are cleared
- All interrupt mask bits are set
- Polarity, sense, external pin, critical interrupt, and activity fields, are reset to their default values.
- TCR, MER and MSR cleared.
- MSG and timer destination fields are set. The IPI dispatch register is cleared.
- All timer base count values are reset to zero and count inhibited
- The CTPR[TASKP] is reset to 0xF, thus disabling interrupt delivery to the processor
- The spurious vector resets to 0xFFFF
- The PMMRs are reset to 0xFFFF.
- The PIC defaults to the pass-through mode
- All other registers remain at their pre-reset programmed values.

The GCR[RST] bit is automatically cleared when the reset sequence is complete.

10.5 Initialization/Application Information

10.5.1 Programming Guidelines

10.5.1.1 PIC Registers

The main accesses to the PIC unit include interrupt and timer initialization and reading the interrupt acknowledge register (IACK) which results in the PIC unit returning the vector associated with the interrupt that is to be serviced.

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

Initialization/Application Information

- IPI dispatch registers and the EOI registers, which return zeros on reads
- Activity bit (A) of the vector/priority registers, which returns the value according to the status of the current interrupt source
- IACK register, which returns the vector of highest priority which is currently pending, or the spurious vector
- Reserved fields always return 0

The following guidelines are recommended when the PIC unit is programmed in mixed mode (GCR[M] is set):

- If the processor's memory management unit (MMU) is enabled, all PIC registers must be located in a cache-inhibited and guarded area.
- 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. Set the processor current task priority register (CTPR) value to zero.
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 FPR[NIRQ]
 - While counter > 0, perform IACK and EOIs to guarantee all the interrupt pending and in-service registers are cleared.
6. Set the processor CTPR value to the desired value.

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 having to handle a false interrupt for this case, software can clear the PIC interrupt pending register 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, low-level if it's a negative-edge source (while the mask bit remains set). Once this is done, configuring the external interrupt source as edge-sensitive will not cause a false interrupt.

10.5.1.2 Changing Interrupt Source Configuration

In order to change the vector, priority, polarity, sense or destination of an active (unmasked) interrupt source, the following sequence should be performed:

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.

10.5.1.3 MSG, IPI, and Timer Priority Levels

The priority value assigned to each active MSG, IPI, and timer channel must be unique, that is, different from the values used for all other interrupt sources.

Chapter 11

I²C Interface

11.1 Overview

The inter-IC (IIC or 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 the system and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus minimizes the interconnections between devices. The synchronous, multi-master bus of the I²C 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.

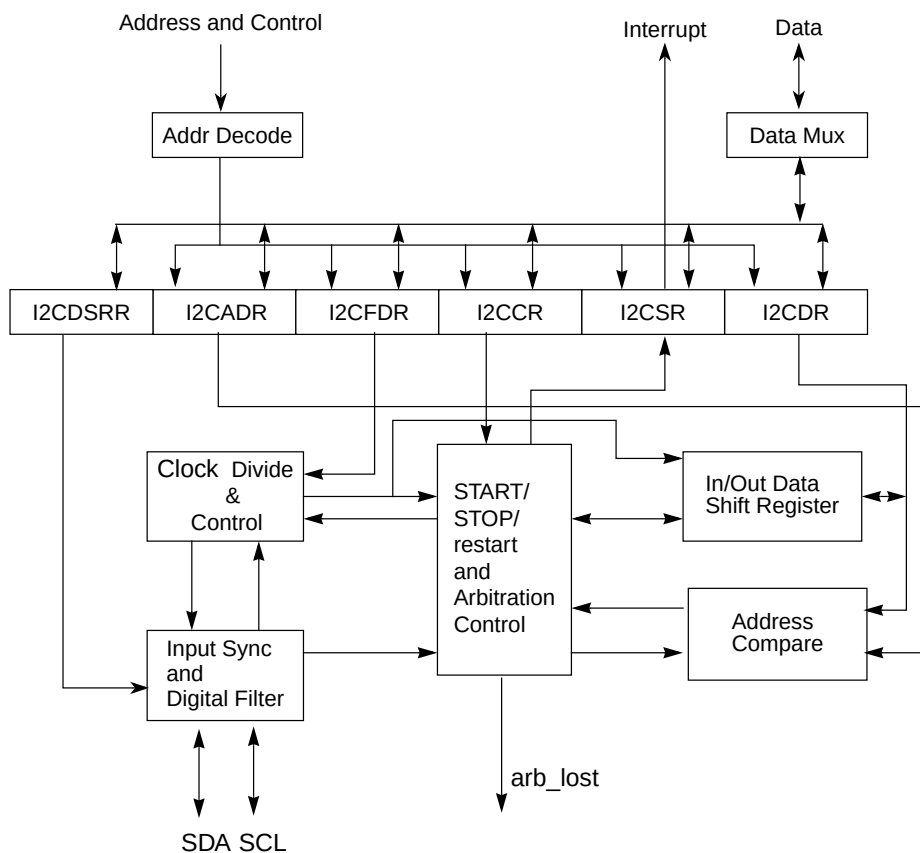


Figure 11-1. I²C Block Diagram

11.1.1.1 Features

The I²C interface includes the following features:

- Two-wire interface
- Multi-master operational
- 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

11.1.2 Modes of Operation

- 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.
- START condition. This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves. This mode is I²C-specific.
- Repeated START condition. A START condition that is generated without a STOP condition to terminate the previous transfer. This mode is I²C-specific.
- STOP condition. The master can terminate the transfer by generating a STOP condition to free the bus. This mode is I²C-specific.
- 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/W bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode. The MPC8540 is in boot sequencer mode when first powered up, unless bypass boot sequencer mode is selected. This mode is used to initialize the configuration registers in the MPC8540 after the MPC8540 is initialized and the boot sequencer mode is disabled.

11.2 External Signal Description

The following sections give an overview of the signals and provide detailed signal descriptions.

11.2.1 Signal Overview

The I²C interface uses the SDA signal and SCL signals for data transfer. All devices that are connected to these two signals must have open-drain or open-collector outputs. A logical AND function is performed on both signals with external pull-up resistors. Note that the signal patterns driven on SDA represent address, data, or read/write information at different stages of the protocol.

Table 11-1 summarizes the two signals that comprise the external I²C interface.

Table 11-1. I²C Interface Signal Description

Signal Name	Idle State	I/O	State Meaning
Serial Clock (SCL)	HIGH	I	When the I ² C 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 I ² C module drives SCL along with SDA when transmitting. As a slave, the I ² C module drives SCL low for data pacing.
Serial Data (SDA)	HIGH	I	When the I ² C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I ² C devices on SDA. The bus is assumed to be busy when SDA is detected low.
		O	When writing as a master or slave, the I ² C module drives data on SDA synchronous to SCL.

11.2.2 Detailed Signal Descriptions

These two signals—SDA and SCL 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 *MPC8540 Hardware Specification* for the electrical characteristics of these signals.

The SDA and SCL signals are described in the following sections.

Table 11-2 contains the detailed descriptions of the external signals of the I²C controller.

Table 11-2. External I²C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description
SDA	I/O	Serial Data. This signal performs as an input when the MPC8540 is in a receiving mode. SDA also performs as an output signal when the MPC8540 is transmitting (as an I ² C master or a slave).
	O	As outputs for the bi-directional serial data, these signals operate as described below.
		State Meaning Asserted/Negated— Data is driven.
	I	As inputs for the bi-directional 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.
SCL	I/O	Serial Clock. This signal performs as an input when the MPC8540 is programmed as an I ² C slave. SCL also performs as an output when the MPC8540 is programmed as an I ² C master.
	O	As outputs for the bi-directional 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 bi-directional serial clock, these signals operate as described below.
		State Meaning Asserted/Negated—The I ² C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.

11.3 Memory Map/Register Definition

Table 11-3 lists the I²C-specific registers and their addresses.

Table 11-3. I²C Memory Map

Address	I ² C Register	Access	Reset	Section/Page
0x0_3000	I ² C address register (I2CADR)	R/W	0x00	8.3.1.1/8-4
0x0_3004	I ² C frequency divider register (I2CFDR)	R/W	0x00	8.3.1.2/8-5
0x0_3008	I ² C control register (I2CCR)	R/W	0x00	8.3.1.3/8-6
0x0_300C	I ² C status register (I2CSR)	R/W	0x81	8.3.1.4/8-8
0x0_3010	I ² C data register (I2CDR)	R/W	0x00	8.3.1.5/8-10
0x0_3014	Digital filter sampling rate register (I2CDFSRR)	R/W	0x00	8.3.1.6/8-10

11.3.1 Register Descriptions

This section describes the I²C registers in detail.

NOTE

Reserved bits should always be written with the value they returned when read. In other words, the register should be programmed by reading the value, modifying the 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).

11.3.1.1 I²C Address Register (I2CADR)

Figure 11-2 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.

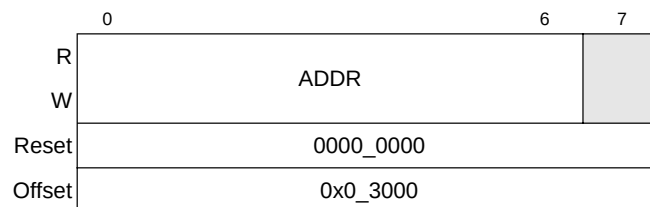


Figure 11-2. I²C Address Register (I2CADR)

Table 11-4 describes the bit settings of I2CADR.

Table 11-4. I2CADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I ² C interface. Note that the default mode of the I ² C interface is slave mode for an address match.
7	—	Reserved

11.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 11-3 shows the sampling rate and the clock bit rate for the I²C unit.

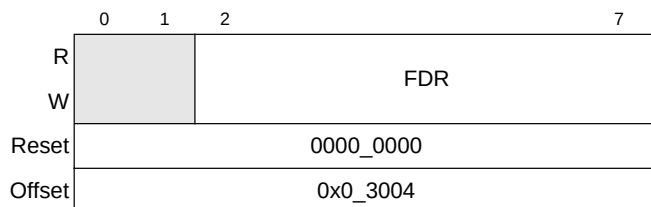
**Figure 11-3. I²C Frequency Divider Register (I2CFDR)**

Figure 11-6 describes the bit settings of I2CFDR. It also maps the I2CFDR[FDR] field to the clock divider values.

Table 11-5. 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 the CCB clock divided by the 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: <table><tr><th>FDR</th><th>Divider (Decimal)</th><th>FDR</th><th>Divider (Decimal)</th></tr><tr><td>0x00</td><td>288</td><td>0x20</td><td>160</td></tr><tr><td>0x01</td><td>320</td><td>0x21</td><td>192</td></tr><tr><td>0x02</td><td>384</td><td>0x22</td><td>224</td></tr><tr><td>0x03</td><td>480</td><td>0x23</td><td>256</td></tr><tr><td>0x04</td><td>576</td><td>0x24</td><td>320</td></tr><tr><td>0x05</td><td>640</td><td>0x25</td><td>384</td></tr><tr><td>0x06</td><td>768</td><td>0x26</td><td>448</td></tr><tr><td>0x07</td><td>960</td><td>0x27</td><td>512</td></tr><tr><td>0x08</td><td>1152</td><td>0x28</td><td>640</td></tr><tr><td>0x09</td><td>1280</td><td>0x29</td><td>768</td></tr><tr><td>0x0A</td><td>1536</td><td>0x2A</td><td>896</td></tr><tr><td>0x0B</td><td>1920</td><td>0x2B</td><td>1024</td></tr><tr><td>0x0C</td><td>2304</td><td>0x2C</td><td>1280</td></tr><tr><td>0x0D</td><td>2560</td><td>0x2D</td><td>1536</td></tr><tr><td>0x0E</td><td>3072</td><td>0x2E</td><td>1792</td></tr><tr><td>0x0F</td><td>3840</td><td>0x2F</td><td>2048</td></tr><tr><td>0x10</td><td>4608</td><td>0x30</td><td>2560</td></tr><tr><td>0x11</td><td>5120</td><td>0x31</td><td>3072</td></tr><tr><td>0x12</td><td>6144</td><td>0x32</td><td>3584</td></tr><tr><td>0x13</td><td>7680</td><td>0x33</td><td>4096</td></tr><tr><td>0x14</td><td>9216</td><td>0x34</td><td>5120</td></tr><tr><td>0x15</td><td>10240</td><td>0x35</td><td>6144</td></tr><tr><td>0x16</td><td>12288</td><td>0x36</td><td>7168</td></tr><tr><td>0x17</td><td>15360</td><td>0x37</td><td>8192</td></tr><tr><td>0x18</td><td>18432</td><td>0x38</td><td>10240</td></tr><tr><td>0x19</td><td>20480</td><td>0x39</td><td>12288</td></tr><tr><td>0x1A</td><td>24576</td><td>0x3A</td><td>14336</td></tr><tr><td>0x1B</td><td>30720</td><td>0x3B</td><td>16384</td></tr><tr><td>0x1C</td><td>36864</td><td>0x3C</td><td>20480</td></tr><tr><td>0x1D</td><td>40960</td><td>0x3D</td><td>24576</td></tr><tr><td>0x1E</td><td>49152</td><td>0x3E</td><td>28672</td></tr><tr><td>0x1F</td><td>61440</td><td>0x3F</td><td>32768</td></tr></table>	FDR	Divider (Decimal)	FDR	Divider (Decimal)	0x00	288	0x20	160	0x01	320	0x21	192	0x02	384	0x22	224	0x03	480	0x23	256	0x04	576	0x24	320	0x05	640	0x25	384	0x06	768	0x26	448	0x07	960	0x27	512	0x08	1152	0x28	640	0x09	1280	0x29	768	0x0A	1536	0x2A	896	0x0B	1920	0x2B	1024	0x0C	2304	0x2C	1280	0x0D	2560	0x2D	1536	0x0E	3072	0x2E	1792	0x0F	3840	0x2F	2048	0x10	4608	0x30	2560	0x11	5120	0x31	3072	0x12	6144	0x32	3584	0x13	7680	0x33	4096	0x14	9216	0x34	5120	0x15	10240	0x35	6144	0x16	12288	0x36	7168	0x17	15360	0x37	8192	0x18	18432	0x38	10240	0x19	20480	0x39	12288	0x1A	24576	0x3A	14336	0x1B	30720	0x3B	16384	0x1C	36864	0x3C	20480	0x1D	40960	0x3D	24576	0x1E	49152	0x3E	28672	0x1F	61440	0x3F	32768
FDR	Divider (Decimal)	FDR	Divider (Decimal)																																																																																																																																			
0x00	288	0x20	160																																																																																																																																			
0x01	320	0x21	192																																																																																																																																			
0x02	384	0x22	224																																																																																																																																			
0x03	480	0x23	256																																																																																																																																			
0x04	576	0x24	320																																																																																																																																			
0x05	640	0x25	384																																																																																																																																			
0x06	768	0x26	448																																																																																																																																			
0x07	960	0x27	512																																																																																																																																			
0x08	1152	0x28	640																																																																																																																																			
0x09	1280	0x29	768																																																																																																																																			
0x0A	1536	0x2A	896																																																																																																																																			
0x0B	1920	0x2B	1024																																																																																																																																			
0x0C	2304	0x2C	1280																																																																																																																																			
0x0D	2560	0x2D	1536																																																																																																																																			
0x0E	3072	0x2E	1792																																																																																																																																			
0x0F	3840	0x2F	2048																																																																																																																																			
0x10	4608	0x30	2560																																																																																																																																			
0x11	5120	0x31	3072																																																																																																																																			
0x12	6144	0x32	3584																																																																																																																																			
0x13	7680	0x33	4096																																																																																																																																			
0x14	9216	0x34	5120																																																																																																																																			
0x15	10240	0x35	6144																																																																																																																																			
0x16	12288	0x36	7168																																																																																																																																			
0x17	15360	0x37	8192																																																																																																																																			
0x18	18432	0x38	10240																																																																																																																																			
0x19	20480	0x39	12288																																																																																																																																			
0x1A	24576	0x3A	14336																																																																																																																																			
0x1B	30720	0x3B	16384																																																																																																																																			
0x1C	36864	0x3C	20480																																																																																																																																			
0x1D	40960	0x3D	24576																																																																																																																																			
0x1E	49152	0x3E	28672																																																																																																																																			
0x1F	61440	0x3F	32768																																																																																																																																			

11.3.1.3 I²C Control Register (I2CCR)

Figure 11-4 shows the modes of the I²C interface that are controlled by I2CCR.

	0	1	2	3	4	5	6	7
R	MEN	MIEN	MSTA	MTX	TXAK			
W						RSTA		BCST
Reset	0000_0000							
Offset	0x0_3008							

Figure 11-4. I²C Control Register (I2CCR)

Table 11-6 describes the bit settings of the I2CCR

Table 11-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I ² C 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 I ² C module is enabled. This bit must be set before any other control register bits have any effect. All I ² C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I ² C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I ² C 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 a 1 to a 0, 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 a 0 to a 1, a START condition is generated on the bus, and the 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 will always be 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 I ² C 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 bit after receiving one byte of data. 1 No acknowledge signal response (high value on SDA) is sent.
5	RSTA	Repeat 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. 0 No START condition is generated 1 Generates repeat START condition.
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability. 1 Enables the I ² C to accept broadcast messages at address zero.

11.3.1.4 I²C Status Register (I2CSR)

The status register, shown in Figure 11-5, 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.

	0	1	2	3	4	5	6	7
R	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
W								
Reset	1000_0001							
Offset	0x0_300C							

Figure 11-5. I²C Status Register (I2CSR)

Table 11-7 describes the bit settings of the I2CSR

Table 11-7. 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 two following conditions— when I2CSR 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[MIE] 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 I ² C bus is idle 1 I ² C 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 will also be set if the I ² C drives an address of all 0s and broadcast mode is enabled.

Table 11-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
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.
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] is set). 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 I2CADDR 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

11.3.1.5 I²C Data Register (I2CDR)

The I2C data register is shown in Figure 11-6.

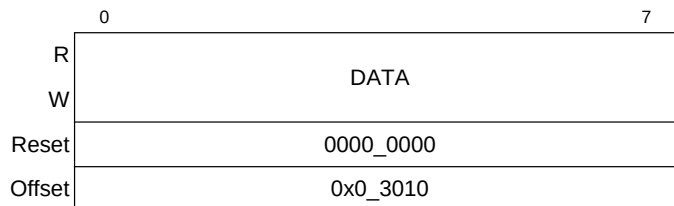
**Figure 11-6. I²C Data Register (I2CDR)**

Table 11-8 shows the bit descriptions for I2CDR.

Table 11-8. 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 I ² C 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 the master receive mode, reading the data register allows the read to occur, but also allows the I ² C module to receive the next byte of data on the I ² C interface. In slave mode, the same function is available after it is addressed.

11.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 11-7.

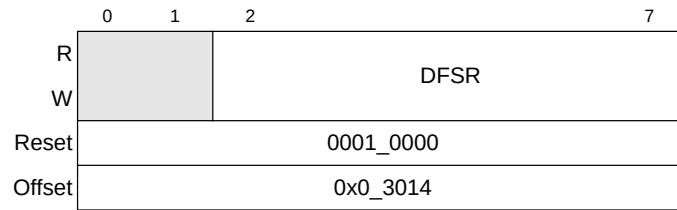


Figure 11-7. I2CDFSRR Format

Table 11-9 shows the field descriptions for I2CDFSRR.

Table 11-9. 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 I ² C bus. The resulting sampling rate is calculated by dividing the platform frequency by the non-zero value of DFSR. If I2CDFSRR is set to zero, the I ² C bus sample points default to the reset divisor 0x10.

11.4 Functional Description

The reset state of the I²C is the boot sequencer mode. After the boot sequencer has completed, the I²C interface will perform as a slave receiver. The I²C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter address.

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 11-8 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 subsections.

Functional Description

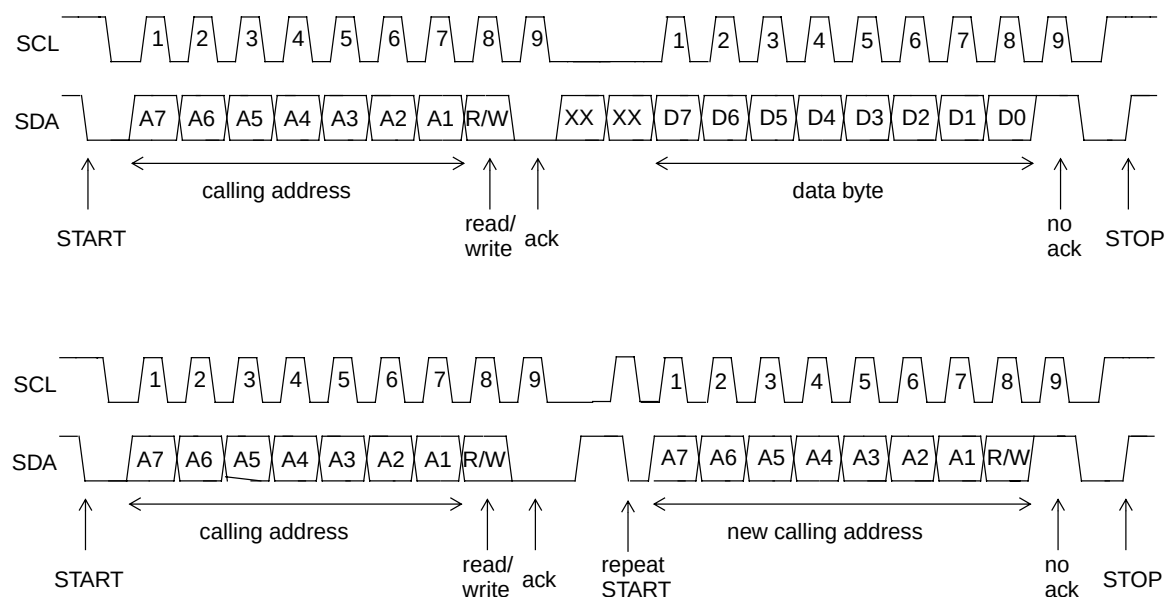


Figure 11-8. I²C Interface Transaction Protocol

11.4.1 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 11-8, 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 the I2CCR[MSTA].

11.4.2 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.

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 11-8. 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 $\overline{R/W}$ 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 will not check the $\overline{R/W}$ bit. The second byte of the broadcast message is the master device ID. 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 device ID is for another receiver device and the third byte is a write command, then software can ignore the third byte during the broadcast. If the device ID 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 11-8. 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 ninth clock. Therefore, one complete data byte transfer takes nine 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.

11.4.3 Repeated START Condition

Figure 11-8 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.

11.4.4 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 11-8. 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 11.4.3, “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.

11.4.5 Arbitration Procedure

The I²C 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 (including the I²C module) clock synchronization procedure 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 I²C 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.

Arbitration is lost (and I2CSR[MAL] is set) in the following circumstances:

- SDA sampled as low when the master drives a high during address or data-transmit cycle.
- SDA sampled as low when the master drives a high during the acknowledge bit of a data-receive cycle.
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.

Note that the I²C module does not automatically retry a failed transfer attempt.

If the I²C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I²C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I²C 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 I²C interface losing arbitration, after which bus operations return to normal.

11.4.6 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, synchronized clock 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.

11.4.7 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold the 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.

11.4.8 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.

11.4.9 Description of I²C Functional Blocks

The following is a description of the I²C blocks shown in Figure 11-1.

11.4.9.1 Clock Control

The clock control block handles requests from clock 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

11.4.9.2 Input Synchronization and Digital Filter

The following sections describe synchronizing of input signal, and filtering of SCL and SDA lines in details.

11.4.9.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

11.4.9.2.2 Filtering of SCL and SDA Lines

The inputs to the SCL and SDA pins are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to 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. The duration of the sampling cycle is controlled by a down counter and a signal that sets at the end of the count. This allows a software write to the frequency register to control the filtered sampling rate.

11.4.9.3 Transaction Monitoring

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.

11.4.9.4 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.

Arbitration by the master in master mode is lost under the following conditions—

- Low detected when high expected (transmit)
- Ack bit, low detected when high expected (receive)

- A START condition is attempted when the bus is busy
- A START condition is attempted when the bus is nearly busy
- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

11.4.9.5 Control Transfer

The I²C 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 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

11.4.9.6 Address Compare

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 four performed address comparisons are described as follows:

- The address sent by the current master matches the general broadcast address
- The address matches either the broadcast address or the slave address; in either case, slave mode results
- 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

11.4.10 Boot Sequencer Mode

At power on reset (POR), the I²C controller defaults to the boot sequencer mode. In this mode, the MPC8540 communicates with one or more EEPROMs through the I²C interface. This EEPROM(s) contains data for the MPC8540 configuration registers. The EEPROM(s) can be programmed to initialize one or more configuration registers.

This is an extension of the standard I²C interface that uses more address bits to allow for EEPROMs that have more than 256 bytes. During this initialization, the extended I²C interface is supported. In this mode, only one EEPROM may be used, and the maximum number of registers is limited by the size of the EEPROM. If the standard I²C interface is used, the I²C module addresses the first EEPROM, and reads 256 bytes. Then it issues a repeated start and addresses the next EEPROM address. This sequence continues until the last register is detected. If the last register is not detected before wrapping back to the first address, an error condition is detected. The I²C continues to read from the EEPROM as long as the continue (CONT) bit is set. This is a field in address/attributes that is transferred from the EEPROM, as described in Section 11.4.10.1, “EEPROM Calling Address,” via a configuration pin on the MPC8540. There should be no other I²C traffic when the boot sequencer is active.

Note that although the I²C block defaults to the boot sequencer mode at POR, the actual boot sequencer used by the MPC8540 at power-up is determined by the LGPL3 and LGPL5 reset configuration pins of the MPC8540. Also note that the default value for the LGPL3 and LGPL5 reset configuration pins is 0b11, which corresponds to the I²C boot sequencer being disabled at power-up. See Section 4.4.3.6, “Boot Sequencer Configuration,” for more information.

11.4.10.1 EEPROM Calling Address

The MPC8540 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 EEPROM(s) are used, they are addressed in sequential order.

11.4.10.2 EEPROM Data Format

The I²C module expects a particular format for data to be programmed in the EEPROM. A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C will check 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 11-9. The first 3 bytes hold the attributes and address offset. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of transaction. Byte enables should be asserted for any byte that will be written, 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 least significant byte of data (data[24:31]).

By asserting 'ACS,' an alternate config space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If the CONT bit is cleared, the first 3 bytes, including ACS, the byte enables, and the address, should be cleared to 0. Also, the data contains the final CRC. A CRC-32 algorithm is used to check the integrity of the data. The following polynomial is used:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

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 I²C asserts an error signal that can be turned into a hard reset request (*hreset_req*) outside of the I²C module. If there is a preamble fail, the boot sequencer may continue to pull the I²C pins low, until the hard reset occurs.

Functional Description

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 11-9. EEPROM Data Format for One Register Preload Command

Figure 11-10 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]								
*								
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]								
CRC[8:15]								
CRC[16:23]								
CRC[24:31]								

Figure 11-10. EEPROM Contents

11.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. It also includes Figure 11-11, a recommended flowchart for the 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. Also, an **msync** assembly instruction should 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.

11.5.1 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 platform clock.
3. Update I2CADDR 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.

11.5.2 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 the data register (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 I2CSR[MIF] = 1 at any time, the I²C interrupt handler should immediately handle the interrupt.

11.5.3 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; an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIE] = 1). In the interrupt handler, software must do the following:

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 11.5.8, “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] should be toggled at this stage. See Section 11.5.8, “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] should 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] = 1), 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 (I2CSR[MAAS] = 0), I2CSR[SRW] is not valid and I2CCR[MTX] should be read to determine the direction of the current transfer. See Section 11.5.8, “Interrupt Service Routine Flowchart,” for more details.

11.5.4 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, which is done by setting the transmit acknowledge (I2CCR[TXAK]) bit 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 will not receive the data acknowledge when I2CCR[TXAK] is set. For 1-byte transfers, a dummy read should be performed by the interrupt service routine. (See

Section 11.5.8, “Interrupt Service Routine Flowchart.”) Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I²C automatically generates a STOP if I2CCR[TXAK] = 1. Therefore, I2CCR[TXAK] must be set to a 1 before allowing the I²C module to receive the last data byte on the I²C bus. Eventually, I2CCR[TXAK] will need to be cleared again for subsequent I2C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

11.5.5 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 by setting I2CCR[RSTA].

11.5.6 Generation of SCL When SDA Low

In some cases it is necessary to force the MPC8540 to become the I²C bus master out of reset and drive the SCL signal (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, the SDA signal can be driven low by another I²C device while the MPC8540 is coming out of reset and will stay low indefinitely. The following procedure can be used to force the MPC8540 to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I²C and set the master bit by setting I2CCR to 0x20.
2. Enable the I²C by setting I2CCR to 0xA0.
3. Read the I2CDR.
4. Return the MPC8540 to slave mode by setting I2CCR to 0x80.

11.5.7 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] = 1, 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 I2CSR[MAAS] automatically. The only time I2CSR[MAAS] is read as set is from the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers will have I2CSR[MAAS] = 0. 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.

11.5.7.1 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] = 1), 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 11.5.8, “Interrupt Service Routine Flowchart.”

11.5.7.2 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 test I2CSR[MAL] first, and the software should clear I2CSR[MAL] if it is set. See Section 11.4.9.4, “Arbitration Control,” for more information.

11.5.8 Interrupt Service Routine Flowchart

Figure 11-11 shows an example algorithm for an I²C interrupt service routine. Deviation from the flowchart may result in unpredictable I²C bus behavior. However, in the slave receive mode (not shown in the flowchart), the interrupt service routine may need to set I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that a **sync** instruction follow each I²C register read or write to guarantee in-order instruction execution.

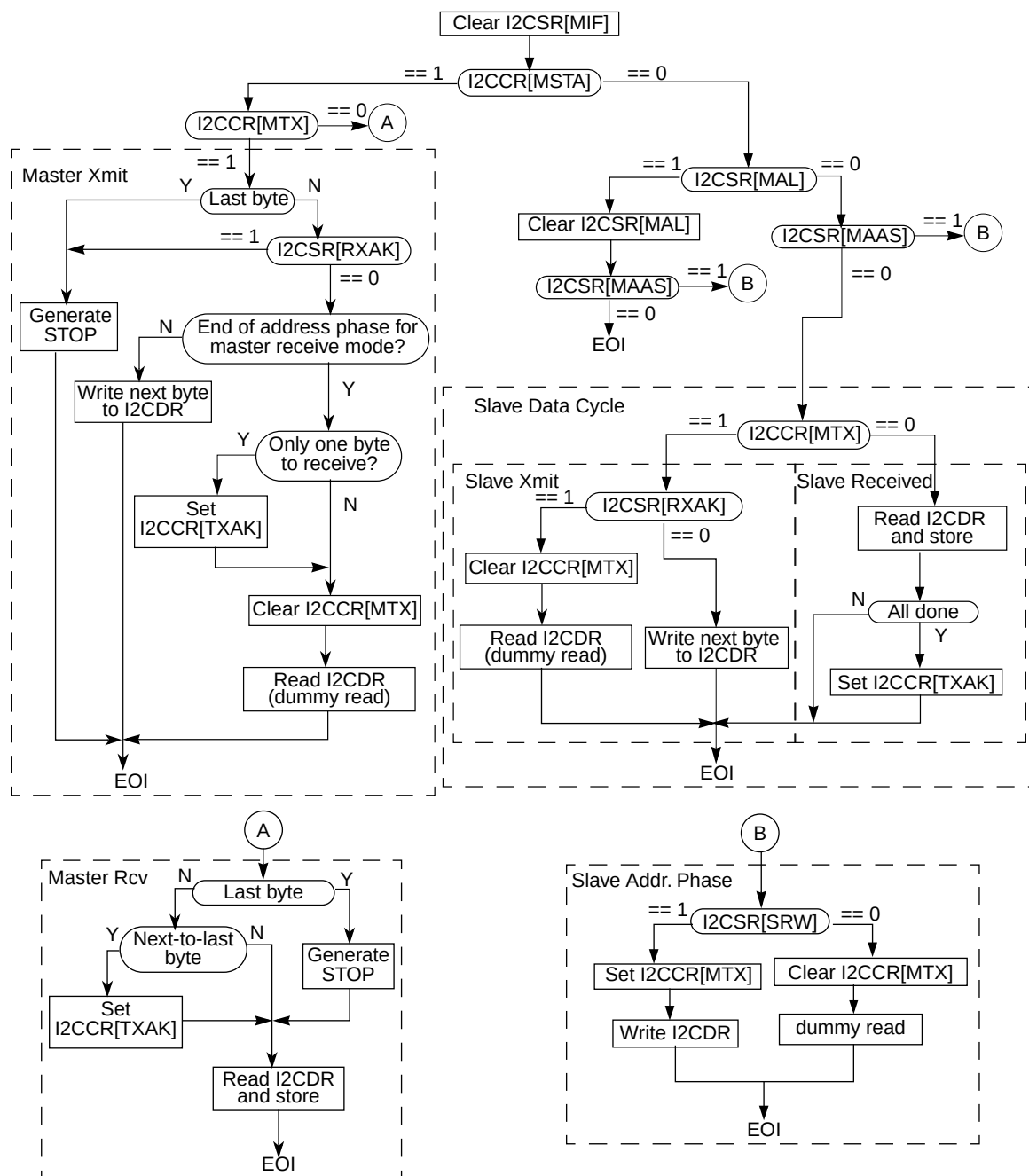


Figure 11-11. Example I²C Interrupt Service Routine Flowchart

Chapter 12

DUART

This chapter describes the two (dual) universal asynchronous receiver/transmitters (UARTs) of the MPC8540. It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

12.1 Overview

The DUART of the MPC8540 consists of two (dual) 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 core complex bus (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 12-1, 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
- Sixteen-bit counter for baud rate generation
- Interrupt control logic

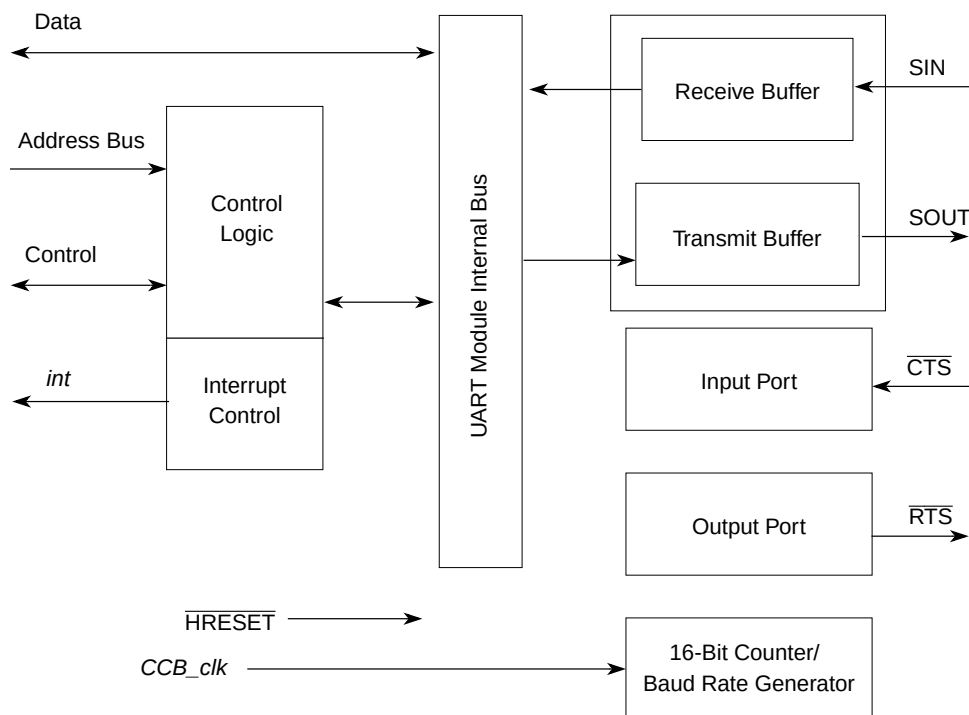


Figure 12-1. DUART Block Diagram

12.1.1 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 CCB clock by 1 to $(2^{16} - 1)$ and generate a 16x clock for the transmitter and receiver engines
- Clear to send ($\overline{CTS}$) and ready to send ($\overline{RTS}$) 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

12.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). The data is placed directly into an internal transmitter shift register of the transmitter FIFO in FIFO mode. 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.

12.2 External Signal Description

This section contains a signal overview and detailed signal descriptions.

12.2.1 Signal Overview

Table 12-1 summarizes the DUART signals.

Table 12-1. DUART Signal Overview

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[0:1]	I	2	1	Serial in data UART0 and UART1
UART_SOUT[0:1]	O	2	1	Serial out data UART0 and UART1
UART_CTS[0:1]	I	2	1	Clear to send UART0 and UART1
UART_RTS[0:1]	O	2	1	Request to send UART0 and UART1

12.2.2 Detailed Signal Descriptions

The eight DUART signals are described in detail in Table 12-2.

Table 12-2. 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 its 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, <i>rxcnt</i> , 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 loop-back 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, <i>rxcnt</i> , uses the frequency of the baud-rate generator to update and drive the data on SOUT.
UART_CTS[0:1]	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective 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 CCB clock
UART_RTS[0:1]	O	Request to send. UART_RT _{Sx} 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 (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 CCB clock

12.3 Memory Map/Register Definition

There are two complete sets of DUART registers (one for UART0 and one for UART1). The two UARTs on the device are identical, except that the registers for UART1 are located at offsets 0x0_4500 (local), and the registers for UART1 are located at offsets 0x0_4600 (local). 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 14 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 12.3.1.7, “Line Control Registers (ULCR0 and ULCR1)” for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Read and writes to these registers must be byte wide operations. Table 12-3 provides a register summary with references to the section and page that contains detailed information about each register.

Table 12-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
0x0_4500	ULCR[DLAB] = 0 UART0 receiver buffer register (URBR)	R	0x00	12.3.1.1/12-6
0x0_4500	ULCR[DLAB] = 0 UART0 transmitter holding register (UTHR)	W	0x00	12.3.1.2/12-6
0x0_4500	ULCR[DLAB] = 1 UART0 divisor least significant byte register (UDLB)	R/W	0x00	12.3.1.3/12-7
0x0_4501	ULCR[DLAB] = 0 UART0 interrupt enable register (UIER)	R/W	0x00	12.3.1.4/12-8
0x0_4501	ULCR[DLAB] = 1 UART0 divisor most significant byte register (UDMB)	R/W	0x00	12.3.1.3/12-7
0x0_4502	ULCR[DLAB] = 0 UART0 interrupt ID register (UIIR)	R	0x01	12.3.1.5/12-9
0x0_4502	ULCR[DLAB] = 0 UART0 FIFO control register (UFCR)	W	0x00	12.3.1.6/12-11
0x0_4502	ULCR[DLAB] = 1 UART0 alternate function register (UAFR)	R/W	0x00	12.3.1.12/12-17
0x0_4503	ULCR[DLAB] = x UART0 line control register (ULCR)	R/W	0x00	12.3.1.7/12-12
0x0_4504	ULCR[DLAB] = x UART0 MODEM control register (UMCR)	R/W	0x00	12.3.1.8/12-13
0x0_4505	ULCR[DLAB] = x UART0 line status register (ULSR)	R	0x60	12.3.1.9/12-14
0x0_4506	ULCR[DLAB] = x UART0 MODEM status register (UMSR)	R	0x00	12.3.1.10/12-16
0x0_4507	ULCR[DLAB] = x UART0 scratch register (USCR)	R/W	0x00	12.3.1.11/12-16
0x0_4510	ULCR[DLAB] = x UART0 DMA status register (UDSR)	R	0x01	12.3.1.13/12-17
0x0_4600	ULCR[DLAB] = 0 UART1 receiver buffer register (URBR)	R	0x00	12.3.1.1/12-6
0x0_4600	ULCR[DLAB] = 0 UART1 transmitter holding register (UTHR)	W	0x00	12.3.1.2/12-6
0x0_4600	ULCR[DLAB] = 1 UART1 divisor least significant byte register (UDLB)	R/W	0x00	12.3.1.3/12-7
0x0_4601	ULCR[DLAB] = 0 UART1 interrupt enable register (UIER)	R/W	0x00	12.3.1.4/12-8
0x0_4601	ULCR[DLAB] = 1 UART1 divisor most significant byte register (UDMB)	R/W	0x00	12.3.1.3/12-7
0x0_4602	ULCR[DLAB] = 0 UART1 interrupt ID register (UIIR)	R	0x01	12.3.1.5/12-9
0x0_4602	ULCR[DLAB] = 0 UART1 FIFO control register (UFCR)	W	0x00	12.3.1.6/12-11
0x0_4602	ULCR[DLAB] = 1 UART1 alternate function register (UAFR)	R/W	0x00	12.3.1.12/12-17
0x0_4603	ULCR[DLAB] = x UART1 line control register (ULCR)	R/W	0x00	12.3.1.7/12-12
0x0_4604	ULCR[DLAB] = x UART1 MODEM control register (UMCR)	R/W	0x00	12.3.1.8/12-13
0x0_4605	ULCR[DLAB] = x UART1 line status register (ULSR)	R	0x60	12.3.1.9/12-14
0x0_4606	ULCR[DLAB] = x UART1 MODEM status register (UMSR)	R	0x00	12.3.1.10/12-16
0x0_4607	ULCR[DLAB] = x UART1 scratch register (USCR)	R/W	0x00	12.3.1.11/12-16
0x0_4610	ULCR[DLAB] = x UART1 DMA status register (UDSR)	R	0x01	12.3.1.13/12-17

12.3.1 Register Descriptions

The following sections describe the MPC8540 UART0 and UART1 registers.

12.3.1.1 Receiver Buffer Registers (URBR0 and URBR1)

These registers contains 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 12.3.1.9, “Line Status Registers (ULSR0 and ULSR1).” Figure 12-3 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

Figure 12-2 shows the bits in the URBRs.

	0	7
R	DATA	
W		
Reset	0x00	
Offset	0x0_4500, 0x0_4600	

Figure 12-2. Receiver Buffer Registers (URBR0 and URBR1)

Table 12-4 describes the fields of URBR.

Table 12-4. URBR Field Definition

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus. [Read only]

12.3.1.2 Transmitter Holding Registers (UTHR0 and UTHR1)

A write to these 8-bit registers will cause the UART devices to transfer 5 to 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 will be the first byte onto the bus. UDSR[TXRDY] indicates when the FIFO is full. Refer to the Table 12-21 and the Table 12-22 for more details.

Figure 12-3 shows the bits in the UTHRs.

	0	7
R		
W	DATA	
Reset	0x00	
Offset	0x0_4500, 0x0_4600	

Figure 12-3. Transmitter Holding Registers (UTHR0 and UTHR1)

Table 12-5 describes the fields of UTHR.

Table 12-5. UTHR Field Definition

Bits	Name	Description
0–7	DATA	Data that is written to UTHR. [Write only]

12.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB, UDLB)

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 x [UDMB:UDLB]). Equivalently, [UDMB:UDLB:4'b0000] = platform clock frequency / desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 12-8.

Figure 12-4 shows the bits in the UDMBs.

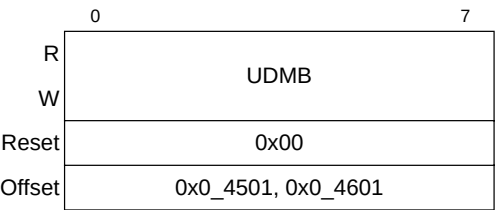


Figure 12-4. Divisor Most Significant Byte Registers (UDMB0 and UDMB1)

Table 12-6 describes the fields of UDMB registers.

Table 12-6. UDMB Field Definition

Bits	Name	Description
0–7	UDMB	Divisor most significant byte

Figure 12-5 shows the bits in the UDLBs.

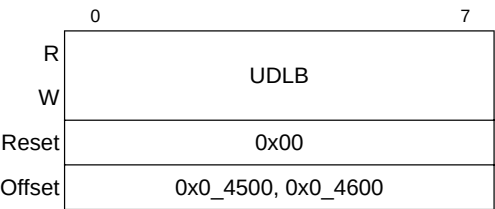


Figure 12-5. Divisor Least Significant Byte Registers (UDLB0 and UDLB1)

Table 12-7 describes the fields of UDLB registers.

Table 12-7. UDLB Field Definition

Bits	Name	Description
0–7	UDLB	Divisor least significant byte. This is concatenated with UDMB.

Table 12-8 shows baud rate when the input clock is at certain frequencies.

Table 12-8. Baud Rate Examples

Baud Rate (Decimal)	Divisor		CCB Frequency (MHz)	Percent Error (Decimal)
	Decimal	Hex		
9,600	1736	6C8	266	0.0064
19,200	868	364	266	0.0064
38,400	434	1B2	266	0.0064
56,000	298	12A	266	0.1280
128,000	130	82	266	0.1600
256,000	65	41	266	0.1600
9,600	2170	87A	333	0.0064
19,200	1085	43D	333	0.0064
38,400	543	21F	333	0.0858
56,000	372	174	333	0.0064
128,000	163	A3	333	0.1472
256,000	81	51	333	0.4672

To get the percent error value, the following three steps are taken:

1. The input clock frequency (ICF) is divided by the actual frequency input (AFI) to get the correct divisor value (ICF/AFI where $AFI = \text{baud rate} \times 16$).
2. The divisor value is subtracted from 1.
3. The result from the step two is multiplied by 100 to calculate the final percent error. The result is calculated in absolute value (no negative numbers).

These steps can be described with the following equation:

$$\text{Percent error value} = (1 - AFI/ICF) \times 100$$

12.3.1.4 Interrupt Enable Register (UIER)

The UIER gives the user the ability to mask specific UART interrupts to the MPC8540 interrupt controller.

Figure 12-6 shows the bits in the UIER.

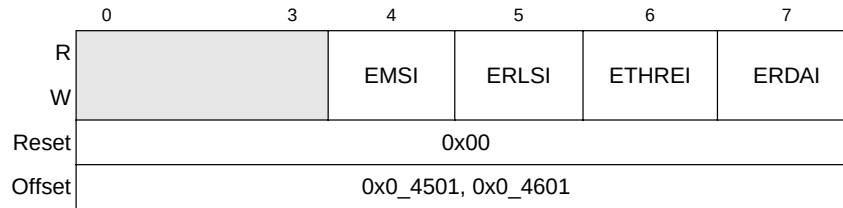
**Figure 12-6. Interrupt Enable Register (UIER)**

Table 12-9 describes the fields of UIER.

Table 12-9. UIER Field Definitions

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.

12.3.1.5 Interrupt ID Registers (UIR0 and UIR1)

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 12-11 for more details.

Memory Map/Register Definition

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 12-7 shows the bits in the UIIR.

	0	1	2	3	4	5	6	7
R	FE	—	IID3	IID2	IID1	IID0		
W								
Reset	0x01							
Offset	0x0_4502, 0x0_4602							

Figure 12-7. Interrupt ID Registers (UIIR)

Table 12-10 describes the fields of the UIIR.

Table 12-10. UIIR Field Definitions

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 12-11. 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 12-11.
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 12-11.

Table 12-11. 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	Reading the line status register
0b0100	Second	Received data available	Receiver Data Available or Trigger Level reached in FIFO mode	Reading the receiver buffer register or if the number of bytes in the receiver FIFO drops below the trigger level.
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time	Reading the receiver buffer register

Table 12-11. UIIR IID Bits Summary

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0010	Third	UTHR empty	Transmitter Holding Register is empty	Reading the UIIR or writing to the UTHR
0b0000	Fourth	MODEM status	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Reading the UMSR

12.3.1.6 FIFO Control Registers (UFCR0 and UFCR1)

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 12-8 shows the bits in the UFCRs.

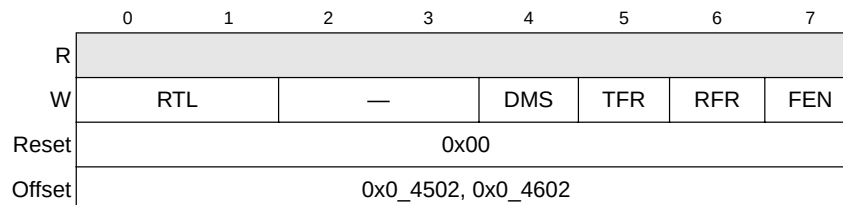
**Figure 12-8. FIFO Control Registers (UFCR)**

Table 12-12 describes the fields of the UFCRs.

Table 12-12. UFCR Field Definitions

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
4	DMS	DMA mode select. See Section 12.4.5.2, “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.

Table 12-12. UFCR Field Definitions

Bits	Name	Description
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.

12.3.1.7 Line Control Registers (ULCR0 and ULCR1)

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 12-14 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 12-9 shows the bits in the ULCRs.

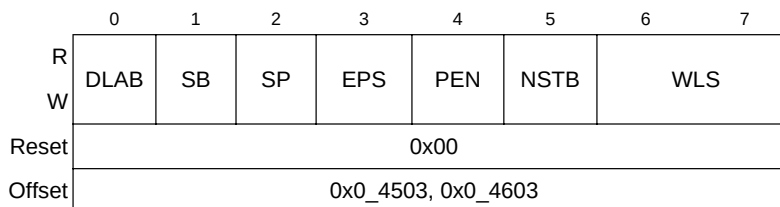
**Figure 12-9. Line Control Register (ULCR)**

Table 12-13 describes the fields of the ULCRs.

Table 12-13. ULCR Field Definitions

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 12-14 for more information. 0 If PEN = 1 and SP = 0 then odd parity is selected. 1 If PEN = 1 and SP = 0 then 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 1/2 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 12-14. 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

12.3.1.8 MODEM Control Registers (UMCR0 and UMCR1)

The UMCRs control the interface with the external peripheral device on the UART bus.

Figure 12-10 shows the bits in the UMCRs.

	0	2	3	4	5	6	7
R			LOOP			RTS	
W							
Reset	0x00						
Offset	0x0_4504, 0x0_4604						

Figure 12-10. Modem Control Register (UMCR)

Table 12-15 describes the fields of UMCRs.

Table 12-15. UMCR Field Definitions

Bits	Name	Description
0–2	—	Reserved
3	LOOP	Local loop-back 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 UART_RTS output. Informs external MODEM or peripheral that the UART is ready for sending/receiving data.
7	—	Reserved

12.3.1.9 Line Status Registers (ULSR0 and ULSR1)

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 12-11 shows the bits in the ULSRs.

	0	1	2	3	4	5	6	7
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0x60							
Offset	0x0_4505, 0x0_4605							

Figure 12-11. Line Status Register (ULSR)

Table 12-16 describes the fields of the ULSRs.

Table 12-16. ULSR Field Definitions

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 (i.e., 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. Note that the ULSR[BI] is set immediately after ULSR is read if bus remains zero and no mark state followed by a valid new character has been detected
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 will receive the following new data.
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.

12.3.1.10 MODEM Status Registers (UMSR0 and UMSR1)

The UMSRs track the status of the MODEM (or external peripheral device) clear to send ($\overline{\text{CTS}}$) signal for the corresponding UART.

Figure 12-12 shows the bits in the UMSRs.

	0	2	3	4	6	7
R	—		CTS	—		DCTS
W						
Reset	0x00					
Offset	0x0_4506, 0x0_4606					

Figure 12-12. Modem Status Register (UMSR)

Table 12-17 describes the fields of the UMSRs.

Table 12-17. UMSR Field Definitions

Bit	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 $\overline{\text{CTS}}$ is negated. 1 Corresponding $\overline{\text{CTS}}$ is asserted. The MODEM or peripheral device is ready for data transfers.
4–6	—	Reserved
7	DCTS	Delta clear to send 0 No change on the corresponding $\overline{\text{CTS}}$ signal since the last read of UMSR[CTS]. 1 The $\overline{\text{CTS}}$ value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition.

12.3.1.11 Scratch Registers (USCR0 and USCR1)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not effect the operation of the DUART.

Figure 12-13 shows the bits in USCRs.

	0	7
R		
W	DATA	
Reset	0x00	
Offset	0x0_4507, 0x0_4607	

Figure 12-13. Scratch Register (USCR)

Table 12-18 describes the fields of the USCRs.

Table 12-18. USCR Field Definitions

Bits	Name	Description
0–7	DATA	Data

12.3.1.12 Alternate Function Registers (UAFR0 and UAFR1)

The UAFRs give software the ability to write to both UART0 and UART1 registers simultaneously with the same write operation. The UAFRs also provide a means for the device's performance monitor to track the baud clock.

Figure 12-14 shows the bits in the UAFRs.

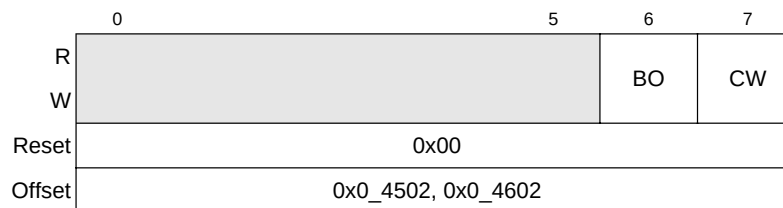
**Figure 12-14. Alternate Function Register (UAFR)**

Table 12-19 describes the fields of the UAFRs.

Table 12-19. UAFR Field Definitions

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.

12.3.1.13 DMA Status Registers (UDSR0 and UDSR1)

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 12-15 shows the bits in UDSRs.

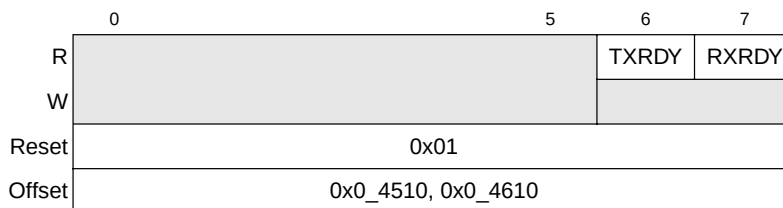
**Figure 12-15. DMA Status Register (UDSR)**

Table 12-20 describes the fields of the UDSRs.

Table 12-20. UDSR Field Definitions

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 12-22 1 This bit is set. As shown in Table 12-21
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status is dependent 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 12-24 1 This bit is set. As shown in Table 12-23

Table 12-21. 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 12-22. 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 12-23. 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 12-24. UDSR[RXRDY] Cleared

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.

12.4 Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock signal.

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). The data is placed directly into an internal transmitter shift register, or into the transmitter FIFO in FIFO mode—see Section 12.4.5, “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. The receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO in FIFO mode. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt-driven.

12.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 12-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

Functional Description

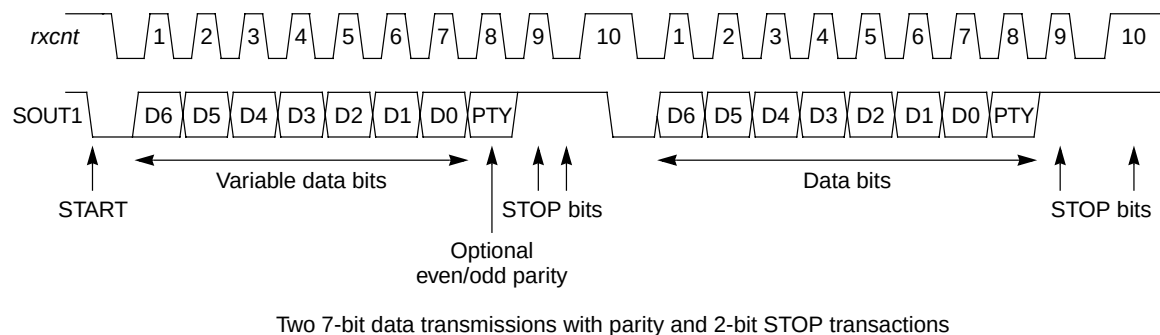


Figure 12-16. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of three or four parts:

- START bit
- data transfer (least significant bit is first data bit on the bus)
- parity bit (optional)
- STOP bit

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 loop-back mode, different errors, and FIFO mode.

12.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 12-16 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.

12.4.1.2 Data Transfer

Each data transfer contains 5, 6, 7, or 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 to 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.

12.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 12.3.1.7, “Line Control Registers (ULCR0 and ULCR1).” 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 12.3.1.9, “Line Status Registers (ULSR0 and ULSR1)”).

12.4.1.4 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.

12.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the CCB 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:

$$\text{Baud rate} = (1/16) \times (\text{CCB clock frequency} / \text{divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

12.4.3 Local Loop-Back Mode

Local loop-back 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 loop-back mode the transmit and receive data paths of the DUART can be verified. Note that in local loop-back mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

12.4.4 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 when the line status register (ULSR) is read.

12.4.4.1 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.

12.4.4.2 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.

12.4.4.3 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.

12.4.5 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.

12.4.5.1 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 12.3.1.4, “Interrupt Enable Register (UIER),” 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 to 1 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 12.3.1.5, “Interrupt ID Registers (UIIR0 and UIIR1),” for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

12.4.5.2 DMA Mode Select

The UDSR[RXRDY] bit reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] = 0), 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] = 1 and UFCR[FEN] = 1), 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] = 0), 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] = 1 and UFCR[FEN] = 1), 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 12.3.1.13, “DMA Status Registers (UDSR0 and UDSR1),” for a complete description of the UDSR[RXRDY] and UDSR[TXRDY] bits.

12.4.5.3 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 12.3.1.4, “Interrupt Enable Register (UIER).”

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.

12.5 DUART Initialization Sequence

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited area.
- All the DUART registers are one byte wide. Read 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 13

Local Bus Controller

13.1 Introduction

Figure 13-1 is a functional block diagram of the local bus controller, which supports three interfaces: a general purpose chip select machine (GPCM) interface, a user programmable machine (UPM) interface, and an SDRAM controller.

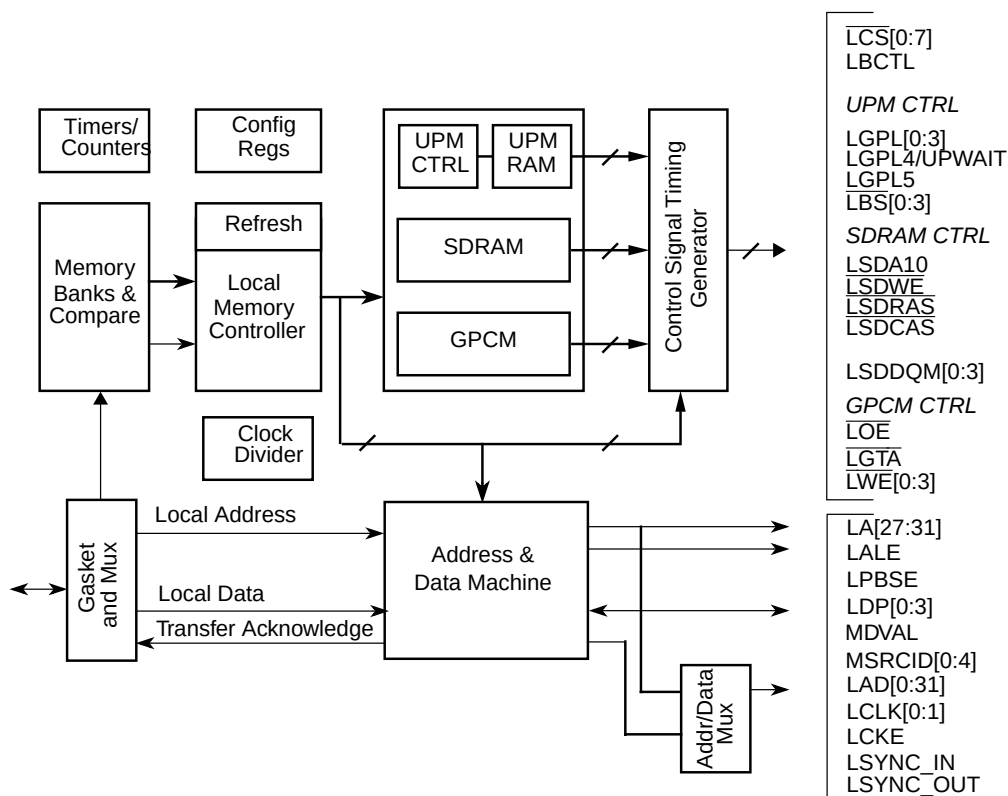


Figure 13-1. Local Bus Controller Block Diagram

13.1.1 Overview

The main component of the local bus controller (LBC) 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 high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals. The external address latch signal (LAL) allows multiplexing of addresses with data signals to reduce the device pin count.

The local bus controller also includes a number of data checking and protection features such as data parity generation and checking, write protection and a bus monitor to ensure that each bus cycle is terminated within a user-specified period.

13.1.2 Features

The local bus controller main features are as follows:

- Memory controller with eight memory banks
 - 34-bit address decoding with mask
 - Variable memory block sizes (32 Kbytes to 16 Gbytes)
 - Selection of control signal generation on a per-bank basis
 - Data buffer controls activated on a per-bank basis
 - Up to 256-byte bursts, arbitrarily aligned
 - Automatic segmentation of large transactions
 - Odd/even parity checking including read-modify-write (RMW) parity for single accesses
 - Write-protection capability
 - Atomic operation
 - Parity byte-select
- Synchronous DRAM (SDRAM) machine
 - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
 - Supports up to 4 concurrent open pages per device
 - Supports SDRAM port size of 32-bit, 16-bit and 8-bit
 - Supports external address and/or command lines buffering
- General-purpose chip-select machine (GPCM)
 - Compatible with SRAM, EPROM, FEPRM, and peripherals

- Global (boot) chip-select available at system reset
- Boot chip-select support for 8-, 16-, 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 ($\overline{\text{LGTA}}$)
- 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-, 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, 256 Mbyte page banks
- Optional monitoring of transfers between local bus internal masters and local bus slaves (local bus error reporting)
- Support for delay locked loop (DLL) with software-configurable bypass for low frequency bus clocks

13.1.3 Modes of Operation

The local bus controller provides one general purpose machine (GPCM), one SDRAM machine and three user-programmable machines (UPMs) for the local bus. There is no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the local bus controller, the memory address is compared with the address information of each bank (chip select). The corresponding machine assigned to that bank (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends. Thus only one of the eight chip selects is active at any given time with the LBC in either GPCM, SDRAM or UPM mode for the duration of the transaction.

13.1.3.1 LBC Bus Clock and Clock Ratios

The local bus controller supports ratios of 2, 4 and 8 between the faster internal (system) clock and slower external bus clock (LCLK[0:1]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). This ratio has no functional effect on operation in SDRAM mode but it affects the resolution of signal timing shifts in GPCM mode and the interpretation of UPM array words in UPM mode. The bus clock is driven identically onto a pair of pins, LCLK[0] and LCLK[1], to allow the clock load to be shared equally across a pair of signal nets, thereby enhancing the edge rates of the bus clock.

13.1.3.2 Source ID Debug Mode

In a special system debug mode, the local bus controller is capable of providing the ID of the transaction source on external device pins and permits valid address and data to be captured. In this mode, the system MSRCID[0:4] pins drive the 5-bit internal source ID of the current transaction, whenever there is a valid address or data. When valid address or data is available on the external pins, the source ID pins (MSRCID[0:4]) are driven with a valid 5-bit value other than 0x1F (which is reserved to indicate invalid address or data). The combination of a valid source ID and the value of external address latch enable (LALE) and data valid (MDVAL) determine the data to be latched 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[0:31]. Note that in SDRAM mode the address vector contains the full address as {row, bank, column, LSBs} where row corresponds to the same row address for the given column address and LSBs are the unconnected LSBs of the address for a given port size.
- If a valid source ID is detected on MSRCID[0:4] and MDVAL is asserted, valid data may be latched from LAD[0:31].

13.1.4 Power-Down Mode

The local bus controller is capable of entering power-down mode when the system stops the internal (system) clock to the block. The LBC recognizes the power-down mode (stopped clock) and allows the DLL sufficient time to recover following the re-application of the system clock. Upon entering power-down mode, LBC will place any SDRAM devices, if used, in self-refresh mode before the bus clock is stopped.

13.1.5 References

- MPC8260 PowerQUICC II User's Manual, Chapters 4, 6, and 10, MPC8260UM/D, Rev.0, Motorola, Inc., May 1999.

13.2 External Signal Description

Table 13-1 contains a list of external signals related to the local bus controller and summarizes their function. The table also shows the reset state of all external signals during assertion of $\overline{\text{HRESET}}$.

Table 13-1. Signal Properties—Summary

Name	# of Signals	Direction	Function	Reset State (outputs)
LALE	1	output	External address latch enable	Low
$\overline{\text{LCS}}$	8	output	Chip selects	All high
$\overline{\text{LWE}}$ / LSDDQM/ LBS	4	output output output	GPCM mode: write enable SDRAM mode: byte lane data mask UPM mode: byte (lane) select	All high
LSDA10/ LGPL0	1	output output	SDRAM mode: row address bit/command bit UPM mode: general purpose line 0	High
$\overline{\text{LSDWE}}$ / LGPL1	1	output output	SDRAM mode: write enable UPM mode: general purpose line 1	High
$\overline{\text{LOE}}$ / $\overline{\text{LSDRAS}}$ / LGPL2	1	output output output	GPCM mode: output enable SDRAM mode: row address strobe UPM mode: general purpose line 2	High
$\overline{\text{LSDCAS}}$ / LGPL3	1	output output	SDRAM mode: column address strobe UPM mode: general purpose line 3	High
$\overline{\text{LGT\!A}}$ / LGPL4/ UPWAIT/ LPBSE	1	input output input output	GPCM mode: transaction termination UPM mode: general purpose line 4 UPM mode: external device wait Local bus parity byte select	high-Z
LGPL5	1	output	UPM mode: general purpose line 5	High
LBCTL	1	output	Data buffer control	High
LA[27:31]	5	output	Local bus non-multiplexed address LSBs	All low
LAD[0:31]	32	input/output	Multiplexed address/data bus	high-Z
LDP	4	input/output	Local bus data parity	high-Z
LCKE	1	output	Local bus clock enable	High
LCLK	2	output	Local bus clocks	driven
LSYNC_IN	1	input	DLL synchronize input	—
LSYNC_OUT	1	output	DLL synchronize output	driven
MDVAL	1	output	In LBC debug mode: local bus data valid	Low
MSRCID	5	output	In LBC debug mode: local bus source ID	All high

Table 13-2 shows the detailed external signal descriptions for the local bus controller.

Table 13-2. 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—LALE is asserted for at least one bus clock cycle for each memory-controller transaction, and, if ORx[EAD] = 1, it is asserted for the number of bus clock cycles defined by the EADC field in the LCRR. Note that no other control signals are asserted during the assertion of LALE. Negated—
		Timing Assertion— Negation—
LCS[0:7]	O	Chip selects. Eight chip selects are provided which are mutually exclusive.
		State Meaning Asserted/Negated—Used to enable specific memory devices or peripherals connected to the LBC. LCS[0:7] are provided on a per-bank basis with LCS0 corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.
		Timing Assertion— Negation—
LWE[0:3]/ LSDDQM[0:3]/ LBS[0:3]	O	GPCM write enable/SDRAM data mask/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 BRx[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—For GPCM operation, LWE[0:3] assert for each byte lane enabled for writing. For SDRAM operation, LSDDQM[0:3] function as the DQM or data mask signals provided by JEDEC-compliant SDRAM devices, with one DQM provided per byte lane. LSDDQM[0:3] are driven high when the LBC wishes to mask a write or disable read data output from the SDRAM. LBS[0:3] are programmable byte-select signals in UPM mode. See Section 13.4.4.4, "The RAM Array," for programming details about LBS[0:3]. Negated—
		Timing Assertion— See Section 13.4.2, "General-Purpose Chip-Select Machine (GPCM), for details regarding the timing of LWE[0:3]. Negation—
LSDA10/ LGPL0	O	SDRAM A10/General purpose line 0
		State Meaning Asserted/Negated—For SDRAM accesses, this signal represents address bit 10. When the row address is driven, this signal drives the value of address bit 10. When the column address is driven, this signal forms part of the SDRAM command. This signal is one of six general purpose signals when in UPM mode and drives a value programmed in the UPM array
		Timing Assertion— Negation—

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LSDWE/ LGPL1	O	SDRAM write enable/General-purpose line 1	
		State Meaning	Asserted/Negated—This signal should be connected to the SDRAM device WE input and acts as the SDRAM write enable when accessing SDRAM. This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array
		Timing	Assertion— Negation—
LOE/ LSDRAS/ LGPL2	O	GPCM output enable/SDRAM RAS/General-purpose line 2	
		State Meaning	Asserted/Negated—This signal controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, this signal is the row address strobe (RAS). This signal is one of six general purpose lines when in UPM mode, and drives a value programmed in the UPM array.
		Timing	Assertion— Negation—
LSDCAS/ LGPL3	O	SDRAM CAS/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe (CAS). This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.
		Timing	Assertion— Negation—
LGTA/ LGPL4/ UPWAIT/ LPBSE	I/O	GPCM terminate access/General-purpose line 4/UPM wait/parity byte select	
		State Meaning	Asserted/Negated—This signal is an input in GPCM mode and is used for transaction termination. This signal 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. This signal can also be configured as LPBSE, and in this case disables any use in GPCM or UPM modes. Systems that use read-modify-write parity require an additional memory device and therefore need to generate a byte-select like a normal data device. ANDing the four bits LBS[0:3] through external logic to achieve the logical function of this byte-select can affect the memory access timing because it adds a delay to the byte-select path. The local bus controller provides this optional byte-select pin that is an internal AND of the four (active low) byte selects, allowing glueless, faster connection to RMW-parity devices.
		Timing	Assertion— Negation—
LGPL5	O	General-purpose line 5	
		State Meaning	Asserted/Negated—This signal is one of six general purpose signals when in UPM mode, and drives a value programmed in the UPM array.
		Timing	Assertion— Negation—

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LBCTL	O		Data buffer control. The memory controller activates a data buffer control signal (LBCTL) for the local bus when a GPCM- or UPM-controlled bank is accessed. Access to an SDRAM machine-controlled bank does not activate the buffer control. The buffer control can be disabled by setting ORx[BCTLD].
		State Meaning	Asserted/Negated—The LBCTL pin normally functions as a write/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 LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
		Timing	Assertion— Negation—
LA[27:31]	O		Local bus nonmultiplexed address LSBs. All bits driven on LA[27:31] are defined for 8-bit port sizes, whereas for 32-bit port sizes, LA[30:31] are undefined, and for 16-bit port sizes LA[31] is undefined.
		State Meaning	Asserted/Negated—Even though the LBC shares an address and data bus, the five least significant bits of the RAM address always appear on the dedicated address pins, LA[27:31]. These may be used, unlatched, in place of LAD[27:31] to connect the five least significant bits of the address for address phases. For some RAM devices, such as fast-page DRAM, LA[27:31] serve as the column address offset during a burst access.
		Timing	Assertion— Negation—
LAD[0:31]	I/O		Multiplexed address/data bus. For configuration of a port size in BRx[PS] as 32 bits, all of LAD[0:31] need to 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[0:31] 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[0:31] are driven with the RAM address for the access to follow. External logic should propagate the address on LAD[0:31] while LALE is asserted, and latch the address upon negation of LALE. After LALE is negated, LAD[0:31] are either driven by write data or are made high-impedance by the LBC in order to sample read data driven by an external device. Following the last data transfer of a write access, LAD[0:31] are again taken into a high-impedance state.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LDP[0:3]	I/O	Local bus data parity. For configuration of a port size in BRx[PS] as 32 bits, all of LAD[0:31] needs to 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—During write accesses, a parity bit is generated for each 8 bits of LAD[0:31], such that LDP[0] 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—LDP[0:3] drive and receive the data parity corresponding with the data phases on LAD[0:31]. 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[0:31]. The LDP[0:3] pins change impedance in concert with the LAD[0:31] signals.
LCKE	O	Local bus clock enable	
		State Meaning	Asserted/Negated—LCKE is the bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. This signal is asserted during normal SDRAM operation.
LCLK[0:1]	O	Local bus clocks	
		State Meaning	Asserted/Negated—LCLK[0:1] drive an identical bus clock signal for distributed loads. If the LBC delay-locked loop (DLL) is enabled (see LCRR[DBYP], Figure 13-19 on page 13-36), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:31] and LCSx) by a time delay matching the delay of the DLL timing loop set up between LSYNC_OUT and LSYNC_IN..
LSYNC_OUT	O	DLL 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 DLL 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[0:1] and clocked drivers in the system. No load other than a timing loop should be placed on LSYNC_OUT.
LSYNC_IN	I	DLL synchronization in.	
		State Meaning	Asserted/Negated—See description of LSYNC_OUT.
MDVAL	O	Local bus data valid (LBC debug mode only)	
		State Meaning	Asserted—For a read, MDVAL asserts for 1 bus cycle in the cycle immediately preceding the sampling of read data on LAD[0:31]. For a write, MDVAL asserts for one bus cycle in the cycle corresponding with the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, MDVAL asserts for each data beat. Negated—
		Timing	Assertion/Negation—MDVAL is valid only while LBC is in system debug mode. In debug mode, MDVAL asserts when the LBC generates a data transfer acknowledge.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all pins of MSRCID[0:4] are driven high unless MSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—MSRCID[0:4] 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 the case of address debug, MSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—albeit with optional padding—for reconstructing the system address presented to the LBC. For example, MSRCID[0:4] will be valid only during CAS phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits will all be present on LAD[0:31] during a CAS cycle
		Timing	Assertion/Negation—

13.3 Memory Map/Register Definition

Table 13-3 shows the memory mapped registers of the LBC.

Table 13-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	Base register 0 (BR0)	R/W	0x0000_RR01 ¹	13.3.1.1/13-11
0x0_5008	Base register 1 (BR1)		0x0000_0000	
0x0_5010	Base register 2 (BR2)			
0x0_5018	Base register 3 (BR3)			
0x0_5020	Base register 4 (BR4)			
0x0_5028	Base register 5 (BR5)			
0x0_5030	Base register6 (BR6)			
0x0_5038	Base register7 (BR7)			
0x0_5004	Options register 0 (OR0)	R/W	0x0000_0FF7	13.3.1.2/13-13
0x0_500C	Options register 1 (OR1)		0x0000_0000	
0x0_5014	Options register 2 (OR2)			
0x0_501C	Options register 3 (OR3)			
0x0_5024	Options register 4 (OR4)			
0x0_502C	Options register 5 (OR5)			
0x0_5034	Options register6 (OR6)			
0x0_503C	Options register7 (OR7)			
0x0_5068	UPM address register (MAR)	R/W	0x0000_0000	13.3.1.3/13-20
0x0_5070	UPMA mode register (MAMR)	R/W	0x0000_0000	13.3.1.4/13-20

Table 13-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5074	UPMB mode register (MBMR)	R/W	0x0000_0000	13.3.1.4/13-20
0x0_5078	UPMC mode register (MCMR)	R/W	0x0000_0000	13.3.1.4/13-20
0x0_5084	Memory refresh timer prescaler register (MRTPR)	R/W	0x0000_0000	13.3.1.5/13-23
0x0_5088	UPM data register (MDR)	R/W	0x0000_0000	13.3.1.6/13-24
0x0_5094	SDRAM mode register (LSDMR)	R/W	0x0000_0000	13.3.1.7/13-25
0x0_50A0	UPM refresh timer (LURT)	R/W	0x0000_0000	13.3.1.8/13-27
0x0_50A4	SDRAM refresh timer (LSRT)	R/W	0x0000_0000	13.3.1.9/13-28
0x0_50B0	Transfer error status register (LTESR)	Read/ Bit-reset	0x0000_0000	13.3.1.10/13-29
0x0_50B4	Transfer error disable register (LTEDR)	R/W	0x0000_0000	13.3.1.11/13-31
0x0_50B8	Transfer error interrupt register (LTEIR)	R/W	0x0000_0000	13.3.1.12/13-32
0x0_50BC	Transfer error attributes register (LTEATR)	R/W	0x0000_0000	13.3.1.13/13-33
0x0_50C0	Transfer error address register (LTEAR)	R/W	0x0000_0000	13.3.1.14/13-34
0x0_50D0	Configuration register (LBCR)	R/W	0x0000_0000	13.3.1.15/13-34
0x0_50D4	Clock ratio register (LCRR)	R/W	0x8000_0008	13.3.1.16/13-35

¹ Port size for BR0 is configured from external pins during reset, hence 'RR' is either 0x08, 0x10, or 0x18.

13.3.1 Register Descriptions

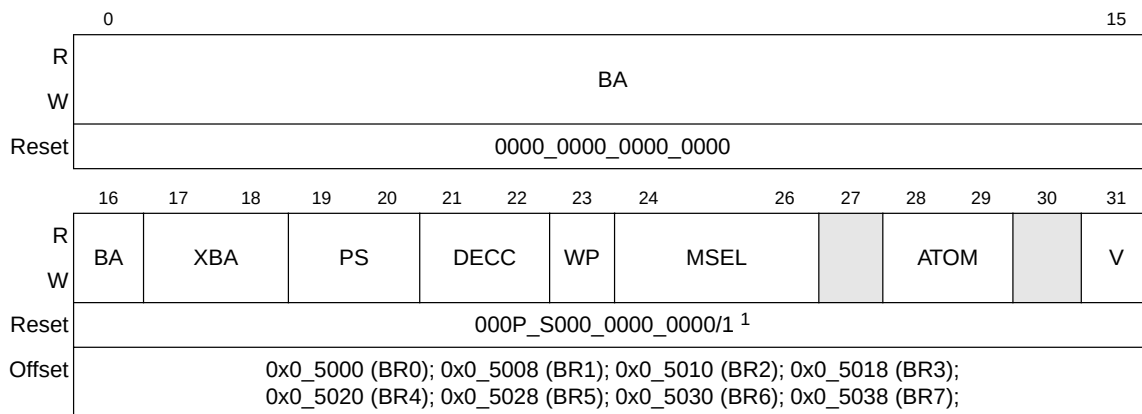
This section provides a detailed description of the local bus controller configuration, status, and control registers with detailed bit and field descriptions.

Address offsets in the LBC address range that are not defined in Table 13-3 should not be accessed for reading or writing. Similarly, only zero bits should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

13.3.1.1 Base Registers (BR0–BR7)

The base registers (BRx) 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) also includes a memory attribute and selects the machine for memory operation handling. Figure 13-2 contains the field descriptions for BRx. Note that after a system reset, the valid bit is set in BR0 (BR0[V] = 1) and cleared in BR1–BR7 (BR1–7[V] = 0), and the value of BR0[PS] reflects the initial port size configured by the boot ROM location pins.

Memory Map/Register Definition



¹ BR0 has its valid bit set during reset. Thus bank 0 is valid with the port size (PS) configured from external boot ROM configuration pins during reset. All other base registers have all bits cleared to zero during reset.

Figure 13-2. Base Registers (BRx)

Table 13-4 describes BRx fields.

Table 13-4. BRx Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base address 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 ORx[AM].
17–18	XBA	Extended base address. Extends BA by a further 2 bits such that 19 bits of each base address register are compared to the 34-bit transaction address. XBA provides the extra 2 MSBs (i.e. {XBA,BA} represents the full base address). Used with the extended address mask bits ORx[XAM].
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, the value of PS is configured from the boot ROM location pins during reset. For all other banks the value is reset to 00 (port size not defined). 00 reserved 01 8-bit 10 16-bit 11 32-bit
21–22	DECC	Specifies the method for data error checking. 00 Data error checking disabled, but normal parity generation 01 Normal parity generation and checking 10 Read-modify-write parity generation and normal parity checking (32-bit port size only) 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 $\overline{\text{LCSx}}$ on write cycles to this memory bank. LTESR[WP] is set (if enabled) if a write to this memory bank is attempted, and a local bus error interrupt is generated (if enabled), terminating the cycle.

Table 13-4. BRx Field Descriptions (continued)

Bits	Name	Description
24–26	MSEL	Machine select. Specifies which machine to use for the memory operations handling. Reset value selects GPCM. 000 GPCM 001 Reserved 010 Reserved 011 SDRAM 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 BRx and ORx pair are valid. The $\overline{\text{LCSx}}$ signal does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

13.3.1.2 Option Registers (OR0–OR7)

The ORx registers define the sizes of memory banks and access attributes. The ORx attribute bits support the following three modes of operation as defined by BRx[MSEL].

- GPCM mode
- UPM mode
- SDRAM mode

The ORx registers are interpreted differently depending on which of the three machine types is selected for that bank.

13.3.1.2.1 Address Mask

The address mask fields of the option registers (ORx[XAM,AM]) mask up to 19 corresponding BRx[BA,XBA]. The 15 lsbs of the 34-bit internal 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

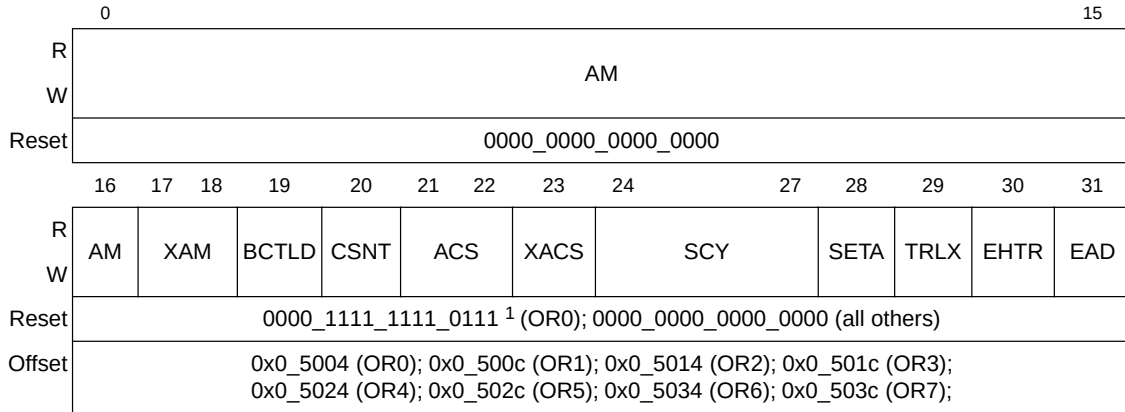
than one area of the address map. Table 13-5 shows memory bank sizes from 256 Kbytes up to 16 Gbytes.

Table 13-5. Memory Bank Sizes in Relation to Address Mask

XAM	AM	Memory Bank Size
00	0000_0000_0000_0000_0	16 Gbytes
10	0000_0000_0000_0000_0	8 Gbytes
11	0000_0000_0000_0000_0	4 Gbytes
11	1000_0000_0000_0000_0	2 Gbytes
11	1100_0000_0000_0000_0	1 Gbyte
11	1110_0000_0000_0000_0	512 Mbytes
11	1111_0000_0000_0000_0	256 Mbytes
11	1111_1000_0000_0000_0	128 MByte
11	1111_1100_0000_0000_0	64 MByte
11	1111_1110_0000_0000_0	32 MByte
11	1111_1111_0000_0000_0	16 Mbyte
11	1111_1111_1000_0000_0	8 Mbyte
11	1111_1111_1100_0000_0	4 Mbyte
11	1111_1111_1110_0000_0	2 Mbyte
11	1111_1111_1111_0000_0	1 Mbyte
11	1111_1111_1111_1000_0	512 Kbyte
11	1111_1111_1111_1100_0	256 Kbyte
11	1111_1111_1111_1110_0	128 Kbyte
11	1111_1111_1111_1111_0	64 Kbyte
11	1111_1111_1111_1111_1	32 Kbyte

13.3.1.2.2 Option Registers (ORx)—GPCM Mode

Table 13-6 shows the bit fields for ORx when the corresponding BRx[MSEL] selects the GPCM machine.



¹ OR0 has this value set during reset (GPCM is the default control machine for all banks coming out of reset). All other option registers have all bits cleared.

Figure 13-3. Option Registers in GPCM Mode

Table 13-6 describes ORx fields for GPCM mode.

Table 13-6. ORx—GPCM Field Descriptions

Bits	Name	Description												
0–16	AM	GPCM address mask. Masks corresponding BRx 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	XAM	Extended address mask. Masks the corresponding XBA bits in the BRx register, effectively extending the address mask (AM) by 2 bits.												
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 $\overline{LCSx}$ and $\overline{LWE}$ are negated during an external memory write access handled by the GPCM, provided that ACS is not set to 00 (when ACS=00, CSNT is interpreted as 0). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{LCSx}$ and $\overline{LWE}$ are negated normally. 1 $\overline{LCSx}$ and $\overline{LWE}$ are negated earlier depending on the value of LCRR[CLKDIV]. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th>LCRR[CLKDIV]</th><th>CSNT</th><th>Meaning</th></tr> <tr> <td style="text-align: center;">x</td><td style="text-align: center;">0</td><td>$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: center;">1</td><td>$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.</td></tr> <tr> <td style="text-align: center;">4 or 8</td><td style="text-align: center;">1</td><td>$\overline{LCSx}$ and $\overline{LWE}$ are negated quarter of a bus clock cycle earlier.</td></tr> </table>	LCRR[CLKDIV]	CSNT	Meaning	x	0	$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.	2	1	$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.	4 or 8	1	$\overline{LCSx}$ and $\overline{LWE}$ are negated quarter of a bus clock cycle earlier.
LCRR[CLKDIV]	CSNT	Meaning												
x	0	$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.												
2	1	$\overline{LCSx}$ and $\overline{LWE}$ are negated normally.												
4 or 8	1	$\overline{LCSx}$ and $\overline{LWE}$ are negated quarter of a bus clock cycle earlier.												

Table 13-6. ORx—GPCM Field Descriptions (continued)

Bits	Name	Description																					
21–22	ACS	Address to chip-select setup. Determines the delay of the $\overline{\text{LCSx}}$ assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, $\text{OR0}[\text{ACS}] = 11$ <table border="1"> <thead> <tr> <th>LCRR[CLKDIV]</th><th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>x</td><td>00</td><td>$\overline{\text{LCSx}}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that $\text{CSNT}=0$.</td></tr> <tr> <td>x</td><td>01</td><td>Reserved.</td></tr> <tr> <td>2</td><td>10</td><td>$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.</td></tr> <tr> <td>2</td><td>11</td><td>$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.</td></tr> <tr> <td>4 or 8</td><td>10</td><td>$\overline{\text{LCSx}}$ is output a quarter bus clock cycle after the address lines.</td></tr> <tr> <td>4 or 8</td><td>11</td><td>$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.</td></tr> </tbody> </table>	LCRR[CLKDIV]	Value	Meaning	x	00	$\overline{\text{LCSx}}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that $\text{CSNT}=0$.	x	01	Reserved.	2	10	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.	2	11	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.	4 or 8	10	$\overline{\text{LCSx}}$ is output a quarter bus clock cycle after the address lines.	4 or 8	11	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.
LCRR[CLKDIV]	Value	Meaning																					
x	00	$\overline{\text{LCSx}}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that $\text{CSNT}=0$.																					
x	01	Reserved.																					
2	10	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.																					
2	11	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.																					
4 or 8	10	$\overline{\text{LCSx}}$ is output a quarter bus clock cycle after the address lines.																					
4 or 8	11	$\overline{\text{LCSx}}$ is output a half bus clock cycle after the address lines.																					
23	XACS	Extra address to chip-select setup. Setting this bit increases the delay of the $\overline{\text{LCSx}}$ assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, $\text{OR0}[\text{XACS}] = 0$. 0 Address to chip-select setup is determined by $\text{ORx}[\text{ACS}]$ and $\text{LCRR}[\text{CLKDIV}]$. 1 Address to chip-select setup is extended (see Table 13-24 and Table 13-24 for $\text{LCRR}[\text{CLKDIV}]=4$ or 8, Table 13-25 and Table 13-26 for $\text{LCRR}[\text{CLKDIV}]=2$).																					
24–27	SCY	Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, $\text{OR0}[\text{SCY}] = 1111$. 0000 No wait states 0001 1 bus clock cycle wait state ... 1111 15 bus clock cycle wait states																					
28	SETA	External address termination. 0 Access is terminated internally by the memory controller unless the external device asserts $\overline{\text{LGTA}}$ 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).																					
29	TRLX	Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals. 0 Normal timing is generated by the GPCM. 1 Relaxed timing on the following parameters: • 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. • $\overline{\text{LCSx}}$ (only if ACS is not equal to 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes.																					

Table 13-6. ORx—GPCM 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. <table border="1"> <thead> <tr> <th>TRLX</th><th>EHTR</th><th>Meaning</th></tr> </thead> <tbody> <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> </tbody> </table>	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.
TRLX	EHTR	Meaning															
0	0	The memory controller generates normal timing. No additional cycles are inserted.															
0	1	1 idle clock cycle is inserted.															
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).															

13.3.1.2.3 Option Registers (ORx)—UPM Mode

Table 13-7 shows the bit fields for ORx when the corresponding BRx[MSEL] selects a UPM machine.

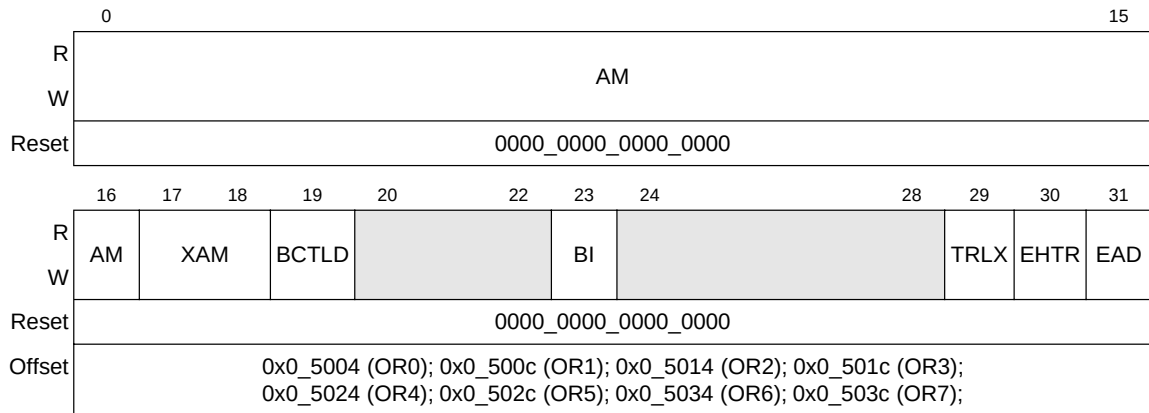
**Figure 13-4. Option Registers in UPM Mode**

Table 13-7 describes BRx fields for UPM mode.

Table 13-7. ORx—UPM Field Descriptions

Bits	Name	Description															
0–16	AM	UPM address mask. Masks corresponding BRx 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	XAM	Extended address mask. Masks the corresponding XBA bits in the BRx register, effectively extending the address mask (AM) by 2 bits.															
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. <table border="1"> <thead> <tr> <th>TRLX</th><th>EHTR</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>Normal timing is generated by the memory controller. 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> </tbody> </table>	TRLX	EHTR	Meaning	0	0	Normal timing is generated by the memory controller. 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.
TRLX	EHTR	Meaning															
0	0	Normal timing is generated by the memory controller. No additional cycles are inserted.															
0	1	1 idle clock cycle is inserted.															
1	0	4 idle clock cycles are inserted.															
1	1	8 idle clock cycles are inserted.															
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).															

13.3.1.2.4 Option Registers (ORx)—SDRAM Mode

Figure 13-5 shows the bit fields for ORx when the corresponding BRx[MSEL] selects the SDRAM machine.

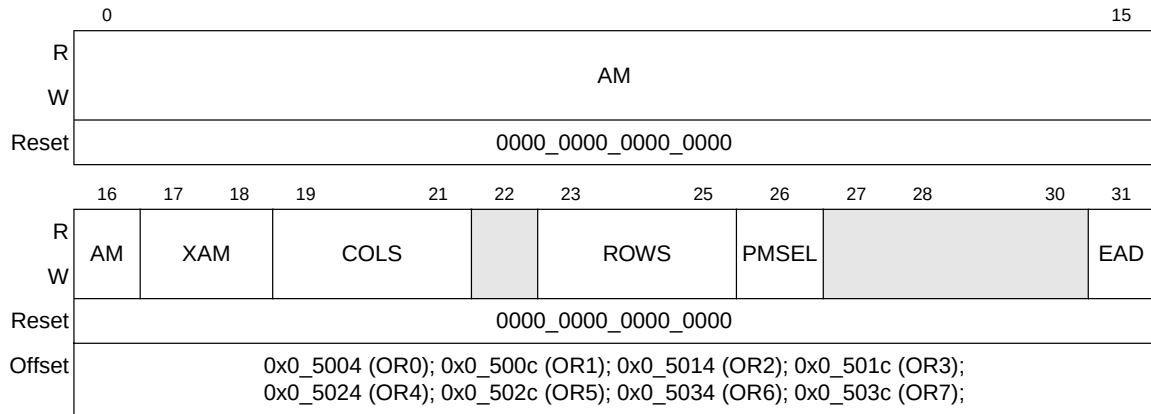
**Figure 13-5. Option Registers in SDRAM Mode**

Table 13-8 describes BRx fields for SDRAM mode.

Table 13-8. ORx—SDRAM Field Descriptions

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BRx 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. AM can be read or written at any time. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address pins.
17–18	XAM	Extended address mask. Masks the corresponding XBA bits in the BRx register, effectively extending the address mask (AM) by 2 bits.
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7 001 8 010 9 011 10 100 11 101 12 110 13 111 14
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 001 10 010 11 011 12 100 13 101 14 110 15 111 reserved
26	PMSEL	Page mode select. Selects page mode for the SDRAM connected to the memory controller bank. 0 Back-to-back page mode (normal operation). Page is closed when the bus becomes idle. 1 Page is kept open until a page miss or refresh occurs.

Table 13-8. ORx—SDRAM Field Descriptions (continued)

Bits	Name	Description
27–30	—	Reserved
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

13.3.1.3 UPM Memory Address Register (MAR)

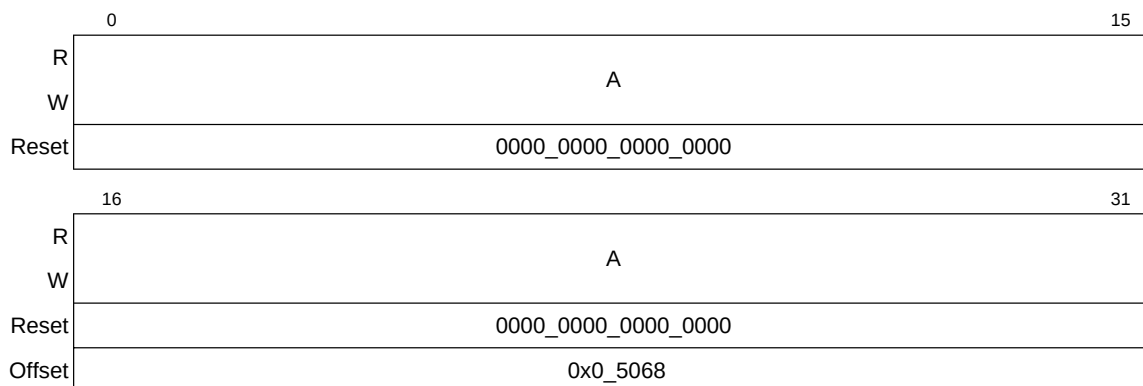
**Figure 13-6. UPM Memory Address Register**

Table 13-9 describes MAR fields.

Table 13-9. MAR Field Descriptions

Bits	Name	Description
0–31	A	The memory address register is used to store an address which can be output to the address signals under control of the AMX bits in the UPM RAM word.

13.3.1.4 UPM Mode Registers (MxMR)

The UPM machine mode registers (MAMR, MBMR and MCMR) contain the configuration for the three UPMs.

	0	1	2	3	4	5	7	8	9	10	12	13	14	15
R		RFEN	OP	UWPL	AM	DS	G0CLx	GPL4	RLFx					
W														
Reset	0000_0000_0000_0000													
	16	17	18		21	22		25	26					31
R	RLFx	WLFx				TLFx				MAD				
W														
Reset	0000_0000_0000_0000													
Offset	0x0_5070 (MAMR); 0x0_5074 (MBMR); 0x0_5078 (MCMR)													

Figure 13-7. UPM Mode Registers

Table 13-10 describes UPM mode fields.

Table 13-10. MxMR 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 will 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 UPMx 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, the MAD field 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, the MAD field 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	UPWAIT polarity active low. Sets the polarity of the UPWAIT pin when in UPM mode. 0 UPWAIT is active high. 1 UPWAIT is active low.

Table 13-10. MxMR Field Descriptions (continued)

Bits	Name	Description																																																																															
5–7	AM	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.																																																																															
		<table><tr><th>Value</th><th>LA0–LA15</th><th>LA16</th><th>LA17</th><th>LA18</th><th>LA19-LA28</th><th>LA29</th><th>LA30</th><th>LA31</th></tr><tr><td>000</td><td>0</td><td>A8</td><td>A9</td><td>A10</td><td>A11-A20</td><td>A21</td><td>A22</td><td>A23</td></tr><tr><td>001</td><td>0</td><td>A7</td><td>A8</td><td>A9</td><td>A10-A19</td><td>A20</td><td>A21</td><td>A22</td></tr><tr><td>010</td><td>0</td><td>A6</td><td>A7</td><td>A8</td><td>A9-A18</td><td>A19</td><td>A20</td><td>A21</td></tr><tr><td>011</td><td>0</td><td>A5</td><td>A6</td><td>A7</td><td>A8-A17</td><td>A18</td><td>A19</td><td>A20</td></tr><tr><td>100</td><td>0</td><td>A4</td><td>A5</td><td>A6</td><td>A7-A16</td><td>A17</td><td>A18</td><td>A19</td></tr><tr><td>101</td><td>0</td><td>A3</td><td>A4</td><td>A5</td><td>A6-A15</td><td>A16</td><td>A17</td><td>A18</td></tr><tr><td>110</td><td colspan="7">Reserved</td></tr><tr><td>111</td><td colspan="7">Reserved</td></tr></table>	Value	LA0–LA15	LA16	LA17	LA18	LA19-LA28	LA29	LA30	LA31	000	0	A8	A9	A10	A11-A20	A21	A22	A23	001	0	A7	A8	A9	A10-A19	A20	A21	A22	010	0	A6	A7	A8	A9-A18	A19	A20	A21	011	0	A5	A6	A7	A8-A17	A18	A19	A20	100	0	A4	A5	A6	A7-A16	A17	A18	A19	101	0	A3	A4	A5	A6-A15	A16	A17	A18	110	Reserved							111	Reserved						
		Value	LA0–LA15	LA16	LA17	LA18	LA19-LA28	LA29	LA30	LA31																																																																							
		000	0	A8	A9	A10	A11-A20	A21	A22	A23																																																																							
		001	0	A7	A8	A9	A10-A19	A20	A21	A22																																																																							
		010	0	A6	A7	A8	A9-A18	A19	A20	A21																																																																							
		011	0	A5	A6	A7	A8-A17	A18	A19	A20																																																																							
		100	0	A4	A5	A6	A7-A16	A17	A18	A19																																																																							
		101	0	A3	A4	A5	A6-A15	A16	A17	A18																																																																							
		110	Reserved																																																																														
111	Reserved																																																																																
8–9	DS	Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by UPMx. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPMx allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPMx 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																																																																															
10–12	G0CLx	General line 0 control. Determines which logical address line can be output to the LGPL0 pin when the UPMx is selected to control the memory access.																																																																															
		000 A12																																																																															
		001 A11																																																																															
		010 A10																																																																															
		011 A9																																																																															
		100 A8																																																																															
		101 A7																																																																															
		110 A6																																																																															
		111 A5																																																																															
13	GPL4	LGPL4 output line disable. Determines how the LGPL4/UPWAIT pin is controlled by the corresponding bits in the UPMx array. See Table 13-30 on page 13-73.																																																																															
		<table><tr><th rowspan="2">Value</th><th rowspan="2">LGPL4/UPWAIT 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>UPWAIT (input)</td><td>DLT3</td><td>WAEN</td></tr></table>	Value	LGPL4/UPWAIT pin function	Interpretation of UPM word bits		G4T1/DLT3	G4T3/WAEN	0	LGPL4 (output)	G4T1	G4T3	1	UPWAIT (input)	DLT3	WAEN																																																																	
		Value			LGPL4/UPWAIT pin function	Interpretation of UPM word bits																																																																											
			G4T1/DLT3	G4T3/WAEN																																																																													
		0	LGPL4 (output)	G4T1	G4T3																																																																												
1	UPWAIT (input)	DLT3	WAEN																																																																														

Table 13-10. MxMR Field Descriptions (continued)

Bits	Name	Description
14–17	RLFx	Read loop field. Determines the number of times a loop defined in the UPMx 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
18–21	WLFx	Write loop field. Determines the number of times a loop defined in the UPMx 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	TLFx	Refresh loop field. Determines the number of times a loop defined in the UPMx 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 UPMx.

13.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

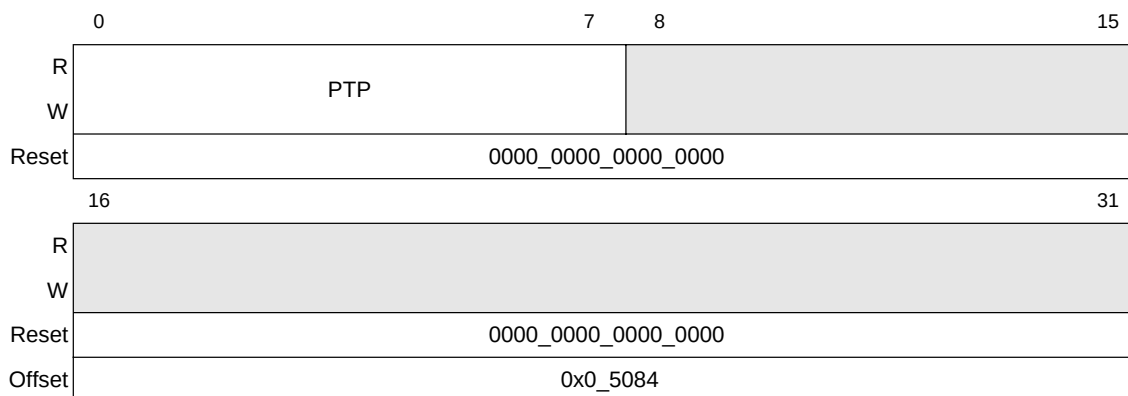


Figure 13-8. Memory Refresh Timer Prescaler Register

Table 13-11 describes MRTPR fields.

Table 13-11. 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

13.3.1.6 UPM Data Register (MDR)

The memory data register (MDR) contains data written to or read from the RAM array for UPM read or write commands. MDR must be set up before issuing a write command to the UPM.

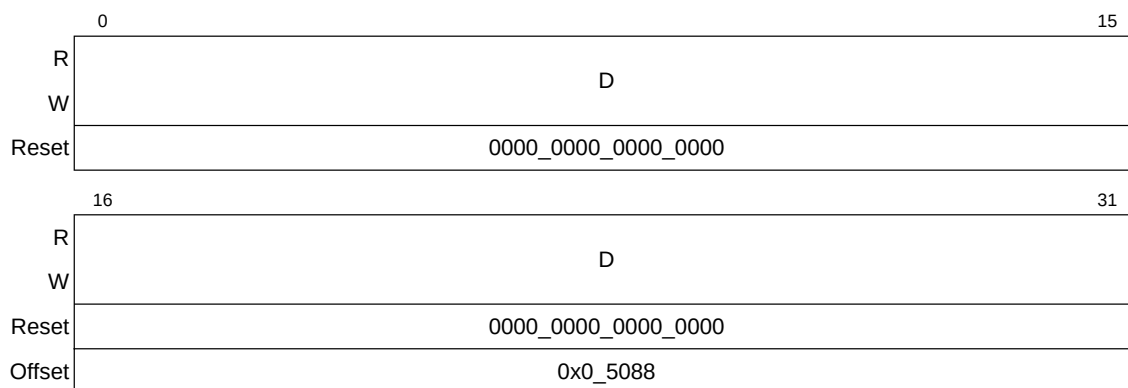


Figure 13-9. UPM Data Register

Table 13-12 describes MDR fields.

Table 13-12. MDR Field Descriptions

Bits	Name	Description
0–31	D	The data to be read or written into the RAM array when a write or read command is supplied to the UPM (MxMR[OP] = 01 or MxMR[OP] = 10).

13.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR) is used to configure operations pertaining to SDRAM.

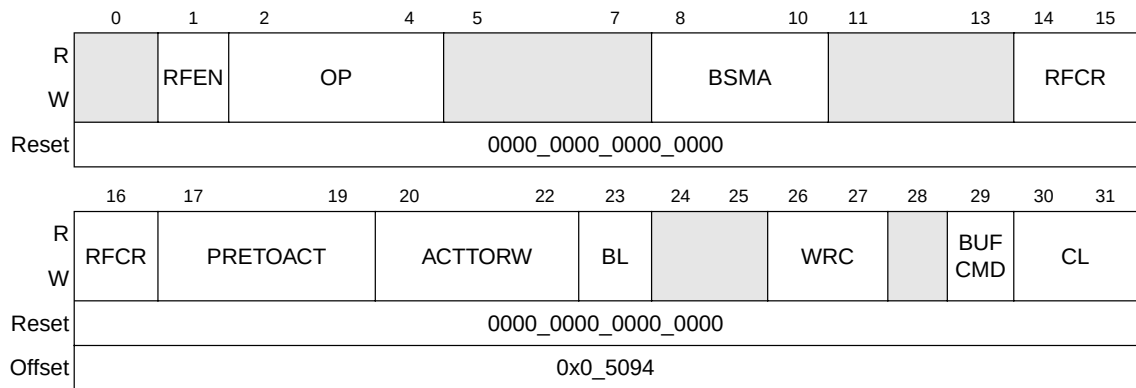
**Figure 13-10. SDRAM Machine Mode Register (LSDMR)**

Table 13-12 describes LSDMR fields.

Table 13-13. LSDMR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the SDRAM requires refresh services. 0 Refresh services are not required. 1 Refresh services are required.

Table 13-13. LSDMR Field Descriptions (continued)

Bits	Name	Description																											
2–4	OP	SDRAM operation. Selects the operation that occurs when the SDRAM device is accessed.} <table> <tr> <th>Value</th><th>Meaning</th><th>Use</th></tr> <tr> <td>000</td><td>Normal operation</td><td>Normal operation</td></tr> <tr> <td>001</td><td>Auto refresh</td><td>Initialization</td></tr> <tr> <td>010</td><td>Self refresh</td><td>Power down mode or Debug</td></tr> <tr> <td>011</td><td>Mode Register write</td><td>Initialization</td></tr> <tr> <td>100</td><td>Precharge bank</td><td>Debug</td></tr> <tr> <td>101</td><td>Precharge all banks</td><td>Initialization</td></tr> <tr> <td>110</td><td>Activate bank</td><td>Debug</td></tr> <tr> <td>111</td><td>Read/write without valid data transfer</td><td>Debug</td></tr> </table>	Value	Meaning	Use	000	Normal operation	Normal operation	001	Auto refresh	Initialization	010	Self refresh	Power down mode or Debug	011	Mode Register write	Initialization	100	Precharge bank	Debug	101	Precharge all banks	Initialization	110	Activate bank	Debug	111	Read/write without valid data transfer	Debug
Value	Meaning	Use																											
000	Normal operation	Normal operation																											
001	Auto refresh	Initialization																											
010	Self refresh	Power down mode or Debug																											
011	Mode Register write	Initialization																											
100	Precharge bank	Debug																											
101	Precharge all banks	Initialization																											
110	Activate bank	Debug																											
111	Read/write without valid data transfer	Debug																											
5–7	—	Reserved																											
8–10	BSMA	Bank select multiplexed address line. Selects which address pins serve as the 2-bit bank-select address for SDRAM. 000 LA12:LA13 001 LA13:LA14 010 LA14:LA15 011 LA15:LA16 100 LA16:LA17 101 LA17:LA18 110 LA18:LA19 111 LA19:LA20																											
11–13	—	Reserved																											
14–16	RFCR	Refresh recovery. Sets the refresh recovery interval in bus clock cycles. Defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command. 000 reserved 001 3 clocks 010 4 clocks 011 5 clocks 100 6 clocks 101 7 clocks 110 8 clocks 111 16 clocks																											
17–19	PRETOACT	Defines the earliest timing for ACTIVATE or REFRESH command after a PRECHARGE command (number of bus clock cycle wait states). 000 8 001 1 010 2 011 3 100 4 101 5 110 6 111 7																											

Table 13-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
20–22	ACTTORW	Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). 000 8 001 reserved 010 2 011 3 100 4 101 5 110 6 111 7
23	BL	Sets the burst length for SDRAM accesses. 0 SDRAM burst length is 4. Use this value if the device port size is 16. 1 SDRAM burst length is 8. Use this value if the device port size is 32 or 8.
24–25	—	Reserved
26–27	WRC	Write recovery time. Defines the earliest timing for PRECHARGE command after the last data is written to the SDRAM.) 00 4 01 Reserved 10 2 11 3
28	—	Reserved
29	BUFCMD	Control line assertion timing. If external buffers are placed on the control lines going to both the SDRAM and address lines, setting BUFCMD causes all SDRAM control lines except $\overline{\text{LCSx}}$, $\overline{\text{LCKE}}$, $\overline{\text{LALE}}$ and $\overline{\text{LSDDQM}}[0:3]$ to be asserted for $\text{LCRR}[\text{BUFCMDC}]$ cycles, instead of one. 0 Normal timing for the control lines 1 All control lines except $\overline{\text{LCSx}}$ are asserted for the number of cycles specified by $\text{LCRR}[\text{BUFCMDC}]$.
30–31	CL	CAS latency. Defines the timing for first read data after a column address is sampled by the SDRAM. 00 Extended CAS latency. According to $\text{LCRR}[\text{ECL}]$. See Table 13-22 on page 13-36. 01 1 10 2 11 3

13.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT) generates a refresh request for all valid banks that selected a UPM machine and are refresh-enabled ($\text{MxMR}[\text{RFEN}] = 1$). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

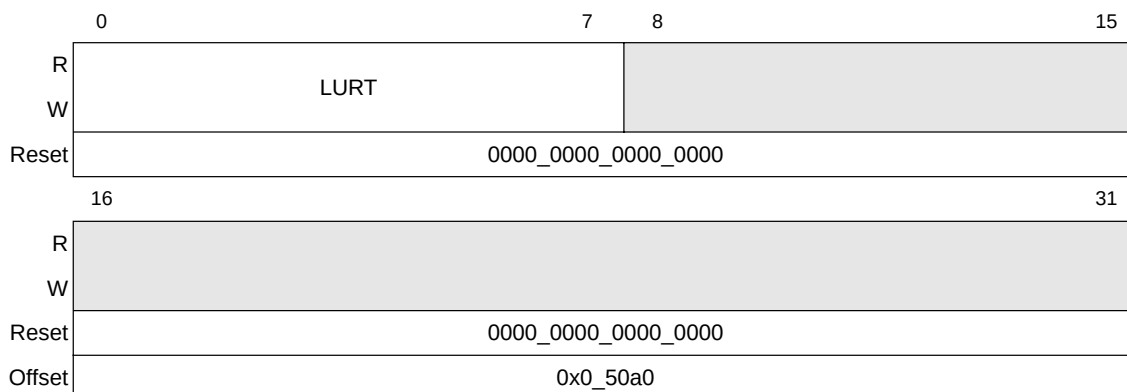
**Figure 13-11. UPM Refresh Timer (LURT)**

Table 13-14 describes LURT fields.

Table 13-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT ¹	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LURT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ Example: For a 266-MHz system clock and a required service rate of 15.6μs, given MRTPR[PTP] = 32, the LURT value should be 128 decimal. 128/(266 MHz/32) = 15.4μs, which is less than the required service period of 15.6μs.
8–31	—	Reserved

¹ Note that a value of 0x00 (reset value) sets the maximum period of 256 x MRTPR[PTP] system clock cycles.

13.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT) generates a refresh request for all valid banks that selected a SDRAM machine and are refresh-enabled (LSDMR[RFEN] = 1). Each time the timer expires, all banks that qualify generate a bank staggering auto refresh request using the SDRAM machine.

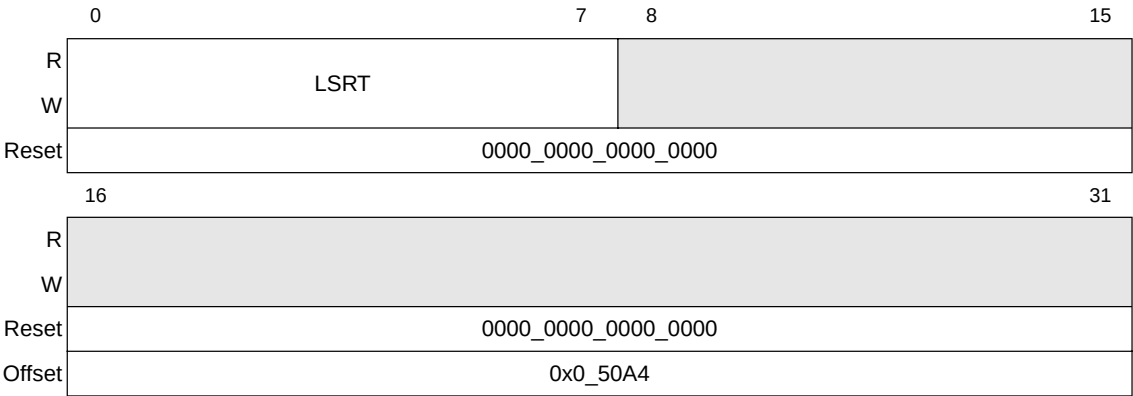


Figure 13-12. LSRT SDRAM Refresh Timer

Table 13-15 describes LSRT fields.

Table 13-15. LSRT Field Descriptions

Bits	Name	Description
0–7	LSRT	SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{F_{\text{systemclock}}}{\text{MRTPR}[\text{PTP}]}\right)}$ Note that a value of 0x00 (reset value) sets the maximum period of 256 x MRTPR[PTP] system clock cycles. Example: For a 266-MHz system clock and a required service rate of 15.6µs, given MRTPR[PTP] = 32, the LSRT value should be 128 decimal. 128/(266 MHz/32) = 15.4µs, which is less than the required service period of 15.6µs.
8–31	—	Reserved

13.3.1.10 Transfer Error Status Register (LTESR)

The LBC has five registers for error management.

- LTESR indicates the cause of an error
- LTEDR is used to enable (and disable) error checking
- LTEIR enables reporting of errors through an interrupt
- LTEATR captures the source attributes of an error
- LTEAR captures the address of a transaction that caused an error

LTESR is a bit-reset type register. Reading from 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 the write protect error bit (LTESR[WP]) and not affect any other bits in the register, the value 0b0400_0000 should be written to the register.

Memory Map/Register Definition

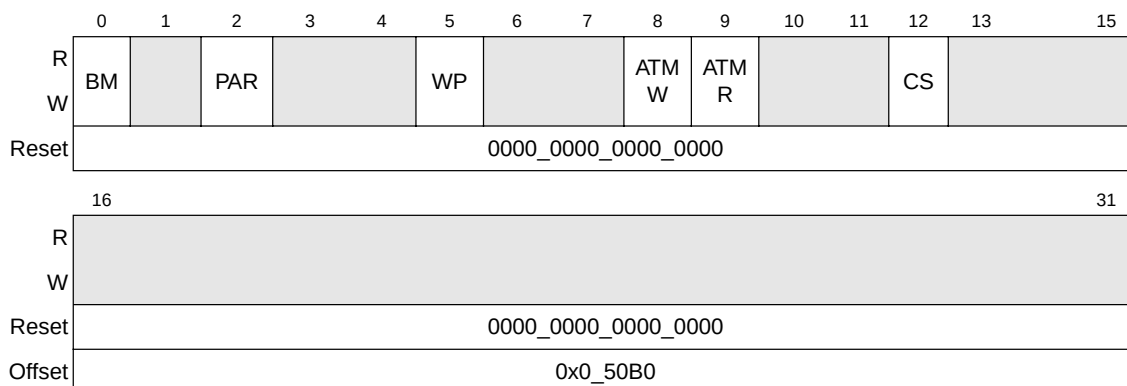


Figure 13-13. Transfer Error Status Register

Table 13-16 describes LTESR fields.

Table 13-16. 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[BM] x 8 bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	0 No local bus parity error 1 Local bus parity error. LTEATR[PB] indicates the byte lane that caused the error and LTEATR[BKN] 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.
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 LBC that did not hit any memory bank.
13–31	—	Reserved

13.3.1.11 Transfer Error Check Disable Register (LTEDR)

This register is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) via the interrupt mechanism.

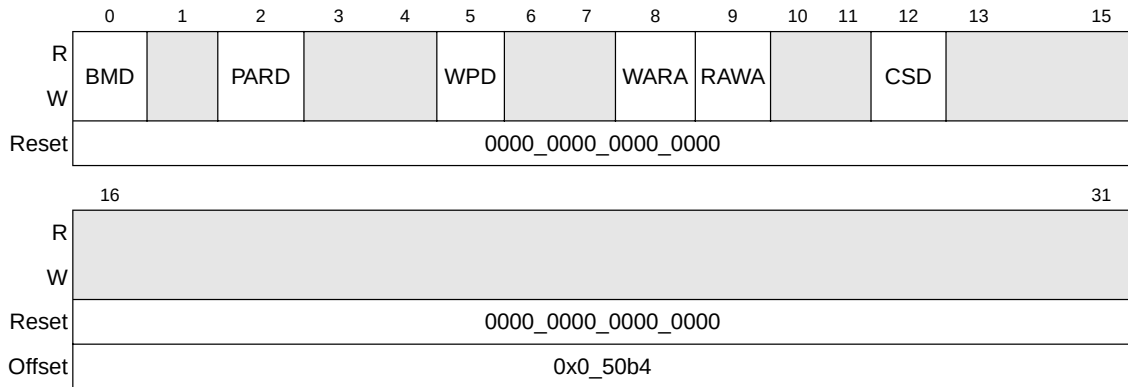


Figure 13-14. Transfer Error Check Disable Register

Table 13-17 describes LTEDR fields.

Table 13-17. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable. 0 Bus monitor is enabled. 1 Bus monitor is disabled.
1	—	Reserved
2	PARD	Parity error checking disabled. 0 Parity error checking is enabled. 1 Parity 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
12	CSD	Chip select error checking disable. 0 Chip select error checking is enabled. 1 Chip select error checking is disabled.
13–31	—	Reserved

13.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

This register is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. Once an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.

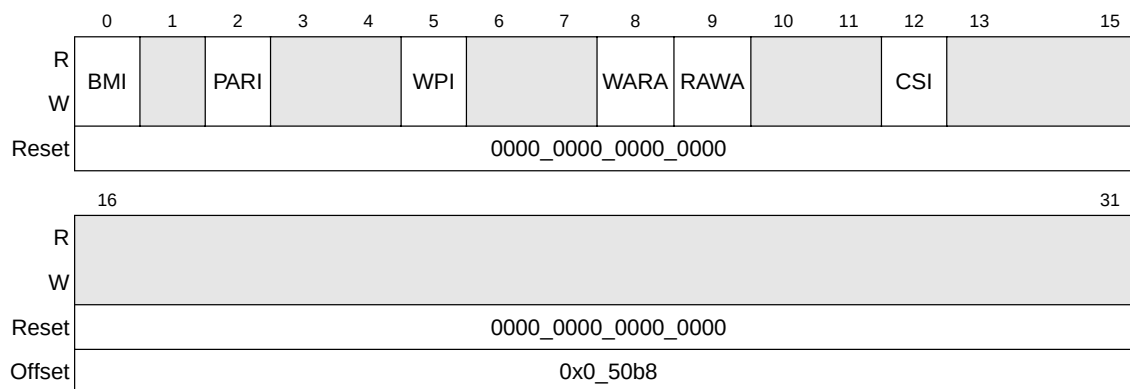


Figure 13-15. Transfer Error Interrupt Enable Register

Table 13-18 describes LTEIR fields.

Table 13-18. 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	—	Reserved
2	PARI	Parity error interrupt enable. 0 Parity error reporting is disabled. 1 Parity error reporting is enabled.
3–4	—	Reserved
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

Table 13-18. LTEIR Field Descriptions (continued)

Bits	Name	Description
12	CSI	Chip select error interrupt enable. 0 Chip select error reporting is disabled. 1 Chip select error reporting is enabled.
13–31	—	Reserved

13.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 13-16 shows the LTEATR. Once LTEATR[V] has been set, software must clear this bit to allow LTEATR and LTEAR to update following any subsequent errors.

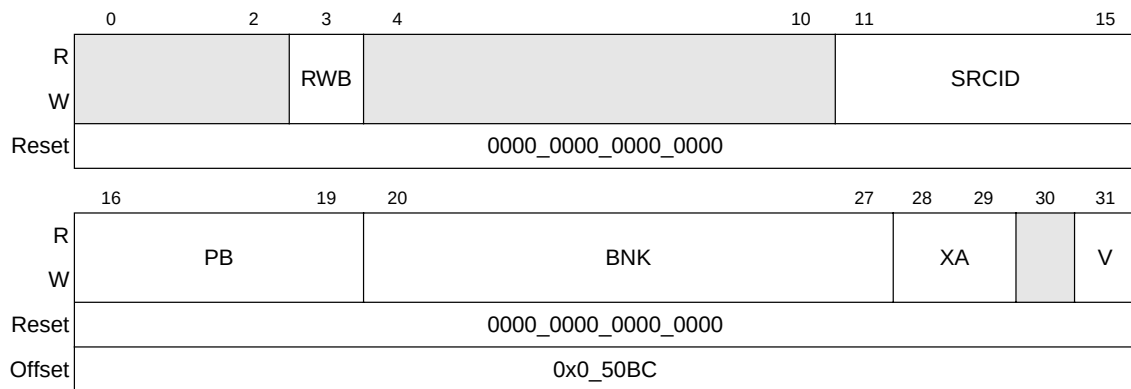
**Figure 13-16. Transfer Error Attributes Register**

Table 13-19 describes LTEATR fields.

Table 13-19. 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	These bits capture the source of the transaction when this information is provided on the internal interface to the LBC.
16–19	PB	Parity error on byte. 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).
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. Note that BNK is invalid if the error was not caused by parity checks.
28–29	XA	Extended address for the error. These bits capture the two most significant bits of the 34-bit address of the transaction resulting in an error.

Table 13-19. LTEATR Field Descriptions

Bits	Name	Description
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.

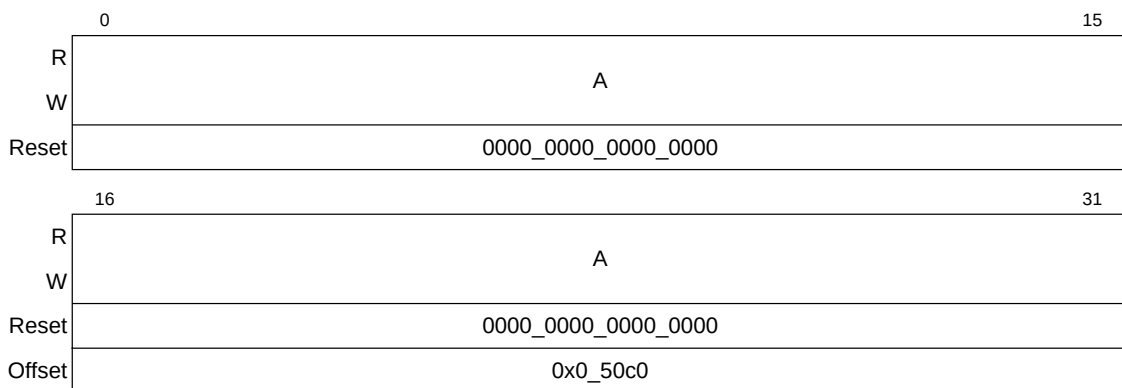
13.3.1.14 Transfer Error Address Register (LTEAR)**Figure 13-17. Transfer Error Address Register**

Table 13-20 describes LTEAR fields.

Table 13-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Captures the 32 lsbs of the 34-bit address of the transaction resulting in an error.

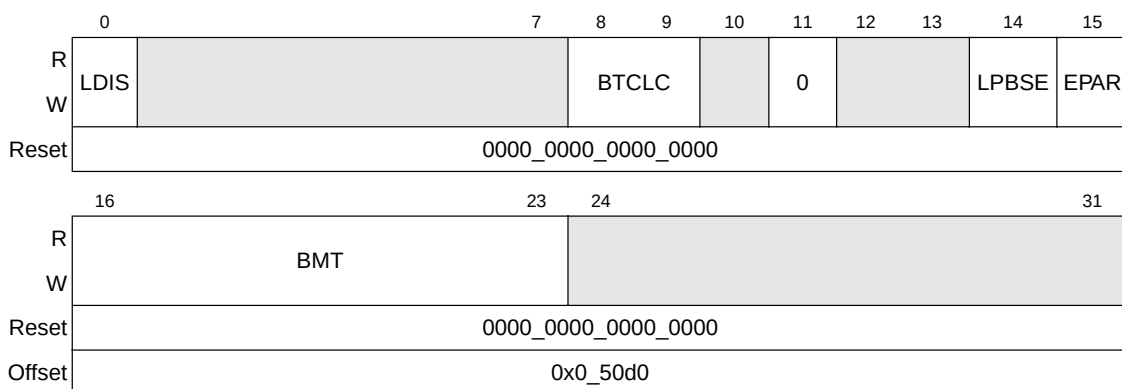
13.3.1.15 Local Bus Configuration Register (LBCR)**Figure 13-18. Local Bus Configuration Register**

Table 13-21 describes LBCR fields.

Table 13-21. 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	BCTL	Defines the use of LBCTL. 00 LBCTL is used as $\overline{W/\overline{R}}$ 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.
10–11	—	Reserved
12–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}/\overline{LGPL4}/UPWAIT/LPBSE$ pin. 0 Parity byte select is disabled. $\overline{LGTA}/\overline{LGPL4}/UPWAIT/LPBSE$ pin is available for memory control as $\overline{LGPL4}$ (output) or $\overline{LGTA}/UPWAIT$ (input). 1 Parity byte select is enabled. $\overline{LGTA}/\overline{LGPL4}/UPWAIT/LPBSE$ pin is dedicated as the parity byte select output, and $\overline{LGTA}/UPWAIT$ is disabled.
15	EPAR	Determines odd or even parity. Writing the memory with EPAR = 1 and reading the memory with EPAR = 0 generates parity errors for testing. 0 Odd parity. 1 Even parity.
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 2048 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 x 8. Apart from BMT = 0x00, the minimum value of BMT is 5, corresponding with 40 bus cycles. Shorter time-outs may result in spurious errors during SDRAM operation.
24–31	—	Reserved

13.3.1.16 Clock Ratio Register (LCRR)

The clock ratio register sets the system clock /LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

Memory Map/Register Definition

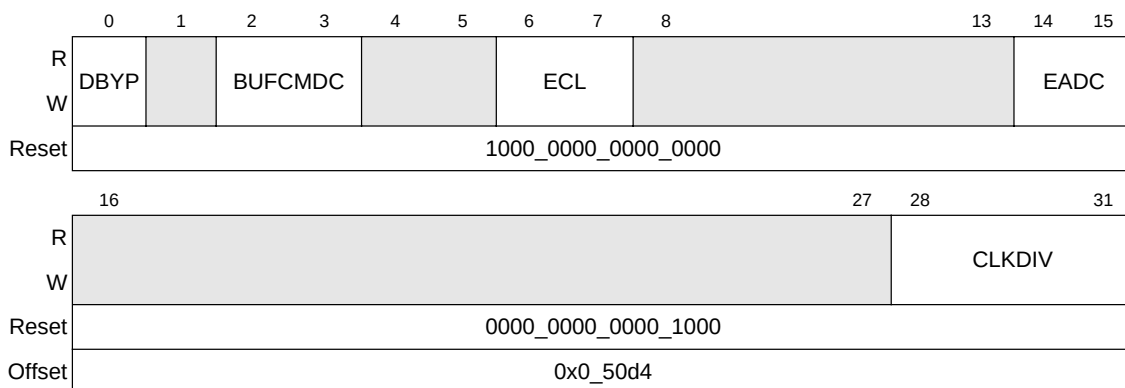


Figure 13-19. Clock Ratio Register (LCRR)

Table 13-22 describes LCRR fields.

Table 13-22. LCRR Field Descriptions

Bits	Name	Description
0	DBYP	DLL bypass. This bit should be set when using low bus clock frequencies if the DLL is unable to lock. When in DLL bypass mode, incoming data is captured in the middle of the bus clock cycle. 0 The DLL is enabled. 1 The DLL is bypassed.
1	—	Reserved
2–3	BUFCMDC	Additional delay cycles for SDRAM control signals. Defines the number of cycles to be added for each SDRAM command when LSDMR[BUFCMD] = 1. 00 4 01 1 10 2 11 3
4–5	—	Reserved
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved
14–15	EADC	External address delay cycles. Defines the number of cycles for the assertion of LALE. 00 4 01 1 10 2 11 3

Table 13-22. LCRR Field Descriptions (continued)

Bits	Name	Description
16–27	—	Reserved
28–31	CLKDIV	System (input) clock divider. Sets the frequency ratio between the (input) system clock and the memory bus clock. Only the values shown in the table below are allowed. 0000 reserved 0001 reserved 0010 2 0011 reserved 0100 4 0101 reserved 0110 reserved 0111 reserved 1000 8 1001–1111 reserved

13.4 Functional Description

The local bus controller allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to synchronous DRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal delay-locked loop (DLL) for bus clock generation ensures improved data set-up margins for board designs.
- The GPCM provides interfacing for simpler, lower-performance memories and memory-mapped devices. The GPCM has inherently lower performance because it does not support bursting. For this reason, GPCM-controlled banks are used primarily for boot-loading and access to low-performance memory-mapped peripherals.
- 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. The UPM can be used to generate flexible, user-defined timing patterns for control signals that govern a memory device. These patterns define how the external control signals behave during a read, write, burst-read, or burst-write access. Refresh timers are also available to periodically initiate user-defined refresh patterns.

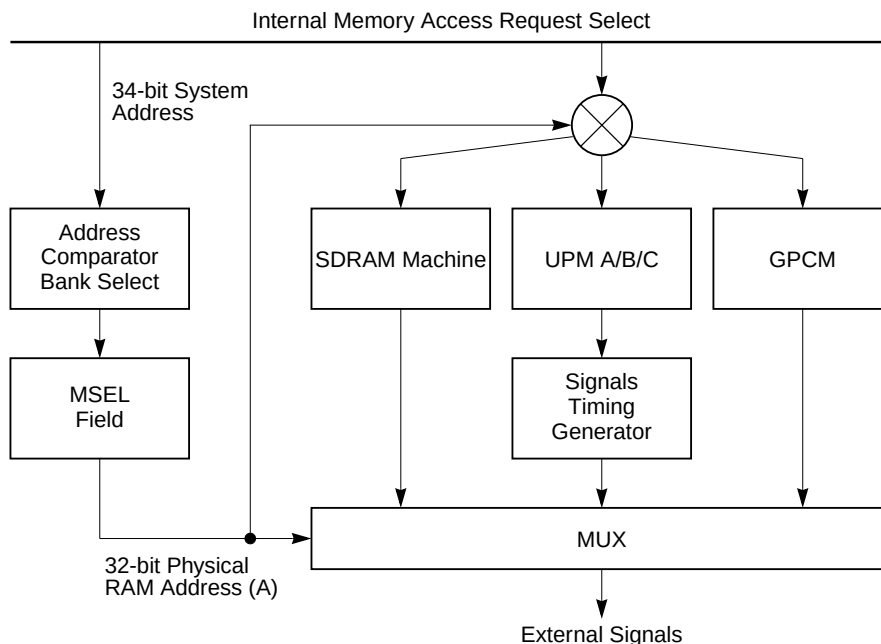


Figure 13-20. Basic Operation of Memory Controllers in LBC

Each memory bank (chip select) can be assigned to any one of these three type of machines via the machine select bits of the base register for that bank (BRx[MSEL]), as illustrated in Figure 13-20. If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

13.4.1 Basic Architecture

13.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BRx registers, while the corresponding address masks are written to the ORx registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the nineteen MSBs of the address, masked by ORx[XAM] and ORx[AM], with the base address for each bank (BRx[XBA] and BRx[BA]). If a match is found on a memory controller bank, the attributes defined in the BRx and ORx for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

13.4.1.2 External Address Latch Enable Signal (LALE)

The local bus uses a multiplexed address/data bus. Therefore the local bus controller must distinguish between address and data phases, which take place on the same bus (LAD[0:31])

pins). The LALE signal, when asserted, signifies an address phase during which the LBC drives the memory address on the LAD[0:31] 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[0:31] 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.

The frequency of LALE assertion varies across the three memory controllers. In the case of GPCM, every assertion of $\overline{\text{LCSx}}$ is considered an independent access, and accordingly, LALE will assert prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and $\overline{\text{LCSx}}$ 32 times in order to satisfy a 32-byte cache line transfer. The SDRAM controller asserts LALE only to initiate a burst transfer with a starting address, therefore no more than one assertion of LALE may be required for SDRAM to transfer a 32-byte cache line via a 32-bit port. 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[27:31] with and without LALE being involved. In general, when using the GPCM and SDRAM controllers it is not necessary to use LA[27:31] if a sufficiently wide latch is used to capture the entire address during LALE phases. UPM may require LA[27:31] if LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by LBC, Figure 13-21 shows LBC signals for GPCM performing a 32-byte write starting at hexadecimal 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.

Functional Description

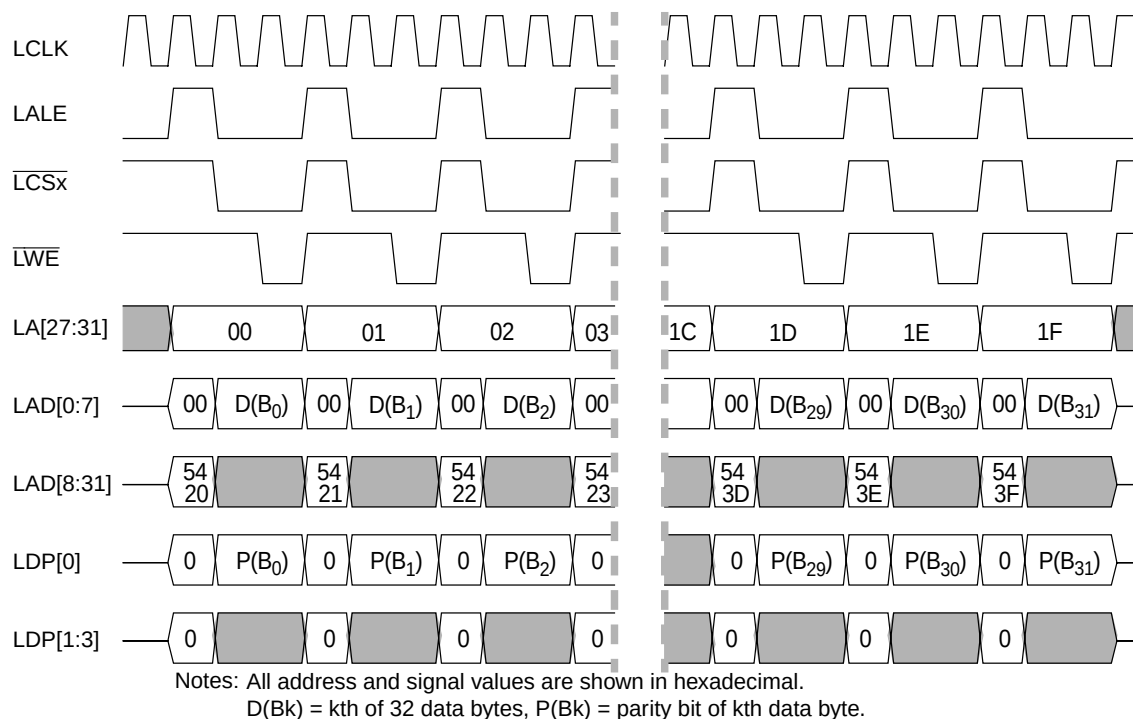


Figure 13-21. Example of 8-bit GPCM writing 32 bytes to address 0x5420.

13.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] 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 TA is asserted internally by the LBC. During LBC debug mode, TA is also visible externally on the MDVAL pin. GPCM and SDRAM controllers automatically generate TA according to the timing parameters programmed for them in option and mode registers; UPM generates TA only when a UPM pattern has the UTA RAM word bit set. An illustration of LALE, TA (internal), and $\overline{\text{LCSx}}$ is shown in Figure 13-22. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCSx}}$ (or any other control signal) remains negated or frozen.

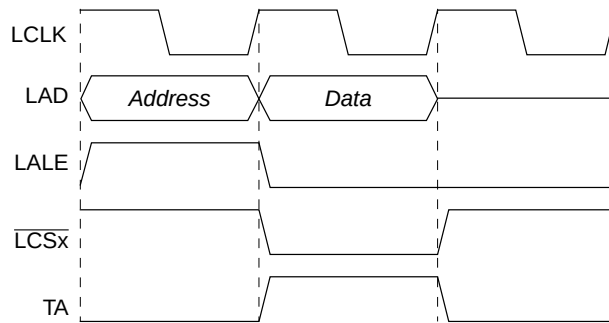


Figure 13-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCSx}}$

13.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. The LBCTL control can be disabled by setting $\text{ORx}[\text{BCTLD}]$. Access to an SDRAM machine controlled bank does not activate the LBCTL control. The LBCTL control can be further configured by $\text{LBCR}[\text{BCTLC}]$ 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 ($\text{LBCR}[\text{BCTLC}] = 00$), 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.

If an external bus transceiver is used, LBCTL should be used to signify the write direction when high. Note that the default (reset and bus idle) value of LBCTL is also high.

13.4.1.5 Atomic Operation

The LBC supports the following kinds of atomic bus operations (set by $\text{BRx}[\text{ATOM}]$):

- Read-after-write (RAWA). When a write access hits a memory bank in which $\text{ATOM} = 01$, the LBC 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). This feature is intended for CAM operations.

- Write-after-read (WARA). When a read access hit a memory bank in which $ATOM = 10$, the LBC 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 do not change reservation status, but are otherwise processed normally. 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).

13.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming $BRx[DECC]$. Parity is generated and checked on a per-byte basis using $LDP[0:3]$ for the bank if $BRx[DECC] = 01$ (normal parity) or $BRx[DECC] = 10$ for read-modify-write (RMW) parity. Byte lane parity on $LDP[0:3]$ is generated regardless of the setting of $BRx[DECC]$. 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).

13.4.1.7 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 ($LBCR[BMT]$) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the count down until the data tenure is completed and then idles if there is no pending transaction. Bus monitor error reporting through $LTESR[BM]$ can be disabled by setting the $LTEDR[BMD]$ bit. However, even if bus monitor error reporting is disabled, the bus monitor is still active and can generate a UPM exception, 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—particularly for SDRAMs—resulting in incomplete data transfers. Accordingly, apart from the reset value of 0x00 (corresponding with the maximum time-out of 2048 bus cycles), $LBCR[BMT]$ must not be set below 0x05 (or 40 bus cycles for time-out) under any circumstances.

13.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPRM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups— BRx and ORx .

Figure 13-23 shows a simple connection between an 8-bit port size SRAM device and the local bus controller in GPCM mode. Byte-write enable signals ($\overline{LWE}$) are available for each byte written to memory. Also, the output enable signal ($\overline{LOE}$) 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.

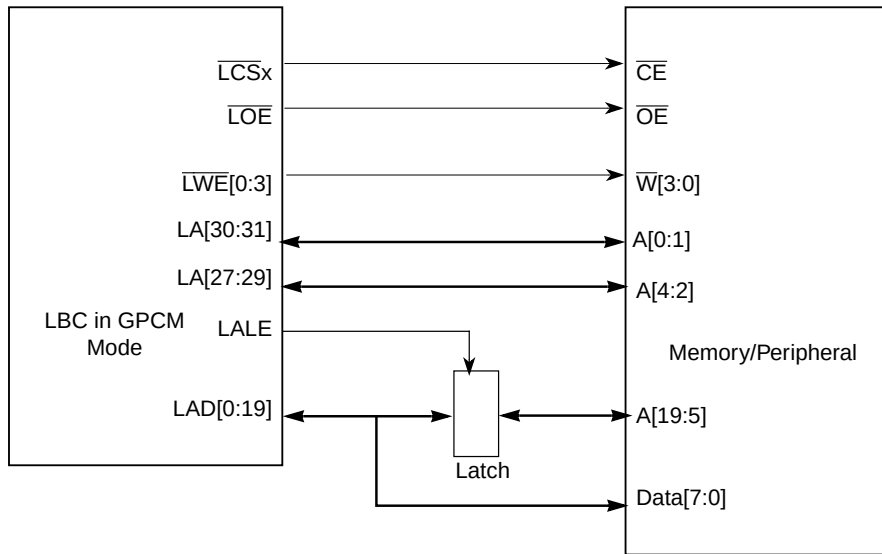


Figure 13-23. Local Bus to GPCM Device Interface

Figure 13-24 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure $\text{ORx}[\text{ACS}]$ to specify $\overline{\text{LCS}}$ to meet this requirement.

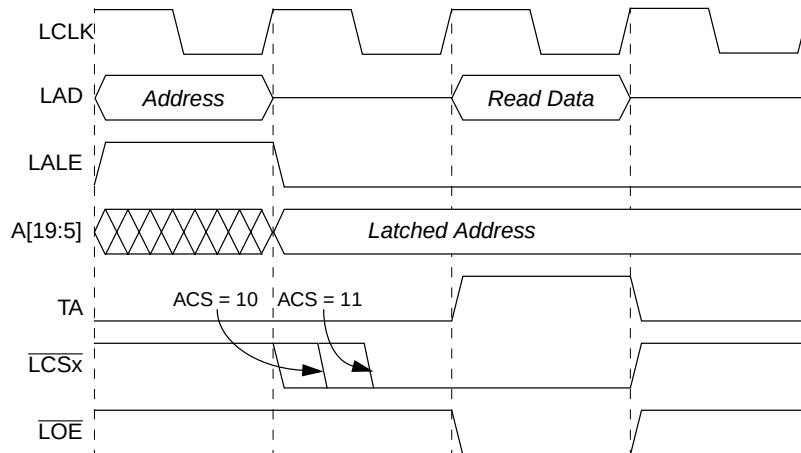


Figure 13-24. GPCM Basic Read Timing
($\text{XACS} = 0$, $\text{ACS} = 1\text{x}$, $\text{TRLX} = 0$, $\text{CLKDIV} = 4,8$)

13.4.2.1 Timing Configuration

If $\text{BRx}[\text{MSEL}]$ selects the GPCM, the attributes for the memory cycle are taken from ORx . These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR and SETA fields. Table 13-35 shows signal behavior and system response for a write access with

Functional Description

LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 13-24 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 13-25 and Table 13-26 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 13-23. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCSx}}$ Asserted	$\overline{\text{LCSx}}$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles ¹
0	0	00	0	0	0	0	3+SCY
0	0	10	0	1/4	0	0	3+SCY
0	0	11	0	1/2	0	0	3+SCY
0	1	00	0	0	0	0	3+SCY
0	1	10	0	1	0	0	3+SCY
0	1	11	0	2	0	0	4+SCY
0	0	00	1	0	0	-1/4	3+SCY
0	0	10	1	1/4	-1/4	-1/4	3+SCY
0	0	11	1	1/2	-1/4	-1/4	3+SCY
0	1	00	1	0	0	-1/4	3+SCY
0	1	10	1	1	-1/4	-1/4	3+SCY
0	1	11	1	2	-1/4	-1/4	4+SCY
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/4	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	-1-1/4	4+2*SCY
1	0	10	1	1+1/4	-1-1/4	-1-1/4	5+2*SCY
1	0	11	1	1+1/2	-1-1/4	-1-1/4	5+2*SCY
1	1	00	1	0	0	-1-1/4	4+2*SCY
1	1	10	1	2	-1-1/4	-1-1/4	5+2*SCY
1	1	11	1	3	-1-1/4	-1-1/4	6+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (ORx[EAD] = 0; ORx[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 13-24. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCSx}}$ Asserted	$\overline{\text{LCSx}}$ Negated to Address Change	Total Cycles ¹
0	0	0	00	0	1	4+SCY
0	0	0	10	1/4	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/4	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/4	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/4	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (ORx[EAD] = 0; ORx[EAD] = 1 and LCRR[EADC] = 01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 13-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (bus clock cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCSx}}$ Asserted	$\overline{\text{LCSx}}$ Negated to Address Change	$\overline{\text{LWE}}$ Negated to Address/Data Invalid	Total Cycles ¹
0	0	00	0	0	0	0	3+SCY
0	0	10	0	1/2	0	0	3+SCY
0	0	11	0	1/2	0	0	3+SCY
0	1	00	0	0	0	0	3+SCY
0	1	10	0	1	0	0	3+SCY
0	1	11	0	2	0	0	4+SCY
0	0	00	1	0	0	0	3+SCY
0	0	10	1	1/2	0	0	3+SCY
0	0	11	1	1/2	0	0	3+SCY
0	1	00	1	0	0	0	3+SCY
0	1	10	1	1	0	0	3+SCY
0	1	11	1	2	0	0	4+SCY
1	0	00	0	0	0	0	3+2*SCY
1	0	10	0	1+1/2	0	0	4+2*SCY
1	0	11	0	1+1/2	0	0	4+2*SCY
1	1	00	0	0	0	0	3+2*SCY
1	1	10	0	2	0	0	4+2*SCY
1	1	11	0	3	0	0	5+2*SCY
1	0	00	1	0	0	-1	4+2*SCY
1	0	10	1	1+1/2	-1	-1	5+2*SCY
1	0	11	1	1+1/2	-1	-1	5+2*SCY
1	1	00	1	0	0	-1	4+2*SCY
1	1	10	1	2	-1	-1	5+2*SCY
1	1	11	1	3	-1	-1	6+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (ORx[EAD]=0; ORx[EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

Table 13-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCSx}}$ Asserted	$\overline{\text{LCSx}}$ Negated to Address Change	Total Cycles ¹
0	0	0	00	0	1	4+SCY
0	0	0	10	1/2	1	4+SCY
0	0	0	11	1/2	1	4+SCY
0	0	1	00	0	1	4+SCY
0	0	1	10	1	1	4+SCY
0	0	1	11	2	1	5+SCY
0	1	0	00	0	2	5+SCY
0	1	0	10	1/2	2	5+SCY
0	1	0	11	1/2	2	5+SCY
0	1	1	00	0	2	5+SCY
0	1	1	10	1	2	5+SCY
0	1	1	11	2	2	6+SCY
1	0	0	00	0	5	8+2*SCY
1	0	0	10	1+1/2	5	9+2*SCY
1	0	0	11	1+1/2	5	9+2*SCY
1	0	1	00	0	5	8+2*SCY
1	0	1	10	2	5	9+2*SCY
1	0	1	11	3	5	10+2*SCY
1	1	0	00	0	9	12+2*SCY
1	1	0	10	1+1/2	9	13+2*SCY
1	1	0	11	1+1/2	9	13+2*SCY
1	1	1	00	0	9	12+2*SCY
1	1	1	10	2	9	13+2*SCY
1	1	1	11	3	9	14+2*SCY

¹ Total cycles when LALE is asserted for one cycle only (ORx[EAD]=0; ORx[EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than one cycle increases the total cycle count accordingly.

13.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM, support an option to drive the $\overline{\text{LCSx}}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCSx}}$ can be driven in any of the following ways:

Functional Description

- Simultaneous with the latched memory address. (Refers to the externally latched address and not the address timing on LAD[0:31]. That is, chip select does not assert during LALE).
- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4, 8).
- One half of a clock cycle later (for LCRR[CLKDIV] = 2, 4, or 8).
- One clock cycle later (for LCRR[CLKDIV] = 4), when ORx[XACS] = 1.
- Two clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when ORx[XACS] = 1.
- Three clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when ORx[XACS] = 1 and ORx[TRLX] = 1.

The timing diagram in Figure 13-24 shows two chip-select assertion timings for the case LCRR[CLKDIV] = 4 or 8. If LCRR[CLKDIV] = 2, $\overline{\text{LCSx}}$ asserts identically for ORx[ACS] = 10 or 11.

13.4.2.2.1 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 ORx[SCY] and ORx[TRLX]. Internal generation of transfer acknowledge is enabled if ORx[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 ORx[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}}$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from ORx[SCY] cycles to 2*ORx[SCY] cycles, allowing a maximum of 30 wait states.

13.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 13-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCSx}}$ 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.

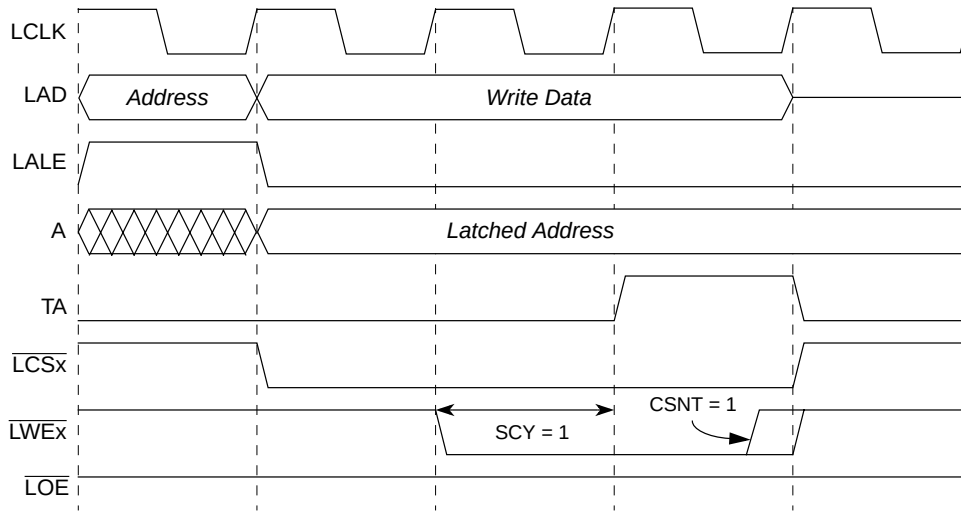


Figure 13-25. GPCM Basic Write Timing
(XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)

As Figure 13-25 shows, the timing for $\overline{\text{LCSx}}$ is the same as for the latched address. The strobes for the transaction are supplied by $\overline{\text{LOE}}$ or $\overline{\text{LWEx}}$, depending on the transaction direction—read or write (write case shown in the figure). $\text{ORx}[\text{CSNT}]$ controls 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 provided that $\text{LCRR}[\text{CLDIV}] = 4$ or 8. For example, when $\text{ACS} = 00$ and $\text{CSNT} = 1$, $\overline{\text{LWEx}}$ is negated one quarter of a clock earlier, as shown in Figure 13-25. If $\text{LCRR}[\text{CLDIV}] = 2$, $\overline{\text{LWEx}}$ is negated either coincident with $\overline{\text{LCSx}}$ or one cycle earlier.

13.4.2.2.3 Relaxed Timing

$\text{ORx}[\text{TRLX}]$ is provided for memory systems that require more relaxed timing between signals. Setting $\text{TRLX} = 1$ 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.
- $\overline{\text{LCSx}}$ 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.

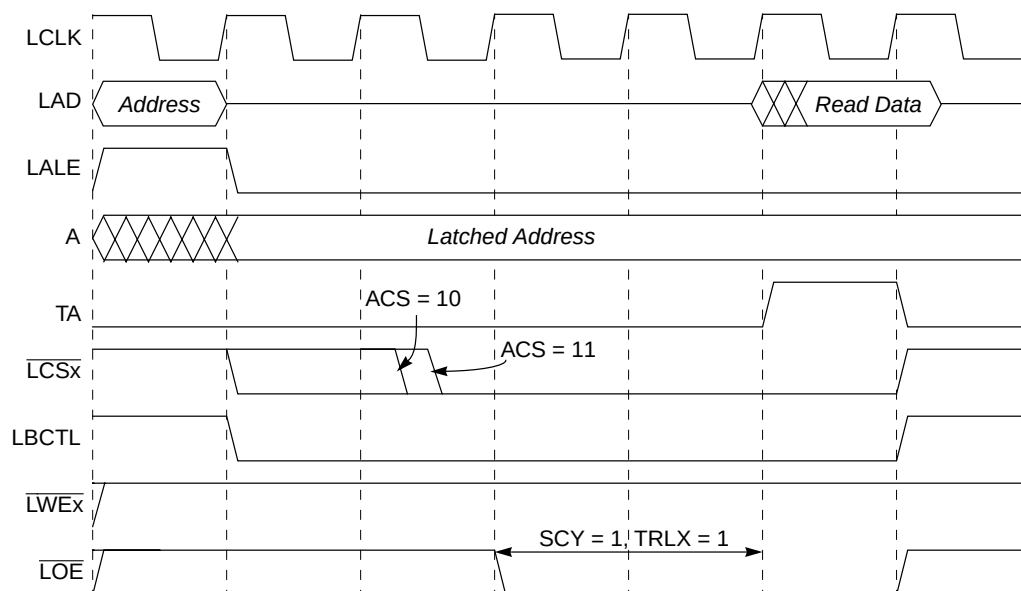


Figure 13-26. GPCM Relaxed Timing Read
(XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

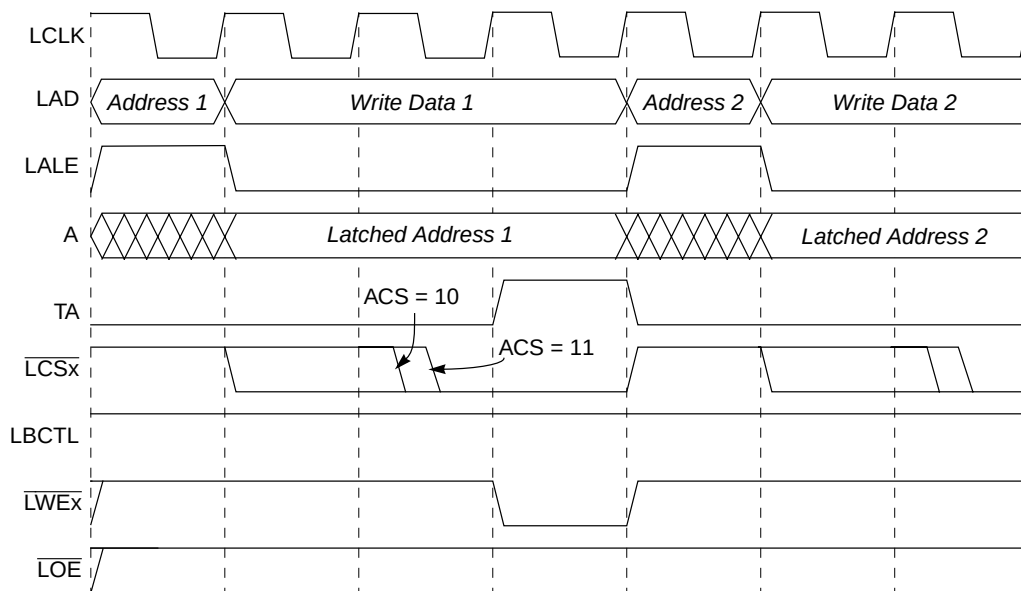


Figure 13-27. GPCM Relaxed Timing Back-to-Back Writes
(XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

Relaxed timing read and write transactions are illustrated in Figure 13-26 and Figure 13-27. The effect of CLKDIV = 2 for these examples is only to delay the assertion of $\overline{\text{LCSx}}$ in the ACS = 10 case to the ACS = 11 case. The example in Figure 13-27 also

shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

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 12-28 and Figure 12-29. If $\text{ACS} \neq 00$, $\overline{\text{LCSx}}$ is also negated one clock earlier.

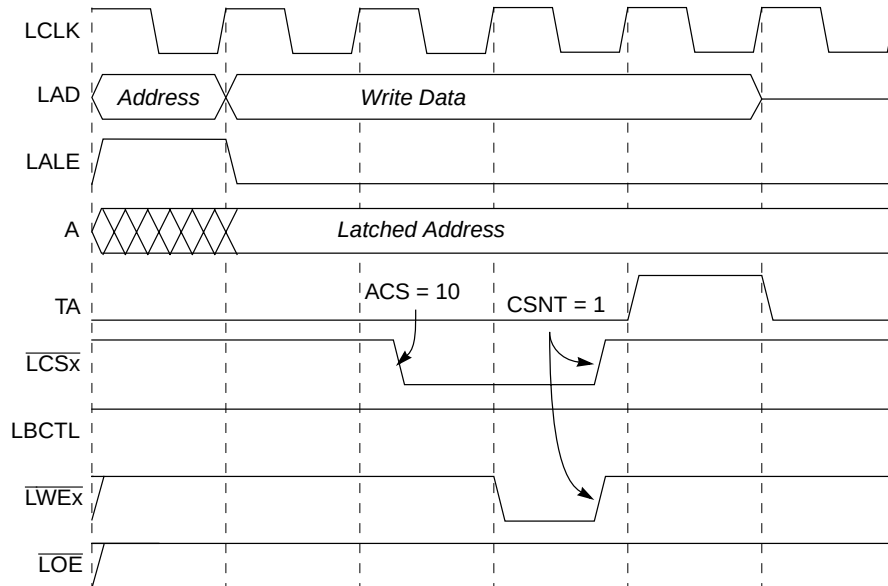


Figure 13-28. GPCM Relaxed Timing Write
(XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

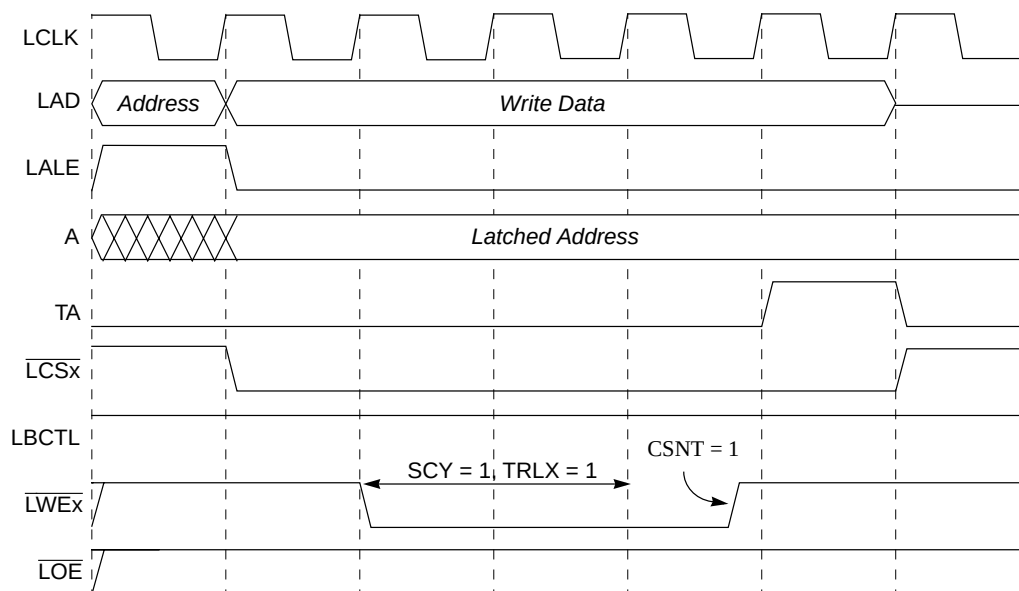


Figure 13-29. GPCM Relaxed Timing Write
 (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

13.4.2.2.4 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 $\overline{\text{LCSx}}$ is asserted or coinciding with $\overline{\text{LCSx}}$ (if XACS = 1 and ACS = 10 or ACS = 11). Accordingly, assertion of $\overline{\text{LOE}}$ can be delayed (along with the assertion of $\overline{\text{LCSx}}$) by programming TRLX = 1. $\overline{\text{LOE}}$ negates on the rising clock edge coinciding with $\overline{\text{LCSx}}$ negation.

13.4.2.2.5 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 ORx[TRLX] and ORx[EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in Table 13-6 in addition to any existing bus turnaround cycle.

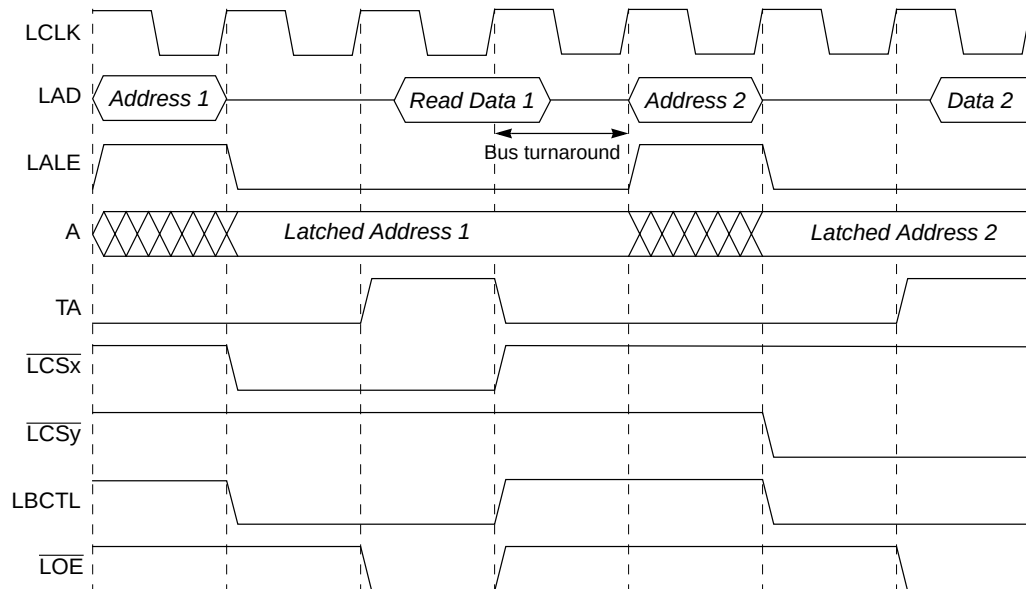


**Figure 13-30. GPCM Read Followed by Read
(TRLX = 0, EHTR = 0, Fastest Timing)**

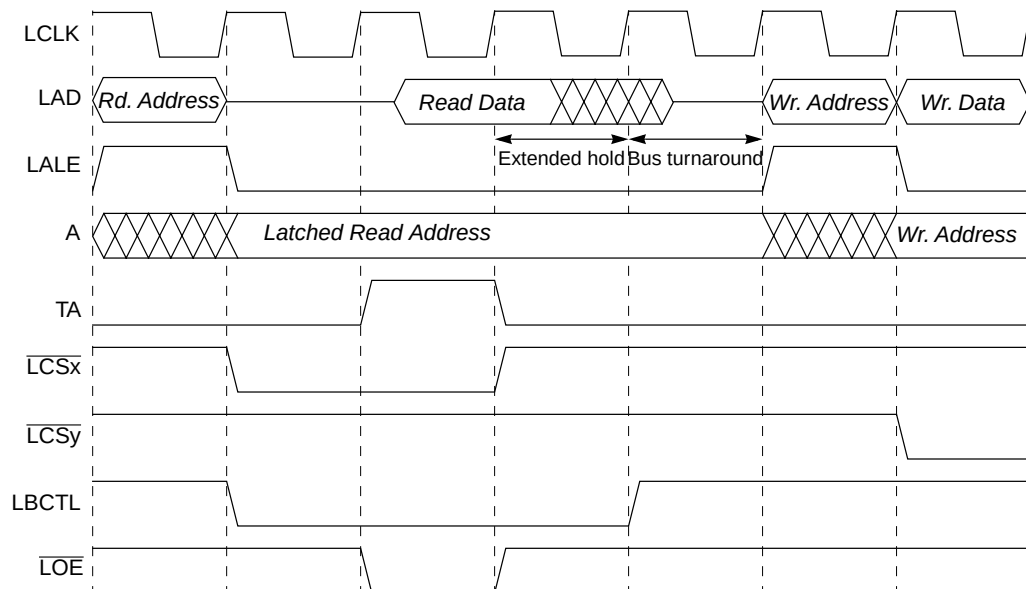


**Figure 13-31. GPCM Read Followed by Write
(TRLX = 0, EHTR = 1, 1-cycle Extended Hold Time on Reads)**

13.4.2.3 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 $\overline{\text{LCSx}}$, 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 $\text{ORx}[\text{SETA}]$). $\overline{\text{LGTA}}$ 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 $\text{ORx}[\text{SETA}]$. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if $\text{ORx}[\text{SETA}] = 0$ (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if $\text{ORx}[\text{SETA}] = 1$. The timing of $\overline{\text{LGTA}}$ is illustrated by the example in Figure 13-32.

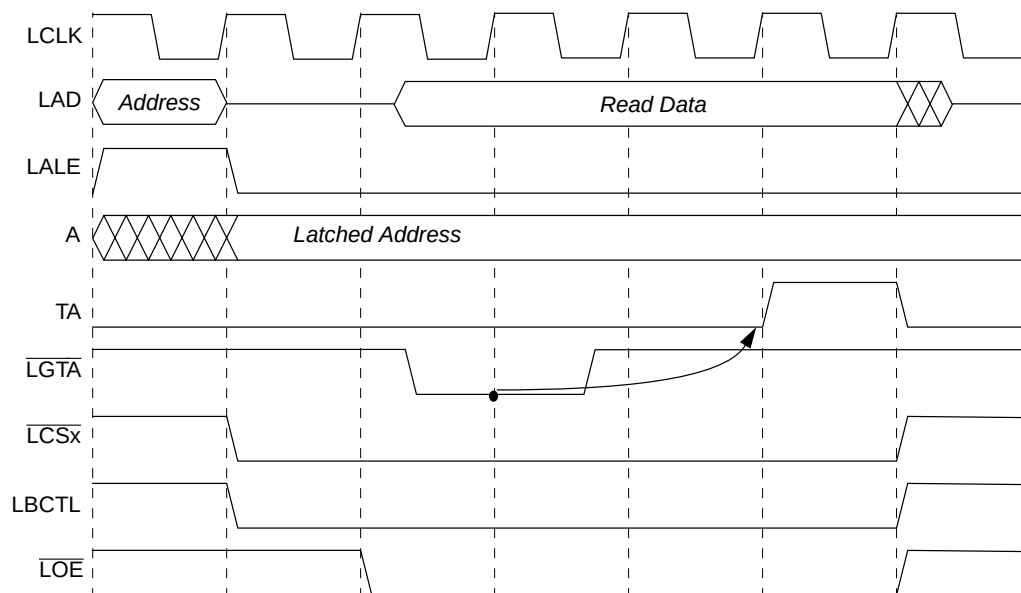


Figure 13-32. External Termination of GPCM Access

13.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. The $\overline{\text{LCS0}}$ signal is the boot chip-select output; its operation differs from the 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 13-27 describes the initial values of the boot bank in the memory controller.

Table 13-27. Boot Bank Field Values after Reset

Register	Field	Setting
BR0	BA	0000_0000_0000_0000_0
	XBA	00
	PS	From pin during reset.
	DECC	00
	WP	0
	MSEL	000
	ATOM	00
	V	1
OR0	AM	0000_0000_0000_0000_0
	XAM	00
	BCTLD	0
	CSNT	1
	ACS	11
	XACS	1
	SCY	1111
	SETA	0
	TRLX	1
	EHTR	1
	EAD	1

13.4.3 SDRAM Machine

The local bus controller provides an SDRAM interface (machine) for the local bus. The machine provides the control functions and signals for Intel PC133 and JEDEC-compliant SDRAM devices. Each bank can control an SDRAM device on the local bus.

13.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR). Figure 13-33 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[4:0] of the SDRAM connect directly to LA[27:31], address pin A10 connects to the LBC's dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

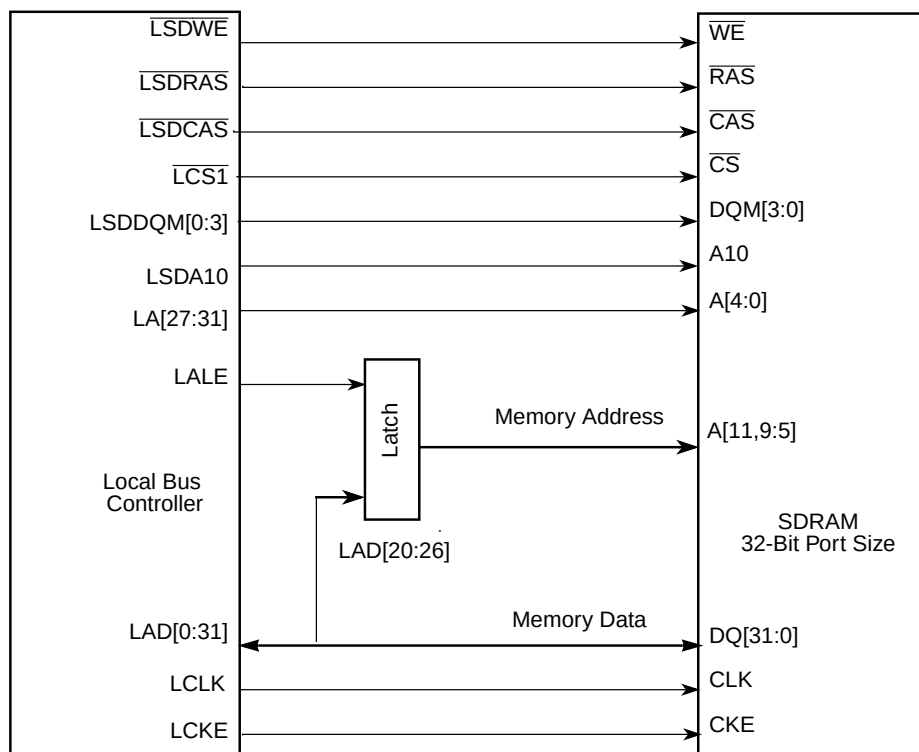


Figure 13-33. Connection to a 32-Bit SDRAM with 12 Address Lines

13.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR_x, BR_x, LSDMR). After all memory parameters are configured, system software should execute the following initialization sequence for each SDRAM device.

- Issue a PRECHARGE-ALL-BANKS command
- Issue eight AUTO-REFRESH commands
- Issue a MODE-SET command to initialize the mode register

The initial commands are executed by setting LSDMR[OP] and accessing the SDRAM with any write that hits the relevant bank.

Note that software should ensure that no memory operations begin until this process completes.

13.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands

The SDRAM machine performs all accesses to SDRAM by using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

The following SDRAM interface commands are provided by setting LSDMR[OP] to a non zero value (LSDMR[OP] = 000 sets normal read/write operation):

Table 13-28. SDRAM Interface Commands

Command (LSDMR[OP])	Description
ACTIVATE (110)	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another ACTIVATE is issued.
MODE-SET (011)	Allows setting of SDRAM options—CAS latency and burst length. CAS latency depends on the SDRAM device used. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the local bus memory controller supports only 8-beat bursts for 8-bit and 32-bit port size, or 4-beat bursts for 16-bit portsize. The LBC does not support burst lengths of 1, 2 and a page for SDRAMs. The mode register data (CAS latency and burst length) is programmed into the LSDMR register by initialization software after reset. After the LSDMR is set, the LBC transfers the information to the SDRAM device by issuing a MODE-SET command.
PRECHARGE (100: single bank) (101: all-banks)	Restores data from the sense amplifiers to the appropriate row in the SDRAM device array. Also initializes the sense amplifiers to prepare for activating another row in the SDRAM device. Note that the LBC uses LSDA10 to distinguish between PRECHARGE-ALL-BANKS (LSDA10 is high) and PRECHARGE-SINGLE-BANK (LSDA10 is low). The SDRAMs must be compatible with this format.
READ (111)	Latches the column address and transfers data from the selected sense amplifier on the SDRAM device, to the output buffer as determined by the column address. During each successive clock, additional data is driven without additional read commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. Read data is discarded by the LBC.
WRITE (111)	Latches the column address and transfers data from the data signals to the selected sense amplifier on the SDRAM device, as determined by the column address. During each successive clock, additional data is transferred to the sense amplifiers from the data signals without additional write commands. At the end of the burst, the page remains open. Burst length is the one set for this bank. LSDDQM[0:3] are inactive and write data is undefined.
AUTO-REFRESH (001)	Causes a row to be read in all memory banks (JEDEC SDRAM) as determined by the refresh row address counter (similar to CBR). The refresh row address counter is internal to the SDRAM device. After being read, a row is automatically rewritten into the memory array. All banks must be in a precharged state before executing refresh.
SELF-REFRESH (010)	Allows data to be retained in the SDRAM device, even when the rest of the LBC is in a power saving mode with clocks turned off. When placed in this mode, the SDRAM device is capable of issuing its own refresh commands, without external clocking from the LBC and the LCKE pin from LBC is deasserted. This command can be issued at any time. Normal operation can be resumed only by setting LSDMR[OP] = 000, and waiting a minimum of 200 bus cycles before issuing reads or writes to LBC.

13.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the ORx register, is used along with the bank size to compare page bits of the address to the page register each time a bus-cycle access is requested. If a match is found, together with a bank match, the bus cycle is defined as a page hit. An open page is automatically closed by the SDRAM machine if the bus becomes idle, unless $ORx[PMSEL] = 0$.

13.4.3.5 Page Management

The LBC is capable of managing a maximum of 4 open pages (one page per SDRAM bank) for a single SDRAM device. After a page is opened, it remains open unless:

- the next access is to a page in a different SDRAM device, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- the next access is to a page in an SDRAM bank which has a different page open on it, in which case the old page is closed with a PRECHARGE-SINGLE-BANK command.
- the current SDRAM device requires refresh services, in which case all open pages on the current device are closed with a PRECHARGE-ALL-BANKS command.
- the bus becomes idle and $ORx[PMSEL] = 0$, in which case all open pages in the current device are closed with a PRECHARGE-ALL-BANKS command.

13.4.3.6 SDRAM Address Multiplexing

The lower bits of the address bus are connected to the memory device's address port with the memory controller multiplexing the row/column and the internal bank select lines. The position of the bank select lines are set according to $LSDMR[BSMA]$. Figure 13-34 shows how the SDRAM controller shifts the row address down to the lower output address signals during activate and shifts the bank select bits up to the address pins specified by $LSDMR[BSMA]$. The LSB of the logical row address (A_x in Figure 13-34) is aligned with the connected least significant bit of LAD (bits 29, 30 and 31 for port sizes of 32-bits, 16-bits and 8-bits respectively).

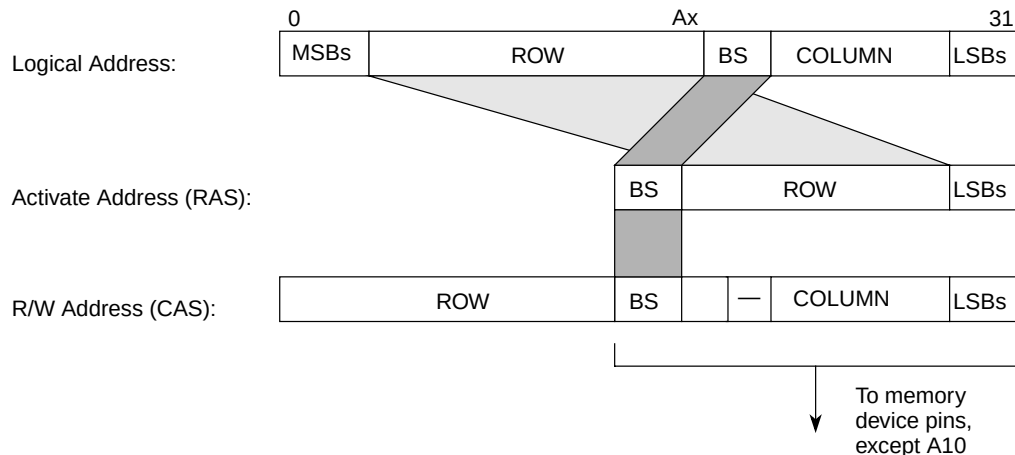


Figure 13-34. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address which includes row and column, is generated on the LAD[0:31] signals. However, address/data signal multiplexing implies that the address must be latched by an external latch that is controlled by LALE. All SDRAM device address signals need to be connected to the latched address bits and burst address bits (LA[27:31]) of the LBC, with the exception of A10, which has a dedicated connection on LSDA10. LSDA10 is driven with the appropriate row address bit for SDRAM commands that require A10 to be an address.

13.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the ORx and LSDMR registers. These parameters include the following:

- Precharge to activate interval (LSDMR[PRETOACT]).
- Activate to read/write interval (LSDMR[ACTTORW]).
- CAS latency, column address to first data out (LSDMR[CL] and LCRR[ECL]).
- Write recovery, last data in to precharge (LSDMR[WRC]).
- Refresh recovery interval (LSDMR[RFRC]).
- External buffers on the control lines present (LSDMR[BUFCMD] and LCRR[BUFCMDC]).

In addition, the LBC hardware ensures a default activate to precharge interval of 10 bus cycles. The following sections describe SDRAM parameters programmed in LSDMR.

13.4.3.7.1 Precharge-to-Activate Interval

This parameter, controlled by LSDMR[PRETOACT] defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

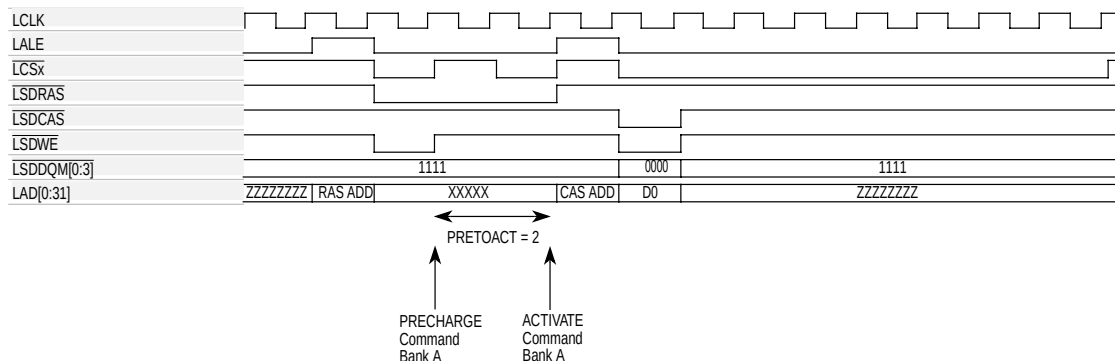


Figure 13-35. PRETOACT = 2 (2 Clock Cycles)

13.4.3.7.2 Activate to Read/Write Interval

This parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

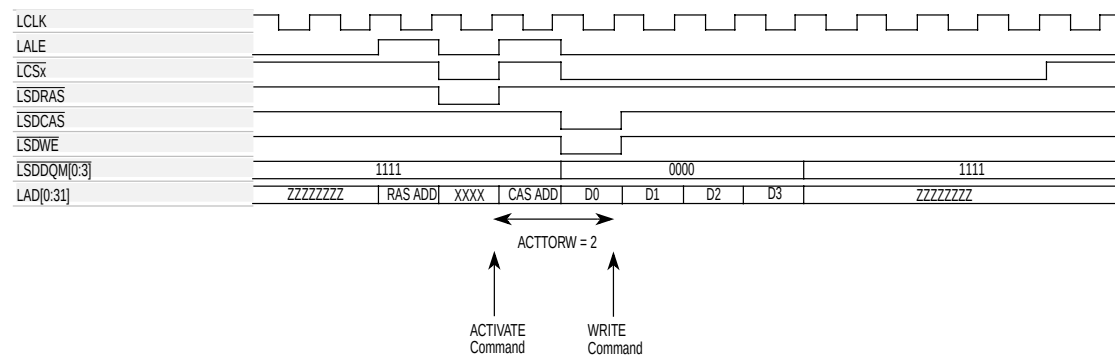


Figure 13-36. ACTTORW = 2 (2 Clock Cycles)

13.4.3.7.3 Column Address to First Data Out—CAS Latency

This parameter, controlled by LSDMR[CL] for latency of 1, 2, or 3 and by LCRR[ECL] for latency of more than 3, defines the timing for first read data after a column address is sampled by the SDRAM.

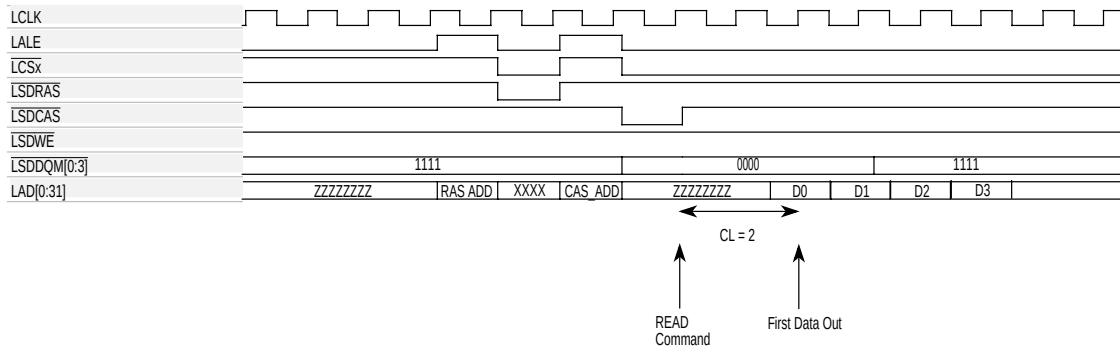


Figure 13-37. CL = 2 (2 Clock Cycles)

13.4.3.7.4 Last Data In to Precharge—Write Recovery

This parameter, controlled by LSDMR[WRC], defines the earliest timing for a PRECHARGE command after the last data was written to the SDRAM.

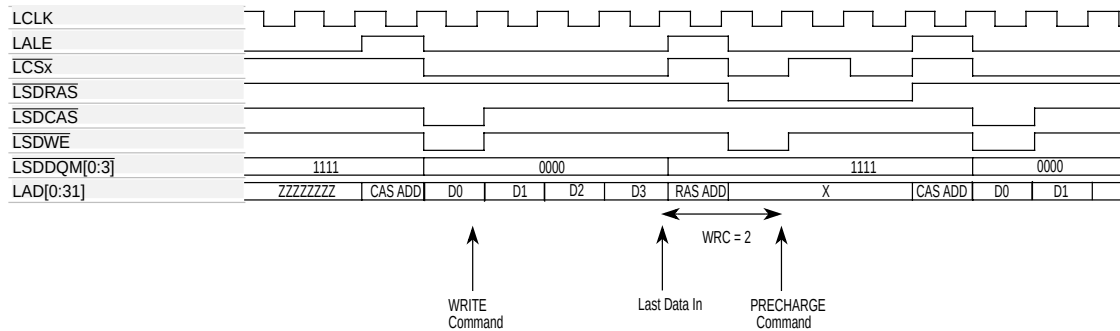


Figure 13-38. WRC = 2 (2 Clock Cycles)

13.4.3.7.5 Refresh Recovery Interval (RFRC)

This parameter, controlled by LSDMR[RFRC], defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command to the same SDRAM device.

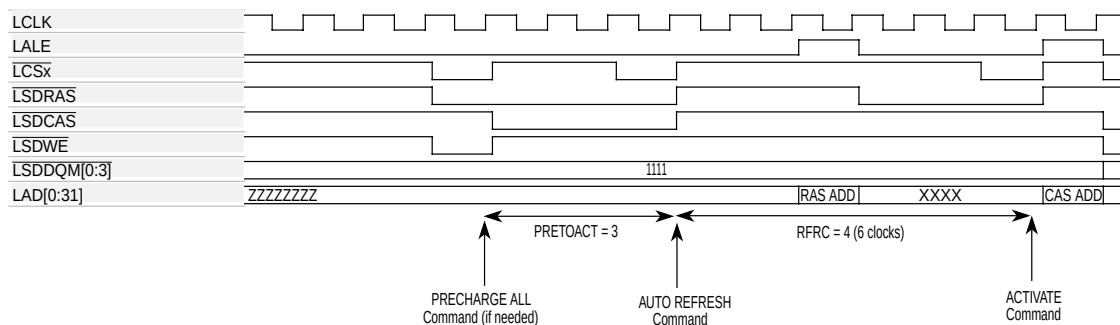


Figure 13-39. RFRC = 4 (6 Clock Cycles)

13.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and LSDA10), is endangering the device setup time, $\text{LSDMR}[\text{BUFCMD}]$ should be set. Setting this bit causes the memory controller to add $\text{LCRR}[\text{BUFCMDC}]$ extra bus cycles to the assertion of SDRAM control signals ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and LSDA10) for each SDRAM command.

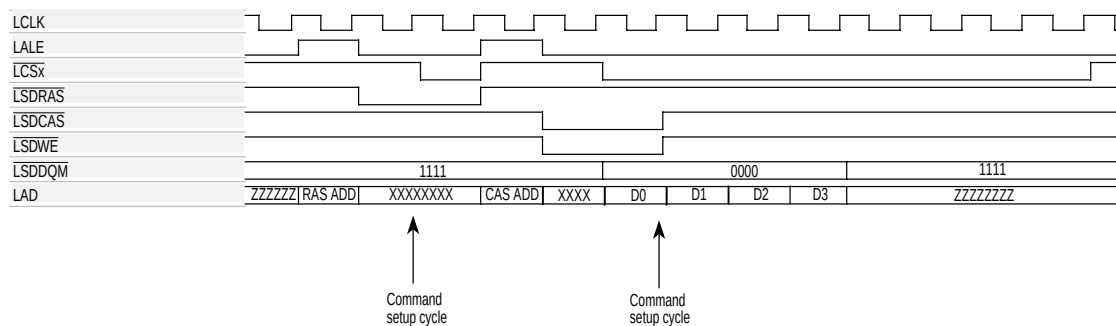


Figure 13-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

13.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

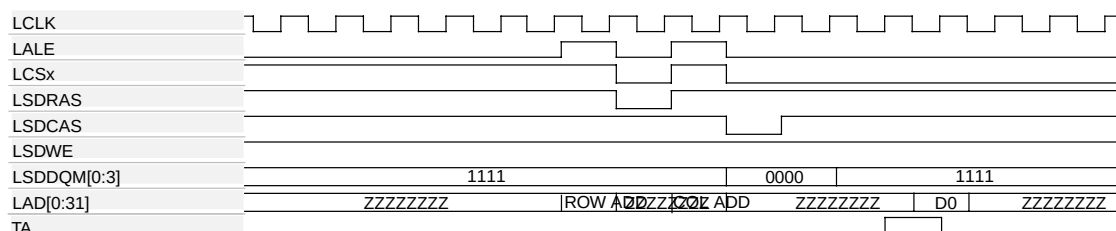


Figure 13-41. SDRAM Single-Beat Read, Page Closed, CL = 3

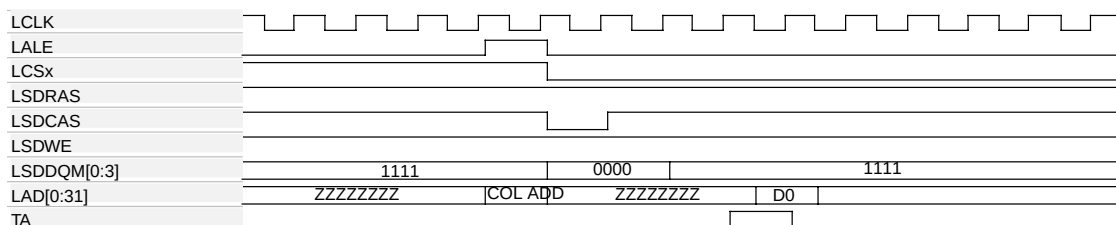


Figure 13-42. SDRAM Single-Beat Read, Page Hit, CL = 3

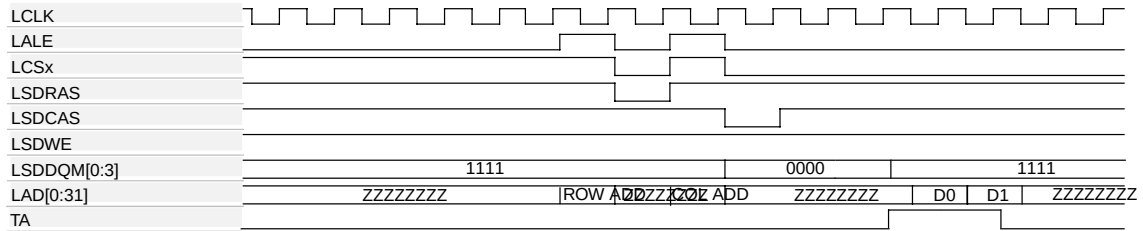


Figure 13-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

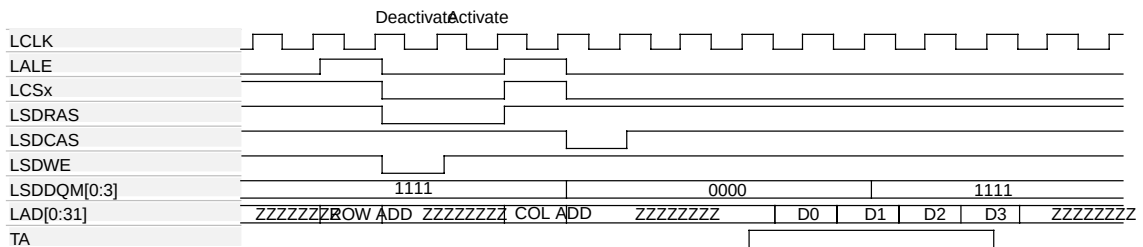


Figure 13-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

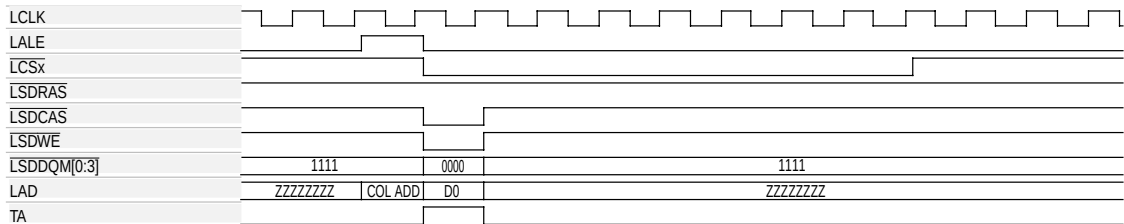


Figure 13-45. SDRAM Single Beat Write, Page Hit.

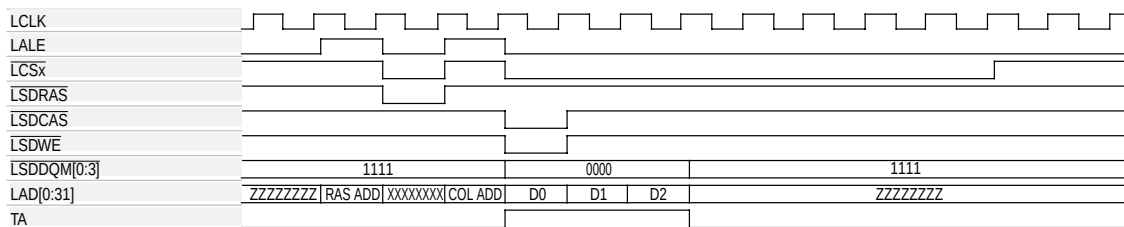


Figure 13-46. SDRAM Three-Beat Write, Page Closed

Functional Description

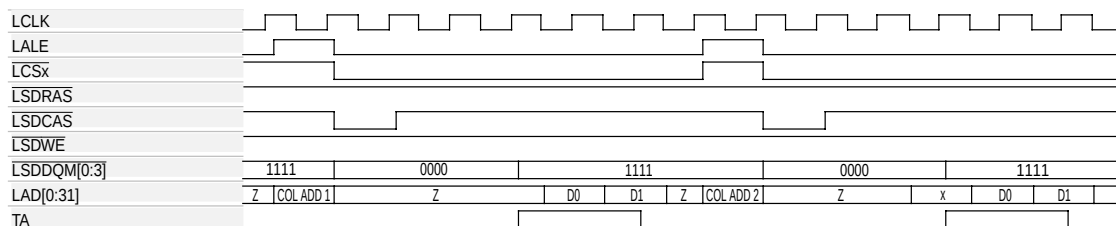


Figure 13-47. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

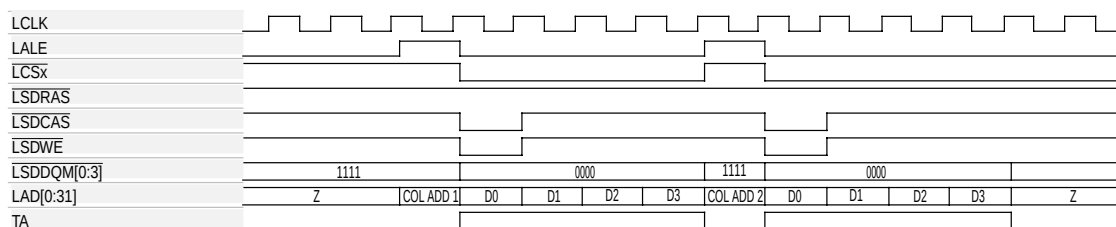


Figure 13-48. SDRAM Write-after-Write Pipelined, Page Hit

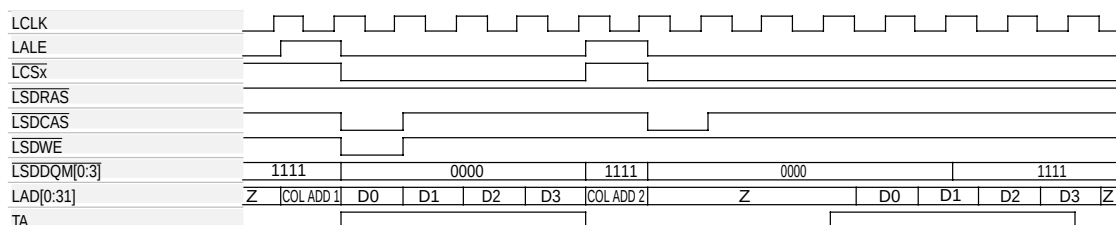


Figure 13-49. SDRAM Read-after-Write Pipelined, Page Hit

13.4.3.9 SDRAM Read/Write Transactions

The SDRAM interface supports read and write transactions of between 1 and 8 data beats for transaction sizes ranging from 1 to 32 bytes. A full burst is performed for each transaction, with the burst length dependent on the port size. A maximum burst of 8 beats is used for an 8-bit or 32-bit port size, while a maximum burst of 4 beats is used for a 16-bit port size, as programmed in LSDMR[BL]. For reads that require less than the full burst length, extraneous data in the burst is ignored and suppressed by the assertion of LSDDQM[0:3]. For writes that require less than the full burst length, the non-targeted addresses are protected by driving corresponding LSDDQM bits high (inactive) on the irrelevant cycles of the burst. However, system performance is not compromised because, if a new transaction is pending, the SDRAM controller begins executing it immediately, effectively terminating the burst early.

13.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 13-50. In this case, the latched address carries the mode bits for the command.

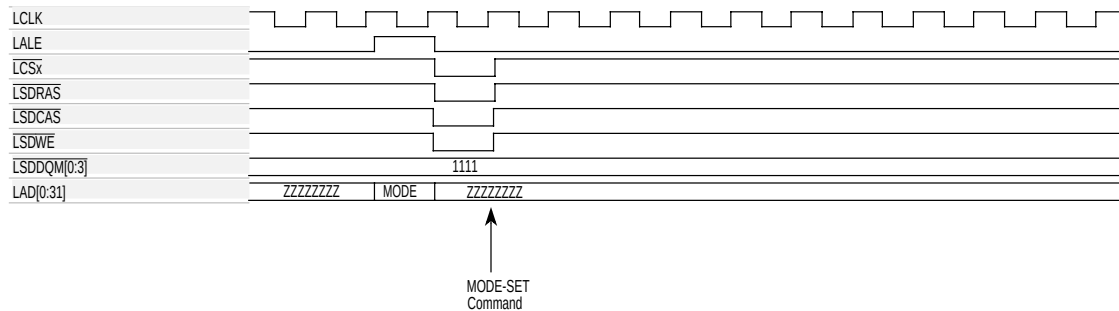


Figure 13-50. SDRAM MODE-SET Command

13.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires; this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

13.4.3.11.1 SDRAM Refresh Timing

The SDRAM memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the SDRAM machine's mode register (LSDMR[RFCR]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the

Functional Description

refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFCR].

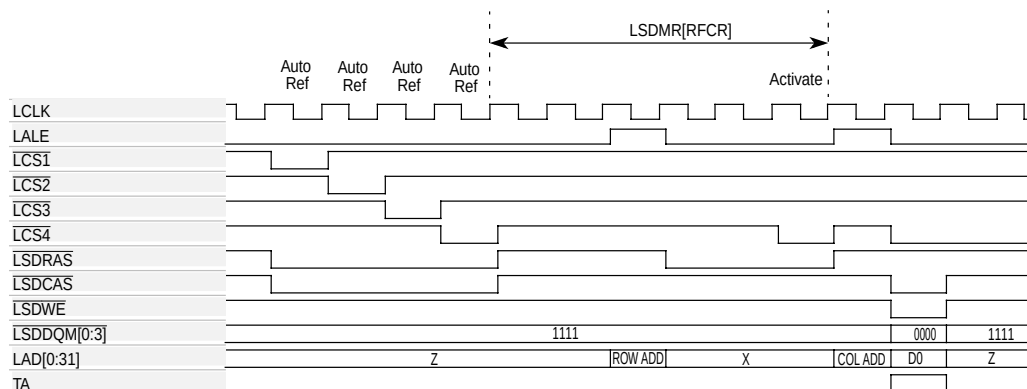


Figure 13-51. SDRAM Bank-Staggered Auto Refresh Timing

13.4.4 User-Programmable Machines (UPMs)

The user-programmable machines (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 ($\overline{LCSx}$, $\overline{LBS}[0:3]$ and $\overline{LGPL}[0:5]$) 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. Figure 13-52 shows the basic operation of each UPM.

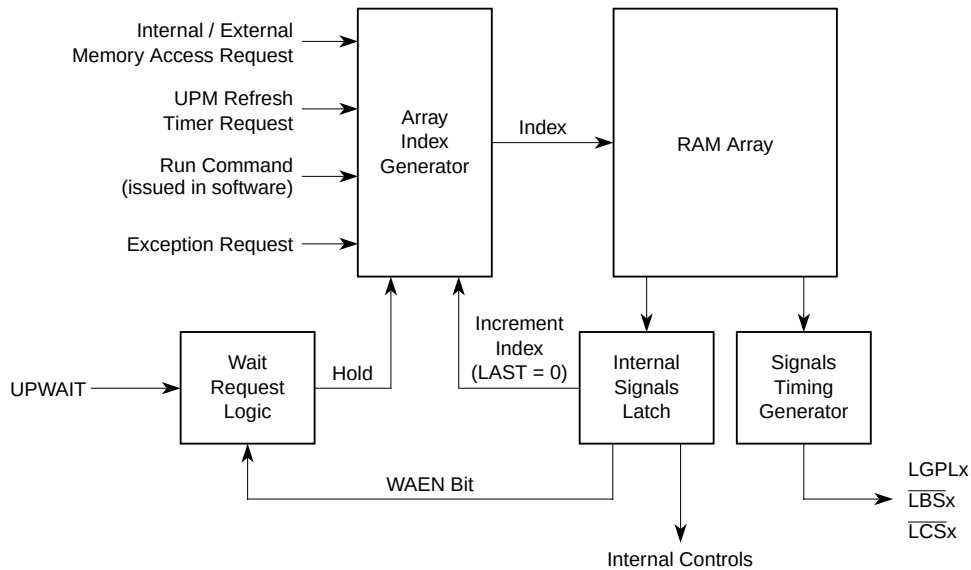


Figure 13-52. User-Programmable Machine Functional Block Diagram

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 generates an exception request

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 UPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

13.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each of the possible UPM requests. An internal device's request for a memory access initiates one of the following patterns (MxMR[OP] = 00):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

A UPM refresh timer request pattern initiates a refresh timer pattern (RTS).

An exception (caused by a bus monitor time-out error) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

Functional Description

Figure 13-53 and Table 13-29 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands ($MxMR[OP] = 11$), however, can initiate patterns starting at any of the 64 UPM RAM words.

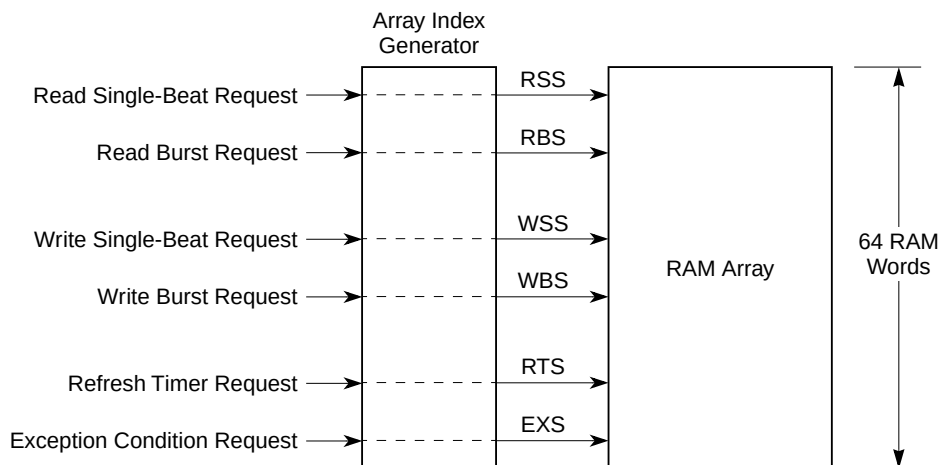


Figure 13-53. RAM Array Indexing

Table 13-29. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

13.4.4.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly four 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 ORx[BI].

13.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 13-54 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

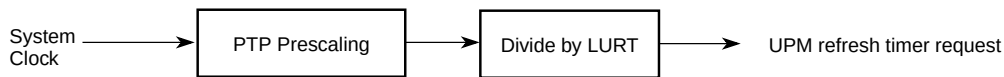


Figure 13-54. Memory Refresh Timer Request Block Diagram

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 refresh pattern if the RFEN bit of the corresponding UPM is set. 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.

Note that the UPM refresh timer request must not be used in a system with SDRAM refresh enabled. The system designer must choose to use either SDRAM refresh or UPM refresh. Using both may result in missing refresh periods to memory.

13.4.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then the RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The run command is issued by setting MxMR[OP] = 11 and accessing the UPMx memory region with any write transaction that hits the corresponding

UPM machine. The starting address in the RAM array for the pattern is set in the MxMR[MAD] field.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

13.4.4.1.4 Exception Requests

When the LBC 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 timing requirements of the device without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

13.4.4.2 Programming the UPMs

The UPM is a microsequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. These are the steps that should be followed to program the UPMs:

1. Set up BRx and ORx registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT and MAMR[RFEN] if refresh is required.
4. Program the machine mode register (MxMR).

Patterns are written to the RAM array by setting MxMR[OP] = 01 and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each time followed by a (dummy) write transaction to the relevant UPM assigned bank.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when MxMR[OP] = 10).

13.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. For LCRR[CLKDIV] = 4 or 8, each bit in the RAM word relating to $\overline{\text{LCSx}}$ and $\overline{\text{LBS}}$ timing specifies the value of the corresponding external signal at each quarter phase of the bus clock. If LCRR[CLKDIV] = 2, the external signal can change value only on each half phase of the bus clock. If the RAM word in this case (LCRR[CLKDIV] = 2) specifies a quarter phase signal change, the signal timing generator interprets this as a half cycle change.

The division of UPM bus cycles into phases is shown in Figure 13-55 and Figure 13-56. If $\text{LCRR}[\text{CLKDIV}] = 2$, the bus cycle comprises only two active phases, T1 and T3, which correspond with the first and second halves of the bus clock cycle, respectively. However, if $\text{LCRR}[\text{CLKDIV}] = 4$ or 8, four phases, T1–T4, define four quarters of the bus clock cycle. Because T2 and T4 are inactive when $\text{LCRR}[\text{CLKDIV}] = 2$, UPM ignores signal timing programmed for assertion in either of these phases in the case $\text{LCRR}[\text{CLKDIV}] = 2$.

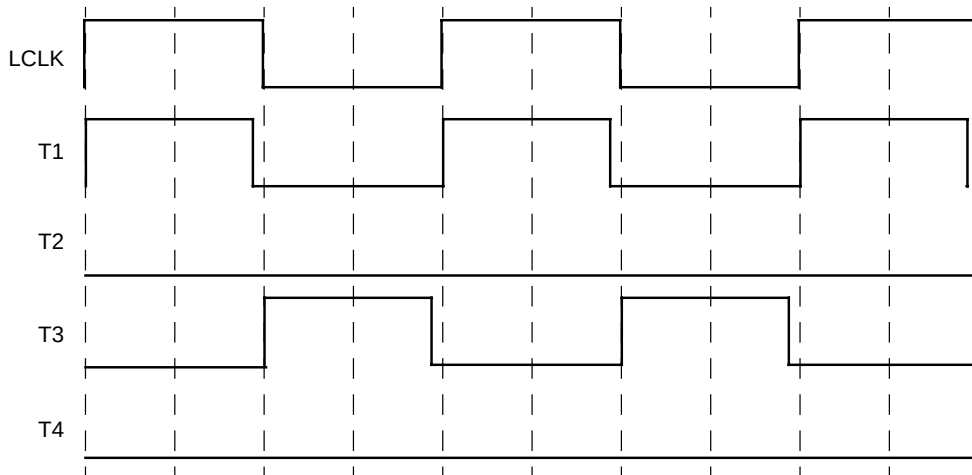


Figure 13-55. UPM Clock Scheme for $\text{LCRR}[\text{CLKDIV}] = 2$

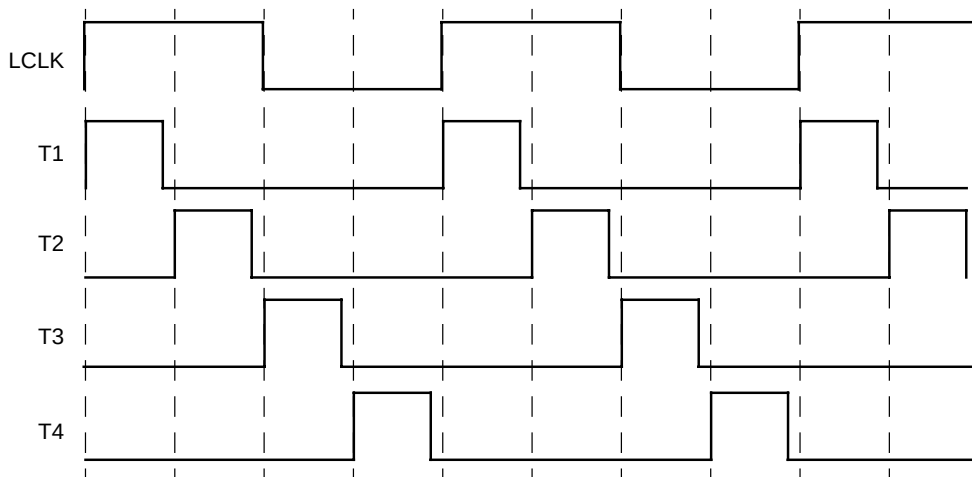


Figure 13-56. UPM Clock Scheme for $\text{LCRR}[\text{CLKDIV}] = 4$ or 8

13.4.4.4 The RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 13-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCSx}}$

Functional Description

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.

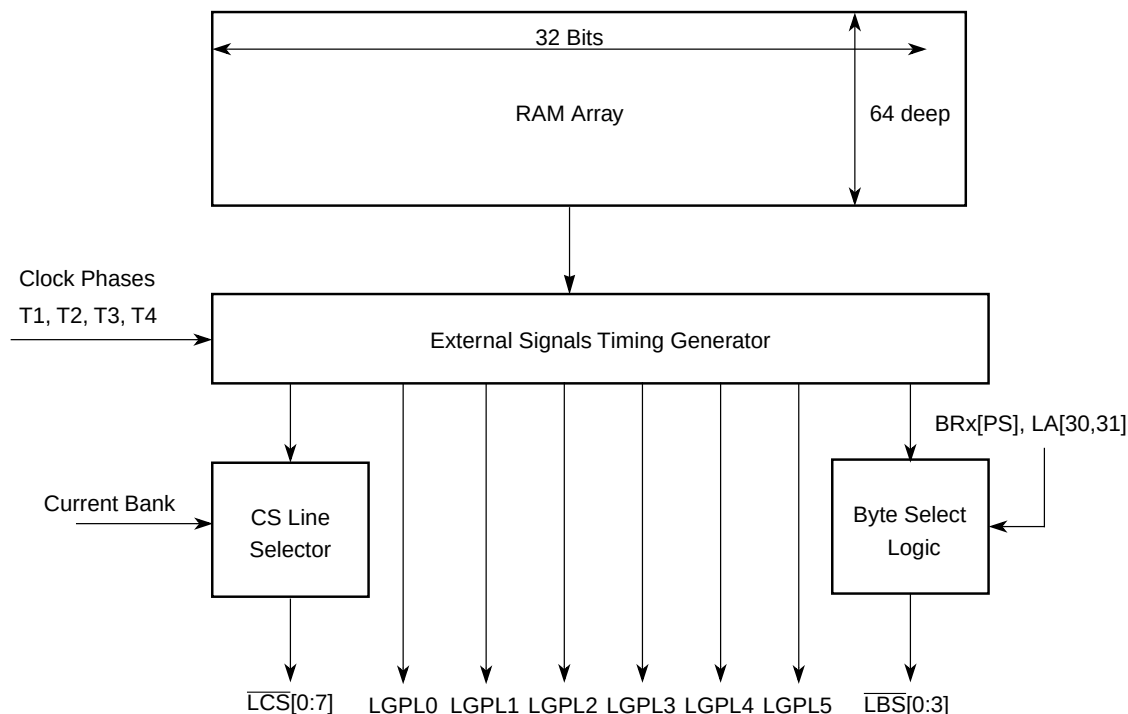


Figure 13-57. RAM Array and Signal Generation

13.4.4.4.1 RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 13-58 shows the RAM word fields. When $\text{LCRR}[\text{CLKDIV}] = 4$ or 8, the CST_x and BST_x bits determine the state of UPM signals $\overline{\text{LCS}}_x$ and $\overline{\text{LBS}}[0:3]$ at each quarter phase of the bus clock. When $\text{LCRR}[\text{CLKDIV}] = 2$, CST_2 and CST_4 are ignored and the external has the values defined by CST_1 and CST_3 but extended to half the clock cycle in duration. The same interpretation occurs for the BST_x bits when $\text{LCRR}[\text{CLKDIV}] = 2$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H	G1T1	G1T3	G2T1	G2T3	
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	G3T1	G3T3	G4T1 DLT3	G4T3 WAEN	G5T1	G5T3	REDO		LOOP	EXEN	AMX	NA	UTA	TODT	LAST	
W																
Reset	0000_0000_0000_0000															
Offset																

Figure 13-58. RAM Word Field Descriptions

Table 13-30 describes RAM word fields.

Table 13-30. RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCSx}}$ during bus clock quarter phase 1 if $\text{LCRR}[\text{CLKDIV}] = 4$ or 8. Defines the state (0 or 1) of LCSx during bus clock half phase 1 if $\text{LCRR}[\text{CLKDIV}] = 2$.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{\text{LCSx}}$ during bus clock quarter phase 2 if $\text{LCRR}[\text{CLKDIV}] = 4$ or 8. Ignored when $\text{LCRR}[\text{CLKDIV}] = 2$.
2	CST3	Chip select timing 3. Defines the state (0 or 1) of $\overline{\text{LCSx}}$ during bus clock quarter phase 3 if $\text{LCRR}[\text{CLKDIV}] = 4$ or 8. Defines the state (0 or 1) of LCSx during bus clock half phase 2 if $\text{LCRR}[\text{CLKDIV}] = 2$.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{\text{LCSx}}$ during bus clock quarter phase 4. if $\text{LCRR}[\text{CLKDIV}] = 4$ or 8. Ignored when $\text{LCRR}[\text{CLKDIV}] = 2$.
4	BST1	Byte select timing 1. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 1 ($\text{LCRR}[\text{CLKDIV}] = 4$ or 8) or bus clock half phase 1 ($\text{LCRR}[\text{CLKDIV}] = 2$), in conjunction with $\text{BRx}[\text{PS}]$ and $\text{LA}[30-31]$.
5	BST2	Byte select timing 2. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 2 ($\text{LCRR}[\text{CLKDIV}] = 4$ or 8), in conjunction with $\text{BRx}[\text{PS}]$ and $\text{LA}[30-31]$. Ignored when $\text{LCRR}[\text{CLKDIV}] = 2$.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3 ($\text{LCRR}[\text{CLKDIV}] = 4$ or 8) or bus clock half phase 2 ($\text{LCRR}[\text{CLKDIV}] = 2$), in conjunction with $\text{BRx}[\text{PS}]$ and $\text{LA}[30-31]$.
7	BST4	Byte select timing 4. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 4 ($\text{LCRR}[\text{CLKDIV}] = 4$ or 8), in conjunction with $\text{BRx}[\text{PS}]$ and $\text{LA}[30-31]$. Ignored when $\text{LCRR}[\text{CLKDIV}] = 2$.
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 $\text{MxMR}[\text{GOCL}]$ 01 reserved 10 0 11 1

Table 13-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
10–11	G0H	General purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by MxMR[G0CL] 01 reserved 10 0 11 1
12	G1T1	General purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).
13	G1T3	General purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
15	G2T3	General purpose line 2 timing 3. Defines the state (0 or 1) of LGPL2 during bus clock quarter phases 3 and 4 (second half phase).
16	G3T1	General purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DLT3	General purpose line 4 timing 1/delay time 3. The function of this bit is determined by MxMR[GPL4]. If MxMR[GPL4] = 0 and LGPL4/UPWAIT 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 MxMR[GPL4] = 1 and LGPL4/UPWAIT functions as an input (UPWAIT), 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. The function of this bit is determined by MxMR[GPL4]. If MxMR[GPL4] = 0 and LGPL4/UPWAIT 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 MxMR[GPL4] = 1 and LGPL4/UPWAIT functions as an input (UPWAIT), G4T3/WAEN is used to enable the wait mechanism: 0 UPWAIT detection is disabled. 1 UPWAIT is enabled. If UPWAIT is detected as being asserted, a freeze in the external signals logical values occurs until UPWAIT 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

Table 13-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
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 MxMR. 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.
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 deassert 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[0:31] during a LALE phase. Any change in the AMX field initiates a new LALE (address) phase. 00 LAD[0:31] is the non-multiplexed address. For example, column address. 01 Reserved 10 LAD[0:31] is the address multiplexed according to MxMR[AMx]. For example, row address. 11 LAD[0:31] is the contents of MAR. Used, for example, to initialize a mode. Note that 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 BRx[PS], the increment value of LA[27:31] 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 MxMR[DSx]. 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, the execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in MxMR[DSx]. 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.

The following sections provide additional information about some of the RAM word fields.

13.4.4.4.2 Chip-Select Signal Timing (CSTn)

If BRx[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCSx}}$ signal for that bank with timing as specified in the UPM RAM word CSTn fields. The selected UPM affects only the assertion and negation of the appropriate $\overline{\text{LCSx}}$ signal.

The state of the selected $\overline{\text{LCSx}}$ signal of the corresponding bank depends on the value of each CSTn bit. Figure 13-59 shows how UPMs control $\overline{\text{LCSx}}$ signals.

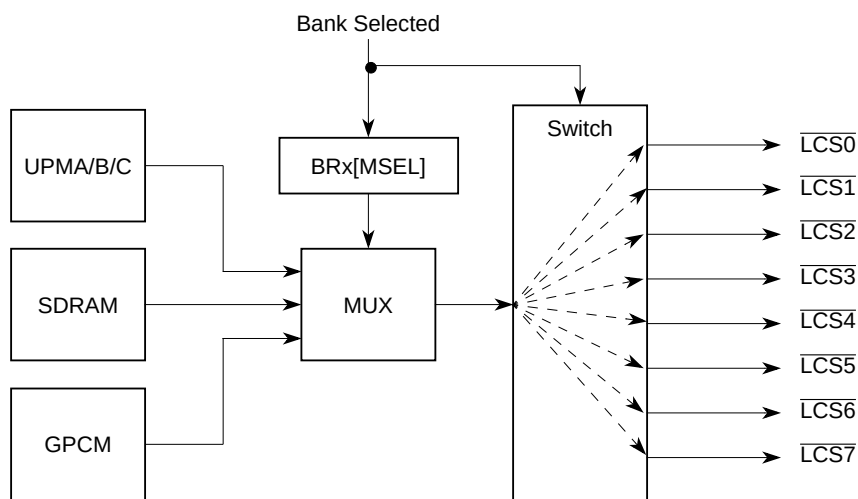


Figure 13-59. $\overline{\text{LCSx}}$ Signal Selection

13.4.4.4.3 Byte-Select Signal Timing (BSTn)

If BRx[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, the $\overline{\text{LBS}}[0:3]$ signals are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed. Figure 13-60 shows how UPMs control $\overline{\text{LBS}}[0:3]$ signals.

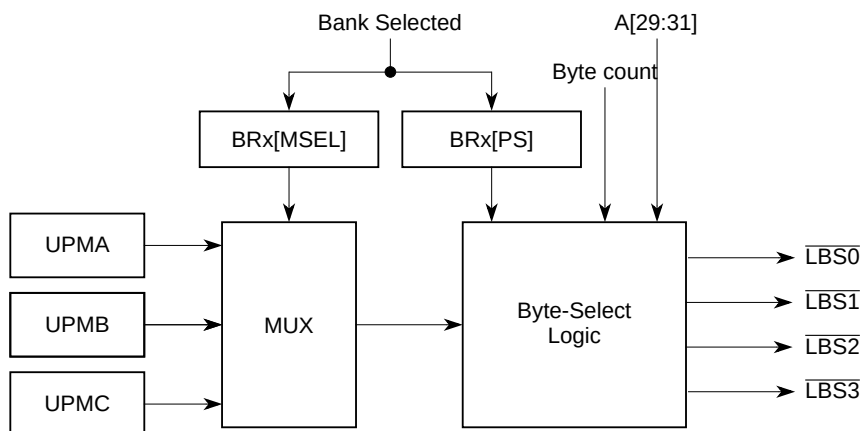


Figure 13-60. $\overline{\text{LBS}}$ Signal Selection

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] contains valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ indicates that LAD[24:31] contains 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, $\overline{\text{LBS}}[0:3]$ signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

13.4.4.4.4 General-Purpose Signals (GxTx, GOx)

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 LGPLx lines.

GPL0 can be controlled by an address line specified in MxMR[G0CLx]. To use this feature, G0H and G0L should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

13.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 13-31. 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 in the following case:

- LAST and LOOP must not be set together.

Table 13-31. MxMR Loop Field Use

Request Serviced	Loop Field
Read single-beat cycle	RLFx
Read burst cycle	RLFx
Write single-beat cycle	WLFx
Write burst cycle	WLFx
Refresh timer expired	TLFx
RUN command	RLFx

13.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

13.4.4.4.7 Address Multiplexing (AMX)

The address lines can be controlled by the pattern the user provides in the UPM. The address multiplex bits can choose between driving the transaction address, driving it according to the multiplexing specified by the MxMR[AMx] field, or driving the MAR contents on the address signals. In all cases, the LA[27:31] signals of the LBC are driven by the 5 lsb's of the address selected by AMX, regardless of whether the 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 shows how MxMR[AMx] settings affect address multiplexing when the RAM word AMX = 10. The 16 most significant bits of the LAD[0:31] bus during an address phase are driven with zero in the AMX = 10 case.

Table 13-32. UPM Address Multiplexing

AMx	LAD[0:31] as Address Signals	A0–A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31
000	Signal driven on external pin when address multiplexing is enabled— RAM word AMX = 10	0	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23
001		0	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22
010		0	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
011		0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20
100		0	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19
101		0	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18

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[0:31] bus with the assertion of LALE for the number of cycles set for LALE in the ORx and LCRR registers. The LGPL[0:5] signals maintain the value specified in the RAM word during the LALE phase.

13.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by LBC), the value of the DLT3 bit in the same RAM word, in conjunction with MxMR[GPLx4DIS], determines when the data input is sampled by LBC as follows:

- If MxMR[GPLx4DIS] = 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 data is sampled by LBC on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.
- If MxMR[GPLx4DIS] = 0 (G4T4/DLT3 functions as G4T4), or if MxMR[GPLx4DIS] = 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 13-61 shows how data sampling is controlled by the UPM.

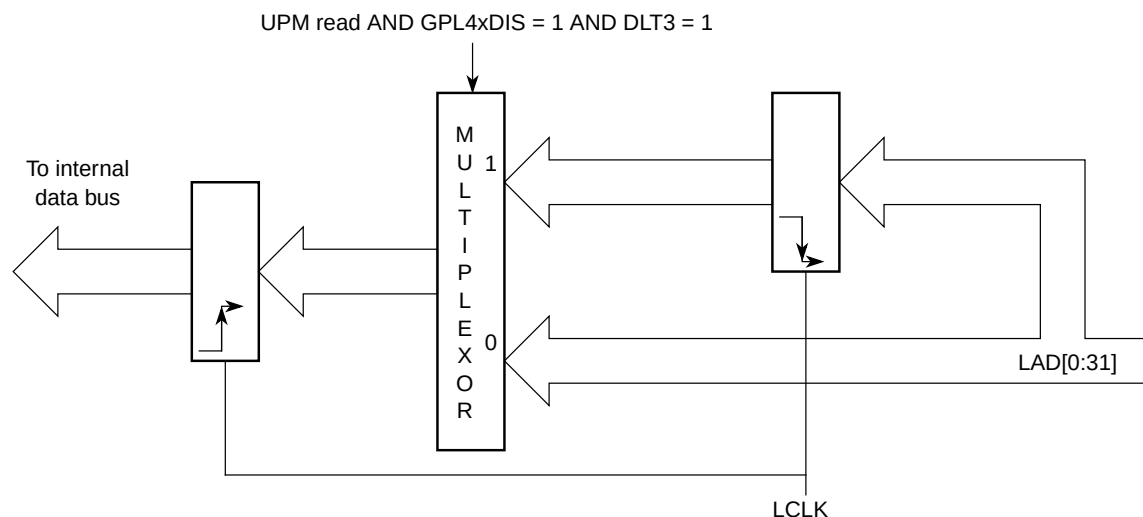


Figure 13-61. UPM Read Access Data Sampling

13.4.4.4.9 $\overline{\text{LGPL}}[0:5]$ 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.

13.4.4.4.10 The 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 UPWAIT 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 UPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If UPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until UPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. When UPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 13-62 shows how the WAEN bit in the word read by the UPM and the UPWAIT signal are used to hold the UPM in a particular state until UPWAIT is negated. As the example shows, the $\overline{\text{LCSx}}$ and $\overline{\text{LGPL1}}$ states and the WAEN value are frozen until UPWAIT is recognized as negated. WAEN is typically set before the line that contain 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 UPM freezes or not

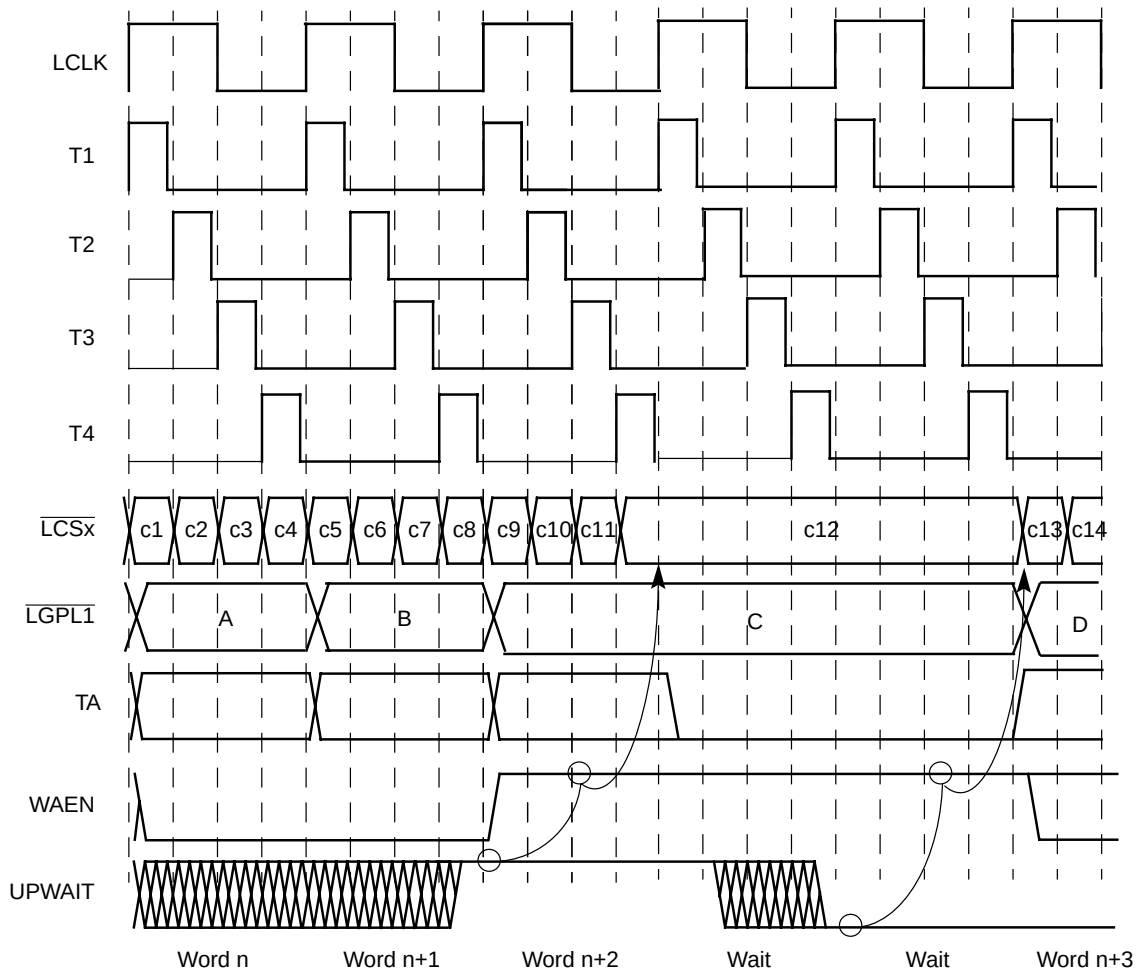


Figure 13-62. Effect of UPWAIT Signal

13.4.4.5 Synchronous Sampling of UPWAIT for Early Transfer Acknowledge

If UPWAIT 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.

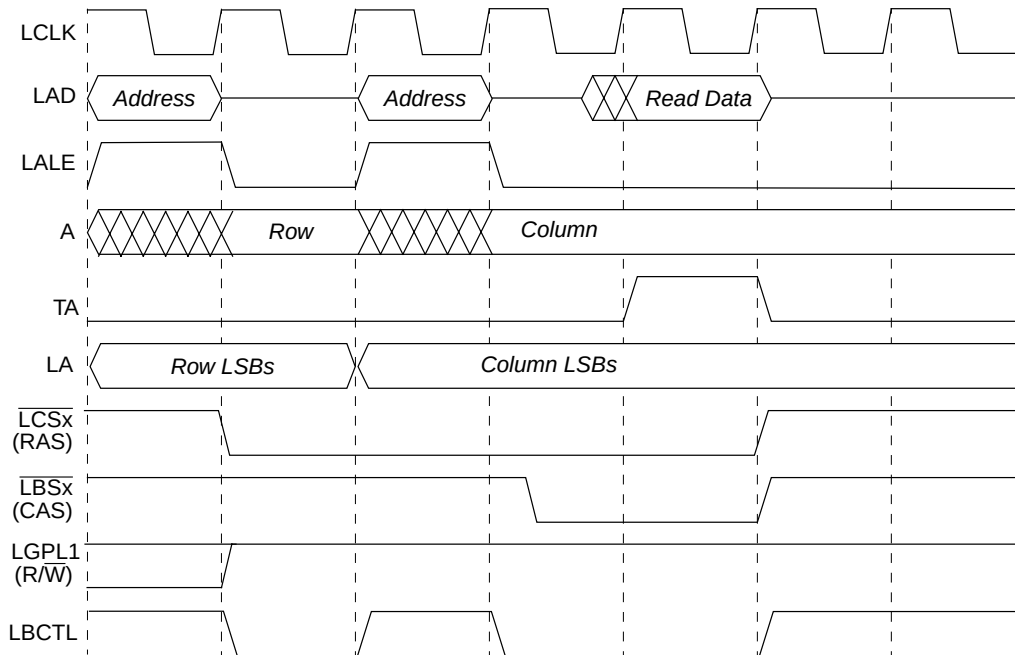
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat UPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples UPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because UPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of UPWAIT to affect a data transfer.

13.4.4.6 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 ORx[TRLX] and ORx[EHTR]. The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the ORx register in addition to any existing bus turn around cycle.

13.4.4.7 Memory System Interface Example Using UPM

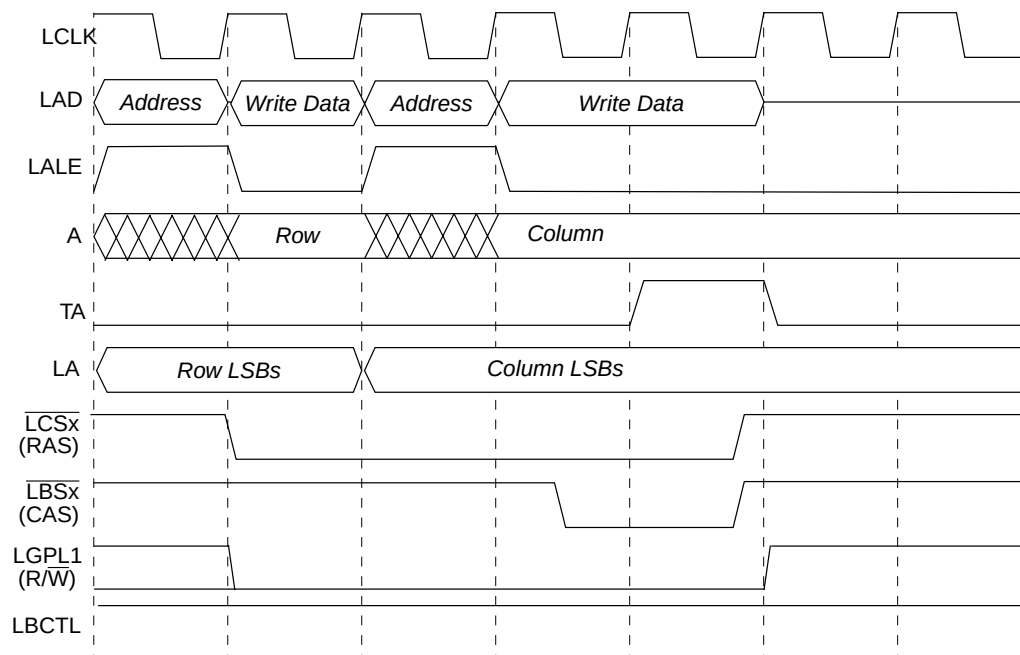
Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section describes timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with LCRR[CLKDIV] = 4 or 8. The examples shown are for illustrative purposes only, and may not represent the timing necessary for any specific device used with the LBC. In the examples, LGPL1 is programmed to drive $\overline{R/W}$ of the DRAM, although any LGPLx signal may be used for this purpose.



cst1	Bit 0	0	LALE pause (due to change in AMX)	0	0
cst2	Bit 1	0		0	0
cst3	Bit 2	0		0	0
cst4	Bit 3	0		0	0
bst1	Bit 4	1		1	0
bst2	Bit 5	1		0	0
bst3	Bit 6	1		0	0
bst4	Bit 7	1		0	0
g0l0	Bit 8				
g0l1	Bit 9				
g0h0	Bit 10				
g0h1	Bit 11				
g1t1	Bit 12	1		1	1
g1t3	Bit 13	1		1	1
g2t1	Bit 14				
g2t3	Bit 15				
g3t1	Bit 16				
g3t3	Bit 17				
g4t1	Bit 18				
g4t3	Bit 19				
g5t1	Bit 20				
g5t3	Bit 21				
redo[0]	Bit 22				
redo[1]	Bit 23				
loop	Bit 24	0		0	0
exen	Bit 25	0		0	0
amx0	Bit 26	1		0	0
amx1	Bit 27	0		0	0
na	Bit 28	0		0	0
uta	Bit 29	0		0	1
todt	Bit 30	0		0	1
last	Bit 31	0		0	1
	RSS	RSS	RSS+1	RSS+1	RSS+2

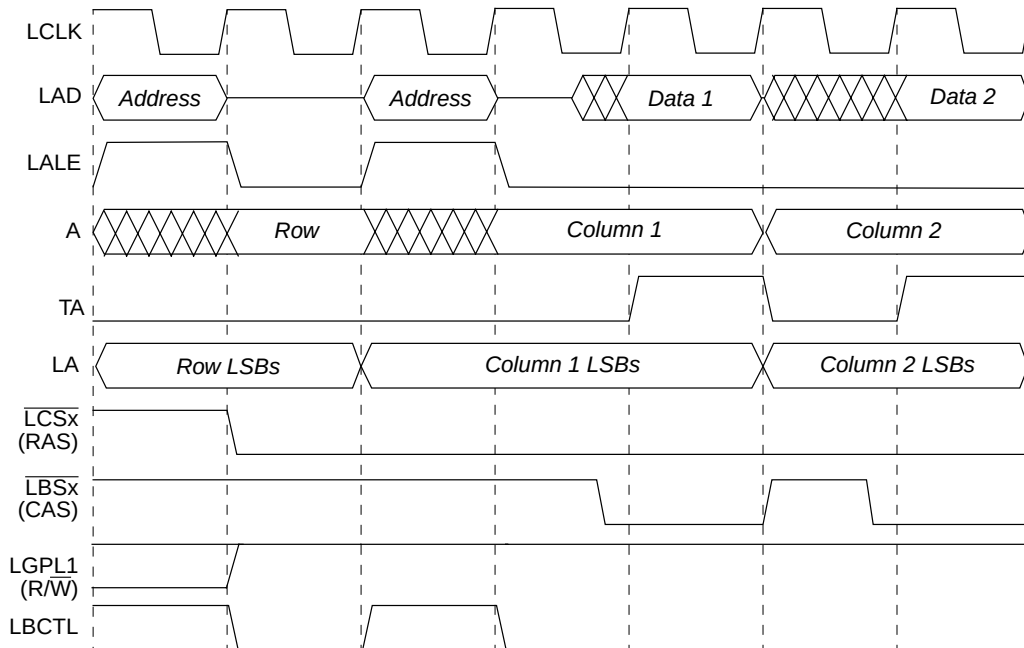
Figure 13-63. Single-Beat Read Access to FPM DRAM

Functional Description



cst1	Bit 0	0	LALE pause (due to change in AMX)	0	0
cst2	Bit 1	0		0	0
cst3	Bit 2	0		0	0
cst4	Bit 3	0		0	1
bst1	Bit 4	1		1	0
bst2	Bit 5	1		1	0
bst3	Bit 6	1		0	0
bst4	Bit 7	1		0	1
g0l0	Bit 8				
g0l1	Bit 9				
g0h0	Bit 10				
g0h1	Bit 11				
g1t1	Bit 12	0		0	0
g1t3	Bit 13	0		0	0
g2t1	Bit 14				
g2t3	Bit 15				
g3t1	Bit 16				
g3t3	Bit 17				
g4t1	Bit 18				
g4t3	Bit 19				
g5t1	Bit 20				
g5t3	Bit 21				
redo[0]	Bit 22				
redo[1]	Bit 23				
loop	Bit 24	0		0	0
exen	Bit 25	0		0	0
amx0	Bit 26	1		0	0
amx1	Bit 27	0		0	0
na	Bit 28	0		0	0
uta	Bit 29	0		0	1
todt	Bit 30	0		0	1
last	Bit 31	0		0	1
	WSS	WSS	WSS+1	WSS+1	WSS+2

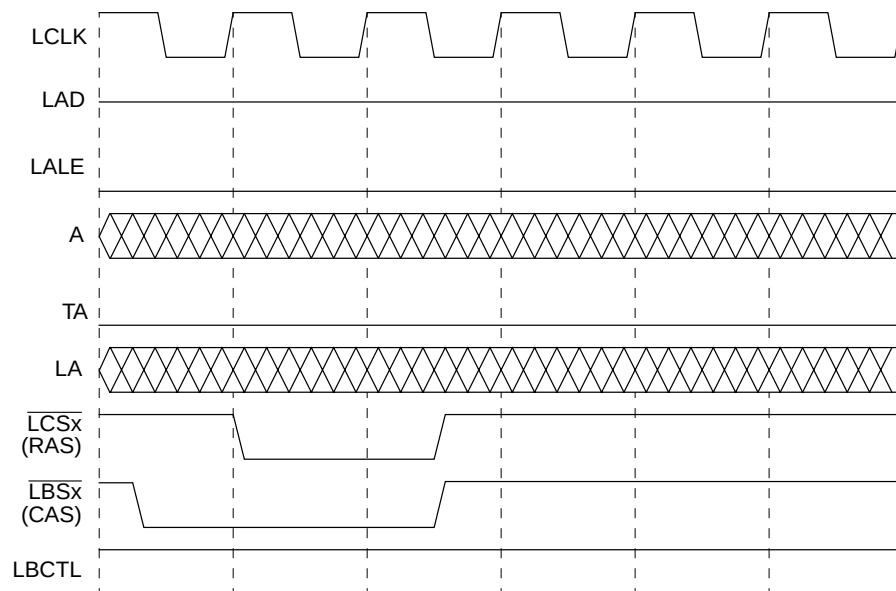
Figure 13-64. Single-Beat Write Access to FPM DRAM



cst1	Bit 0	0	LALE pause (due to change in AMX)	0	0	1
cst2	Bit 1	0		0	0	1
cst3	Bit 2	0		0	0	1
cst4	Bit 3	0		0	0	1
bst1	Bit 4	1		1	0	1
bst2	Bit 5	1		1	0	1
bst3	Bit 6	1		1	0	1
bst4	Bit 7	1		0	0	1
g0l0	Bit 8					
g0l1	Bit 9					
g0h0	Bit 10					
g0h1	Bit 11					
g1t1	Bit 12	1		1	1	1
g1t3	Bit 13	1		1	1	1
g2t1	Bit 14					
g2t3	Bit 15					
g3t1	Bit 16					
g3t3	Bit 17					
g4t1	Bit 18					
g4t3	Bit 19					
g5t1	Bit 20					
g5t3	Bit 21					
redo[0]	Bit 22					
redo[1]	Bit 23					
loop	Bit 24	0		1	1	0
exen	Bit 25	0		0	1	0
amx0	Bit 26	1		0	0	0
amx1	Bit 27	0		0	0	0
na	Bit 28	0		0	1	0
uta	Bit 29	0		0	1	0
todt	Bit 30	0		0	0	1
last	Bit 31	0		0	0	1
	RBS	RBS		RBS+1	RBS+2	RBS+3

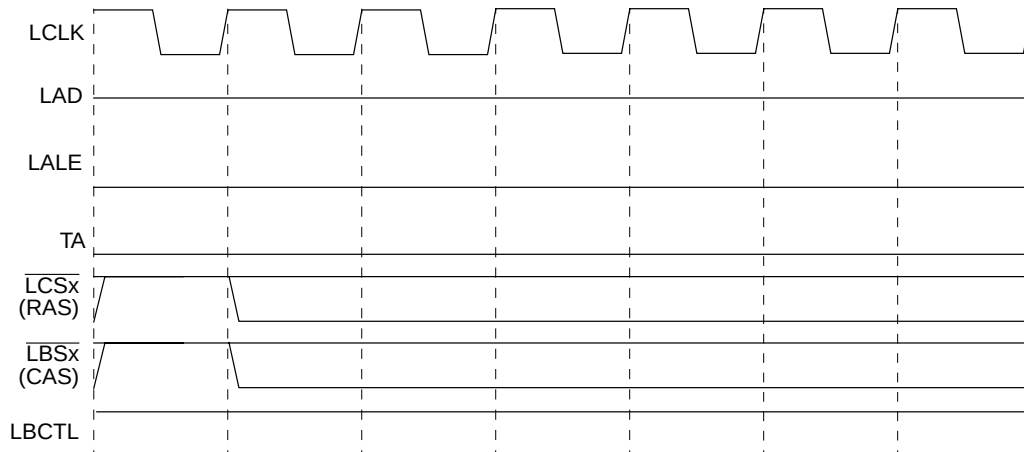
Figure 13-65. Burst Read Access to FPM DRAM Using LOOP (2 Beats Shown)

Functional Description



cst1	1	0	0	Bit 0
cst2	1	0	0	Bit 1
cst3	1	0	1	Bit 2
cst4	1	0	1	Bit 3
bst1	1	0	0	Bit 4
bst2	0	0	0	Bit 5
bst3	0	0	1	Bit 6
bst4	0	0	1	Bit 7
g0l0				Bit 8
g0l1				Bit 9
g0h0				Bit 10
g0h1				Bit 11
g1t1				Bit 12
g1t3				Bit 13
g2t1				Bit 14
g2t3				Bit 15
g3t1				Bit 16
g3t3				Bit 17
g4t1				Bit 18
g4t3				Bit 19
g5t1				Bit 20
g5t3				Bit 21
redo[0]				Bit 22
redo[1]				Bit 23
loop	0	0	0	Bit 24
exen	0	0	0	Bit 25
amx0	0	0	0	Bit 26
amx1	0	0	0	Bit 27
na	0	0	0	Bit 28
uta	0	0	0	Bit 29
todt	0	0	1	Bit 30
last	0	0	1	Bit 31
	PTS	PTS+1	PTS+2	

Figure 13-66. Refresh Cycle (CBR) to FPM DRAM



cst1	1	Bit 0
cst2	1	Bit 1
cst3	1	Bit 2
cst4	1	Bit 3
bst1	1	Bit 4
bst2	1	Bit 5
bst3	1	Bit 6
bst4	1	Bit 7
g0l0		Bit 8
g0l1		Bit 9
g0h0		Bit 10
g0h1		Bit 11
g1t1		Bit 12
g1t3		Bit 13
g2t1		Bit 14
g2t3		Bit 15
g3t1		Bit 16
g3t3		Bit 17
g4t1		Bit 18
g4t3		Bit 19
g5t1		Bit 20
g5t3		Bit 21
redo[0]		Bit 22
redo[1]		Bit 23
loop	0	Bit 24
exen	0	Bit 25
amx0	0	Bit 26
amx1	0	Bit 27
na	0	Bit 28
uta	0	Bit 29
todt	1	Bit 30
last	1	Bit 31
EXS		

Figure 13-67. Exception Cycle

13.5 Initialization/Application Information

13.5.1 Interfacing to Peripherals

13.5.1.1 Multiplexed Address/Data Bus and Unmultiplexed Address Signals

To save pins on the local bus, address and data are multiplexed onto the same 32 bit bus. An external latch is needed to unmultiplex and reconstruct the original address. No external intelligence is needed, because the LALE signal provides the correct timing to control a standard logic latch. The LAD pins can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8,16, and 32 bit. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven unmultiplexed. For 8-bit devices, LA[30:31] should be used and for 16 bit devices, LA[30] should be used. 32-bit devices use neither of these signals.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[27:29] are the burst addresses within a natural 32-byte burst. To minimize the amount of address phases needed on the local bus and to optimize the throughput, those signals are driven separately and should be used whenever a device requires the 5 least significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

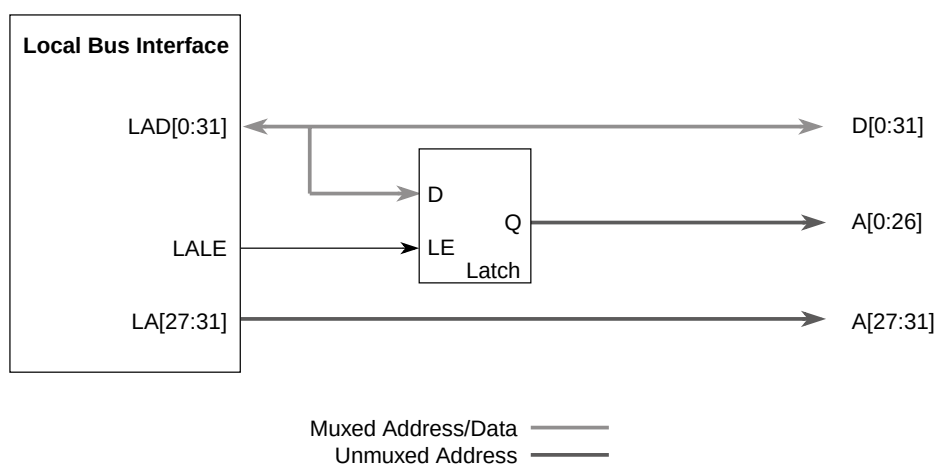


Figure 13-68. Multiplexed Address/Data Bus

13.5.1.2 Peripheral Hierarchy on the Local Bus

In order to achieve high bus speed interfaces for synchronous SRAMs or SDRAMs a hierarchy of the memories/peripherals connected to the local bus is suggested. Figure 13-69 shows an example of such a hierarchy.

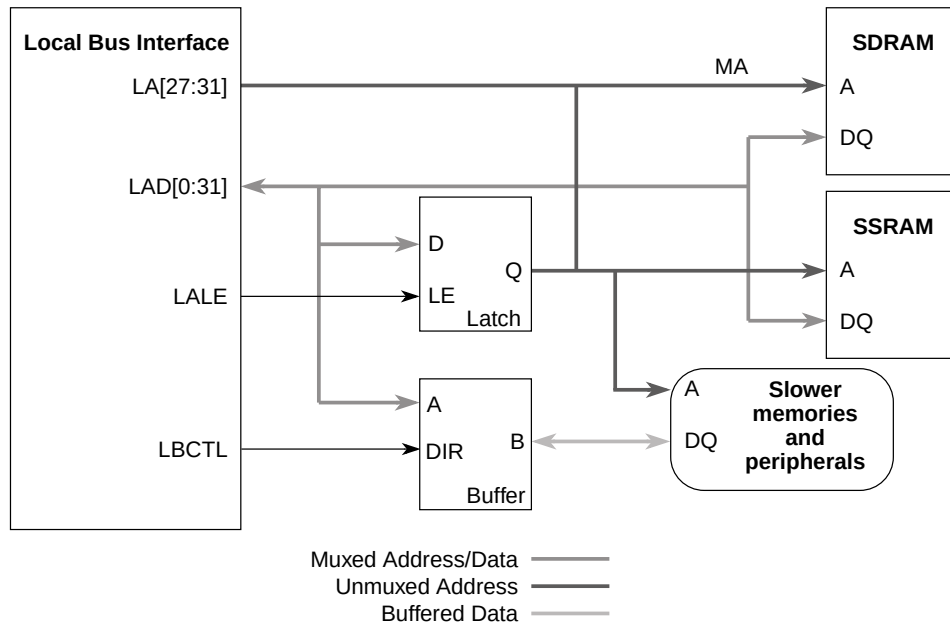


Figure 13-69. Local Bus Peripheral Hierarchy

The multiplexed address/data bus sees the capacitive loading of the data pins of the fast SDRAMs or synchronous SRAMs plus one load for an address latch plus one load for a buffer to the slow memories. The loadings of all other memories and peripherals are hidden behind the buffer and the latch. The system designer needs to investigate the loading scenario and ensure that I/O timings can be met with the loading determined by the connected components.

13.5.1.3 Peripheral Hierarchy on the Local Bus for Very 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 local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used. Figure 13-70 shows an example of such a hierarchy. This section is only a guideline and the board designer must simulate the electric characteristics of his scenario to determine the maximum operating frequency.

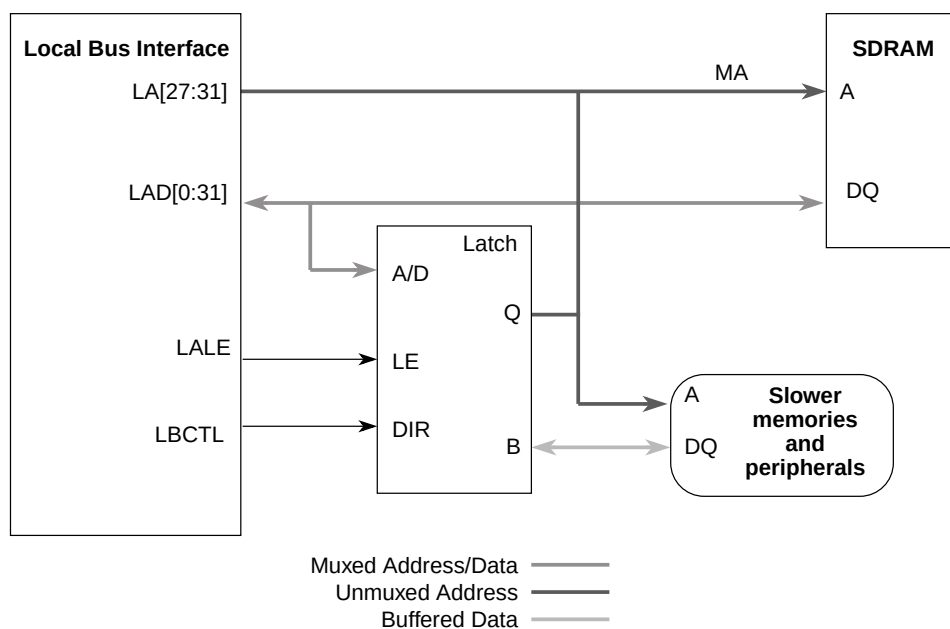


Figure 13-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

13.5.1.4 GPCM Timings

In case a system contains a memory hierarchy with high speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (i.e. FLASH EPROM, peripherals,...) the GPCM controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations.

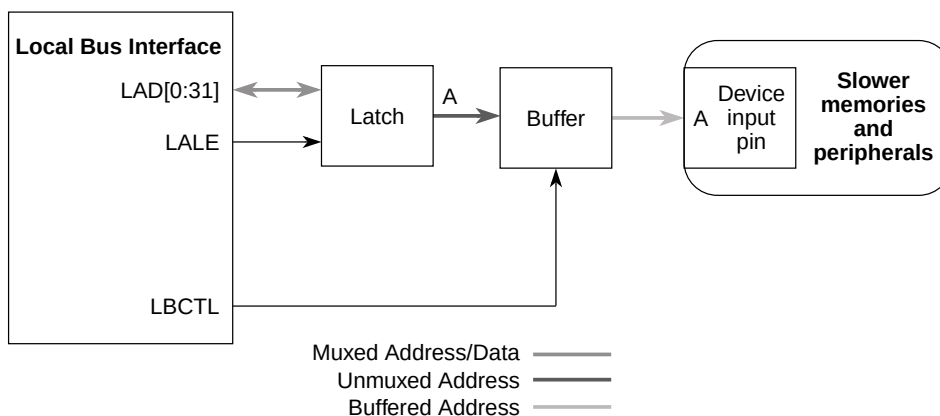


Figure 13-71. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory 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 166 MHz bus frequency $\overline{LCS}$ should arrive in 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.

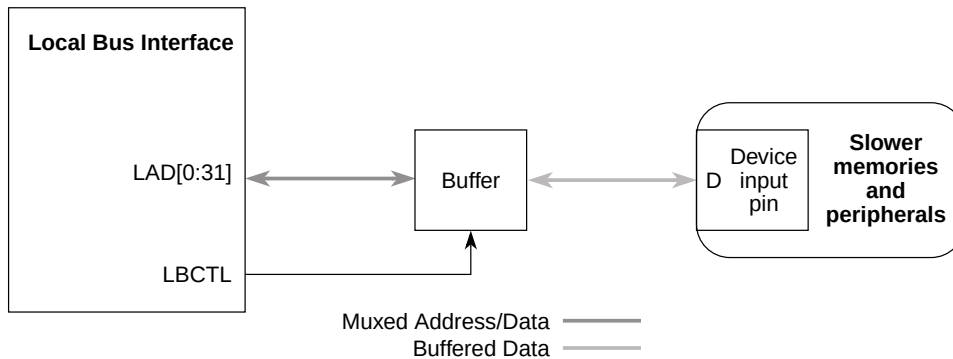


Figure 13-72. GPCM Data timings

13.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

Because the bus does not change direction, these cases do not require special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

13.5.2.1 Address Phase after Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC 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 LBC then inserts a bus turnaround cycle to avoid contention. The

external device has now already placed its data signals in high impedance and no bus contention will occur.

13.5.2.2 Read Data Phase after Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving bus transceivers in the same direction as during a write. After the end of the address phase LBCTL goes low and turns around the direction of the bus transceiver. The LBC places the LAD signals in high impedance after its $t_{dis}(LB)$. The LBCTL will have its new state after $t_{en}(LB)$ and, because this is an asynchronous input, the transceiver starts to drive those signals after its $t_{en}(\text{transceiver})$ time. The system designer has to ensure, that $[t_{en}(LB) + t_{en}(\text{transceiver})]$ is larger than $t_{dis}(LB)$ to avoid bus contention.

13.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle will have a new address phase in any case, this basically is the same case as an address phase after a previous read.

13.5.2.4 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern. The LBC 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.

Bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

13.5.3 Interface to Different Port-Size Devices

The LBC can support 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 D[0:31], a 16-bit port must reside on D[0:15], and an 8-bit port must reside on D[0:7]. The local bus always tries to transfer the maximum amount of data on all bus cycles.

Figure 13-73 shows the device connections on the data bus. Table 13-33 lists the bytes required on the data bus for read cycles.

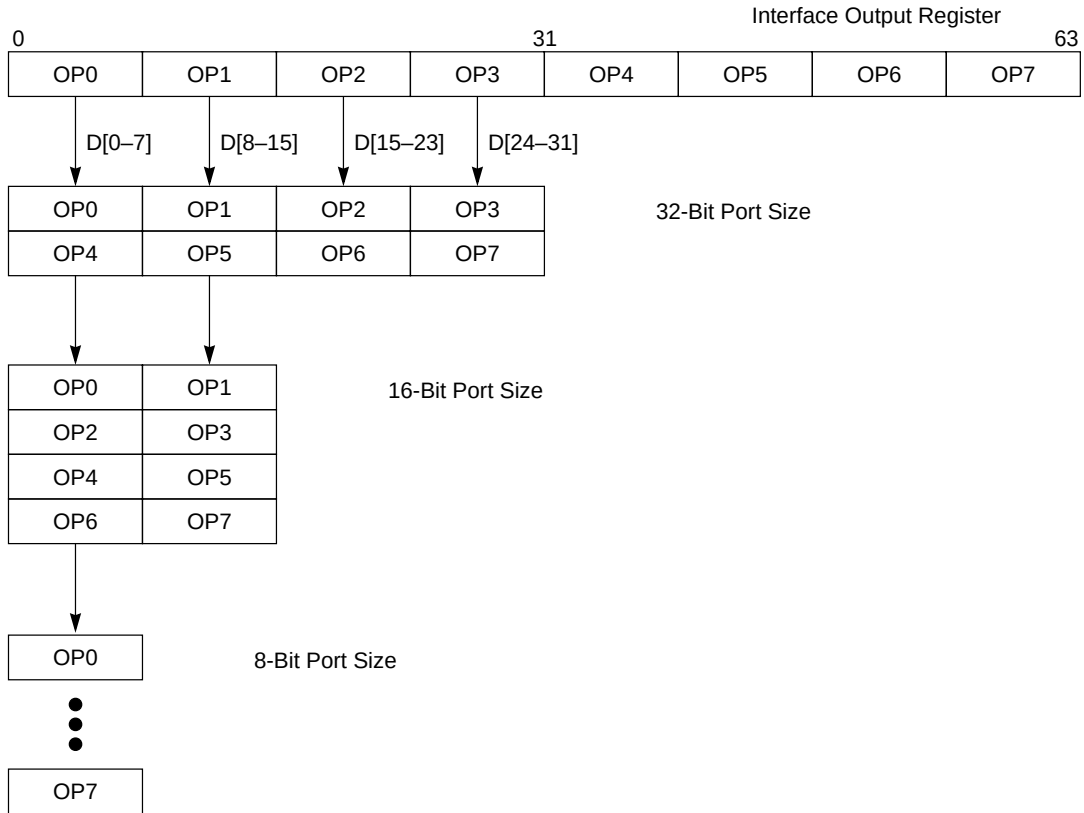


Figure 13-73. Interface to Different Port-Size Devices

Table 13-33. Data Bus Requirements For Read Cycle

Transfer Size	Address State ¹ A[29-31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0-7	8-15	16-23	24-31	0-7	8-15	0-7
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 13-33. Data Bus Requirements For Read Cycle (continued)

Transfer Size	Address State ¹ A[29–31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	0–7	8–15	0–7
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 OPn: These lanes are read or written during that bus transaction. OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.

³ — Denotes a byte not driven during that write cycle.

13.5.4 Interfacing to SDRAM

13.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects ($\overline{\text{LCSx}}$) can be programmed to support multiple SDRAM devices. Note that no limitation exists on the number of chip selects that can be programmed for SDRAM. This means that $\overline{\text{LCS}}[1:7]$ can be programmed to support SDRAM, assuming $\overline{\text{LCS0}}$ is reserved for the general-purpose chip-select machine (GPCM) to connect to Flash memory.

If multiple chip selects are configured to support SDRAM on the local bus, each SDRAM device should have the same port size and timing parameters. This means that all option registers (ORn) for the SDRAM chip selects should be programmed exactly the same.

NOTE

Although in principal it is possible to mix different port sizes and timing parameters, combinations are limited and this operation is not recommended.

All the chip selects share the same local bus SDRAM mode register (LSDMR) for initialization along with the local bus-assigned SDRAM refresh timer register (LSRT) and the memory refresh timer prescaler register (MPTPR) for refresh.

For refresh, the memory controller supplies auto refresh to SDRAM according to the time interval specified in LSRT and MPTPR as follows:

$$\text{Refresh Period} = \frac{(\text{LSRT} + 1) \times (\text{MPTPR}[\text{PTP}] + 1)}{\text{System Frequency}}$$

This represents the time period required between refreshes. When the refresh timer expires, the memory controller issues a CBR to each chip select. Each CBR is separated by one clock. A refresh timing diagram for multiple chip selects is shown in Figure 13-51 “SDRAM Bank-Staggered Auto Refresh Timing.”

During a memory transaction dispatched to the local bus, the memory controller compares the memory address with the address information of each chip select (programmed with BR_n and OR_n). If the comparison matches a chip select that is controlled by SDRAM, the memory controller requests service to the local bus SDRAM machine, depending on the information in BR_n. Although multiple chip selects may be programmed for SDRAM, only one chip select is active at any given time; thus, multiple chip selects can share the same SDRAM machine.

13.5.4.2 Maximum Amount of SDRAM Supported

Table 13-34 summarizes information based on SDRAM data sheets supplied by Micron.

Table 13-34. Micron SDRAM Devices

SDRAM Device	64 Mbit				128 Mbit				256 Mbit				512 Mbit			
I/O Port	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
Bank	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Row	12	12	12	11	12	12	12	12	13	13	13	13	13	13	13	TBD
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	TBD

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires four SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, one chip select provides 128-Mbit/device x 4 devices = 64 Mbytes. If 4 chip selects are programmed for SDRAM use, the result is 64 Mbytes x 4 = 256 Mbyte. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbyte. Consequently 512-Mbit devices allow for 1 Gbyte.

Although there is no technical difficulty in supporting multiple chip select configurations, in practice, the user may want to maximize the amount of SDRAM assigned to each chip select to minimize cost.

13.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

13.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select Mux

LSDMR[BSMA] is used to multiplex the bank select address. The BSMA field and corresponding multiplexed address are shown below:

000 LA12–LA13

001 LA13–LA14

...

111 LA19–LA20

Note that LA12 is the latched value of LAD12.

The highest address pins that the bank selects can be multiplexed with are LA[12:13], which limits the pins for the row address to LA[14:31]. For a 32-bit port, the maximum width of the local bus, LA[30:31] are not connected, and the maximum row is LA[14:29]. The local bus SDRAM machine supports 15 rows, which is sufficient for all devices.

13.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address pins to leave room for future upgrades. For example, a user could multiplex the bank select signals to LA[14:15], leaving LA16 to connect to the address pin for a larger memory size.

This allows the system designer to design one board that can be used with a current generation of SDRAM devices and upgraded to the next generation without requiring a new board layout.

13.5.4.3.3 SDRAM of 128 MBytes

Figure 13-74 shows an SDRAM of 128 Mbytes. Note that all circuit diagrams are principal connection diagrams and do not show any means of signal integrity.

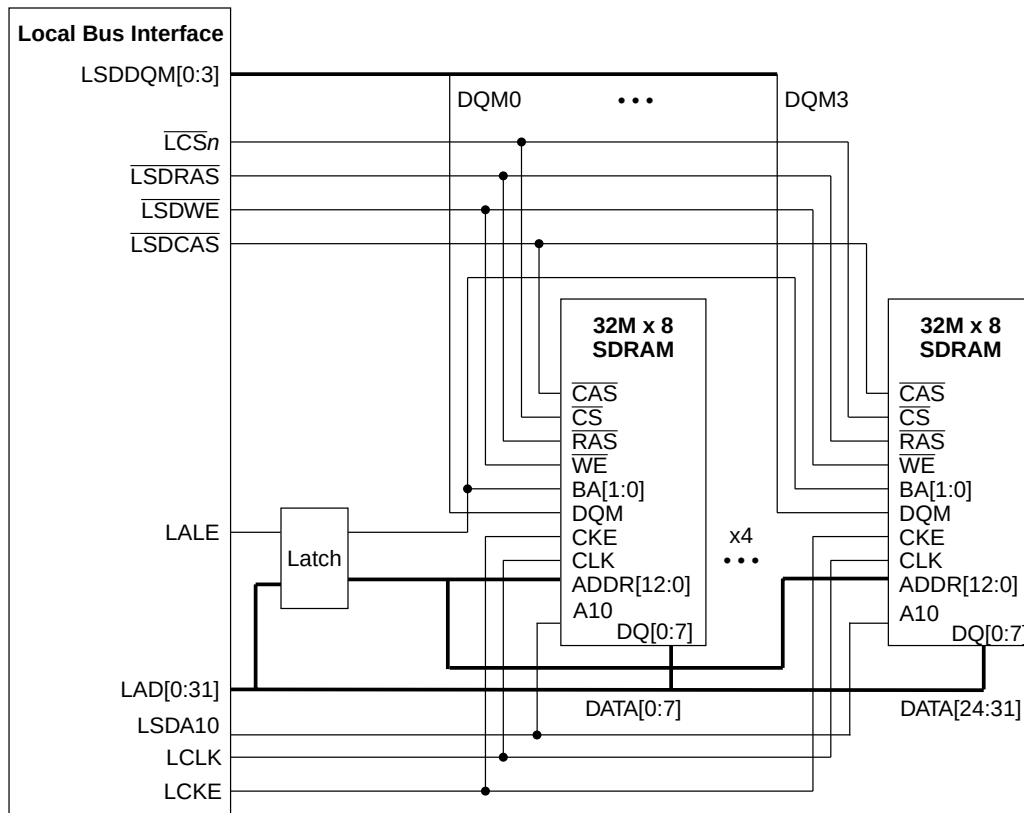


Figure 13-74. 128-Mbyte SDRAM Diagram

Consider the following SDRAM organization:

- The 32-bit port size is organized as 8 x 8 x 32 Mbit.
- Each device has four internal banks, 13 row address lines, and 10 column address lines.

The logical address is partitioned as shown in Table 13-35.

Table 13-35. Logical Address Bus Partitioning

A[0:4]	A[5:17]	A[18:19]	A[20:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters are extracted:

- COLS = 011, 10 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as shown in Table 13-36.

Table 13-36. SDRAM Device Address Port During Address Phase

LA[0:14]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank-select A[18:19]	Row A[5:17]	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

Table 13-37 shows the address port configuration during a READ/WRITE command.

Table 13-37. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17, 18]	LA[19]	LA[20:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

Table 13-38 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the device's specific refresh requirements.

Table 13-38. Register Settings for 128-MByte SDRAMs

Register	Field	Value
BRx	BA	Base address
	XBA	Ext. Base Address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
ORx	AM	11_1111_1000_0000_0000_0
	XAM	11
	COLS	011
	ROWS	100
LSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

13.5.4.3.4 256-MByte SDRAM

This example uses the same Micron SDRAM as in the previous example, but doubles the number of devices connected and therefore uses two chip selects.

13.5.4.3.5 512-MByte SDRAM Memory

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is 8 x 4 x 64 Mbit x 2 chip select lines.
- Each device has 4 internal banks, 13 row address lines, and 11 column address lines.

The logical address is partitioned as shown in Table 13-39.

Table 13-39. Logical Address Partitioning

A[0:3]	A[4:16]	A[17:18]	A[19:29]	A[30:31]
msb of start address	Row	Bank select	Column	lsb

The following parameters can be extracted:

- COLS = 100, 11 column lines
- ROWS = 100, 13 row lines

During the address phase, the SDRAM address port is set as in Table 13-40.

Table 13-40. SDRAM Device Address Port During Address Phase

LA[0:13]	LA[15:16]	LA[17:29]	LA[30:31]
—	Internal bank select (A[17:18])	Row (A[4:16])	no-connect

Because the internal bank selects are multiplexed over LA[15:16], LSDMR[BSMA] must be set to 011.

Table 13-41 shows the address port settings during a READ/WRITE command.

Table 13-41. SDRAM Device Address Port During READ/WRITE Command

LA[0:14]	LA[15:16]	LA[17]	LA[18]	LA[19:29]	LA[30:31]
msb of start address	Internal bank select	Don't care	AP	Column	no-connect

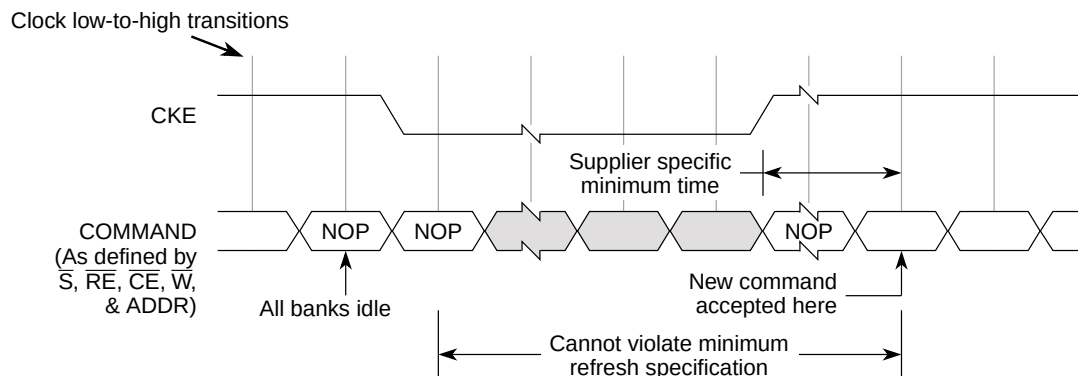
Table 13-42 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 13-42. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BRx	BA	Base address
	XBA	ext. Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
ORx	AM	11_1110_0000_0000_0000_0
	XAM	11
	BPD	01
	COLS	100
	ROWS	100
PSDMR	RFEN	1
	OP	000
	BSMA	011
	RFRC	From device data sheet
	PRETOACT	From device data sheet
	ACTTOROW	From device data sheet
	BL	0
	WRC	From device data sheet
	BUFCMD	0
	CL	From device data sheet

13.5.4.3.6 Power-Down Mode

SDRAMs offer a power-down mode, during which the device is not refreshed, therefore data is not maintained. This mode is invoked by driving CKE low, while all internal banks are idle; note that they must be precharged first. Figure 13-75 shows the timing.

**Figure 13-75. SDRAM Power Down Timing**

CKE remains low, as long as the device is powered down. After CKE transitions to high, the SDRAM exits the power down mode.

13.5.4.3.7 Self-Refresh

In order to be able to stop activity on the local bus (for power save or debug), while the content of the SDRAM is maintained, the self-refresh mode is supported. This mode is invoked by issuing a self-refresh command to the SDRAM. The LBC applies the same timing as for the auto refresh, but also pulls SDRAM CKE (LCKE) low in the same cycle. This can only be done with all banks being idle; the SDRAM machine must precharge them ahead of this. As long as CKE stays low, the device refreshes itself and does not need to see any refreshes from the local bus. To exit self refresh, CKE simply has to be pulled high. Note that after returning from self-refresh mode the SDRAM needs a supplier-specific time before it can accept new commands and the auto-refresh mechanism has to be started again. Figure 13-76 shows this timing. The SDRAM controller always uses 200 local bus clocks, which should satisfy any SDRAM requirements.

Refer to Section 13.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,” for SDRAM interface commands and information on the self-refresh command.

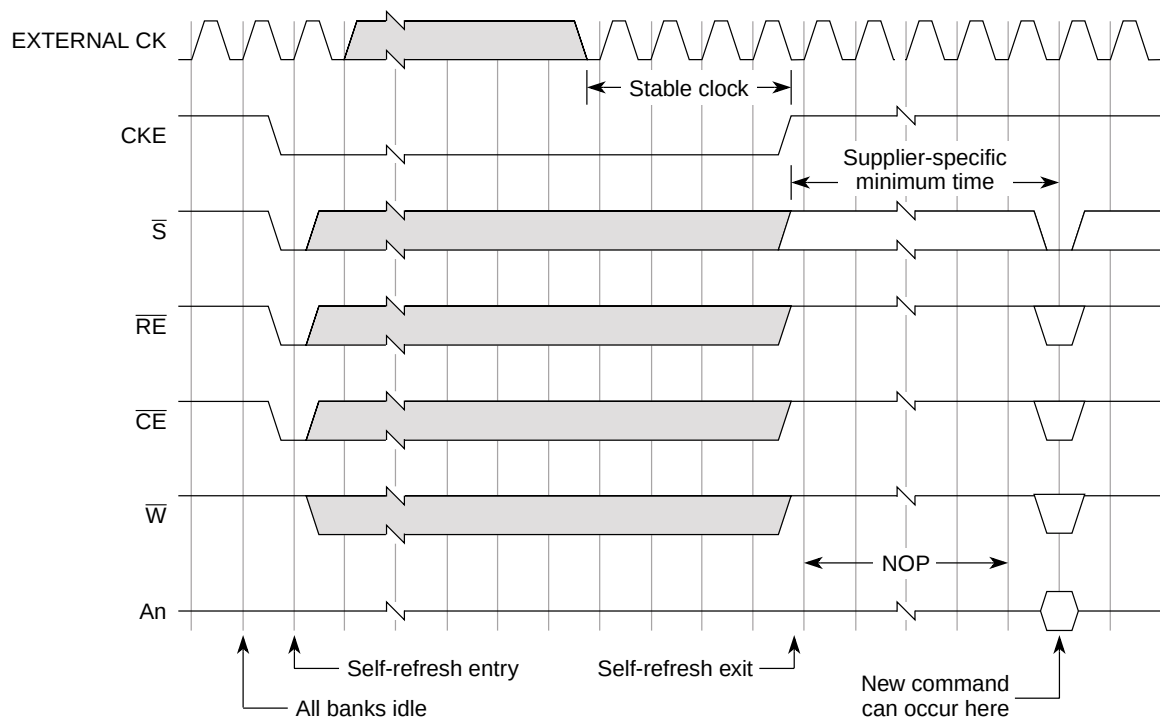


Figure 13-76. SDRAM Self-Refresh Mode Timing

13.5.4.3.8 SDRAM Timing

To allow for very high speeds on the memory bus, the capacitive loading on the local bus must be taken into consideration.

Table 13-43. SDRAM Capacitance

Pin	Min	Max	Unit
CLK	2.0	4.0	pf
$\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WE}}$, $\overline{\text{CS}}$, CKE, DQM	2.0	5.0	pf
Address	2.0	5.0	pf
DQ ₀ -DQ ₃₁	3.5	6.5	pf

Note: Capacitance values compiled from worst case numbers from various datasheets from Samsung and Micron

To implement a system using the hierarchy described earlier for two synchronous memory banks, one address latch and one buffer loading the multiplexed address/data bus sees a loading of four loads of about 6.5 pF maximum. 30 pF can be used as a nominal load.

Table 13-44. SDRAM AC Characteristics

Parameter		Device Speed				Unit
		166 MHz		133 MHz		
		Min	Max	Min	Max	
CLK cycle time	CAS latency = 3	6	1000	7.5	1000	ns
	CAS latency = 2	—		7.5		
CLK to valid output delay	CAS latency = 3	—	5	—	5.4	ns
	CAS latency = 2	—	—	—	5.4	
Output data hold time	CAS latency = 3	2.5	—	3	—	ns
	CAS latency = 2	-	—	3	—	
Input setup time		1.5	—	2	—	ns
input hold time		1	—	1	—	ns
CLK to output in Hi-Z	CAS latency = 3	—	5	5.4	—	ns
	CAS latency = 2	—	—	5.4	—	

Note: AC Characteristics compiled from worst case numbers from various datasheets from Samsung and Micron

Setup and Hold Timing Calculations:

Address TOF (time of flight): board layout delay

Data TOF (time of flight): board layout delay

Clock skew (time of flight): clock skew between the LBC and the clock at the memory device. The local bus DLL feedback mechanism must be used to control this skew to optimize the timing margins. .

Address setup margin = cycle time - local bus address CTQ - SDRAM address input setup time - address TOF + clock skew

Address hold margin = local bus address output hold time + address TOF - SDRAM address input hold time - clock skew

Data write to SDRAM setup margin = cycle time - local bus data CTQ - SDRAM data input setup time - data TOF + clock skew

Data write to SDRAM hold margin = local bus data output hold time + data TOF - SDRAM data input hold time - clock skew

Data read from SDRAM setup margin = cycle time - SDRAM data CTQ - local bus data input setup time - data TOF - clock skew

Data read from SDRAM hold margin = SDRAM data output hold time + data TOF - local bus data input hold time + clock skew

To improve the timing margins a DLL is used to generate external clocks, which minimize the skew between the local bus and the memory clock. Figure 13-77 shows relative timings for local bus clock DLL.

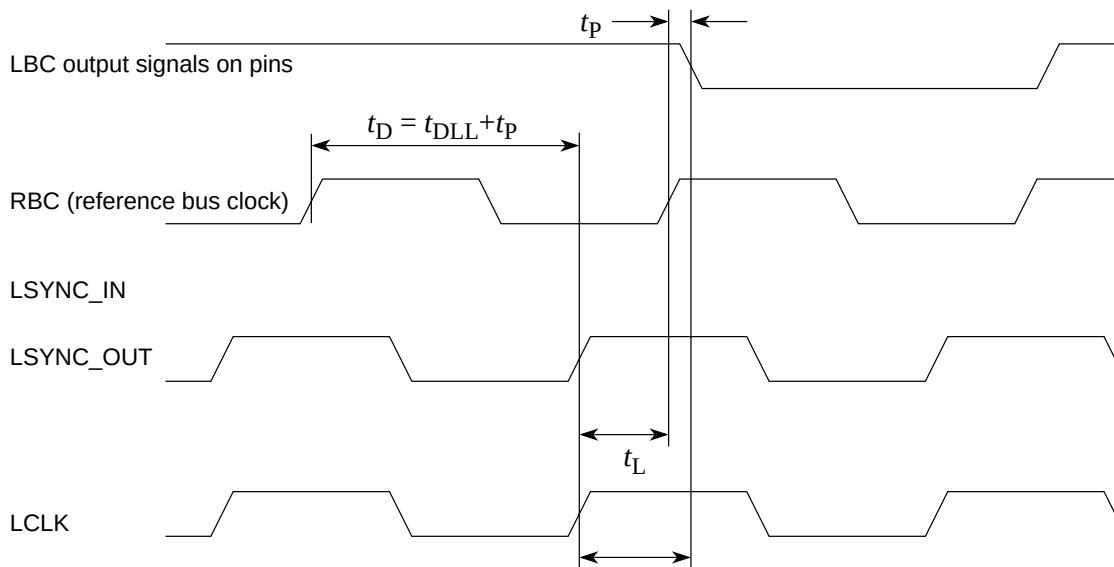


Figure 13-77. Local Bus DLL operation

13.5.4.4 Parity Support for SDRAM

Contrary to older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16 or x32. There are no mainstream devices that include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Since the local bus allows for SDRAM accesses with less than the full port size, read-modify-write cycles are supported for SDRAM write cycles.

Figure 13-78 shows a connection diagram.

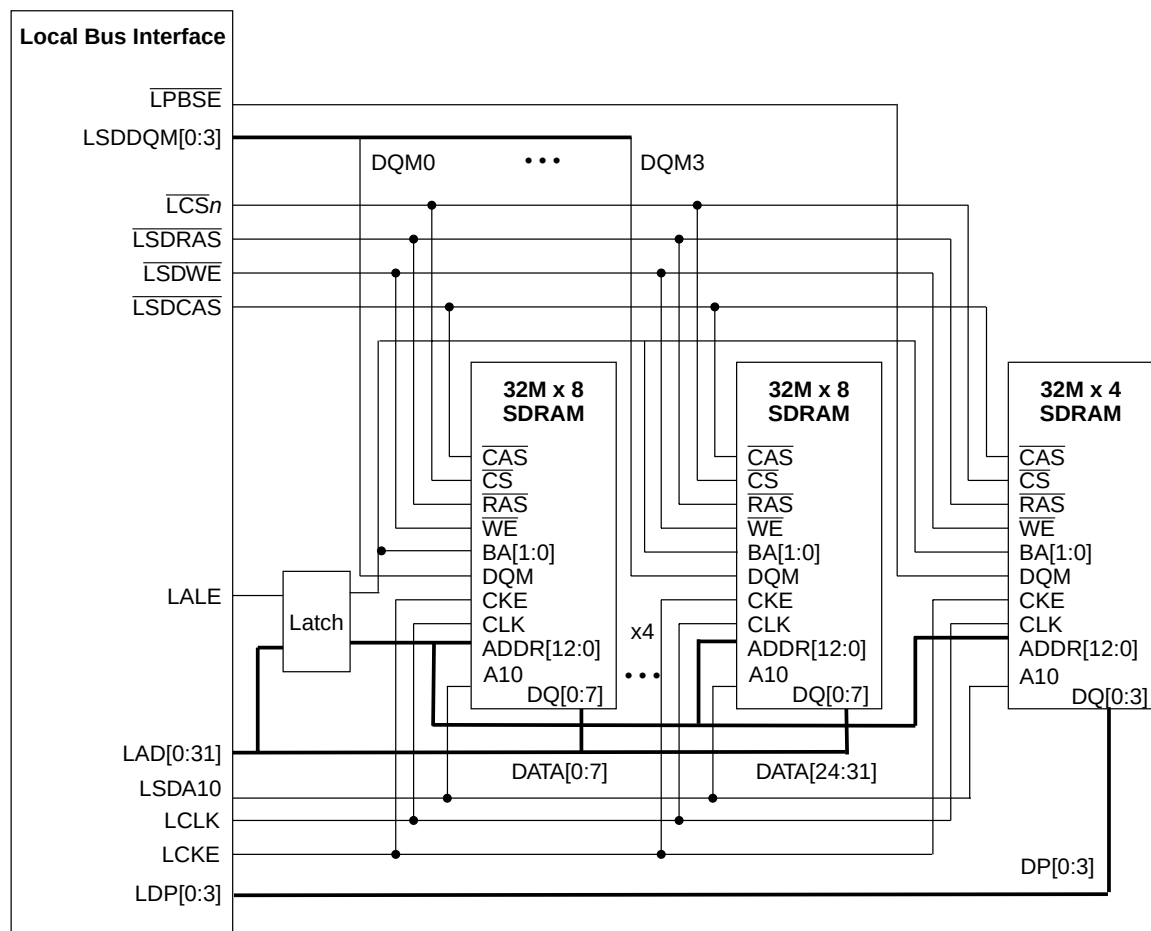


Figure 13-78. Parity Support for SDRAM

13.5.5 Interfacing to ZBT SRAM

In many applications SDRAM provides sufficient performance for the local bus. However, especially in networking applications, the memory access patterns are often very random and SDRAM is not really optimized for that case. ZBT SRAMs have been designed to optimize the performance in networking applications. This section describes how to interface to ZBT SRAMs. Figure 13-79 shows the connections. The UPM of the local bus

controller 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.

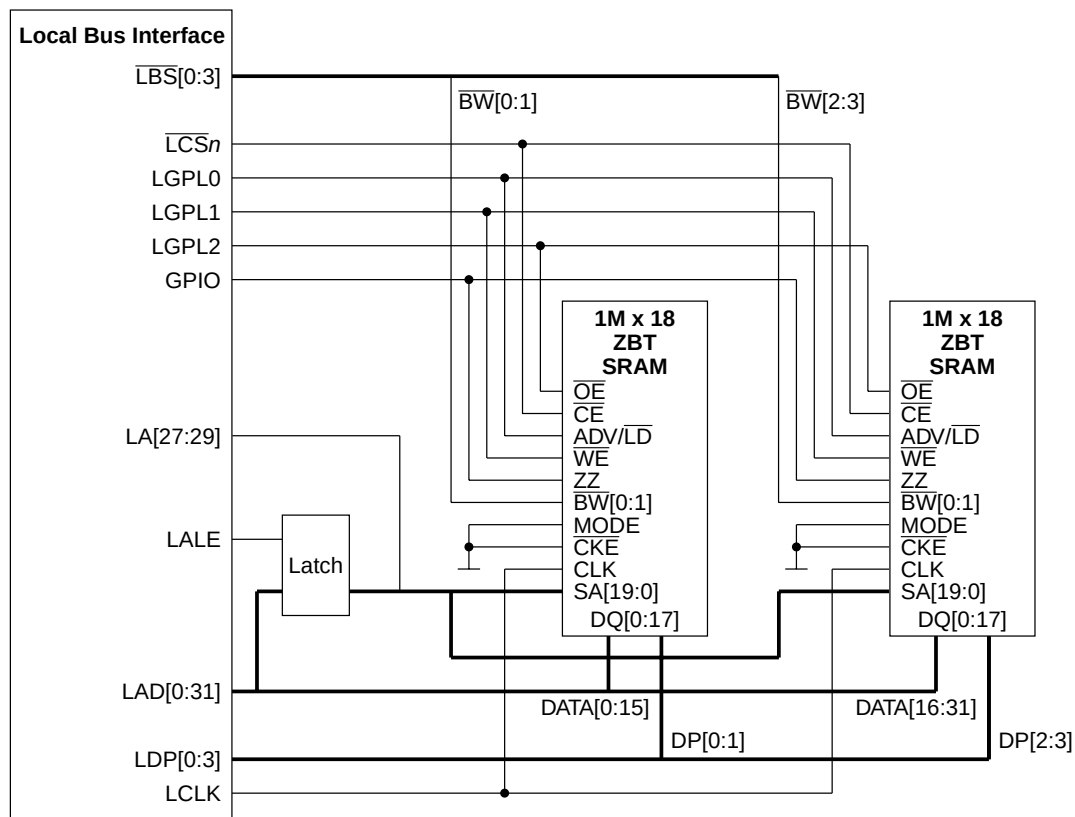


Figure 13-79. Interface to ZBT SRAM

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.

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. The $\overline{\text{CKE}}$ pin should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the local bus controller generates 8-beat transactions (for 32 bit ports) the UPM breaks down each burst into 2 consecutive 4-beat bursts. The internal address generator of the local bus controller generates the new A27 for the second burst.

Because we use linear burst on the SRAM, the device will itself burst 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 = 0 for the first burst and A27 = 1 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 $\overline{WE}$). The UPM controller basically has to wait for the end of the SRAM burst to avoid bus contention with further bus activities.

ZBT SRAMs have a power down mode, which is invoked by the ZZ pin. Connecting a GPIO pin to ZZ allows use of that power down mode; however accesses to the SRAM while in power down mode do not create valid results. This should be taken care of by the system software.

Another observation is that SRAMs are available with natural parity. In the example, we use a x18 SRAM, which holds two data bytes and two parity bits. While for the support of parity on SDRAM banks the local bus has to use read-modify-write cycles and compromise performance, SRAM banks can be used with natural parity and do not compromise performance for parity support.

13.5.6 Interfacing to DSP Host Ports

In many applications, an integrated communications processor aggregates traffic for DSPs and distributes that traffic to the DSPs. The local bus allows connection to a variety of different DSP host ports and this section gives some information on how to interface to some example DSPs.

13.5.6.1 Interfacing to MSC8101 HDI16

This section describes how to interface to the HDI16 peripheral interface of the MSC8101. After initial set-up of the HDI16 interface, the communication between the host and HDI16 device can be achieved either by a read and write transaction from the core or, if the setup on the DSP and the host are implemented appropriately, by DMA transfers of the host DMA controller, which can be triggered automatically by signals generated by the HDI16 peripheral.

13.5.6.1.1 HDI16 Peripherals

The host interface (HDI16) is a 16-bit wide, full-duplex, double-buffered parallel port that can directly connect to the data bus of a host processor. The HDI16 supports a variety of buses and gluelessly connects with a number of industry-standard microcomputers, microprocessors, and DSPs. The HDI16 also supports the 8-bit host data bus, which makes it fully compatible with the DSP56300 HI08 (as viewed by the host side, not from the DSP side).

The host bus can operate asynchronously to the SC140 core clock, and the HDI16 registers are divided into two banks. The host register bank is accessible to the external host, and the core register bank is accessible to the SC140 core.

The MSC8101HDI16 host port peripheral has two sets of 16-bit wide registers—one set is only visible internally to the DSP, while the other set is visible only to the external host processor. Figure 13-81 illustrates the relationship between the two sides.

All of the HDI16 peripheral's registers are mapped directly onto the MSC8101's QBus, as defined by the *MSC8101 Reference Manual, 16-bit Digital Signal Processor* (order number MSC8101RM/D), and the transmit and receive FIFOs are mapped onto the DMA data bus such that the DMA controller can access them directly without core intervention. The addressing for each of these registers is defined in Section 13.5.6.1.2, "Physical Interconnections."

The HDI16 host port itself is a 16-bit wide parallel port with various strobe and multiplexing options.

The most important HDI16 host port facet is that it is specified as an asynchronous interface and so reduces concerns over clock skew between the HDI16 host port and the host device's buses. Furthermore, with all the host port registers being accessed with a single chip select and 4 address lines, as far as the local bus is concerned, the DSP host port is akin to an asynchronous memory mapped region. So, for the HDI16 port in single strobe mode, the host device asserts a chip select, a single data strobe and a read/write line to select HDI16 read or write bus operations.

The read and write strobes are also used as the data latch control to complete the bus transactions, obviating the need for any handshake termination signal from the DSP. The UPM programmer is responsible for satisfying the AC timings of the HDI16 transactions.

Through appropriate mode selection, the HDI16 peripheral's feature set can be fully supported by the local bus's UPM controlled signals. The UPM defined interface can be used with any of the local bus's 8 chip selects to give the 16-bit port size and strobe generation that matches that of the HDI16 host port.

Figure 13-80 shows the internal register diagram of the HDI16.

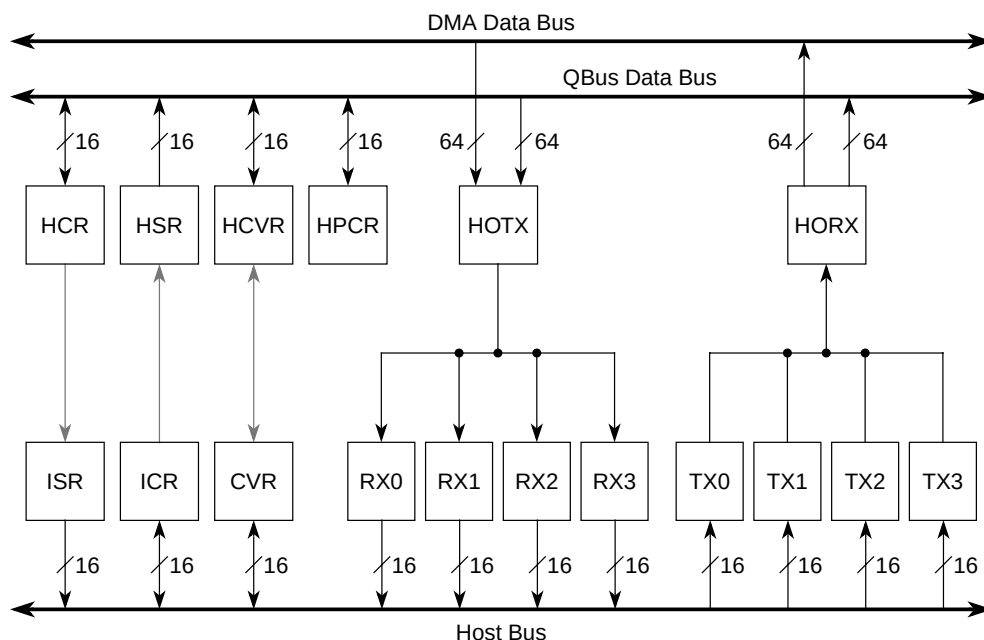


Figure 13-80. MSC8101 HDI16 Peripheral Registers

13.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 13-45 and Figure 13-81.

Table 13-45. Local Bus to MSC8101 HDI16 Connections

MSC8101 Signal(s)	Type	Description	Connect with Local Bus Signal
HD[0:15]	I/O/Z	Host data bus	LAD[0:15]
HA[0]	I	Host address Line	either LA[27] or latched A25
HA[1]	I	Host address Line	either LA[28] or latched A26
HA[2]	I	Host address Line	LA[29]
HA[3]	I	Host address Line	LA[30]
HCS1	I	Host Chip select 1	LCSx
HCS2	I	Host Chip select 2	tie this to V_{DD}
HRDRW	I	Host $R/\overline{W}$ signal	LGPL1 or inverted LBCTL signal
HDS	I	Host Data stobe signal	LGPLY
HRRQ/HACK	O	Receive Host Request OP	as required in application
HTRQ/HREQ	O	Transmit Host Request OP	as required in application

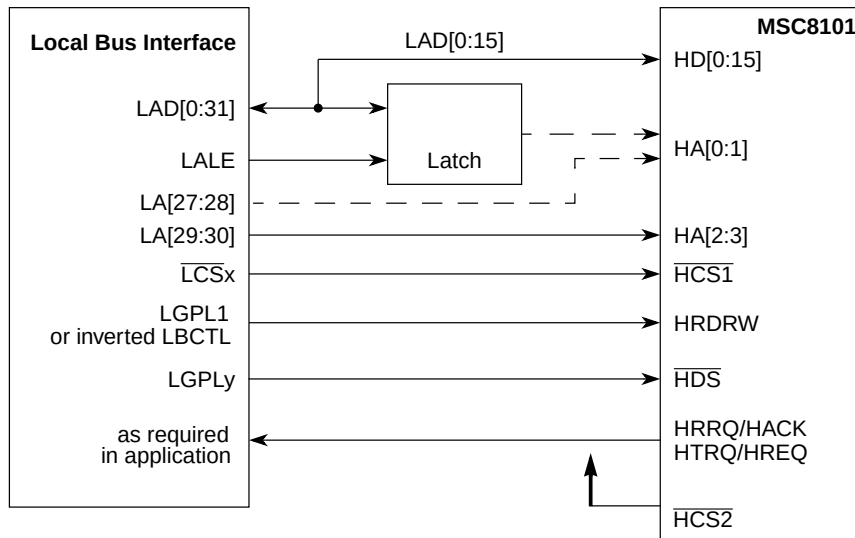


Figure 13-81. Interface to MSC8101 HDI16

The connections are specified as follows:

- One chip select, LCSx (whatever is available in the system), is used to memory map accesses from the host local bus to the HDI16 MSC8101's HDI16 peripheral, and is connected to the HDI16 chip select line (HCS).
- This interface uses two separate general-purpose strobe lines (LGPL1 and LGPLy):
 - LGPL1 is programmed to generate the HDI16 read/write signal (HRDRW), which is typically high for a read access and low for write access. In any case, the host port requires a HR/ $\overline{W}$ signal. This can be generated by using LGPL1, and allows it to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM patten to satisfy the AC timings at the DSP host port.
 - LGPLy is programmed to generate the HDI16 data strobe (HDS), which must be asserted every 16-bit read or write transaction.
- Data lines—The bus data lines (LADx) are directly connected to the HDI16 data lines (HDx).
- DMA request/Service request signals (HRRQ & HTRQ)—as appropriate in the application
- Address lines—The address lines between the LBC and the HDI16 MSC8101 can either be connected in a straightforward or a specific manner to enable burst transfers across the HDI16. The connections are defined as follows and are described in more detail in the following sub-section:
 - either LA[27] or latched value of A25 -> HA0

- either LA[28] or latched value of A26 -> HA1
- LA[29] -> HA2
- LA[30] -> HA3
- Ground lines—In order to provide the best ground plane, it is highly advised than all grounds are common and connected together.

13.5.6.1.3 Supporting Burst Transfers

As mentioned previously, to facilitate burst transfers the host's local bus address lines can be connected in a very specific way. First, the local bus A31 signal is not required, as the HDI16 registers are 16-bit word addressed. Secondly, local bus LA[27] and LA[28] signals can be eliminated, so that the host side transmit and receive registers wrap around the same four 16-bit word addresses.

For example, host transmit register 0 on the HDI16 peripheral (address 0x04) can be obtained by the host by accessing any of the following memory mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses as the source or destination addresses increment after each 16-bit access to the interface for all 16 transactions within that burst. By using the addressing as defined, if the first access is at 0x20, the last will be at 0x3e but, more importantly, the four host Tx/Rx registers will have been looped around four times.

13.5.6.1.4 Host 60x Bus:HDI16 Peripheral Interface Hardware Timings

The host UPM controlled local bus and the HDI16 MSC8101's HDI16 host interface are both programmable. Careful programming of the host chip select registers and UPM can meet the HDI16 MSC8101 host port timings.

On any bus access the critical timing for both read and write is typically around the data latch point. For the UPM based read access, the host has the flexibility to latch data on a rising or falling LCLK edge. The falling LCLK edge is used here to latch the HDI16 data into the host MSC8101 at its earliest convenience.

After the data is latched, appropriate HDI16 port data hold time is ensured before the data strobe (DS) and chip select (CS1) are negated.

On a UPM write cycle, the critical action is in enveloping the DS assertion with CS asserted to ensure proper write data hold time after latching by the HDI16 host port.

Special attention needs to be given to both the host read and write access strobe (DS) negation times (HDS assert).

The HDI16 MSC8101 specifies some restrictions for consecutive register access, which results in a hold off negation time for the read and write access strobes.

Rather than restrict the firmware to avoid consecutive bus accesses to host port registers, the negation hold off times should be accommodated in the UPM hardware interface

settings. Additional clocks must be built into the end of UPM based cycle giving appropriate time before the next bus cycle starts.

The timings can be readily adapted to allow external decode logic to be added to support chip selects for a larger number of DSP HDI16 host ports.

13.5.6.2 Interfacing to MSC8102 DSI

The MSC8102 direct-slave interface (DSI) gives an external host direct access to the MSC8102. It provides the following slave interfaces to an external host:

- Asynchronous SRAM-like interface giving the host single accesses (with no external clock).
- Synchronous SSRAM-like interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8102 internal bus clock.

The DSI supports 32 or 64 bit data bus. For connection to the local bus the DSI has to be configured in 32 bit mode. This is achieved through the DSP reset configuration.

The DSI supports two addressing modes, which are determined during the MSC8102 boot sequence. Refer to details in the MSC8102 documentation.

- Full address bus mode with HA[11–29] used in both 32-bit data mode and 64-bit data mode.
- Sliding window mode with HA[14–29] used in both 32-bit data mode and 64-bit data mode.

13.5.6.2.1 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. As an example the dual strobe configuration is shown below.

Figure 13-82 shows the interface to the MSC8102 DSI for asynchronous mode.

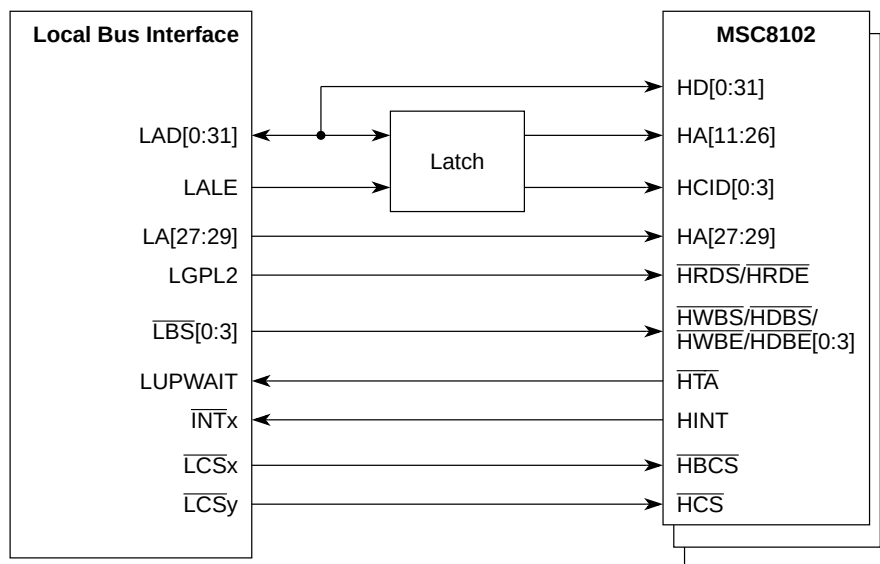


Figure 13-82. Interface to MSC8102 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used, if only relatively small amounts of data are transferred between the communications controller and the MSC8102. To allow for maximum timing flexibility, the UPM machine of the local bus controller should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. The flexibility of the UPM allows sufficient control to satisfy any requirements here.

Figure 13-83 shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0–3]) on the first falling edge of the host write byte strobe signals ($\overline{\text{HWBS}}$) on which the host chip select signal ($\overline{\text{HCS}}$) is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. The DSI will signal with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$), whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating $\overline{\text{HWBS}}$. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the UPM mode register (MxMR) has to be cleared to interpret the correct polarity of the $\overline{\text{HTA}}$ signal. The DSI samples the host address bus (HA) and the host data bus (HD) on the rising edge of $\overline{\text{HWBS}}$. In addition the assertion of $\overline{\text{HWBS}}[0:3]$ are sampled at the end and are part of the access attributes.

Because the UPM is used for this mode, the DCR[4]:HTAAD should be set to 1 and DCR[9-10]:HTADT should be defined to a value different than 00. This mode is to be used in implementations with a pull-up resistor on $\overline{\text{HTA}}$. The host can start its next access (back-to-back accesses) without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then to prevent contention on the $\overline{\text{HTA}}$ signal, the host must wait

until the previous DSI stops driving $\overline{\text{HTA}}$ before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive accesses before $\overline{\text{HTA}}$ is actively driven to a value of 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

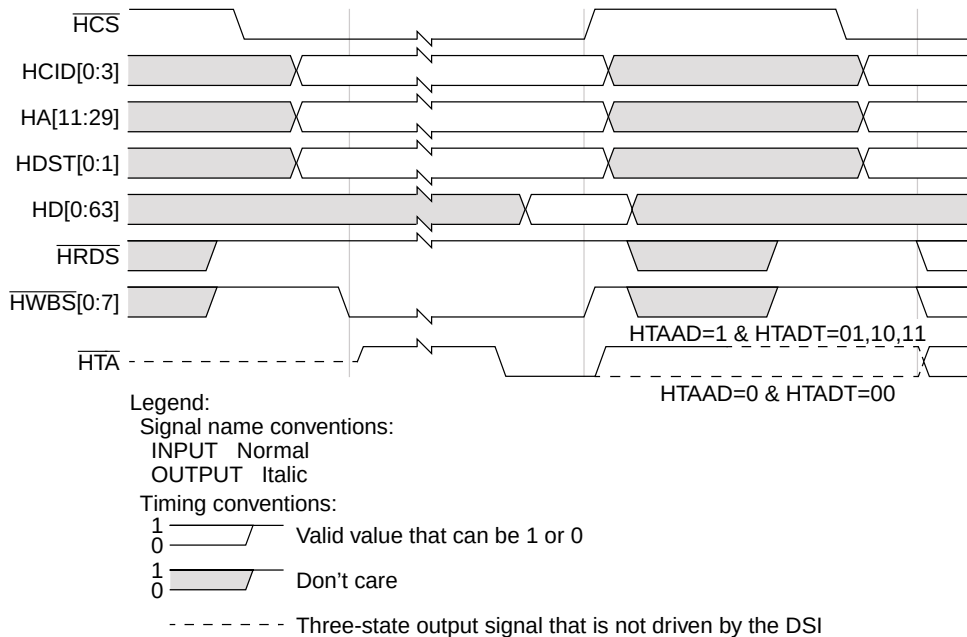


Figure 13-83. Asynchronous Write to MSC8102 DSI

Figure 13-84 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ($\overline{\text{HRDS}}$) on which the $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. When the DCR[8]:RPE bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. The DSI signals (with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$)) that data is valid, and the host can sample the host data bus (HD) and terminate the access by negating $\overline{\text{HRDS}}$. If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of $\overline{\text{HTA}}$ is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the UPM mode register (MxMR) has to be cleared to interpret the correct polarity of the $\overline{\text{HTA}}$ signal.

Because the UPM is used in the mode, the DCR[4]:HTAAD should be set to 1 and the drive time control field, DCR[9-10]:HTADT, should be defined to a value different than 00. This mode is specially designed to be used for implementations with a pull-up resistor on the $\overline{\text{HTA}}$ signal.

The host can start its next access (back-to-back accesses) without negating the $\overline{\text{HCS}}$ signal between accesses. If the next access is not to the same MSC8102, then to prevent contention

on the $\overline{\text{HTA}}$ signal, the host must wait until the previous DSI stops driving the $\overline{\text{HTA}}$ signal before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive access before the $\overline{\text{HTA}}$ signal is actively driven to 1 by the previous access. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ will be inactive.

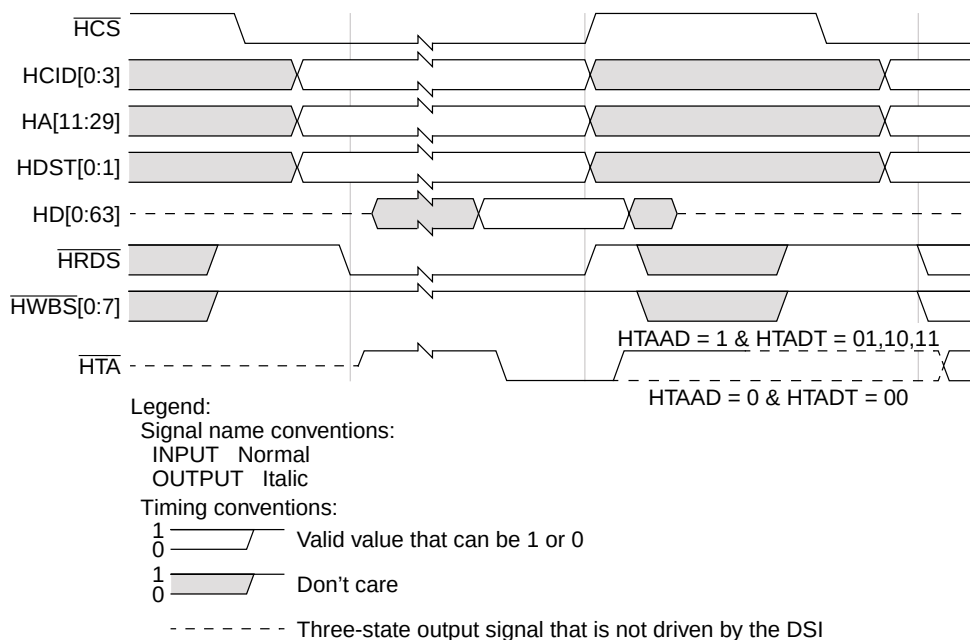


Figure 13-84. Asynchronous Read from MSC8102 DSI

13.5.6.2.2 DSI in Synchronous Mode

The synchronous SSRAM-like mode of the DSI is inherently faster than the asynchronous mode and should be used if larger amounts of data are transferred between the communications controller and the MSC8102. This will optimize the bus utilization, especially if several MSC8102s are connected to one local bus. The UPM machine of the LBCr must be used to implement this interface.

Figure 13-85 shows the interface for synchronous mode.

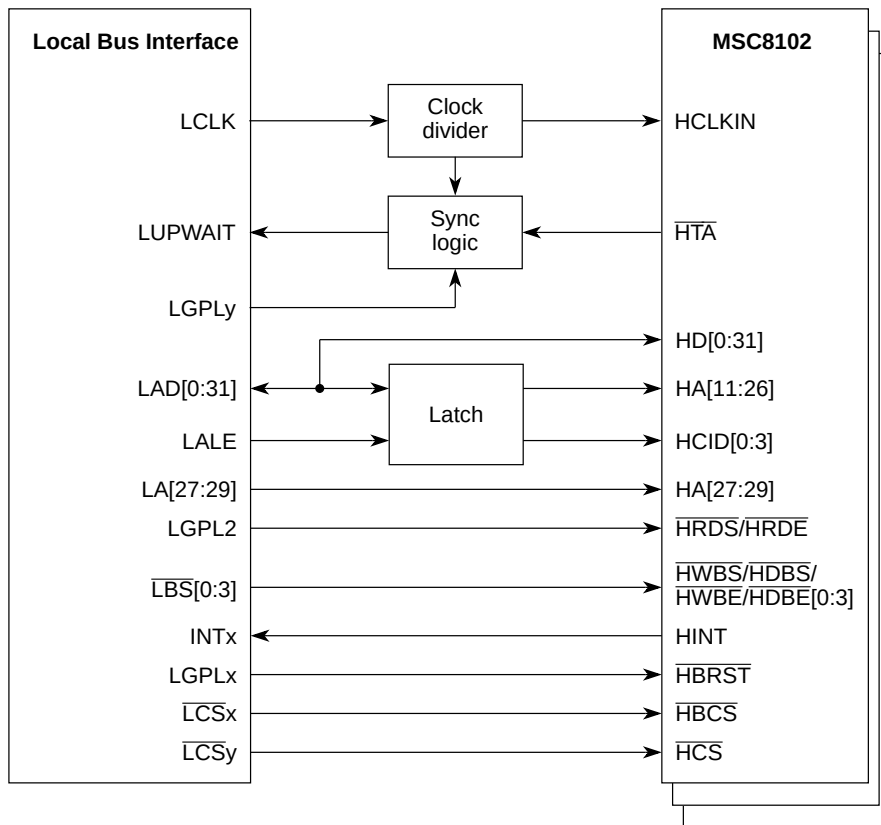


Figure 13-85. Interface to MSC8102 DSI in Synchronous Mode

Because the DSI will assert and negate $\overline{HTA}$ in synchronous mode even within a burst transfer on a clock-by-clock basis and because the DSI expects the host to react within one clock cycle, some tricks can be implemented to support the synchronous mode.

$\overline{HTA}$ drives UPWAIT of the UPM. The UWPL bit in the UPM mode register (MxMR) has to be cleared to interpret the correct polarity of the $\overline{HTA}$ signal. Because this signal influences the internal state machine of the local bus clock, the local bus can not react to the changes of the $\overline{HTA}$ signal correctly within one local bus clock. Refer to Section 13.4.4.4.10, “The Wait Mechanism (WAEN),” for more detailed information.

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3 or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3 or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1x, 2x, 3x or 4x, which should be sufficient for any divider ratio that would be used in this case.

This solution allows the local bus to react within multiple local bus clocks to the $\overline{\text{HTA}}$ signal and be still within one DSI clock. Typically, UPWAIT is synchronized internally, and only 2 clocks after UPWAIT changes, new data can be sampled or presented. For example, if the local bus clock ratio is 3x the DSI clock, data can be sampled in the third local bus sub-clock, which is the last third of the DSI clock. If the local bus clock ratio is only 2x the DSI clock, there is a special mode, where the UPWAIT is NOT synchronized. Refer to Section 13.4.4.5, “Synchronous Sampling of UPWAIT for Early Transfer Acknowledge,” for more detailed information. In this mode, data is sampled in the 2nd subclock, which is the second half of the DSI clock. AC timing of UPWAIT must be met in this mode; otherwise undeterministic behavior may occur.

The remaining issue is the synchronization of the UPM cycles to the beginning of the DSI clock cycle. Because the UPM executes n cycles for every cycle of the DSI, a mechanism must be used to ensure that the UPM changes transitions in a way that is synchronized to the DSI clock. The solution is to use a special synchronize cycle at the beginning of the pattern. A GPL signal is used to control a multiplexor and to activate external synchronization logic, which uses the DSI clock to stall the UPM by asserting UPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexor connects UPWAIT to $\overline{\text{HTA}}$ instead, for the rest of the bus cycle. Note that the GPL signals should be used in the inverted state of their inactive state (GPL[0:4] are 1 when inactive, GPL5 is 0 when inactive) to start the synchronization process.

Figure 13-86 shows an example for a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary in length from one to n (clock ratio) local bus clocks.

The second column (compensation cycle) is intended to compensate for the reaction time of UPWAIT to get in lockstep with the DSI clock. For example, if the clock divider ratio is 1:3 and the UPWAIT reaction time is two local bus clocks, because UPWAIT is synchronized, then one local bus clock should be inserted.

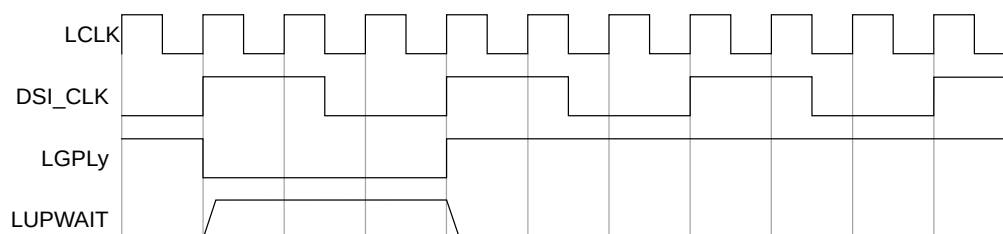


Figure 13-86. UPM Synchronization Cycle

Table 13-46. UPM Synchronization Cycles

	Sync CYCLE	Compensation CYCLE	dsi CYCLE 1	Bits
cst1–cst4				0–3
bst1–bst4				4–7
g0xx				8–11
g1tx				12–13
g2tx				14–15
g3tx				16–17
g4t1				18
g4t3	1	0		19
g5tx				20–21
redo[0]	0	0	1	22
redo[1]	0	0	0	23
loop	0	0		24
exen	0			25
amx0	0	0		26
amx1	0	0		27
na	0			28
uta	0			29
todt	0			30
last	0			31

This section describes synchronous single write and read, and synchronous burst write and read operations.

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. The dual strobe configuration is shown as an example.

Synchronous Single Write

Figure 13-87 shows a synchronous single write access. The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0–3] match the CHIPID value, the DSI is accessed. At least one $\overline{\text{HWBE}}$ signal is asserted, and $\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are negated. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current access and the host must terminate this access. Because $\overline{\text{HTA}}$ is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. Typically, $\overline{\text{HTA}}$ is asserted immediately. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. $\overline{\text{HTA}}$ is asserted for one HCLKIN cycle, driven to logic 1 in the next cycle, and stops being driven on the next rising

edge of HCLKIN. The host can start its next access to the same MSC8102 immediately on the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ will be inactive.

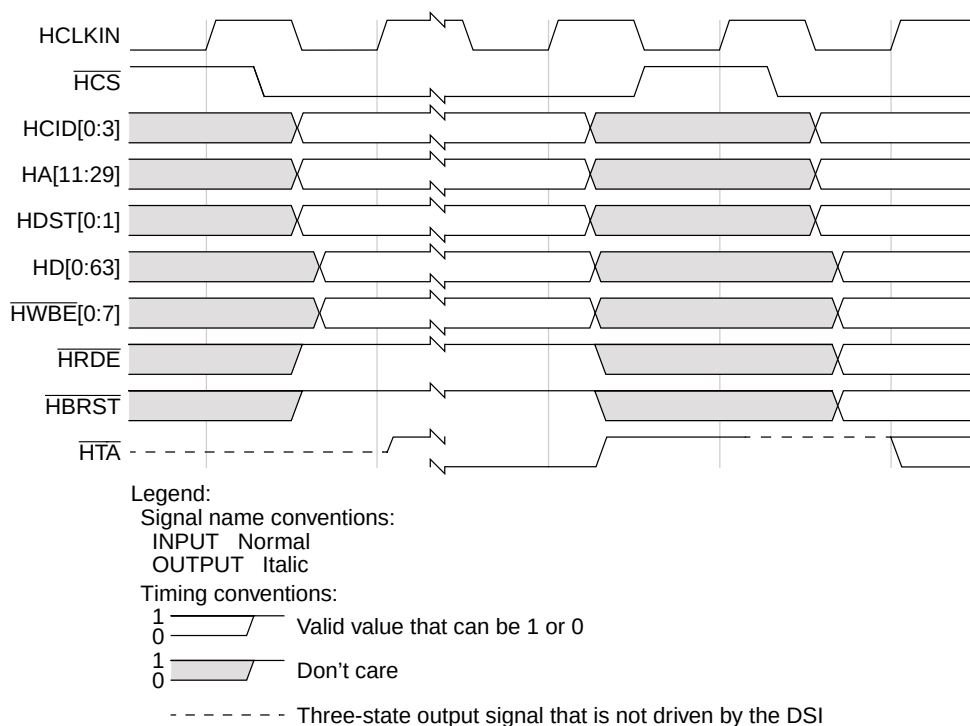


Figure 13-87. Synchronous Single Write to MSC8102 DSI

Synchronous Single Read

Figure 13-88 shows a synchronous single read access. The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. $\overline{\text{HRDE}}$ is asserted, and $\overline{\text{HWBE}}$ and $\overline{\text{HBRST}}$ are negated. When the DCR[8]:RPE bit is set (see MSC8102 documentation), read access to the memory space (not to the register space) initiates prefetching data from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid and the host must sample the HD and terminate the access. Because $\overline{\text{HTA}}$ is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. $\overline{\text{HTA}}$ is asserted for one HCLKIN cycle and driven to logic 1 in the next cycle. It stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If

the next access is not to the same MSC8102, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ will be inactive.

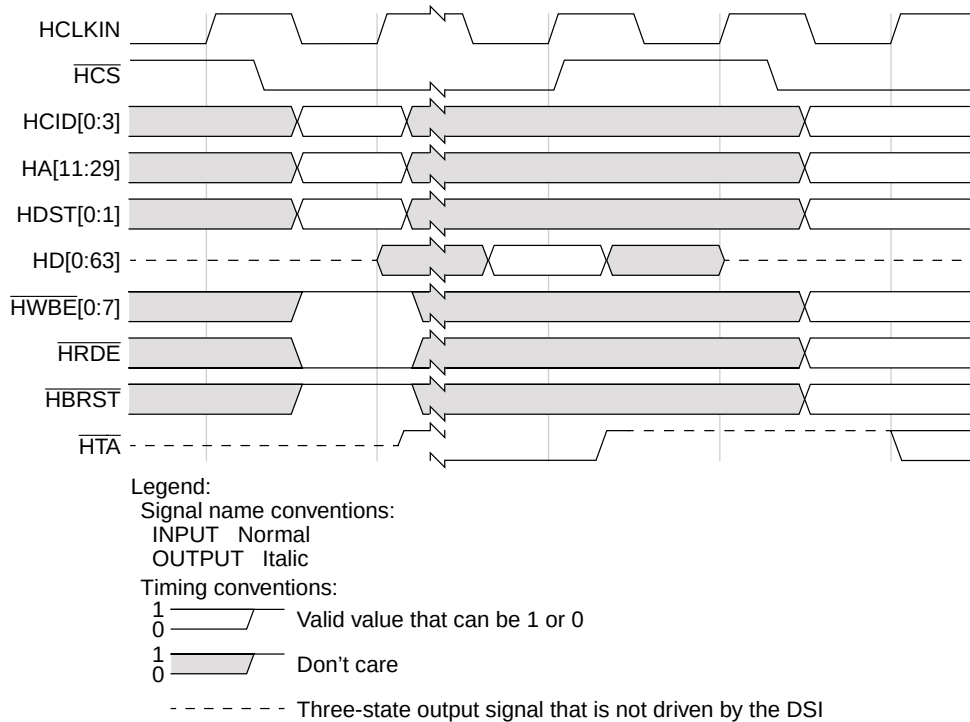


Figure 13-88. Synchronous Single Read to MSC8102 DSI

Synchronous Burst Write

Figure 13-89 shows a synchronous burst write access. The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. $\overline{\text{HWBE}}$ are asserted, $\overline{\text{HBRST}}$ is asserted, and $\overline{\text{HRDE}}$ is negated. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically $\overline{\text{HTA}}$ is asserted immediately for each beat of the access. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. Because $\overline{\text{HTA}}$ is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. After the last beat of the access, $\overline{\text{HTA}}$ is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest

way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ will be inactive.

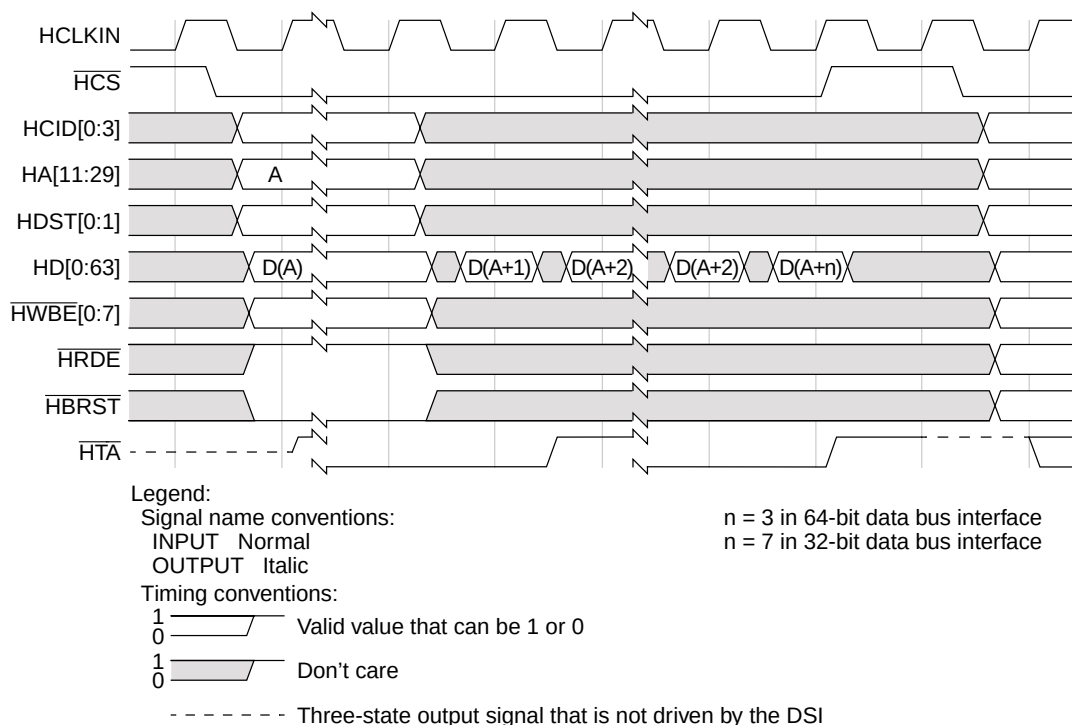


Figure 13-89. Synchronous Burst Write to MSC8102 DSI

Synchronous Burst Read

Figure 13-90 shows a synchronous burst read access. The DSI samples $\overline{\text{HA}}$, $\overline{\text{HDST}}$, $\overline{\text{HCID}}$, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first $\overline{\text{HCLKIN}}$ rising edge on which $\overline{\text{HCS}}$ is asserted. If $\overline{\text{HCID}}[0:3]$ match the CHIPID value, the DSI is accessed. $\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When the DCR[8]:RPE bit (see the MSC8102 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because $\overline{\text{HTA}}$ is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of $\overline{\text{HCLKIN}}$. The host can start its next access to the same MSC8102 immediately in the next $\overline{\text{HCLKIN}}$ rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next

device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee, that $\overline{\text{HTA}}$ will be inactive.

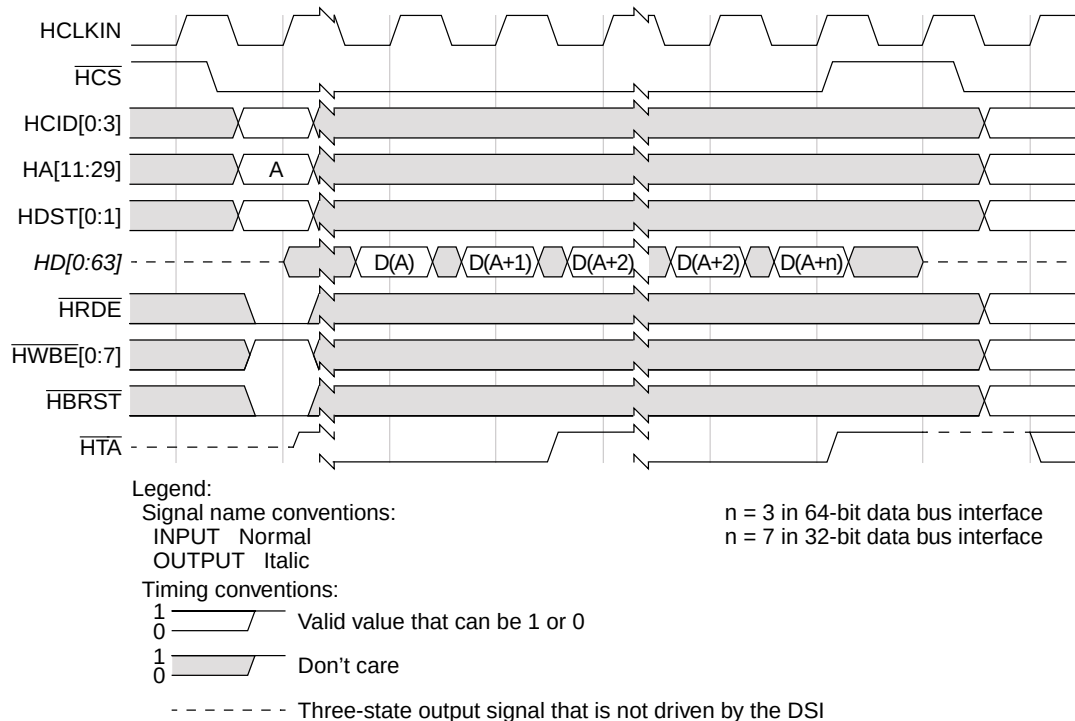


Figure 13-90. Synchronous Burst Read from MSC8102 DSI

13.5.6.2.3 Broadcast Accesses

Using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8102 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode $\text{HCID}[0:3]$.

Note that broadcasting is allowed only for write accesses.

The DSI sets the DSI error register (DER) OVF bit if there is an overflow during broadcast accesses. This bit can be cleared by writing a value of 1 to it.

NOTE

To avoid overflow when accessing DSI registers during broadcast accesses, wait at least 10 host clock cycles (in synchronous mode) or 8 internal clock cycles (in asynchronous mode) in between each DSI register accesses.

To avoid data corruption, when the DER[0]:OVF bit is set, any broadcast access is not written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, the DER[0]:OVF bit must first be read and reset if it is set.

NOTE

In asynchronous mode, write data from a previous access (even if it is from a previous normal write access) may be lost due to overflow during broadcast accesses. To prevent such a loss, ensure that previous access data has propagated to the FIFO or DSI registers, depending on the type of previous access. This can be achieved by performing a read access prior to the first broadcast access.

In broadcast accesses, the host must comply with the following rules:

- In asynchronous mode, $\overline{HWBS}[0:3]/\overline{HDBS}[0:3]$ assertion time should be at least the minimum, which is defined in the AC characteristics section of the *MSC8102 Technical Data* sheet.
- In synchronous mode single access, the host must wait 1 cycle before terminating the access. Access signals must be in the same valid state during two positive edges of the host clock cycles. Access duration is 2 clock cycles (the DSI may translate accesses lasting longer than 2 clock cycles as two or more back-to-back accesses).
- In synchronous mode burst accesses, broadcast accesses are not allowed.

13.5.6.3 Interfacing to EHPI from Texas Instruments TMS320Cxxxx DSPs

The enhanced host port interface (EHPI) on DSPs from Texas Instruments provides a 16-bit-wide parallel port through which a host processor can directly access the memory of the DSP. The host and the DSP can exchange information via memory internal or external to the DSP and within the address reach of the EHPI. The EHPI uses 23-bit addresses, where each address is assigned to a 16-bit word in memory.

The EHPI has two modes, one for multiplexed address/data and one with separate buses.

To allow the connection of multiple DSPs and other peripherals on the local bus, the use of the EHPI non-multiplexed mode is recommended.

The EHPI uses the following signals:

Table 13-47. EHPI Signals

Signals	Type	Description	Connect with
HD[15:0]	I/O/Z	Host data bus non-muxed mode: data only	LAD[0:15]
HA[19:4]	I	Host address bus non-muxed mode: addresses	latched A[11:26]
HA[3:0]		Host address bus: lsbs	LA[27:30]
HBE[1:0]	I	Host byte-enable signals 00:Word, 01:MSB, 10:LSB, 11: Reserved	$\overline{\text{LBS}}[0:1]$
HCS	I	Chip select signal	$\overline{\text{LCSx}}$
HR/ $\overline{\text{W}}$	I	R/ $\overline{\text{W}}$ signal	LGPLY or inverted LBCTL
$\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$	I	Data strobe signals must be at least 2 DSP clocks wide - Host has separate active-low read and write strobe pins: connect one to $\overline{\text{HDS1}}$, one to $\overline{\text{HDS2}}$ - Host has one active-low strobe pin: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 1 - Host has one active high-strobe pin: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 0	LGPLx
HRDY	O	EHPI ready signal	LUPWAIT
HCNTLO	I	EHPI control signals non-muxed mode: - HCNTL=1: access DSP data memory - HCNTL=0: access EHPI control register	application specific: either GPIO pin or latched address lines
HAS	I	Address strobe signal	
HMODE	I	EHPI mode signal High: non-muxed mode Low: muxed mode	high
RST_MODE	I	Reset mode signal	
HINT	O	DSP to host interrupt signal	INTx

To achieve the timings required by the DSP host port and optimize the bus usage, the use of one of the UPMs is recommended.

Note that the DSP address signals reflect 16-bit addresses, whereas local bus address signals reflect byte addresses. This essentially shifts address signals by one bit.

The EHPI host port has two different strobe pins, $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ that allow different options to control transfers. The UPM allows the flexibility to use any of those options; however, the easiest is to use the single active low strobe mode and connect one UPM LGPL signal (whatever is available) to $\overline{\text{HDS}}$.

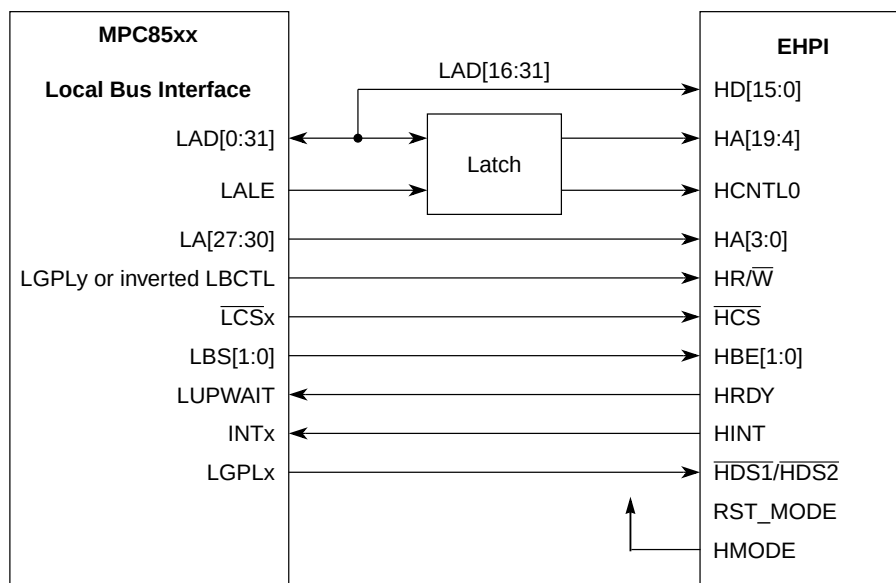
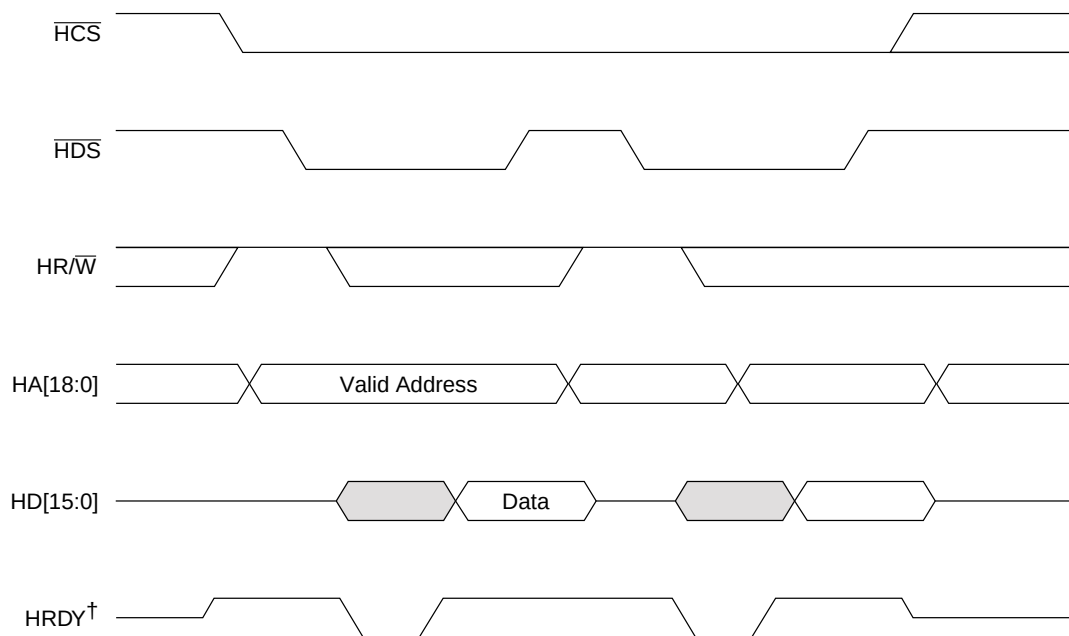


Figure 13-91. Interface to Texas Instruments EHPI in Non-Multiplexed Mode

In any case the host port requires a $HR/\overline{W}$ signal. This can be generated by using another LGPL signal and allows to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM pattern to satisfy AC timings at the DSP host port.

The DSP does not necessarily have deterministic access times. HRDY indicates whether the EHPI is ready for an access. If the signal is low, wait states must be inserted in the cycle. The UPWAIT function of the UPM provides this mechanism. The UWPL bit in the UPM mode register (MxMR) must be set to connect to this active-low signal.

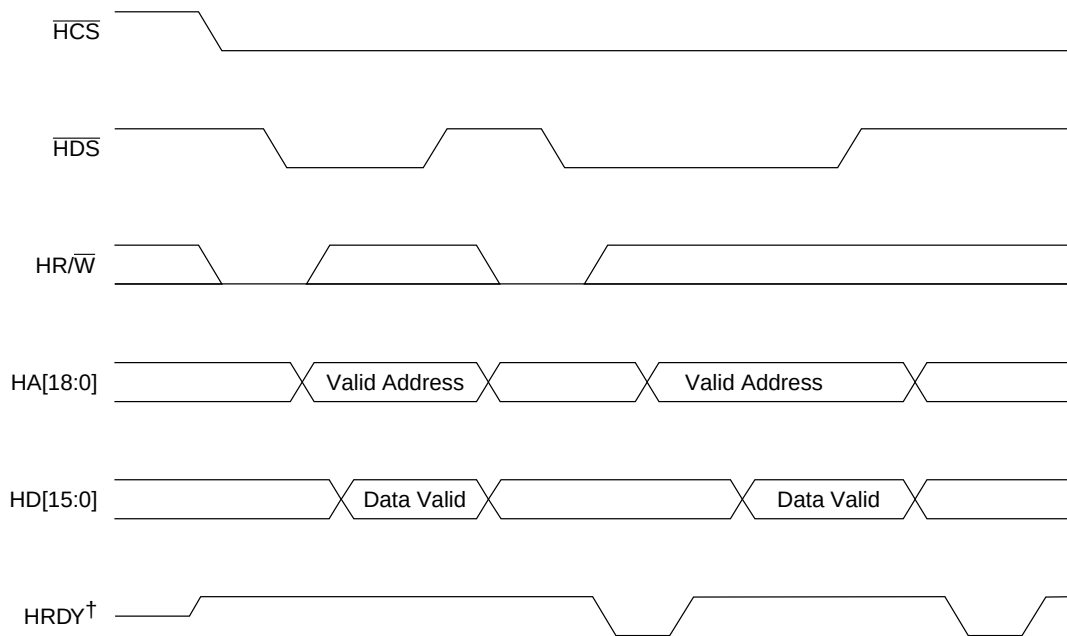
Figure 13-92 shows read timing required by the EHPI in non-multiplexed mode.



[†] While $\overline{HCS}$ is not selected, $HRDY$ is in Hi-Z state.

Figure 13-92. EHPI Non-Multiplexed Read Timings

Figure 13-93 shows write timing required by the EHPI in non-multiplexed mode.



[†] While $\overline{HCS}$ is not selected, $HRDY$ is in Hi-Z state.

Figure 13-93. EHPI Non-Multiplexed Write Timings

13.5.6.3.1 Expansion to Multiple DSPs

The connection shown above can be adapted easily to interface to multiple DSPs instead of only one. Each DSP host port needs to receive its own chip select and interrupt signals, and HRDY must be connected differently, depending on which DSP is used. All other signals can be connected to all host ports in parallel.

HRDY signals can be bussed together for certain DSPs (such as, TMS320VC5510); for others (such as, TMS320VC5509), an external multiplexor must be used to decide which HRDY signal is routed to the local bus. This multiplexor can be controlled by the respective chip selects.

For a larger number of DSPs, this scheme must be extended with external logic, which uses additional address signals to generate multiple DSP HCSs for one local bus chip select and to multiplex the HINT and HRDY signals. The flexibility of the UPM allows for additional delay for that external logic.

Chapter 14

Three-Speed Ethernet Controller

14.1 Introduction

The Ethernet IEEE 802.3 protocol is a widely used LAN 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 standard implemented TSEC is Gigabit Ethernet, which builds on top of the Ethernet protocol, but increases speed tenfold over 10/100 Ethernet to 1000 Mbps, or one gigabits per second (Gbps).

Gigabit Ethernet looks identical to Ethernet from the data link layer upward but uses the ANSI X3T11 FiberChannel FC-0 (interface and media) and FC-1 (encode/decode) for the PHY Layer. In this manner, the standard takes advantage of the existing high-speed physical interface technology of FiberChannel while maintaining the IEEE 802.3 Ethernet frame format, backward compatibility for installed media, and the use of full- or half-duplex CSMA/CD.

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 14-1 depicts the typical Ethernet protocol stack and the relationship to the OSI model.

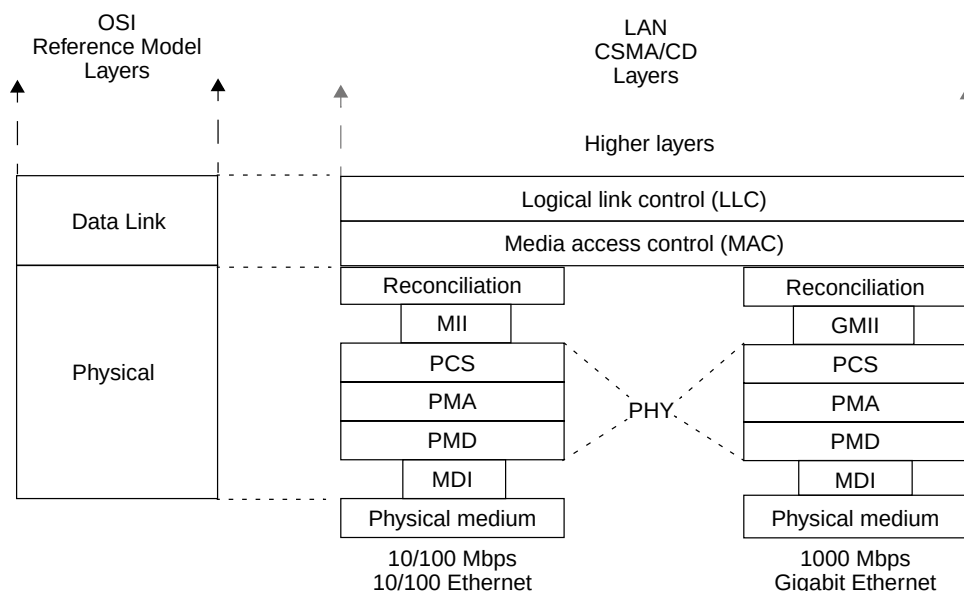


Figure 14-1. Ethernet Protocol in Relation to the OSI Protocol Stack

Gigabit 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.
- **GMII (gigabit media-independent interface) sublayer**—The GMII sublayer provides a standard interface between the MAC layer and the physical layer for 1000 Mbps operation. 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 coding and encoding data stream to and from the MAC sublayer. Medium (1000BASEX) 8B/10B coding is used for fiber. Medium (1000BASET) 8B1Q coding is used for unshielded twisted pair (UTP).
- **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 and vice versa (SERDES). Synchronization is also performed for proper data decoding in this sublayer. For fiber medium (1000BASEX) the interface on the

PHY side is a one-bit 1250 MHz signal, while on the other side is a ten-bit interface (TBI) at 125 MHz. The TBI is an alternative to the GMII interface. If the TBI is used the gigabit Ethernet controller must be capable of performing the PCS function. For UTP medium the PHY side interface consist of four pair of 62.5 MHz PAM5 encoded signals, while the other side provides the 1250 Mbps input to a 8B1Q4 PCS.

- 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.

Figure 14-2 describes the different physical interface standards.

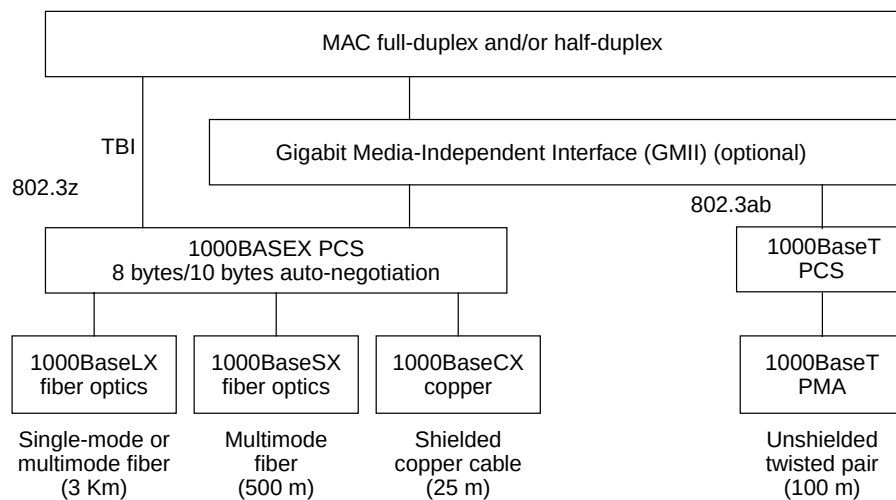


Figure 14-2. 802.3z and 802.3ab Physical Standards

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 14-3. The term “packet” is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

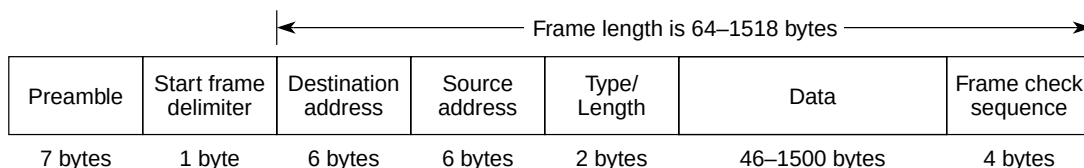


Figure 14-3. Ethernet/IEEE 802.3 Frame Structure

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 1,536 (0x5DD through 0x600) 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 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 time 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 14-4 provides additional details of the Ethernet/IEEE 802.3 frame structure.

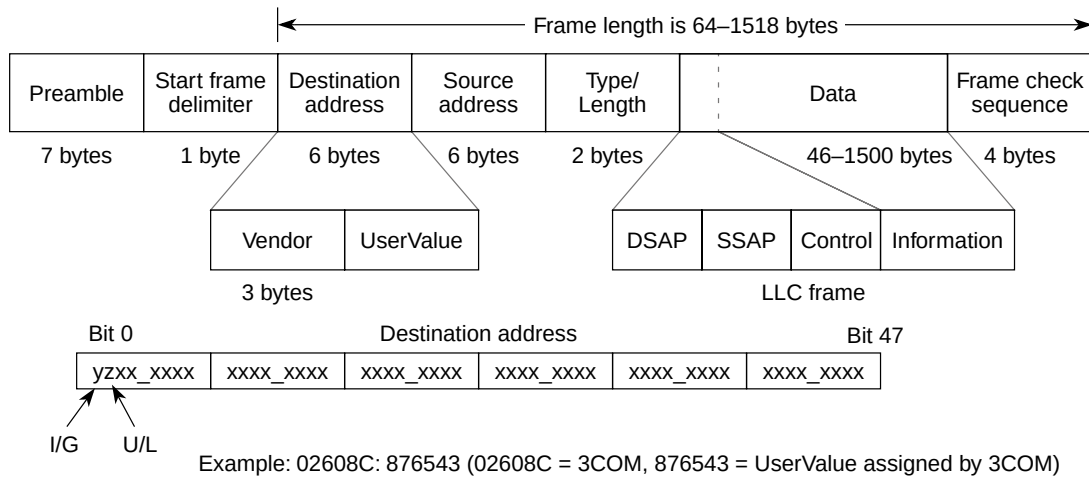


Figure 14-4. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 14-4, 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 14-4 (02608C:876543) which would normally be written as

0000 0010 0110 0000 1000 1100 1000 0111 0110 0101 0100 0011

becomes

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 via 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.

Introduction

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 = 00-00-00, or they become a SNAP protocol identifier if SNAP OUI is non zero.

The three-speed Ethernet controller (TSEC) allows the flexibility to accelerate the identification and retrieval of all the standard and non-standard protocols mentioned above. Figure 14-5 depicts the block diagram of the TSEC.

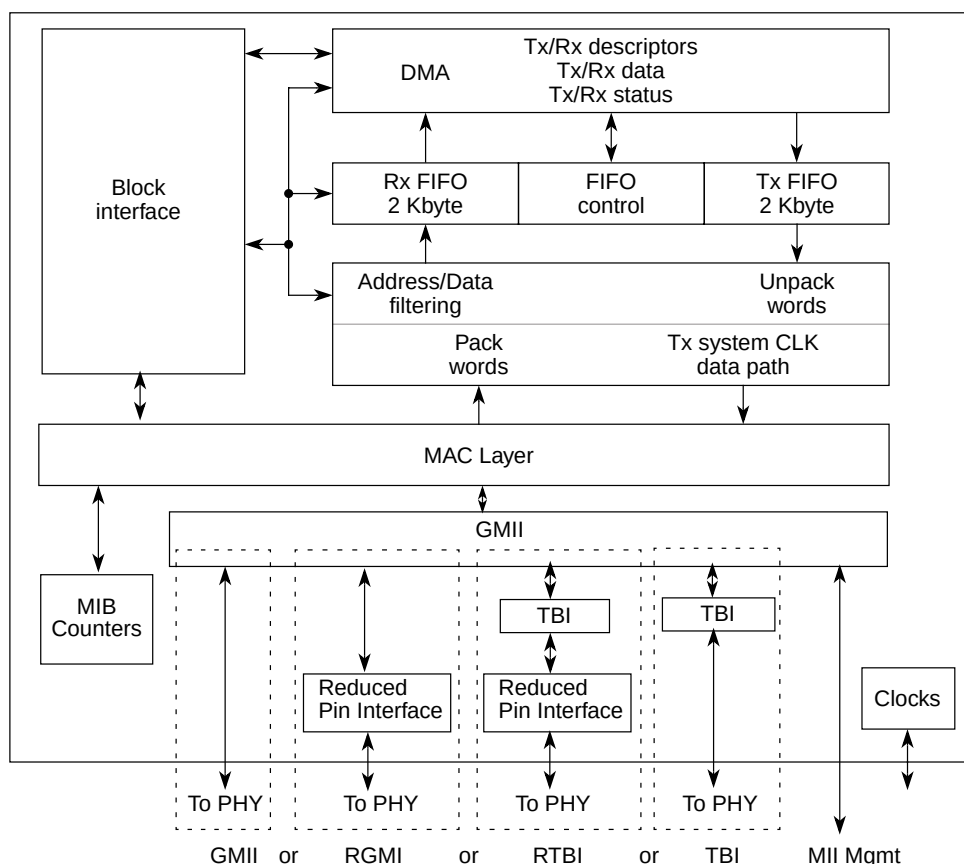


Figure 14-5. TSEC Block Diagram

14.1.1 Three-Speed Ethernet Controller Overview

The TSEC is designed to support 10, 100 and 1000 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
- Ten-bit interface (TBI)
- Register-based statistical module that supports management information base (MIB) remote monitoring (RMON)

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 TSEC is designed for single MAC applications. The TSEC supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver:

- MII interface running at 10/100 Mbps
- GMII interface running at 1000 Mbps
- TBI interface that can be connected to a SERDES device for fibre channel applications.
- Reduced pin count versions of the GMII (RGMII) and ten-bit (RTBI) interfaces

While most of this document refers to the non-reduced pin count interfaces, it must be understood that these references also apply to the reduced pin count interfaces.

The TSEC software programming model is similar to the MPC8260 (PowerQUICC II) device. Hence, it enables Motorola customers to leverage already implemented Ethernet drivers, reducing the software development cycle.

14.2 Features

The MPC8450 TSEC includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant
- Support for different Ethernet physical interfaces:
 - 10/100/1000 Mbps IEEE 802.3 GMII
 - 10/100 Mbps IEEE 802.3 MII
 - 10 Mbps IEEE 802.3 MII
 - 1000 Mbps IEEE 802.3z TBI
 - 10/100 Mbps RGMII
 - 1000 Mbps full-duplex RGMII

Modes of Operation

- 1000 Mbps RTBI
- Full and half-duplex 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)
- Support for out-of-sequence transmit queue (for initiating flow-control)
- Programmable maximum frame length supports jumbo frames (up to 9.6k) and IEEE 802.1 virtual local area network (VLAN) tags and priority
- Retransmission from transmit FIFO following a collision
- Support CRC generation and verification of inbound/outbound packets
- Address recognition
 - Each exact match can be programmed to be accepted or rejected
 - 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 processor's L2 cache to reduce access latency
- RMON statistics support

14.3 Modes of Operation

The TSEC's primary 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 (only 10 and 100 Mbps operation; MACCFG2 register's full-duplex bit is cleared), the MAC complies with the IEEE CSMA/CD access method.

If configured in full-duplex mode (10/100/1000 Mbps operation; 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 Mbps 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 TSEC_n_TX_CLK and TSEC_n_RX_CLK pins, which are driven by the transceiver. The transceiver either auto-negotiates the speed or it may be controlled by software via the serial management interface (EC_MDC/EC_MDIO pins) to the transceiver.

- 1000 Mbps GMII and TBI interface operation

The MAC-PHY interface operates in GMII mode by configuring bits MACCFG2[22:23] = 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 ten-bit interface which contains PCS functions (ten-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 TSEC muxes and decodes the input pins and provides the MAC with the expected interface.

TSEC provides the TSEC_n_GTX_CLK to the PHY in either GMII or TBI mode of operation.

- 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 14.5.3.8.1, “Individual Address Registers 0–7 (IADDR_n),” Section 14.5.3.8.2, “Group Address Registers 0–7 (GADDR_n),” Section 14.5.3.4.1, “Receive Control Register (RCTRL),” and Section 14.6.2.6, “Frame Recognition.”

- RMON support

See Section 14.6.2.5, “RMON Support.”

- Internal loop back

Internal loop back mode is selected via the loop back bit in the MACCFG1 register. See Section 14.6.2.10, “Internal and External Loop Back,” for details.

14.4 External Signals Description

This section defines the TSEC-to-chip 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). Notice that except for external physical interfaces the buses and registers follow a big-endian format.

The TSEC network interface supports multiple options:

- The MII option requires 18 I/O pins (including the EC_MDIO and EC_MDC MII management interface) and supports both a data and a management interface to the

External Signals Description

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 pins with the GMII interface signals.
- Finally, RGMII and RTBI options are reduced pin implementations of the GMII and TBI interfaces.

Table 14-1 lists the network interface signals.

Table 14-1. Signal Properties

Signal Name	Function	Reset State	Pull up
TSECn_COL	MII—collision, input		
TSECn_CRS	MII—carrier sense, input TBI— signal detect, input		
TSECn_GTX_CLK	TBI,RGMII, GMII— transmit clock, output		
EC_GTX_CLK125	Oscillator source for GMII, TBI, RGMII, RTBI transmit clock, input		
EC_MDC	TBI,RGMII, GMII, MII—management clock, output	0	
EC_MDIO	TBI, RGMII, GMII, MII—management data, bidirectional	Hi-Z (input)	
TSECn_RX_CLK	GMII, MII, RGMII—receive clock, input TBI—PMA receive clock 0, input		
TSECn_RX_DV	GMII, MII, RGMII—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		
TSECn_RXD[7:4]	GMII—receive data bit 7:4 input TBI—RCG bit 7:4, input		
TSECn_RXD[3:0]	GMII, MII—receive data bit 3:0 input TBI—RGC bit 3:0, input RGMII (RX_CLK rising) —receive data bit 3:0, input RGMII (RX_CLK falling)—receive data bit 7:4, input		
TSECn_RX_ER	GMII, MII—receive error, input TBI—RGC bit 9, input	0	
TSECn_TX_CLK	MII—transmit clock, input TBI—PMA receive clock 1, input		
TSECn_TXD[7:4]	GMII—transmit data bit 7:4, output TBI—transmit code group (TCG) bit 7:4, output		
TSECn_TXD[3:0]	GMII, MII—transmit data bit 3:0, output TBI—TCG bit 3:0, output RGMII (TX_CLK rising)—transmit data bit 3:0, output RGMII (TX_CLK falling)—transmit data bit 7:4, output		

Table 14-1. Signal Properties (continued)

Signal Name	Function	Reset State	Pull up
TSEc _n _TX_EN	GMII, MII—transmit data valid, output TBI—TCG bit 8, output RGMII (TX_CLK rising)—transmit data enabled, output RGMII (TX_CLK falling)—transmit error, output	0	
TSEc _n _TX_ER	GMII, MII—transmit error, output TBI—TCG bit 9, output	0	

14.4.1 Detailed Signal Descriptions

Below is a description of the TSEC 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. All other modes follow the IEEE 802.3 standard, 2000 Edition.

14.4.1.1 TSEc_n_COL—Collision (Input)

In GMII or MII mode this signal is asserted upon detection of a collision, and remains asserted while the collision persists. The behavior of this signal is not specified while in full-duplex mode. This signal is not required to transition synchronously with TSEc_n_TX_CLK or TSEc_n_RX_CLK.

14.4.1.2 TSEc_n_CRS—Carrier Sense (Input)

In MII or GMII mode TSEc_n_TX_CLK is asserted while the transmit or receive medium is not idle. In the event of a collision, TSEc_n_CRS remains asserted for the duration of the collision. This signal is not required to transition synchronously with TSEc_n_TX_CLK or TSEc_n_RX_CLK.

In TBI mode this pin can be used as SDET (signal detect), an optional signal that some PHYs generate in TBI mode.

14.4.1.3 TSEc_n_GTX_CLK—Gigabit Transmit Clock (Output)

In GMII or TBI mode TSEc_n_GTX_CLK is a 125 MHz clock which provides a timing reference for TX_EN, TXD, and TX_ER.

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 is an output from the TSEC into the PHY.

14.4.1.4 EC_GTX_CLK125—Gigabit Transmit 125 MHz Source (Input)

In GMII, RGMII, RTBI or TBI mode EC_GTX_CLK125 is a 125 MHz input into the TSEC and is used to generate all 125-MHz related signals and clocks. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY.

14.4.1.5 EC_MDC—Management Data Clock (Output)

In GMII, MII, RGMII, RTBI or TBI mode this signal is a clock (typically 2.5 MHz) supplied by the MAC. (IEEE set minimum period of 400 ns or 2.5 MHz, but part 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].

14.4.1.6 EC_MDIO—Management Data Input/Output, BIDI (Input/Output)

In GMII, MII, RGMII, RTBI or TBI mode EC_MDIO is a bi-directional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles.

14.4.1.7 TSEC_n_RX_CLK—Receive Clock (Input)

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, or it is a 125 MHz receive clock.

14.4.1.8 TSEC_n_RX_DV—Receive Data Valid (Input)

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 represent the 10-bit encoded symbol of GMII receive signals.

14.4.1.9 TSEC_n_RXD[7:0]—Receive Data (Input)

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 TSECN_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 mode TSECN_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSECN_RX_DV is asserted. A completely formed SFD must be passed across the MII. While TSECN_RX_DV is not asserted, TSECN_RXD has no meaning.

In RGMII or RTBI mode TSECN_RXD[3:0] are received on the rising edge of TSECN_RX_CLK and TSECN_RXD[7:4] are received on the falling edge of TSECN_RX_CLK.

In TBI mode TSECN_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals.

14.4.1.10 TSECN_RX_ER—Receive Error (Input)

In GMII or MII mode if TSECN_RX_ER and TSECN_RX_DV are asserted, the PHY has detected an error in the current frame.

In TBI mode TSECN_RX_ER represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.

14.4.1.11 TSECN_TX_CLK—Transmit Clock (Input)

In MII mode TSECN_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSECN_TX_EN, TSECN_TXD, and TSECN_TX_ER signals.

In GMII mode TSECN_TX_CLK 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; TSECN_GTX_CLK (125 MHz) becomes the timing reference. The TSECN_GTX_CLK is generated in the TSEC and provided to the PHY and the MAC. The TSECN_TX_CLK is generated in the PHY and provided to the MAC.

In TBI mode this signal is PMA receive clock 1 62.5 MHz, split phase with PMA_RX_CLK1, and is supplied by the SERDES.

14.4.1.12 TSECN_TXD[7:0]—Transmit Data (Output)

In GMII mode TSECN_TXD[7:4], together with TSECN_TXD[3:0], represent one complete octet of data to be sent from the MAC to the PHY when TSECN_TX_DV is asserted and has no meaning while TSECN_TX_EN is negated.

In TBI mode TSECN_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 TSECN_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSECN_TX_EN is asserted and have no meaning while TSECN_TX_EN is negated.

In RGMII or RTBI mode TSECN_TXD[3:0] are transmitted on the rising edge of TSECN_TX_CLK, and TSECN_TXD[7:4] are transmitted on the falling edge of TSECN_TX_CLK.

In TBI mode TSECN_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol.

14.4.1.13 TSECN_TX_EN—Transmit Data Valid (Output)

In GMII or MII mode if TSECN_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII or the MII.

In RGMII mode TSECN_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this pin on rising and falling edges of the TSECN_TX_CLK, respectively.

In TBI mode TSECN_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol.

14.4.1.14 TSECN_TX_ER—Transmit Error (Output)

In GMII or MII mode, assertion of TSECN_TX_ER for one or more clock cycles while TSECN_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSECN_TX_ER has no effect while operating at 10Mbps or while TSECN_TX_EN is negated. This signal transitions synchronously with respect to TSECN_TX_CLK.

In TBI mode TSECN_TX_ER represents TCG[9]. Together, with TCG[8:0], they represent the 10-bit encoded symbol.

14.5 Memory Map/Register Definition

The TSEC uses a software model similar to that employed by the Fast Ethernet function supported on the Motorola MPC8260 CPM FCC and in the FEC of the MPC860T.

The TSEC device is programmed by a combination of control/status registers (CSR) 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 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 14.6.3, “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 as the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 14.5.4, “Ten-Bit Interface (TBI).”

14.5.1 Top-Level Module Memory Map

The TSEC 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
- Event/statistic counters held in the MIB block
- Hash function registers

Table 14-2 defines the top-level memory map.

Table 14-2. Module Memory Map Summary

Address	Function
000–0FF	TSEC general control/status registers
100–2FF	TSEC transmit control/status registers
300–4FF	TSEC receive control/status registers
500–5FF	MAC registers
600–7FF	RMON MIB registers
800–8FF	HASH function registers
900–FFF	Future expansion space

14.5.2 Detailed Memory Map—Control/Status Registers

Table 14-2 lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for both TSECs. That is, TSEC1 starts at 0x2_4000 address offset and TSEC2 starts at 0x2_5000 address offset. The registers for TSEC1 are listed in Table 14-5, but the registers for TSEC2 are not. Note in Table 14-5 that the registers are the same for TSEC2 except that the offset changes from 0x2_4nnn to 0x2_5nnn.

Table 14-3. Module Memory Map

Offset	Name	Access ¹	Reset	Section/Page
TSEC1 General Control and Status Registers				
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	Interrupt event register (IEVENT)	R/W	0x0000_0000	14.5.3.1.1/14-21
0x2_4014	Interrupt mask register (IMASK)	R/W	0x0000_0000	14.5.3.1.2/14-24
0x2_4018	Error Disabled register (EDIS)	R/W	0x0000_0000	14.5.3.1.3/14-25
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	Ethernet control register (ECNTRL)	R/W	0x0000_0000	14.5.3.1.4/14-26
0x2_4024	Minimum frame length register (MINFLR)	R/W	0x0000_0040	14.5.3.1.5/14-28
0x2_4028	Pause time value register (PTV)	R/W	0x0000_0000	14.5.3.1.6/14-28
0x2_402C	DMA control register (DMACTRL)	R/W	0x0000_0000	14.5.3.1.7/14-29
0x2_4030	TBI PHY address register (TBIPA)	R/W	0x0000_0000	14.5.3.1.8/14-31
0x2_4034–0x2_403C	Reserved	R	0x0000_0000	—
0x2_4040–0x2_4088	Reserved	R	0x0000_0000	—
TSEC FIFO Control and Status Registers				14.5.3.2/14-31
0x2_408C	FIFO transmit threshold register (FIFO_TX_THR)	R/W	0x0000_0100	14.5.3.2.1/14-32
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO transmit starve register (FIFO_TX_STARVE)	R/W	0x0000_0080	14.5.3.2.2/14-33
0x2_409C	FIFO transmit starve shutoff register (FIFO_TX_STARVE_SHUTOFF)	R/W	0x0000_0100	14.5.3.2.3/14-33
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	Transmit control register (TCTRL)	R/W	0x0000_0000	14.5.3.3.1/14-34
0x2_4104	Transmit status register (TSTAT)	R/W	0x0000_0000	14.5.3.3.2/14-35
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TxBD data length (TBDLEN)	R/W	0x0000_0000	14.5.3.3.3/14-36
0x2_4110–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	Current TxBD pointer (CTBPTR)	R/W	0x0000_0000	14.5.3.3.4/14-36
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TxBD pointer (TBPTR)	R/W	0x0000_0000	13.5.3.3.7/13-65

Table 14-3. Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_4188– 0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TxBD base address (TBASE)	R/W	0x0000_0000	14.5.3.3.5/14-37
0x2_4208– 0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	Out-of-sequence TxBD register (OSTBD)	R/W	0x0000_0000	14.5.3.3.6/14-38
0x2_42B4	Out-of-sequence Tx data buffer pointer register (OSTBDP)	R/W	0x0000_0000	14.5.3.3.7/14-40
0x2_42B8– 0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	Receive control register (RCTRL)	R/W	0x0000_0000	14.5.3.4.1/14-41
0x2_4304	Receive status register (RSTAT)	R/W	0x0000_0000	14.5.3.4.2/14-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RxBD data length (RBDLEN)	R/W	0x0000_0000	14.5.3.4.3/14-42
0x2_4310– 0x2_431C	Reserved	R	0x0000_0000	—
0x2_4320	Reserved	R/W	0x0000_0000	—
0x2_4324	Current RxBD pointer (CRBPTR)	R/W	0x0000_0000	14.5.3.4.4/14-43
0x2_4328– 0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	Maximum receive buffer length register (MRBLR)	R/W	0x0000_0000	14.5.3.4.5/14-43
0x2_4344– 0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RxBD pointer (RBPTR)	R/W	0x0000_0000	14.5.3.4.6/14-44
0x2_4388– 0x2_4400	Reserved	R/W	0x0000_0000	—
0x2_4404	RxBD base address (RBASE)	R/W	0x0000_0000	14.5.3.4.7/14-45
0x2_4408– 0x2_44FC	Reserved	R/W	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MAC configuration #1 (MACCFG1)	R/W, R	0x0000_0000	14.5.3.6.1/14-48
0x2_4504	MAC configuration #2 (MACCFG2)	R/W	0x0000_7000	14.5.3.6.2/14-50
0x2_4508	IPGIFGI	R/W	0x4060_5060	14.5.3.6.3/14-51
0x2_450C	Half-duplex (HAFDUP)	R/W	0x00A1_F037	14.5.3.6.4/14-52
0x2_4510	Maximum frame (MAXFRM)	R/W	0x0000_0600	14.5.3.6.5/14-54
0x2_4514	Reserved	R	0x0000_0000	—

Table 14-3. Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_4518	Reserved	R	0x0000_0000	—
0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MII Mgmt: configuration (MIIMCFG)	R/W	0x0000_0000	14.5.3.6.6/14-54
0x2_4524	MII Mgmt: command (MIICOM)	R/W	0x0000_0000	14.5.3.6.7/14-55
0x2_4528	MII Mgmt: address (MIIMADD)	R/W	0x0000_0000	14.5.3.6.8/14-56
0x2_452C	MII Mgmt: control (MIIMCON)	WO	0x0000_0000	14.5.3.6.9/14-57
0x2_4530	MII Mgmt: status (MIIMSTAT)	R	0x0000_0000	14.5.3.6.10/14-57
0x2_4534	MII Mgmt: indicators (MIIMIND)	R	0x0000_0000	14.5.3.6.11/14-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	Interface status (IFSTAT)	R,R/LH	0x0000_0001	14.5.3.6.12/14-58
0x2_4540	Station Address, part 1 (MACSTNADDR1)	R/W	0x0000_0000	14.5.3.6.13/14-59
0x2_4544	Station Address, part 2 (MACSTNADDR2)	R/W	0x0000_0000	14.5.3.6.14/14-60
0x2_4548– 0x2_45FC	Reserved	R	0x0000_0000	—
0x2_4600– 0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64 transmit and receive 64-byte frame counter	R/W	0x0000_0000	14.5.3.7.1/14-61
0x2_4684	TR127 transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	14.5.3.7.2/14-62
0x2_4688	TR255 transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	14.5.3.7.3/14-62
0x2_468C	TR511 transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	14.5.3.7.4/14-63
0x2_4690	TR1K transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	14.5.3.7.5/14-64
0x2_4694	TRMAX transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	14.5.3.7.6/14-64
0x2_4698	TRMGV transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	14.5.3.7.7/14-65
TSEC1 Receive Counters				
0x2_469C	RBYT receive byte counter	R/W	0x0000_0000	14.5.3.7.8/14-66
0x2_46A0	RPKT receive packet counter	R/W	0x0000_0000	14.5.3.7.9/14-66
0x2_46A4	RFCS receive FCS error counter	R/W	0x0000_0000	14.5.3.7.10/14-67
0x2_46A8	RMCA receive multicast packet counter	R/W	0x0000_0000	14.5.3.7.11/14-68

Table 14-3. Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_46AC	RBCA receive broadcast packet counter	R/W	0x0000_0000	14.5.3.7.12/14-68
0x2_46B0	RXCF receive control frame packet counter	R/W	0x0000_0000	14.5.3.7.13/14-69
0x2_46B4	RXPf receive PAUSE frame packet counter	R/W	0x0000_0000	14.5.3.7.14/14-70
0x2_46B8	RXUO receive unknown OP code counter	R/W	0x0000_0000	14.5.3.7.15/14-70
0x2_46BC	RALN receive alignment error counter	R/W	0x0000_0000	14.5.3.7.16/14-71
0x2_46C0	RFLR receive frame length error counter	R/W	0x0000_0000	14.5.3.7.17/14-72
0x2_46C4	RCDE receive code error counter	R/W	0x0000_0000	14.5.3.7.18/14-72
0x2_46C8	RCSE receive carrier sense error counter	R/W	0x0000_0000	14.5.3.7.19/14-73
0x2_46CC	RUND receive undersize packet counter	R/W	0x0000_0000	14.5.3.7.20/14-74
0x2_46D0	ROVR receive oversize packet counter	R/W	0x0000_0000	14.5.3.7.21/14-74
0x2_46D4	RFRG receive fragments counter	R/W	0x0000_0000	14.5.3.7.22/14-75
0x2_46D8	RJBR receive jabber counter	R/W	0x0000_0000	14.5.3.7.23/14-76
0x2_46DC	RDRP receive drop	R/W	0x0000_0000	14.5.3.7.24/14-76
Transmit Counters				
0x2_46E0	TBYT transmit byte counter	R/W	0x0000_0000	14.5.3.7.25/14-77
0x2_46E4	TPKT transmit packet counter	R/W	0x0000_0000	14.5.3.7.26/14-78
0x2_46E8	TMCA transmit multicast packet counter	R/W	0x0000_0000	14.5.3.7.27/14-78
0x2_46EC	TBCA transmit broadcast packet counter	R/W	0x0000_0000	14.5.3.7.28/14-79
0x2_46F0	TXPF transmit PAUSE control frame counter	R/W	0x0000_0000	14.5.3.7.29/14-80
0x2_46F4	TDFR transmit deferral packet counter	R/W	0x0000_0000	14.5.3.7.30/14-80
0x2_46F8	TEDF transmit excessive deferral packet counter	R/W	0x0000_0000	14.5.3.7.31/14-81
0x2_46FC	TSCL transmit single collision packet counter	R/W	0x0000_0000	14.5.3.7.32/14-82
0x2_4700	TMCL transmit multiple collision packet counter	R/W	0x0000_0000	14.5.3.7.33/14-82
0x2_4704	TLCL transmit late collision packet counter	R/W	0x0000_0000	14.5.3.7.34/14-83
0x2_4708	TXCL transmit excessive collision packet counter	R/W	0x0000_0000	14.5.3.7.35/14-84
0x2_470C	TNCL transmit total collision counter	R/W	0x0000_0000	14.5.3.7.36/14-84
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP transmit drop frame counter	R/W	0x0000_0000	14.5.3.7.37/14-85
0x2_4718	TJBR transmit jabber frame counter	R/W	0x0000_0000	14.5.3.7.38/14-86
0x2_471C	TFCS transmit FCS error counter	R/W	0x0000_0000	14.5.3.7.39/14-86
0x2_4720	TXCF transmit control frame counter	R/W	0x0000_0000	14.5.3.7.40/14-87
0x2_4724	TOVR transmit oversize frame counter	R/W	0x0000_0000	14.5.3.7.41/14-88
0x2_4728	TUND transmit undersize frame counter	R/W	0x0000_0000	14.5.3.7.42/14-88

Table 14-3. Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_472C	TFRG transmit fragments frame counter	R/W	0x0000_0000	14.5.3.7.43/14-89
General Registers				
0x2_4730	CAR1 carry register one register ²	R	0x0000_0000	14.5.3.7.44/14-90
0x2_4734	CAR2 carry register two register ²	R	0x0000_0000	14.5.3.7.45/14-91
0x2_4738	CAM1 carry register one mask register	R/W	0xFE01_FFFF	14.5.3.7.46/14-92
0x2_473C	CAM2 carry register two mask register	R/W	0x000F_FFFF	14.5.3.7.47/14-93
0x2_4740– 0x2_47FC	Reserved	R	0x0000_0000	—
Hash Function Registers				
0x2_4800	Individual address register 0 (IADDR0)	R/W	0x0000_0000	14.5.3.8.1/14-95
0x2_4804	Individual address register 1 (IADDR1)	R/W	0x0000_0000	
0x2_4808	Individual address register 2 (IADDR2)	R/W	0x0000_0000	
0x2_480C	Individual address register 3 (IADDR3)	R/W	0x0000_0000	
0x2_4810	Individual address register 4 (IADDR4)	R/W	0x0000_0000	
0x2_4814	Individual address register 5 (IADDR5)	R/W	0x0000_0000	
0x2_4818	Individual address register 6 (IADDR6)	R/W	0x0000_0000	
0x2_481C	Individual address register 7 (IADDR7)	R/W	0x0000_0000	
0x2_4820– 0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	Group address register 0 (GADDR0)	R/W	0x0000_0000	14.5.3.8.2/14-95
0x2_4884	Group address register 1 (GADDR1)	R/W	0x0000_0000	
0x2_4888	Group address register 2 (GADDR2)	R/W	0x0000_0000	
0x2_488C	Group address register 3 (GADDR3)	R/W	0x0000_0000	
0x2_4890	Group address register 4 (GADDR4)	R/W	0x0000_0000	
0x2_4894	Group address register 5 (GADDR5)	R/W	0x0000_0000	
0x2_4898	Group address register 6 (GADDR6)	R/W	0x0000_0000	
0x2_489C	Group address register 7 (GADDR7)	R/W	0x0000_0000	
0x2_48A0– 0x2_4BF4	Reserved	R	0x0000_0000	—
0x2_4BF8	Attribute register (ATTR)	R	0x0000_0000	14.5.3.8.3/14-96
0x2_4BFC	Attribute EL & EI register (ATTRELI)	R	0x0000_0000	14.5.3.8.4/14-97
TSEC1 Future Expansion Space				

Table 14-3. Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_4C00– 0x2_4FFC	Reserved	R	0x0000_0000	—
0x2_5000– 0x2_5FFF	TSEC2 REGISTERS ³			14.5.3/14-21

¹ R = means read-only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing,

² Cleared on read

³ TSEC2 has the same memory-mapped registers that are described for TSEC1 from 0x2_4000 to 0x2_4FFF except the offsets are from 0x2_5000 to 0x2_5FFF.

14.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the TSEC registers. Because all of the TSEC registers are 32 bits wide, only 32-bit register accesses are supported.

14.5.3.1 TSEC 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.

14.5.3.1.1 Interrupt Event Register (IEVENT)

If an event occurs that sets a bit in the interrupt event (IEVENT) register, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. The bit in the interrupt event 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 MII

Interrupts resulting from errors/problems detected in the network or transceiver are:

- BABR, BABT, LATE_COL and COL_RETRY_LIM

Interrupts resulting from internal errors are:

- EBERR, XFIFO_UN 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 via the MIB counters.

Figure 14-8 describes the definition for the IEVENT register.

Memory Map/Register Definition

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	0	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	0	LC	CRL	XFUN
W																
Reset	0000_0000_0000_0000															
	16	17					22	23	24	25						31
R	RXB0	0	0	0	0	0	0	GRSC	RXF0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1: 0x2_4010; TSEC2:0x2_5010															

Figure 14-6. IEVENT Register Definition

Table 14-4 describes the fields of the IEVENT register.

Table 14-4. 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[HFE] is set. 0 Excessive frame not received. 1 Excessive frame received.
1	RXC	Receive control interrupt. A control frame was received while MACCFG1[RFCE] 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	Ethernet bus error. This bit indicates that a system bus error 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). If the EBERR is set while receiving a frame, the DMA discards the frame. 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. 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 graceful stop, which was initiated by setting TCTRL[TFC_PAUSE], is now complete. 0 No graceful stop interrupt. 1 Graceful stop requested.

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
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[HFE] 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.
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 TSEC. 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. 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. 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.
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	RXB0	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

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
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	RXF0	Receive frame interrupt. These bits indicate that a frame was received that the last receive buffer descriptor (RxBD) in that 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

14.5.3.1.2 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 remains asserted until the IEVENT bit is cleared either by writing a 1 to it, or by writing a 0 to the corresponding IMASK bit.

Figure 14-7 describes the definition for the IMASK register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BRE	RXC	BSY	EBE	0	MS	GTS	BTE	TXCE	TXEE	TXBE	TXFE	0	LCEN	CRLEN	XFU
W	N	EN	EN	RREN		ROEN	CEN	N	N	N	N	N				NEN
Reset	0000_0000_0000_0000															
	16	17					22	23	24	25						31
R	RXB	0	0	0	0	0	GRS	RXFE	0	0	0	0	0	0	0	
W	EN0						CEN	N0								
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4014; TSEC2:0x2_5014															

Figure 14-7. IMASK Register Definition

Table 14-5 describes the fields of the IMASK register.

Table 14-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable

Table 14-5. IMASK Field Descriptions (continued)

Bits	Name	Description
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	CRLEN	Collision retry limit enable
15	XFUNEN	Transmit fifo underrun enable
16	RXBEN0	Receive buffer interrupt enable
17–22	—	Reserved
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN0	Receive frame interrupt enable
25–31	—	Reserved

14.5.3.1.3 Error Disabled Register (EDIS)

Figure 14-8 describes the definition for the EDIS register. The error disabled register allows the user to disable an error interruption.

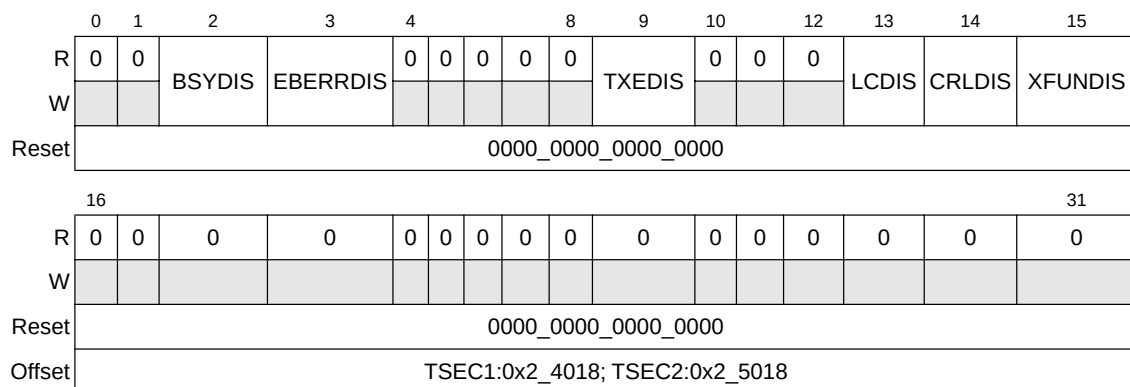
**Figure 14-8. EDIS Register Definition**

Table 14-6 describes the fields of the EDIS register.

Table 14-6. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable. 0 Allow TSEC 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 TSEC 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.
4–8	—	Reserved
9	TXEDIS	Transmit error disable. 0 Allow TSEC 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 TSEC 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	CRLDIS	Collision retry limit disable. 0 Allow TSEC 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 TSEC 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.
16–31	—	Reserved

14.5.3.1.4 Ethernet Control Register (ECNTRL)

ECNTRL is a register writable by the user to reset, configure, and initialize the TSEC. The ECNTRL[TBIM] and ECNTRL[RPM] bits are read-only. Figure 14-9 describes the definition for the ECNTRL register.

	0														15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0	0	0	0	0	0	TBIM	RPM		0	0	0
W		CLRCNT	AUTOZ	STEN									R100M			
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4020; TSEC2:0x2_5020															

Figure 14-9. ECNTRL Register Definition

Table 14-7 describes the fields of the ECNTRL register.

Table 14-7. ECNTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	CLRCNT	Clear all statistics counters 0 Allow MSTAT counters to continue to increment. 1 Reset all MSTAT counters. This bit is self-resetting.
18	AUTOZ	Automatically zero addressed statistical counter values, input to MSTAT module. 0 The user must write the addressed counter zero after a host read. 1 The addressed counter value is zeroed on 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	Statistics enabled, input to MSTAT module. 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.
20–25	—	Reserved
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 using the TSEC _n _GTX_CLK pin. 0 GMII or MII mode interface 1 TBI mode interface
27	RPM	Reduced pin mode. If this bit is set, a reduced pin interface is expected. ECNTRL[TBI Mode] selects GMII or TBI. If RPM is selected, the MAC must be in GMII or TBI mode (RMII is not supported). This register can be pin configured at reset to 0 or 1 with the EC_MDC pin. 0 GMII or TBI in non-reduced pin mode configuration 1 RGMII or RTBI reduced pin mode
28	R100M	RGMII 100 mode. This bit is ignored unless RPM is set and MACCFG2[I/F Mode] is assigned to 10/100 (01). If this bit is set, the TSEC interface is in 100 Mbps speed. 0 RGMII is in 10 Mbps mode 1 RGMII is in 100 Mbps mode
29–31	—	Reserved

14.5.3.1.5 Minimum Frame Length Register (MINFLR)

MINFLR is written by the user. It tells TSEC the smallest packet to accept and place in a receive buffer pointed to by the RxBD. Figure 14-10 describes the definition for the MINFLR register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	31					
R	0	0	0	0	0	0	0	0	0	MINFLR						
W																
Reset	0000_0000_0100_0000															
Offset	TSEC1:0x2_4024; TSEC2:0x2_502															

Figure 14-10. MINFLR Register Definition

Table 14-8 describes the fields of the MINFLR register.

Table 14-8. 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. Unlike the MPC8260, in which PADs are added to make the transmit frame equal to MINFLR bytes, if padding is requested, TSEC always PADS transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

14.5.3.1.6 Pause Time Value Register (PTV)

PTV is a 32-bit register written by the user to store the pause duration used when the TSEC initiates a pause frame via TCTRL[TFC_PAUSE]. The low-order 16 bits (PT) represent the the pause time and the high-order 16 bits (PTE) represent the extended pause time. 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 14.6.2.7, “Flow Control,” on page 14-126 for additional details. Figure 14-11 describes the definition for the PTV register.

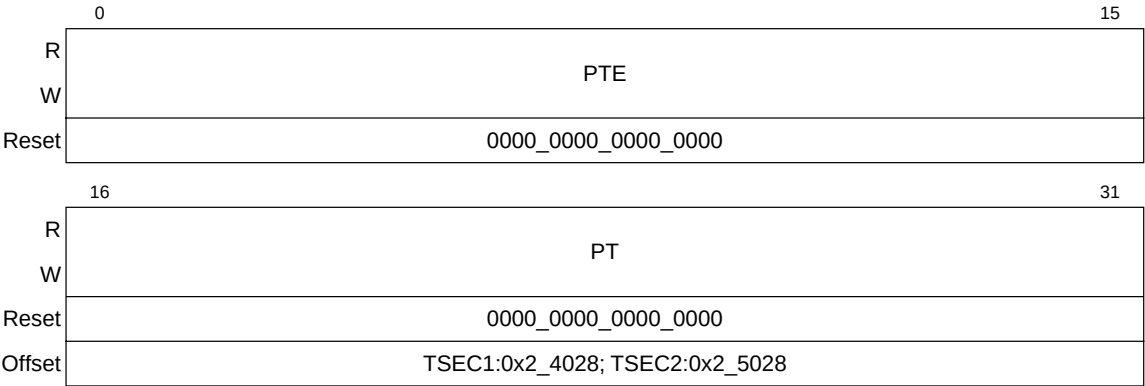


Figure 14-11. PTV Register Definition

Table 14-9 describes the fields of the PTV register.

Table 14-9. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause time value
16–31	PT	Pause time value

14.5.3.1.7 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. Figure 14-12 describes the definition for the DMACTRL register.

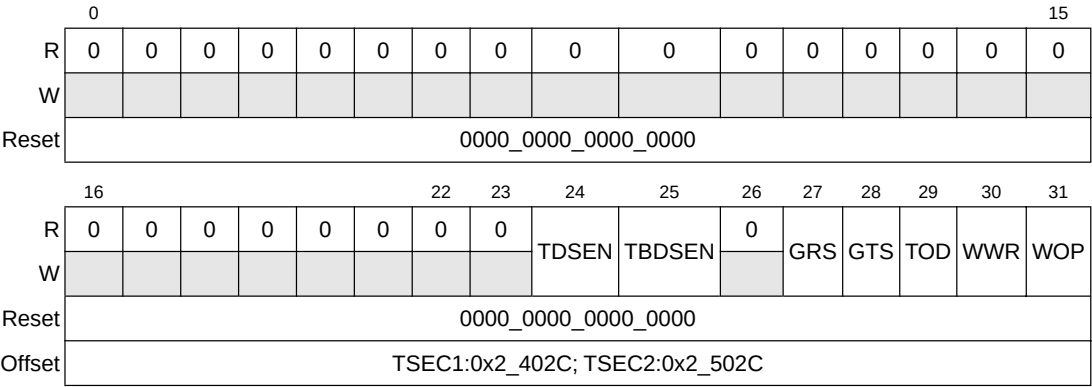


Figure 14-12. DMACTRL Register Definition

Table 14-10 describes the fields of the DMACTRL register.

Table 14-10. 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 receive enable bit of the MAC may still be set, the MAC may continue to receive but the TSEC ignores the receive data until GRS is cleared. If this bit is cleared, the TSEC 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 TSEC hardware without fear of conflict. 0 TSEC scans input data stream for valid frame. 1 TSEC 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	TOD	Transmit on demand. If this bit is set, the TSEC gives high priority to the current TxBD and begins sending the frame without waiting the normal polling time to check the TxBD's R bit. TOD is cleared after transmission begins, but TOD mode continues until an unprepared BD (TxBD[R] is cleared) is reached, the DMACTRL[GTS] is set, or TSTAT[THLT] is set. New TxBDs are processed in order. This bit is ignored if DMACTRL[WOP] is set. 0 TSEC continues. 1 TSEC begins high-priority frame transmission.
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, RXB0, RXF0, the TSEC 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 TSEC to periodically poll TxBD or to wait for software to tell TSEC to fetch a buffer descriptor. While operating in the “Wait” mode, the TSEC 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 every 512 clocks. 1 Do not poll, but wait for a write to TSTAT[THLT].

14.5.3.1.8 TBI Physical Address Register (TBIPA)

This register 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 14.5.4, “Ten-Bit Interface (TBI).” Figure 14-13 describes the definition for the TBIPA register.

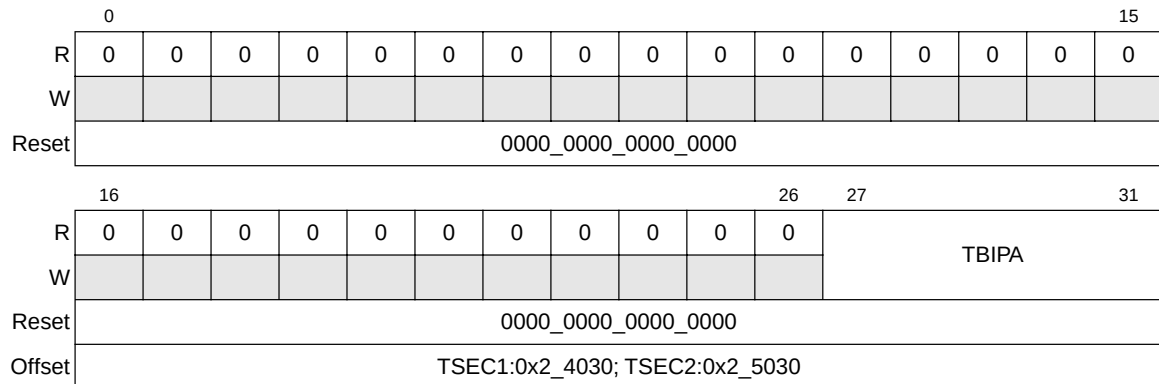


Figure 14-13. TBIPA Register Definition

Table 14-11 describes the fields of the TBIPA register.

Table 14-11. TBIPA Field Descriptions

Bit	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. Refer to Section 14.5.3.6.8, “MII Management Address Register (MIIMADD).”

14.5.3.2 TSEC 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 TSEC 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 TSEC 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 256 entries or 1 Kbytes) indicates the amount of data required to be in the FIFO before starting the transmission of a frame. The FIFO_TX_STARVE (default is 128 entries or 512 Kbyte) 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 256 entries or 1 Kbyte) 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 TSEC fetches from memory.

14.5.3.2.1 FIFO Transmit Threshold Register (FIFO_TX_THR)

The main purpose of the threshold registers is to trigger the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0-511 for 2K FIFO) to trigger the threshold function. If the number of valid entries in the FIFO is equal to or greater than the threshold register, then transmission can begin. This register is read/write by software and is initialized to 0000_0000_0000_0000_0000_0001_0000_0000 at system reset. Figure 14-14 describes the definition for the FIFO_TX_THR register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16						22	23								31
R	0	0	0	0	0	0	0	FIFO_TX_THR								
W																
Reset	0000_0001_0000_0000															
Offset	TSEC1:0x2_408C; TSEC2:0x2_508C															

Figure 14-14. FIFO_TX_THR Register Definition

Table 14-12 describes the fields of the FIFO_TX_THR register.

Table 14-12. FIFO_TX_THR Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_THR	FIFO transmit threshold. These bits mark the number of entries in the transmit FIFO that, if reached, trigger the unloading of frame data into the MAC.

14.5.3.2.2 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register is to inform the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0-511 for 2K 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. This register is read/write by software. Figure 14-15 describes the definition for the FIFO_TX_STARVE register.

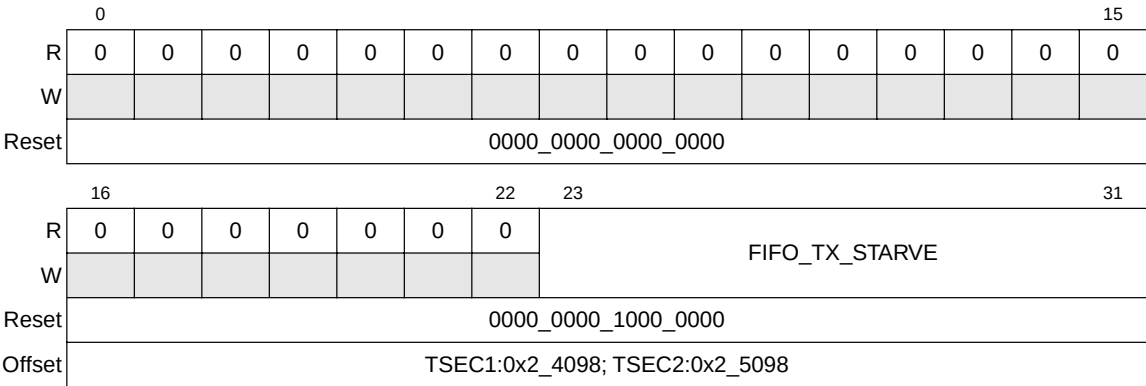


Figure 14-15. FIFO_TX_STARVE Register Definition

Table 14-13 describes the fields of the FIFO_TX_STARVE register.

Table 14-13. 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 if the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

14.5.3.2.3 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff registers 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_TX starve shutoff register, the starve condition ends. This register is read/write by software. Figure 14-16 describes the definition for the FIFO_TX starve shutoff register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16						22				23					31
R	0	0	0	0	0	0	0	FIFO_TX_STARVE_SHUTOFF								
W																
Reset	0000_0001_0000_0000															
Offset	TSEC1:0x2_409C; TSEC2:0x2_509C															

Figure 14-16. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 14-14 describes the fields of the FIFO_TX starve shutoff register.

Table 14-14. FIFO_TX_STARVE_SHUTOFF Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE_SHUTOFF	FIFO transmit starve shutoff. These bits indicate 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.

14.5.3.3 TSEC 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.

14.5.3.3.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. Figure 14-17 describes the definition for the TCTRL register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16	19			20	21	26					27	28		29	31
R	0	0	0	0	THDF	0	0	0	0	0	0	RFC_PAUSE	TFC_PAUSE	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4100; TSEC2:0x2_5100															

Figure 14-17. TCTRL Register Definition

Table 14-15 describes the fields of the TCTRL register.

Table 14-15. 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 TSEC). 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 current transmission completes. Next, the GTSC interrupt in the IEVENT register is asserted. 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 MAC continues. 1 Pause frame is transmitted.
29–31	—	Reserved

14.5.3.3.2 Transmit Status Register (TSTAT)

This register is read/write and is written by TSEC to convey DMA status information. Figure 14-18 describes the definition for the TSTAT register.

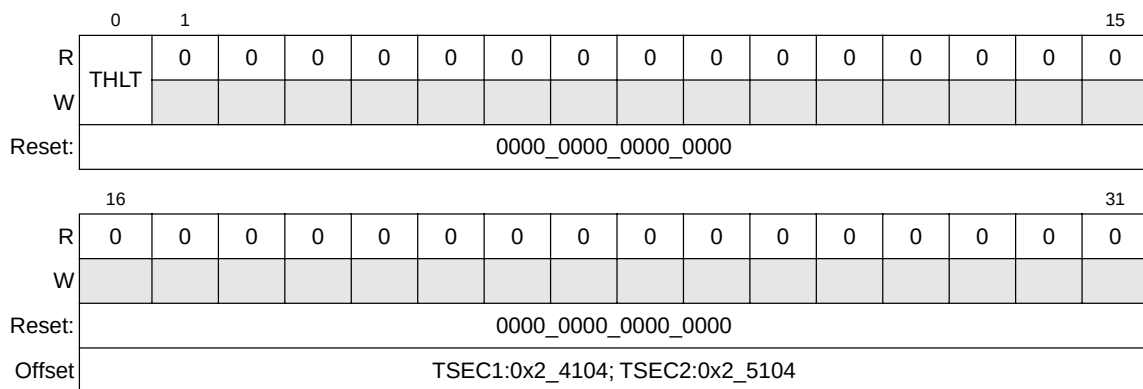


Figure 14-18. TSTAT Register Definition

Table 14-16 describes the fields of the TSTAT register.

Table 14-16. TSTAT Field Descriptions

Bits	Name	Description
0	THLT	Transmit halt. 0 No hardware initiated transmission halt. 1 TSEC transmission function halted. This bit is written by TSEC 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

14.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 14-19 describes the definition for the TBDLEN register.

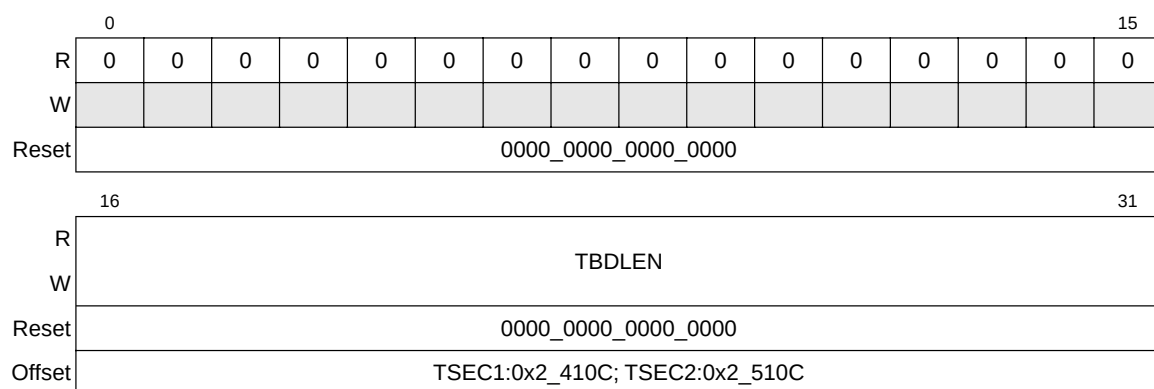


Figure 14-19. TBDLEN Register Definition

Table 14-17 describes the fields of the TBDLEN register.

Table 14-17. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	The TBDLEN is internally written by the DMA module. The transmit channel remains active until TBDLEN is 0.

14.5.3.3.4 Current Transmit Buffer Descriptor Pointer (CTBPTR)

CTBPTR contains the address of the transmit buffer descriptor either currently being processed, or processed most recently. Figure 14-20 describes the definition for the CTBPTR register.

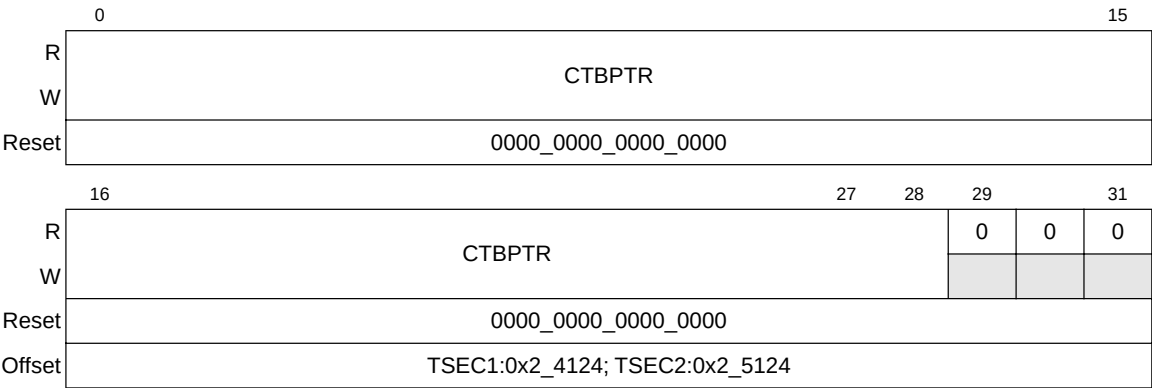


Figure 14-20. CTBPTR Register Definition

Table 14-18 describes the fields of the CTBPTR register.

Table 14-18. CTBPTR Field Descriptions

Bits	Name	Description
0–28	CTBPTR	The CTBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.3.5 Transmit Descriptor Base Address (TBASE)

The TBASE register 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 14-21 describes the definition for the TBASE register.

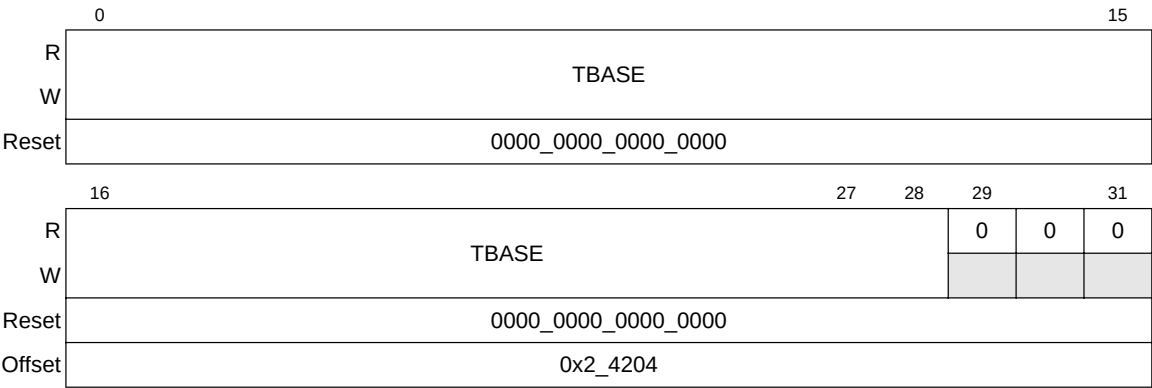


Figure 14-21. TBASE Register Definition

Table 14-19 describes the fields of the TBASE register.

Table 14-19. TBASE Field Descriptions

Bits	Name	Description
0–28	TBASE	Transmit base. TBASE defines the starting location in the memory map for the TSEC 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 TSEC transmit function.
29–31	—	Reserved

14.5.3.3.6 Out-of-Sequence TxBD Register (OSTBD)

OSTBD is the out-of-sequence TxBD register. This register includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. The OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. The user must set OSTBD[L] while preparing this BD. 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 TSEC 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 14-22 describes the definition for the OSTBD register.

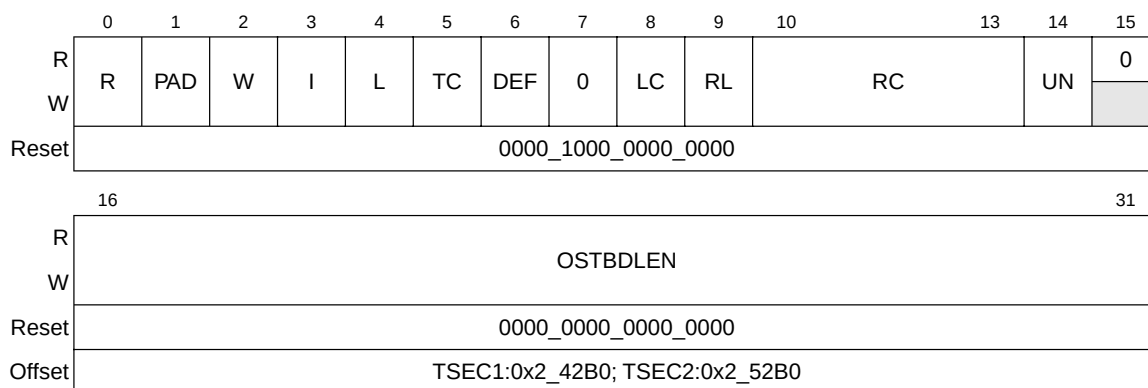
**Figure 14-22. OSTBD Register Definition**

Table 14-20 describes the fields of the OSTBD register.

Table 14-20. OSTBD Field Descriptions

Bits	Name	Description
0	R	Ready, written by TSEC 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 TSEC 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	Padding for short frames. (Valid only while one TxBD is used (L is set), MACCFG2 register's PAD/CRC Enable bit is cleared, and it's CRC Enable bit is cleared; otherwise pads are added to short frames.) 0 Do not add PADS to short frames unless TxBD[TC] is set. 1 Add PADS to short frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADS up to MINFLR value, TSEC always PADS up to the IEEE minimum frame length of 64 bytes.
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[TXF] is set after this buffer is serviced. This bit can cause an interrupt if IEVENT[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 one TxBD is used (L is set), TxBD[PAD] is zero, the MACCFG2 register's PAD/CRC Enable bit is cleared, and its CRC Enable bit is cleared; otherwise, a CRC is added to all frames.) 0 End transmission immediately after the last data byte, unless TxBD[PAD] is set. 1 Transmit the CRC sequence after the last data byte.
6	DEF	DEF. Defer indication, written by hardware. 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	—	Reserved
8	LC	Late collision, written by TSEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The TSEC terminates the transmission and updates LC.
9	RL	Retransmission limit, written by TSEC. 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 TSEC terminates the transmission and updates RL.
10–13	RC	Retry count, written by TSEC. 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.

Table 14-20. OSTBD Field Descriptions (continued)

Bits	Name	Description
14	UN	Underrun, written by TSEC. 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 TSEC 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 TSEC should transmit from this BD's data buffer. It is never modified by the TSEC.

14.5.3.3.7 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

OSTBDP is the out-of-sequence Tx data buffer pointer register. This out-of-sequence Tx data buffer pointer register is written by the user and contains the data buffer pointer fields in the same format as a regular TxBD and together with OSTBD provides the complete 8-byte descriptor. This area must be cleared while not in use. Figure 14-23 describes the definition for the OSTBDP register.

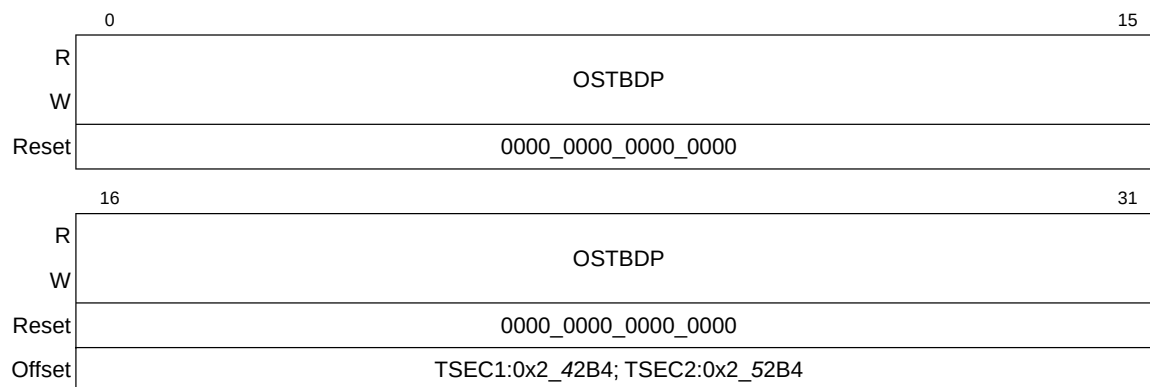
**Figure 14-23. OSTBDP Register Definition**

Table 14-21 describes the fields of the OSTBDP register.

Table 14-21. 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.

14.5.3.4 TSEC 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.

14.5.3.4.1 Receive Control Register (RCTRL)

The RCTRL register is programmed by the user and 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 14-24 describes the definition for the RCTRL register.

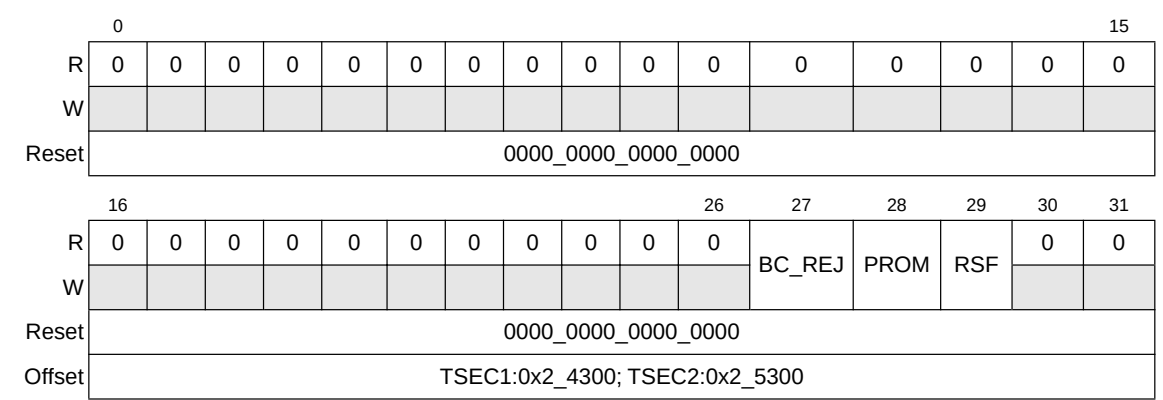


Figure 14-24. RCTRL Register Definition

Table 14-22 describes the fields of the RCTRL register.

Table 14-22. 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.
28	PROM	Promiscuous mode. All frames, regardless of the addresses, are accepted.
29	RSF	Receive short frame mode. Enables the reception of frames shorter than MINFLR bytes.
30–31	—	Reserved

14.5.3.4.2 Receive Status Register (RSTAT)

TSEC writes to this register under the following conditions:

- The receiver runs out of descriptors
- The receiver was halted because an error condition was encountered while receiving a frame

Software should clear the RHLT bit to take TSEC’s receiver function out of a halt state. Figure 14-25 describes the definition for the RSTAT register.

	0							7	8	9						15
R	0	0	0	0	0	0	0	0	QHLT	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4304; TSEC2:0x2_5304															

Figure 14-25. RSTAT Register Definition

Table 14-23 describes the fields of the RSTAT register.

Table 14-23. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. 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

14.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer. Figure 14-26 describes the definition for the RBDLEN register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16															31
R	RBDLEN															
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_430C; TSEC2:0x2_530C															

Figure 14-26. RBDLEN Register Definition

Table 14-24 describes the fields of the RBDLEN register.

Table 14-24. 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. The user should avoid writing to this register.

14.5.3.4.4 Current Receive Buffer Descriptor Pointer (CRBPTR)

CRBPTR contains the address of the receive buffer descriptor either currently being processed, or processed most recently. Figure 14-27 describes the definition for the CRBPTR register. This register is writable, but should never be written by user.

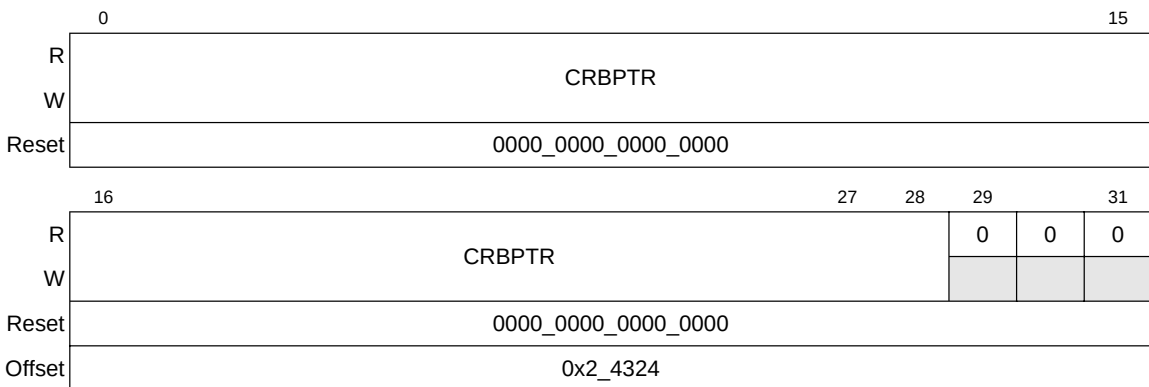
**Figure 14-27. CRBPTR Register Definition**

Table 14-25 describes the fields of the CRBPTR register.

Table 14-25. CRBPTR Field Descriptions

Bit	Name	Description
0–28	CRBPTR	The CRBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.4.5 Maximum Receive Buffer Length Register (MRBLR)

The MRBLR register is written by the user. It informs the TSEC how much space is in the receive buffer pointed to by the RxBD. Figure 14-28 describes the definition for the MRBLR.

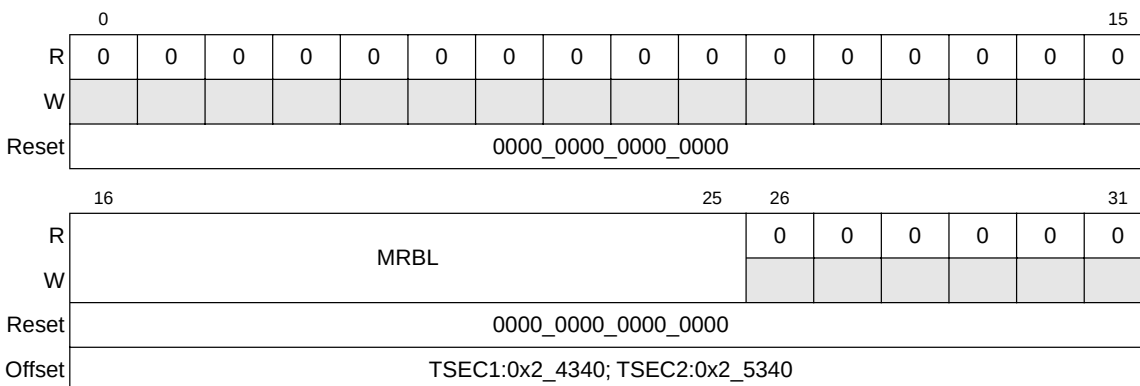


Figure 14-28. MRBL Register Definition

Table 14-26. 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 TSEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. TSEC 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 should be cleared.

14.5.3.4.6 Receive Buffer Descriptor Pointer (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 14-29 describes the definition for the RBPTR register. This register takes on the value of RBASE when the RBASE register is written by software.

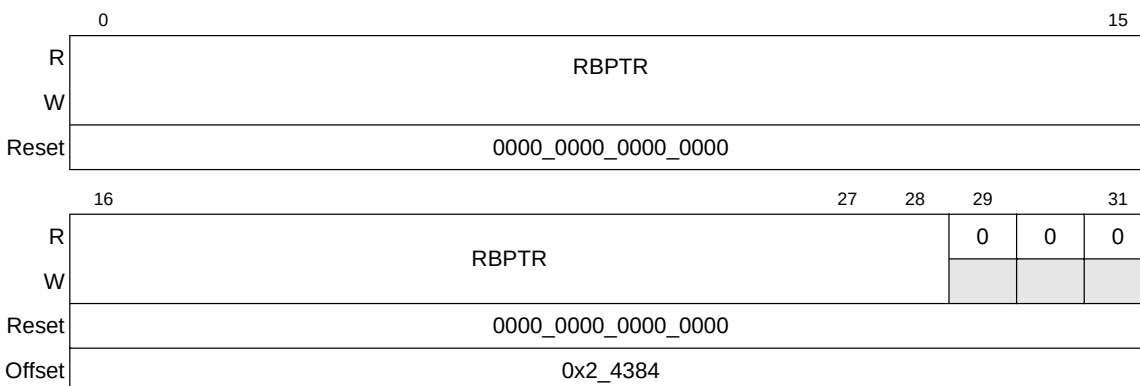


Figure 14-29. Receive Buffer Descriptor Pointer Definition

Table 14-27 describes the fields of the RBPTR register.

Table 14-27. RBPTR Field Descriptions

Bit	Name	Description
0–28	RBPTR	The RBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

14.5.3.4.7 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 14-30 describes the definition for the RBASE register.

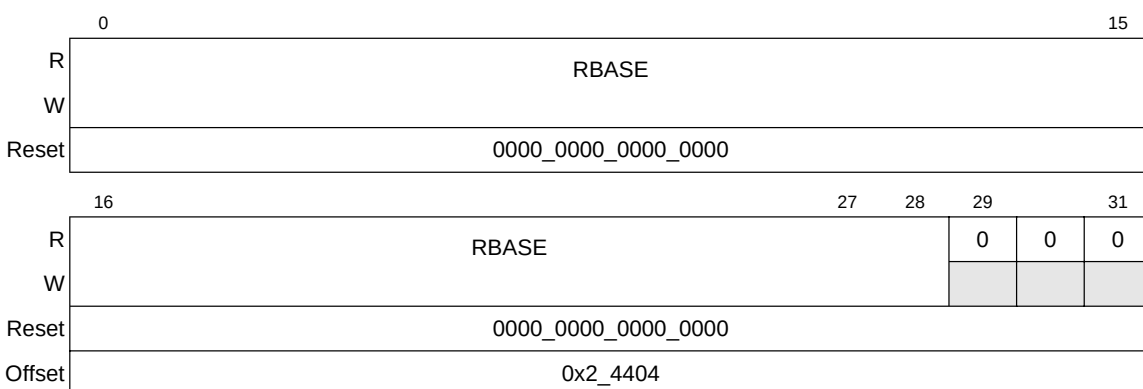
**Figure 14-30. RBASE Register Definition**

Table 14-28 describes the fields of the RBASE register.

Table 14-28. RBASE Field Descriptions

Bits	Name	Description
0–28	RBASE	Receive base low. RBASE defines the starting location in the memory map for the TSEC RxBD. This field must be 8-byte aligned.
29–31	—	Reserved

14.5.3.5 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.

14.5.3.5.1 Configuring the MAC

The MAC configuration registers 1 and 2 provide the means of 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.
- Varying pad/CRC combinations – Three different pad/CRC combinations are provided to handle a variety of system requirements. Most simple are frames that already have a valid FCS field. In this case, the CRC is checked and reported via 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.

14.5.3.5.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the TSEC. Half-duplex is only supported for 10 Mbps 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 TSEC 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.

14.5.3.5.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the TSEC 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 TSEC 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 TSEC is deferring to a carrier, the TSEC 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 TSEC 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:

$$0 < \text{or } = r < \text{or } = 2^k \text{ where } k = \min(n, 10).$$

So, after the first collision, TSEC backs-off either 0 or 1 slot times. After the fifth collision, TSEC 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.

14.5.3.5.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within TSEC. 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. 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 TSEC 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 TSEC 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, TSEC 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 TSEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If cleared, the TSEC 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 TSEC drops carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If, while applying back pressure, the TSEC 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. Collisions for packets while THBP is asserted are counted. TSEC 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.

14.5.3.5.5 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 TSEC 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 TSEC. While enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

14.5.3.6 MAC Registers

Following are the MAC registers.

14.5.3.6.1 MAC Configuration 1 Register (MACCFG1)

The MACCFG1 register is written by the user. Figure 14-31 describes the definition for the MACCFG1 register.

	0	1	2								11	12	13	14	15
R	Soft_Reset	0	0	0	0	0	0	0	0	0	0	Reset Rx MC	Reset Tx MC	Reset Rx Fun	Reset Tx Fun
W															
Reset	0000_0000_0000_0000														

	16						22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	Loop Back	0	0	Rx_Flow	Tx_Flow	Sync'd Rx EN	Rx_EN	Sync'd Tx EN	Tx_EN
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4500; TSEC2:0x2_5500															

Figure 14-31. MACCFG1 Register Definition

Table 14-29 describes the fields of the MACCFG1 register.

Table 14-29. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 14.6.2.2, “Soft Reset and Reconfiguring Procedure,” on page 14-119 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.
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. 0 Normal operation. 1 Loop back the MAC transmit outputs to the MAC receive inputs.
24–25	—	Reserved

Table 14-29. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
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. 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. 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

14.5.3.6.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. Figure 14-32 describes the definition for the MACCFG2 register.

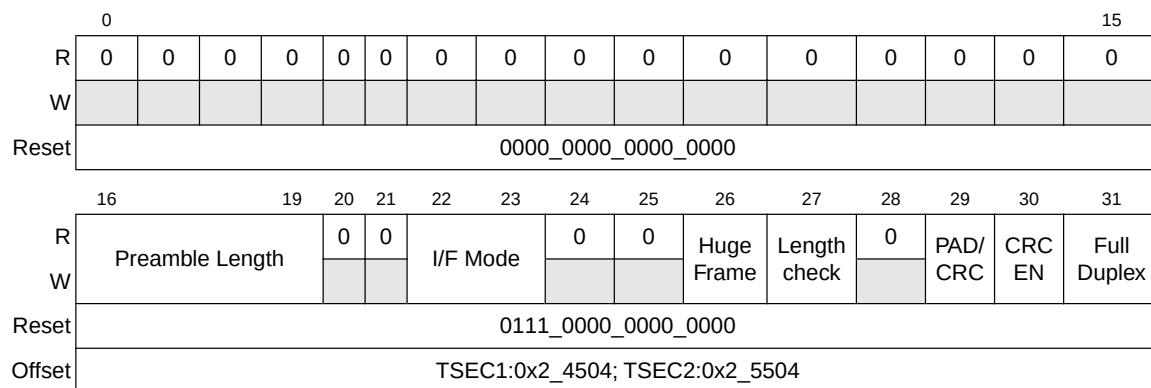
**Figure 14-32. MACCFG2 Register Definition**

Table 14-30 describes the fields of the MACCFG2 register.

Table 14-30. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble Length	This field 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	This field determines the type of interface to which the MAC is connected. Its default is 00. 00 Reserved 01 Nibble mode (MII) 10 Byte mode (GMII/TBI) 11 Reserved
24–25	—	Reserved
26	Huge Frame	Huge frame enable. This bit is 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.
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 to operate in half-duplex mode only. 1 The MAC operates in full-duplex mode.

14.5.3.6.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. Figure 14-33 describes the definition for the IPGIFG register.

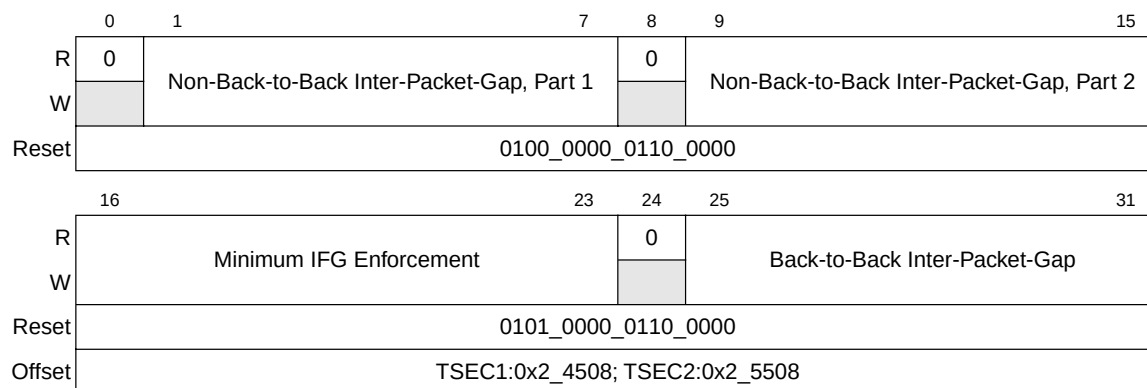
**Figure 14-33. IPGIFG Register Definition**

Table 14-31 describes the fields of the IPGIFG register.

Table 14-31. 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.
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.

14.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 14-34 describes the definition for the HAFDUP register.

	0							7	8	11			12	13		14	15
R	0	0	0	0	0	0	0	0	Alternate BEB Truncation			Alt BEB	BP No BackOff	No BackOff	Excess Defer		
W																	
Reset	0000_0000_1010_0001																
	16			19		20	25				26	31					
R	Retransmission Maximum					0	0	0	0	0	0	Collision Window					
W																	
Reset	1111_0000_0011_0111																
Offset	TSEC1:0x2_450C; TSEC2:0x2_550C																

Figure 14-34. Half-Duplex Register Definition

Table 14-32 describes the fields of the HAFDUP register.

Table 14-32. 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 2 ¹⁰ 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.
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.
20–25	—	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.

14.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 14-35 describes the definition for the MAXFRM register.

	0														15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
	16														31
R	Maximum Frame														
W															
Reset	0000_0110_0000_0000														
Offset	TSEC1:0x2_4510; TSEC2:0x2_5510														

Figure 14-35. Maximum Frame Length Register Definition

Table 14-33 describes the fields of the MAXFRM register.

Table 14-33. MAXFRM 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 Enable].)

14.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 14-36 describes the definition for the MIIMCFG register.

	0	1														15
R	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	Mgmt															
Reset	0000_0000_0000_0000															
	16										26	27	28	29		31
R	0	0	0	0	0	0	0	0	0	0	0	No		Mgmt Clock Select		
W												Pre				
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4520; TSEC2:0x2_5520															

Figure 14-36. MII Management Configuration Register Definition

Table 14-34 describes the fields of the MIIMCFG register.

Table 14-34. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII MGMT to perform mgmt 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 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	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (EC_MDC). Its default value is 000. The source clock frequency is equal to the core complex bus clock divided by eight. 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

14.5.3.6.7 MII Management Command (MIIMCOM)

The MIIMCOM register is written by the user. Figure 14-37 describes the definition for the MIIMCOM register.

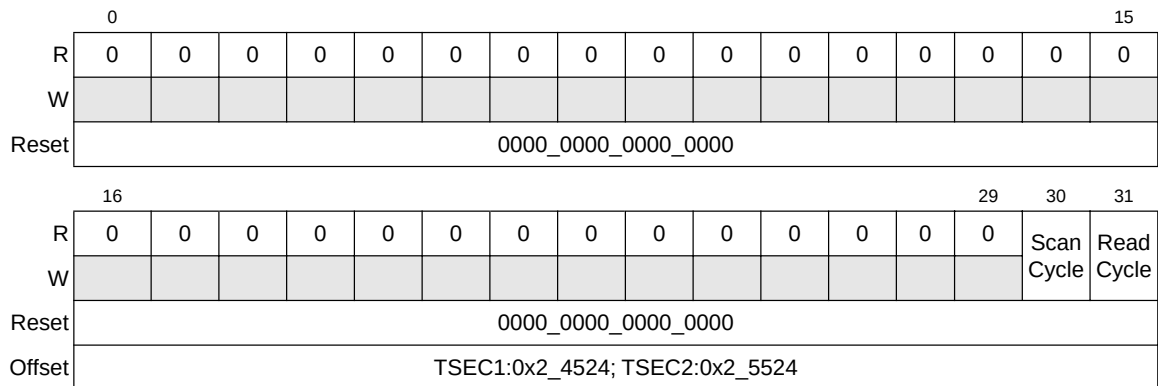
**Figure 14-37. MIIMCOM Register Definition**

Table 14-35 describes the fields of the MIIMCOM register.

Table 14-35. 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. 0 Normal operation. 1 The MII management performs a single read cycle using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The read data is returned in register MIIMSTAT[PHY Status].

14.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 14-38 describes the definition for the MIIMADD register.

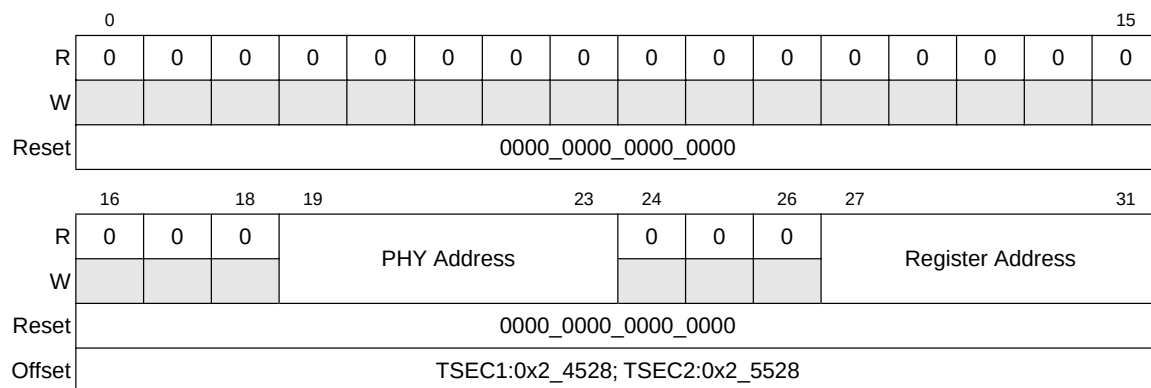
**Figure 14-38. MIIMADD Register Definition**

Table 14-36 describes the fields of the MIIMADD register.

Table 14-36. 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.
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.

14.5.3.6.9 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 14-39 describes the definition for the MIIMCON register.

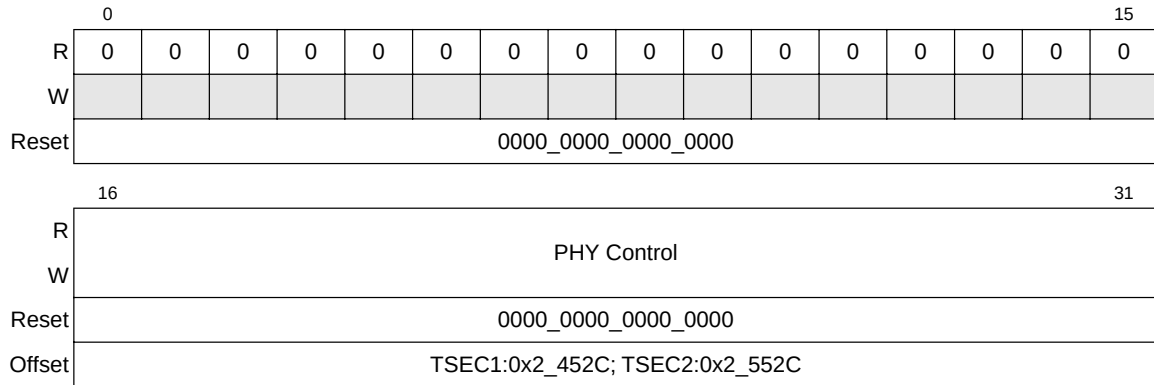


Figure 14-39. MII Mgmt Control Register Definition

Table 14-37 describes the fields of the MIIMCON register.

Table 14-37. 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.

14.5.3.6.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read-only by the user. Figure 14-40 describes the definition for the MIIMSTAT register.

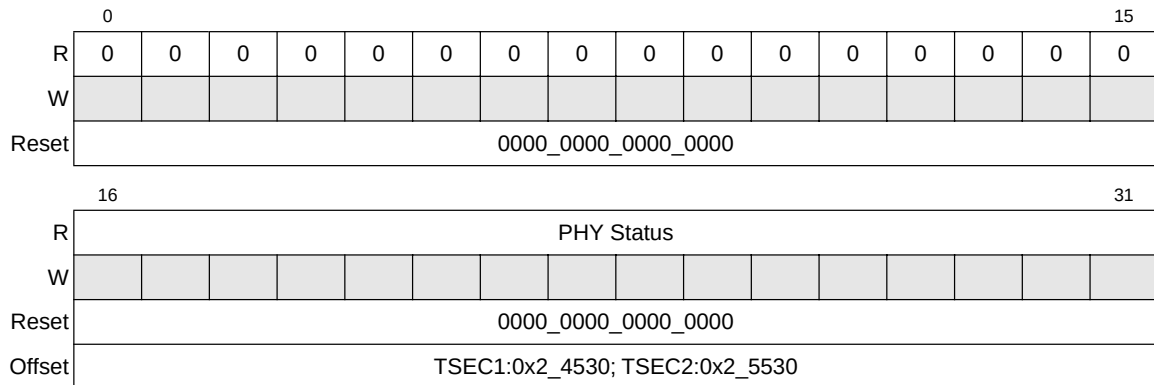


Figure 14-40. MIIMSTAT Register Definition

Table 14-38 describes the fields of the MIIMSTAT register.

Table 14-38. 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.

14.5.3.6.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 14-41 describes the definition for the MIIMIND register.

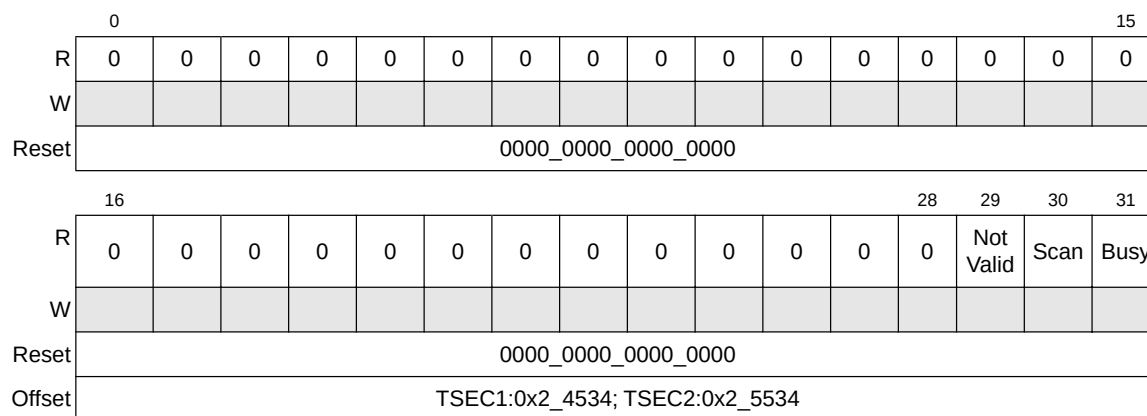


Figure 14-41. MII Mgmt Indicator Register Definition

Table 14-39. 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.

14.5.3.6.12 Interface Status Register (IFSTAT)

The IFSTAT register is written by the user. Figure 14-42 describes the definition for the IFSTAT register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16					21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	Excess Defer	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0001															
Offset	TSEC1:0x2_453C; TSEC2:0x2_553C															

Figure 14-42. Interface Status Register Definition

Table 14-40 describes the fields of the FSTAT register.

Table 14-40. 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

14.5.3.6.13 Station Address Part 1 Register (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 14-43 describes the definition for the MACSTNADDR1 register. 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.

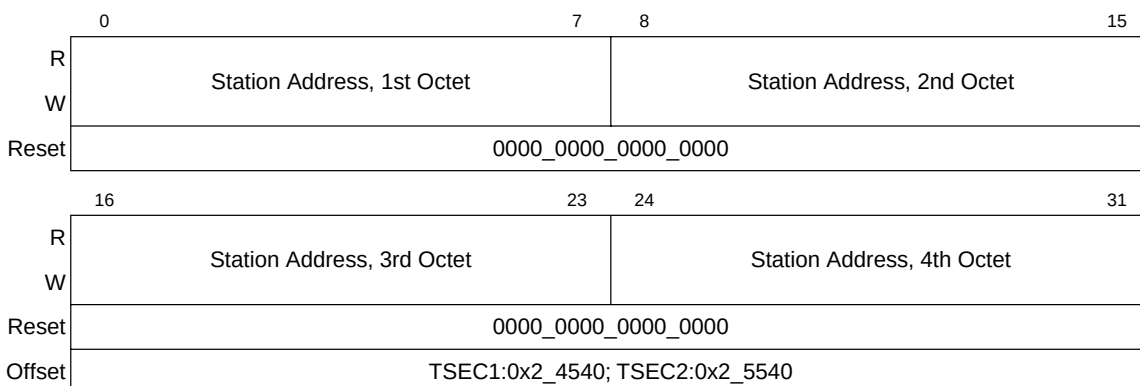
**Figure 14-43. Station Address Part 1 Register Definition**

Table 14-41 describes the fields of the MACSTNADR1 register.

Table 14-41. MACSTNADR1 Field Descriptions

Bit	Name	Description
0–7	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet is stored in 0–7 and defaults to a value of 0x00.
8–15	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet is stored in 8–15 and defaults to a value of 0x00.
16–23	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet is stored in 16–23 and defaults to a value of 0x00.
24–31	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet is stored in 24–31 and defaults to a value of 0x00.

14.5.3.6.14 Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 14-44 describes the definition for the MACSTNADDR2 register.

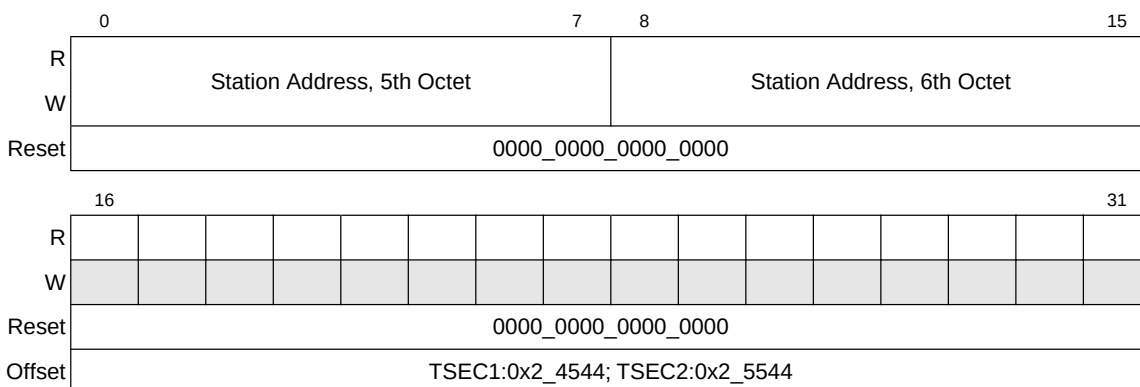
**Figure 14-44. Station Address Part 2 Register Definition**

Table 14-42 describes the fields of the MACSTNADDR2 register.

Table 14-42. MACSTNADDR2 Field Descriptions

Bit	Name	Description
0–7	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet is stored in 0–7 and defaults to a value of 0x00.
8–15	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet is stored in 8–15 and defaults to a value of 0x00.
16–31	—	Reserved

14.5.3.7 MIB Registers

This section describes the MIB registers.

The TSEC MSTAT 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, RMON MIB group 3, RMON MIB group 9, RMON MIB 2, and the 802.3 Ethernet MIB.

The detection of one or more of these statistical events triggers the MSTAT module to update its statistics counters. These counters are stored in internal data registers. The user may access the internal data registers at any time. An interrupt can be generated upon any one counter’s rollover condition via a carry interrupt output from the MSTAT. Each counter’s rollover condition can be discreetly 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 assertion of an external module input pin. Figure 14-45 describes the definition for the TR64 register.

14.5.3.7.1 Transmit and Receive 64-Byte Frame Counter (TR64)

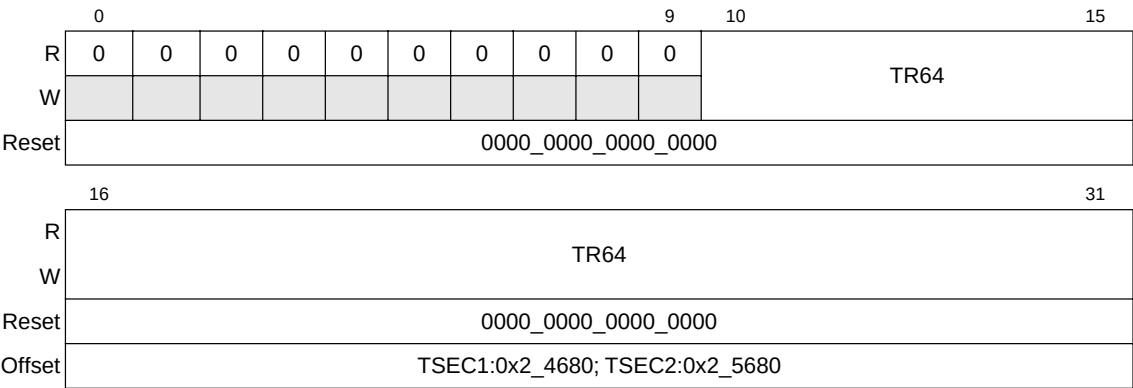


Figure 14-45. Transmit and Receive 64-Byte Frame Register Definition

Table 14-43 describes the fields of the TR64 register.

Table 14-43. 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).

14.5.3.7.2 Transmit and Receive 65- to 127-Byte Frame Counter (TR127)

Figure 14-46 describes the definition for the TR127 register.

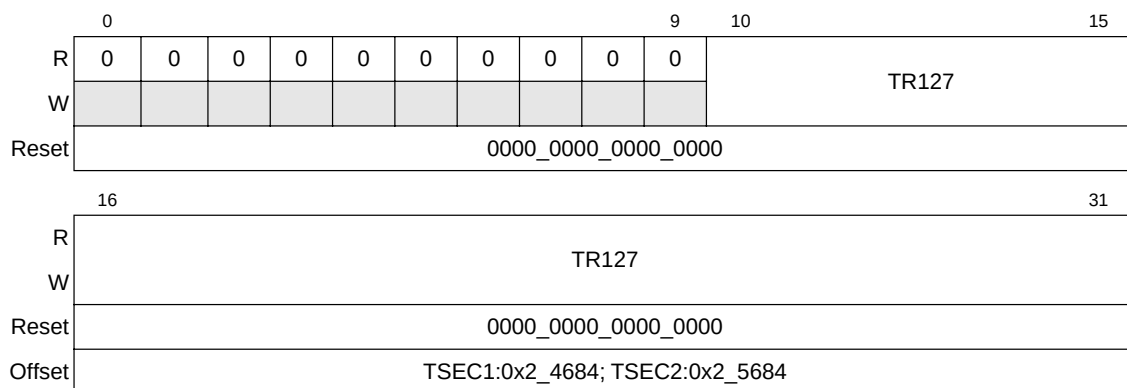
**Figure 14-46. Transmit and Receive 65- to 127-Byte Frame Register Definition**

Table 14-44 describes the fields of the TR127 register.

Table 14-44. TR127 Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	TR127	Transmit and receive 65- to 127-byte frame counter—Increment for each good or bad frame transmitted and received which is 65 to 127 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.3 Transmit and Receive 128- to 255-Byte Frame Counter (TR255)

Figure 14-47 describes the definition for the TR255 register.

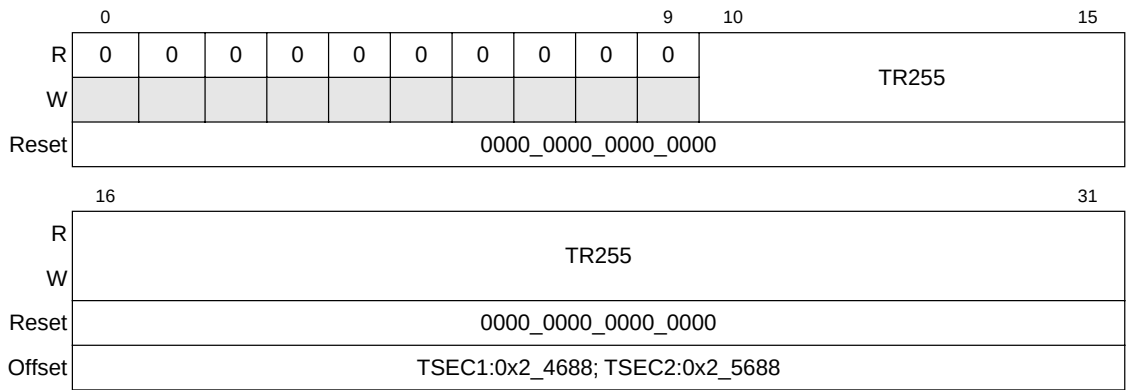


Figure 14-47. Transmit and Received 128- to 255-Byte Frame Register Definition

Table 14-45 describes the fields of the TR255 register.

Table 14-45. 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 to 255 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.4 Transmit and Receive 256- to 511-Byte Frame Counter (TR511)

Figure 14-48 describes the definition for the TR511 register.

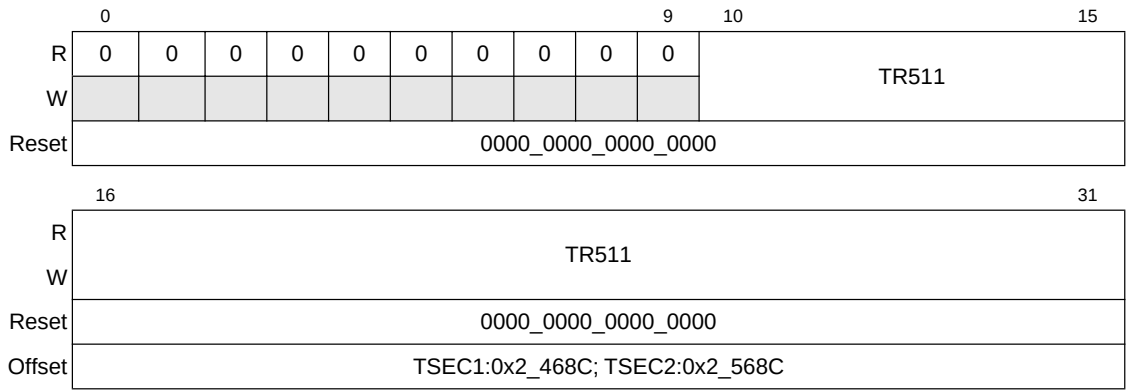


Figure 14-48. Transmit and Received 256- to 511-Byte Frame Register Definition

Table 14-46 describes the fields of the TR511 register.

Table 14-46. 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 to 511 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.5 Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)

Figure 14-49 describes the definition for the TR1K register.

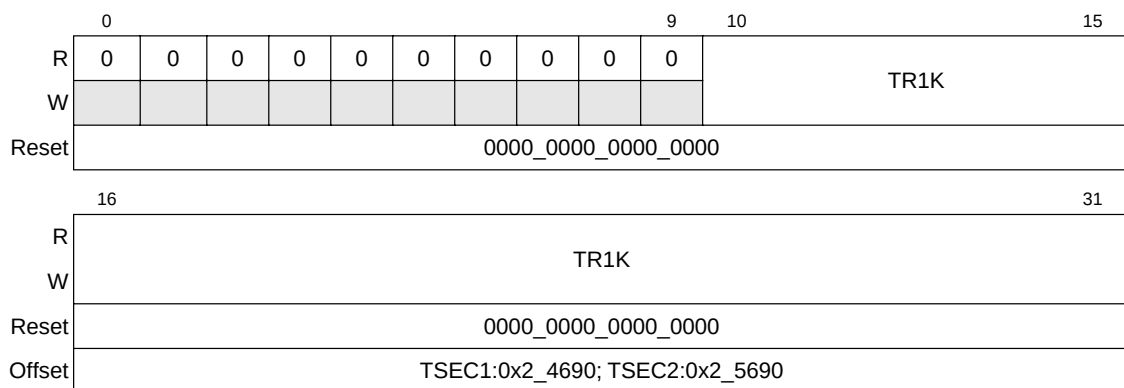
**Figure 14-49. Transmit and Received 512- to 1023-Byte Frame Register Definition**

Table 14-47 describes the fields of the TR1K register.

Table 14-47. 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 to 1023 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.6 Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)

Figure 14-50 describes the definition for the TRMAX register.

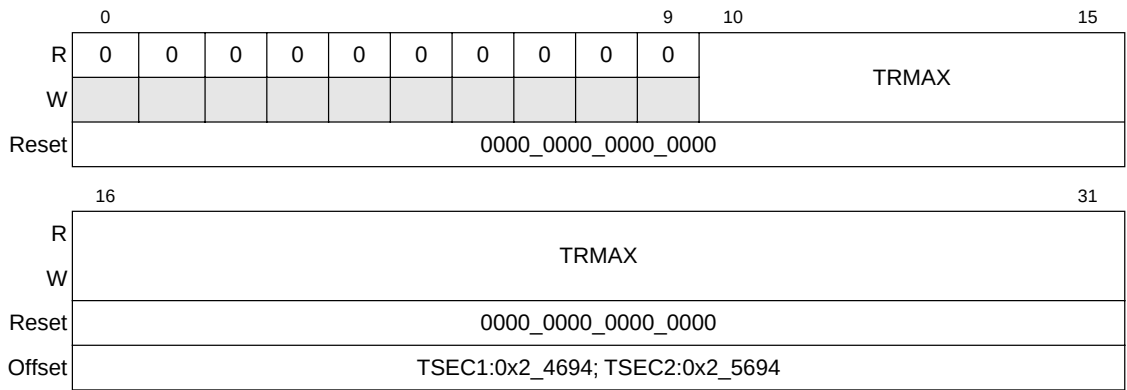


Figure 14-50. Transmit and Received 1024- to 1518-Byte Frame Register Definition

Table 14-48 describes the fields of the TRMAX register.

Table 14-48. 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 to 1518 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.7 Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)

Figure 14-51 describes the definition for the TRMGV register.

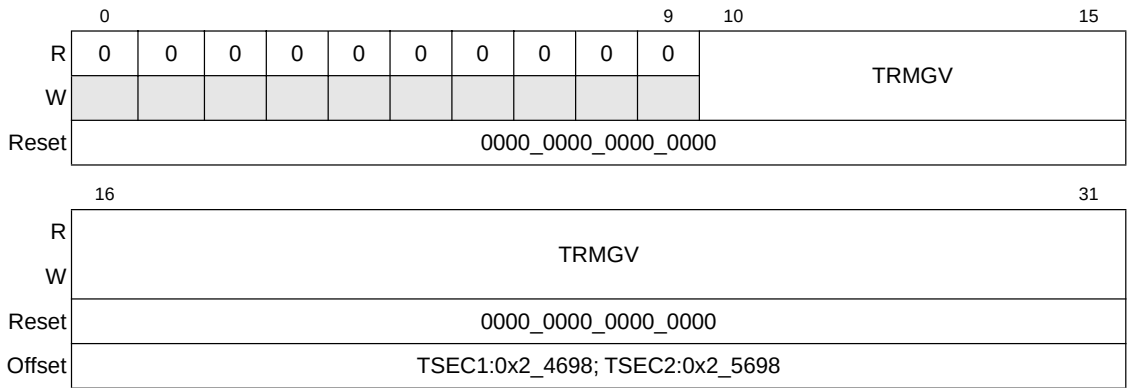


Figure 14-51. Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition

Table 14-49 describes the fields of the TRMGV register.

Table 14-49. 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 to 1522 bytes in length, inclusive (excluding Preamble and SFD but including FCS bytes).

14.5.3.7.8 Receive Byte Counter (RBYT)

Figure 14-52 describes the definition for the RBYT register.

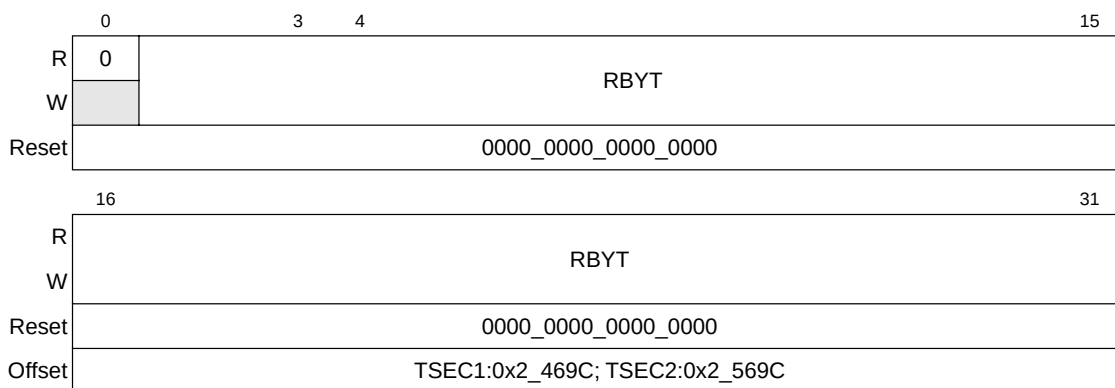
**Figure 14-52. Receive Byte Counter Register Definition**

Table 14-50 describes the fields of the RBYT register.

Table 14-50. RBYT Field Descriptions

Bits	Name	Description
0	—	Reserved
1–31	RBYT	Receive byte counter. The statistic counter register increments by the byte count of frames received with 0 to 1518 bytes, including those in bad packets, excluding Preamble and SFD but including FCS bytes.

14.5.3.7.9 Receive Packet Counter (RPKT)

Figure 14-53 describes the definition for the RPKT register.

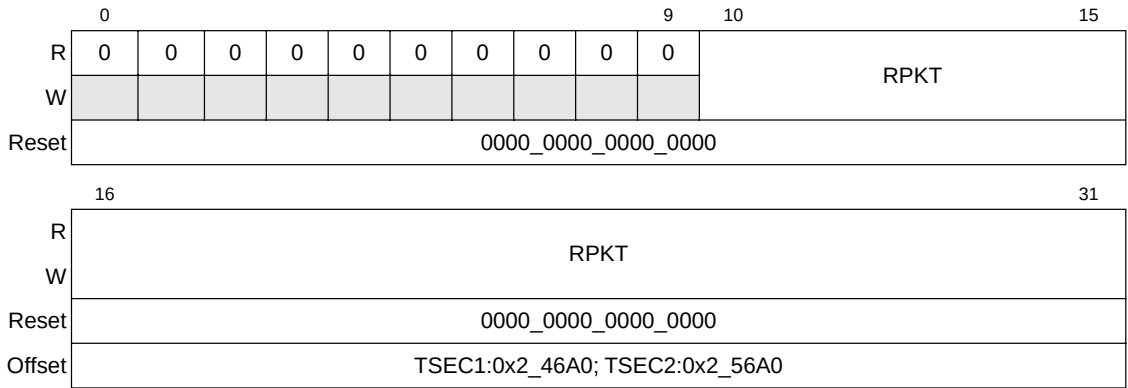


Figure 14-53. Receive Packet Counter Register Definition

Table 14-51 describes the fields of the RPKT register.

Table 14-51. 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).

14.5.3.7.10 Receive FCS Error Counter (RFCS)

Figure 14-54 describes the definition for the RFCS register.

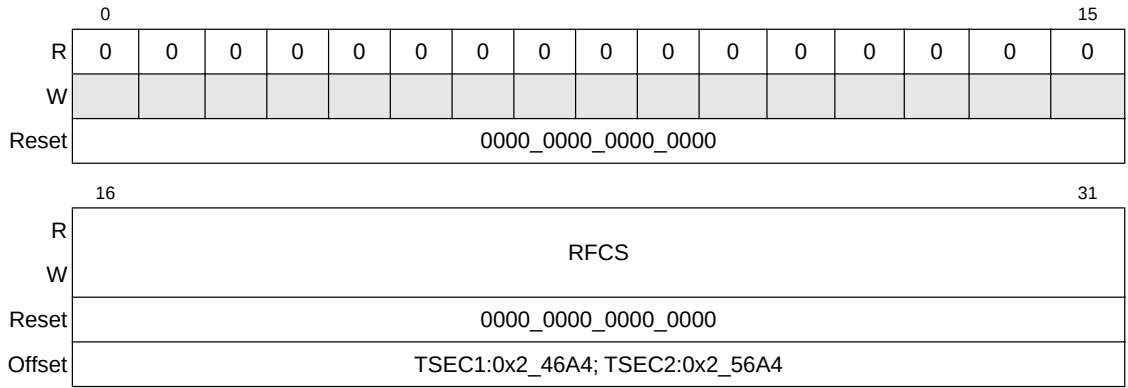


Figure 14-54. Receive FCS Error Counter Register Definition

Table 14-52 describes the fields of the RFCS register.

Table 14-52. RFCS Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RFCS	Receive FCS error counter. Increments for each frame received that has an integral 64 to 1518 length and contains a frame check sequence error.

14.5.3.7.11 Receive Multicast Packet Counter (RMCA)

Figure 14-55 describes the definition for the RMCA register.

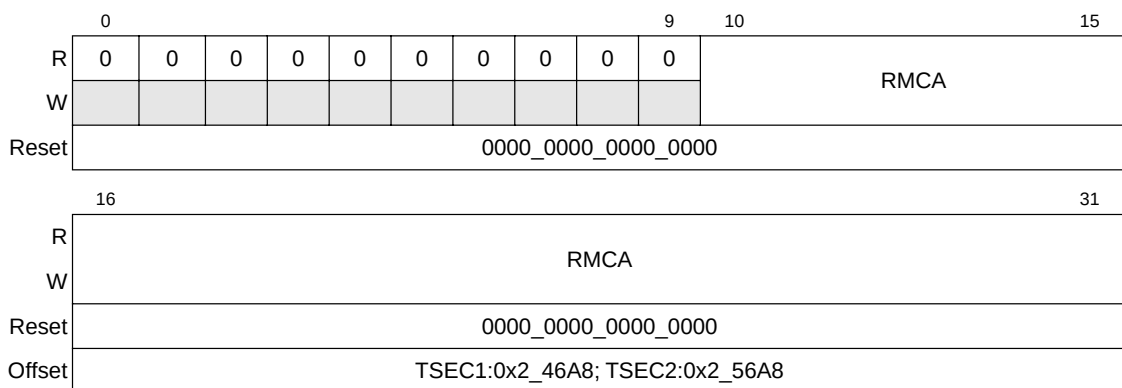
**Figure 14-55. Receive Multicast Packet Counter Register Definition**

Table 14-53 describes the fields of the RMCA register.

Table 14-53. RMCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RMCA	Receive multicast packet counter. Increments for each multicast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding broadcast frames. This count does not include range/length errors.

14.5.3.7.12 Receive Broadcast Packet Counter (RBCA)

Figure 14-55 describes the definition for the RBCA register.

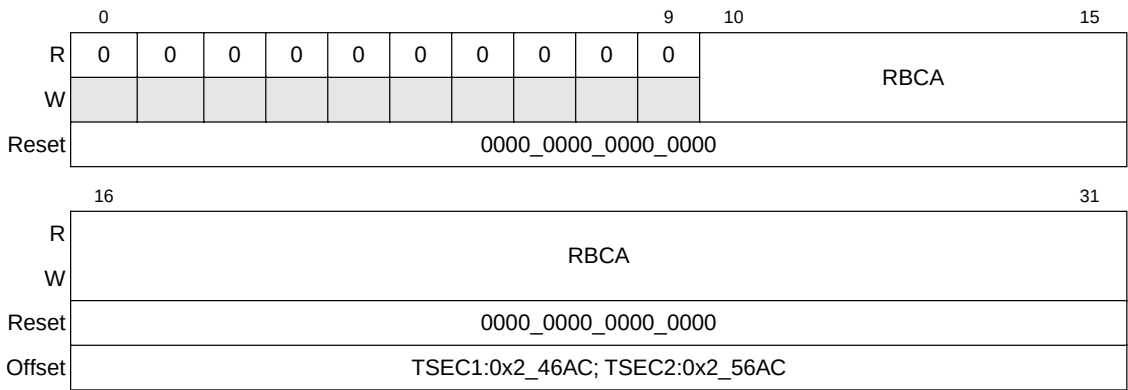


Figure 14-56. Receive Broadcast Packet Counter Register Definition

Table 14-54 describes the fields of the RBCA register.

Table 14-54. RBCA Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–31	RBCA	Receive broadcast packet counter. Increments for each broadcast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN), excluding multicast frames. Does not include range/length errors.

14.5.3.7.13 Receive Control Frame Packet Counter (RXCF)

Figure 14-57 describes the definition for the RXCF register.

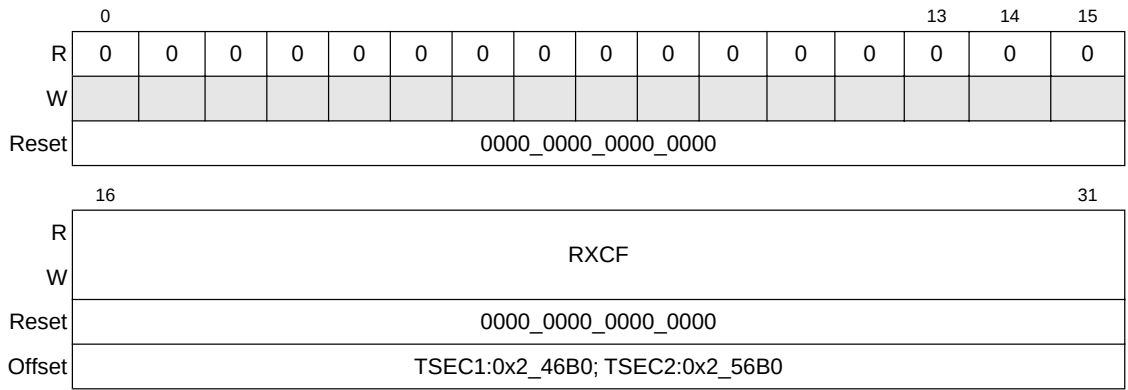


Figure 14-57. Receive Control Frame Packet Counter Register Definition

Table 14-55 describes the fields of the RXCF register.

Table 14-55. 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).

14.5.3.7.14 Receive Pause Frame Packet Counter (RXPF)

Figure 14-58 describes the definition for the RXPF register.

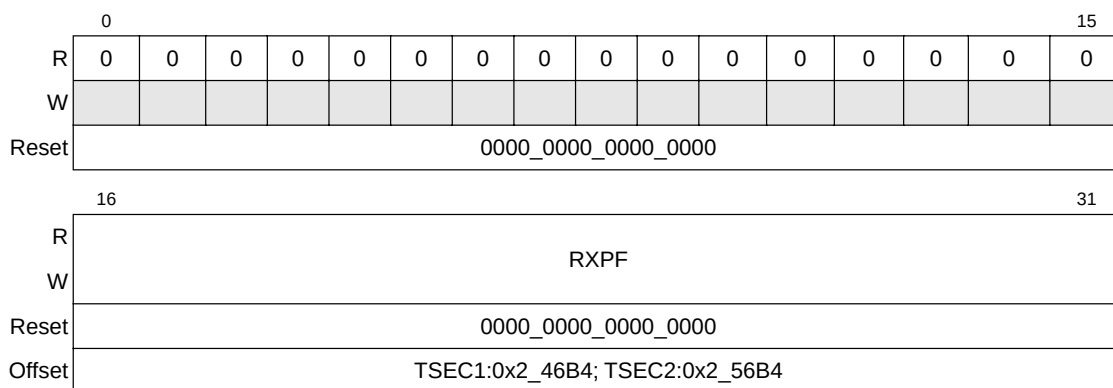
**Figure 14-58. Receive Pause Frame Packet Counter Register Definition**

Table 14-56 describes the fields of the RXPF register.

Table 14-56. RXPF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXPF	Receive PAUSE frame packet counter. Increments each time a valid PAUSE MAC control frame is received.

14.5.3.7.15 Receive Unknown OPCode Packet Counter (RXUO)

Figure 14-59 describes the definition for the RXUO register.

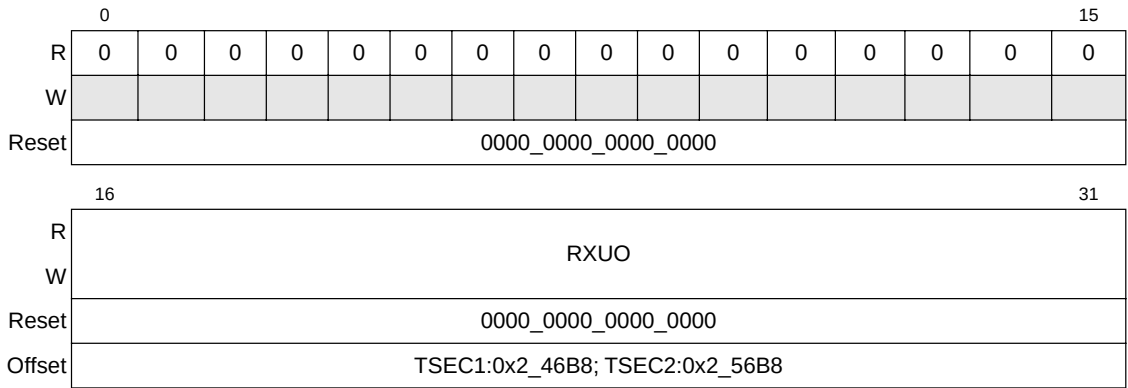


Figure 14-59. Receive Unknown OPCode Packet Counter Register Definition

Table 14-57 describes the fields of the RXUO register.

Table 14-57. 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 a PAUSE.

14.5.3.7.16 Receive Alignment Error Counter (RALN)

Figure 14-60 describes the definition for the RALN register.

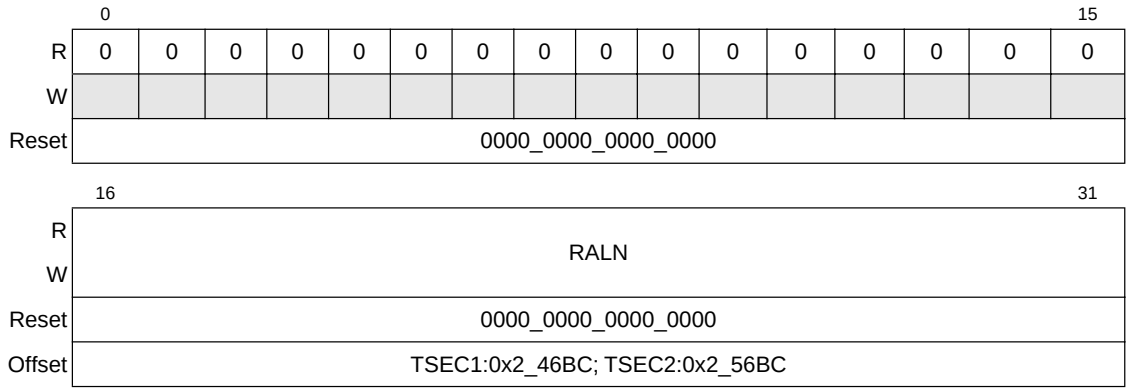


Figure 14-60. Receive Alignment Error Counter Register Definition

Table 14-58 describes the fields of the RALN register.

Table 14-58. 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.

14.5.3.7.17 Receive Frame Length Error Counter (RFLR)

Figure 14-61 describes the definition for the RFLR register.

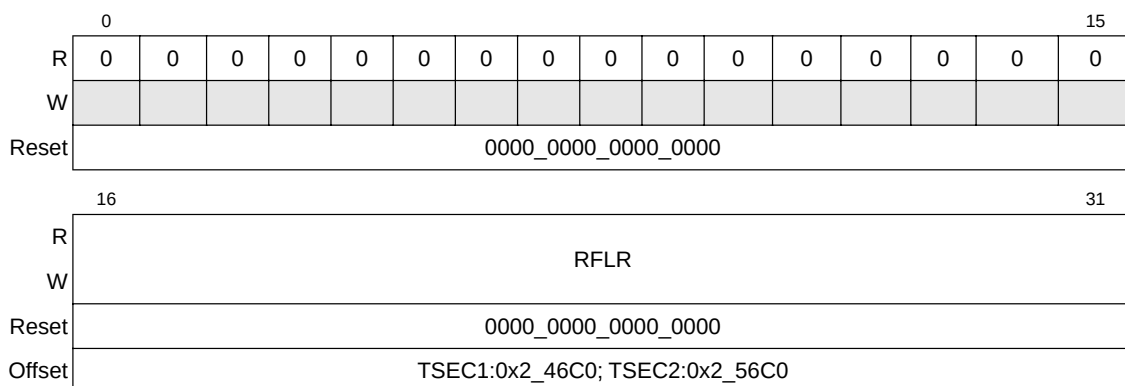
**Figure 14-61. Receive Frame Length Error Counter Register Definition**

Table 14-59 describes the fields of the RFLR register.

Table 14-59. 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 ethernet value.

14.5.3.7.18 Receive Code Error Counter (RCDE)

Figure 14-62 describes the definition for the RCDE register.

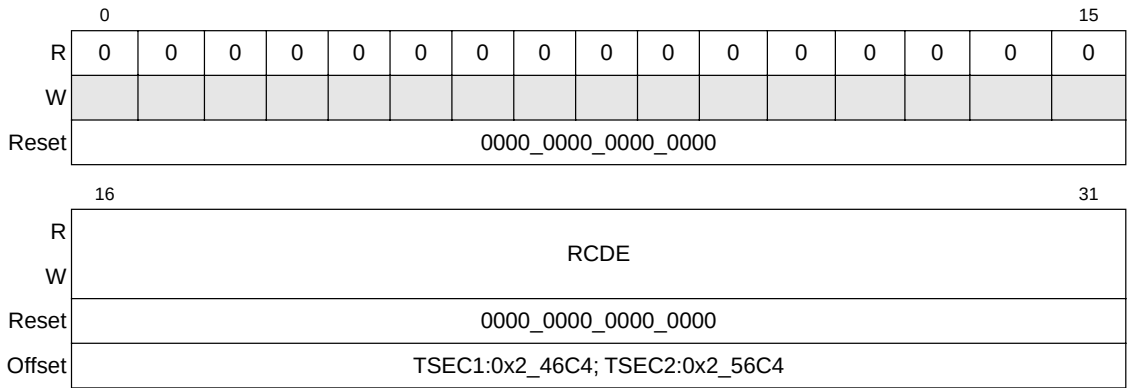


Figure 14-62. Receive Code Error Counter Register Definition

Table 14-60 describes the fields of the RCDE register.

Table 14-60. 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.

14.5.3.7.19 Receive Carrier Sense Error Counter (RCSE)

Figure 14-63 describes the definition for the RCSE register.

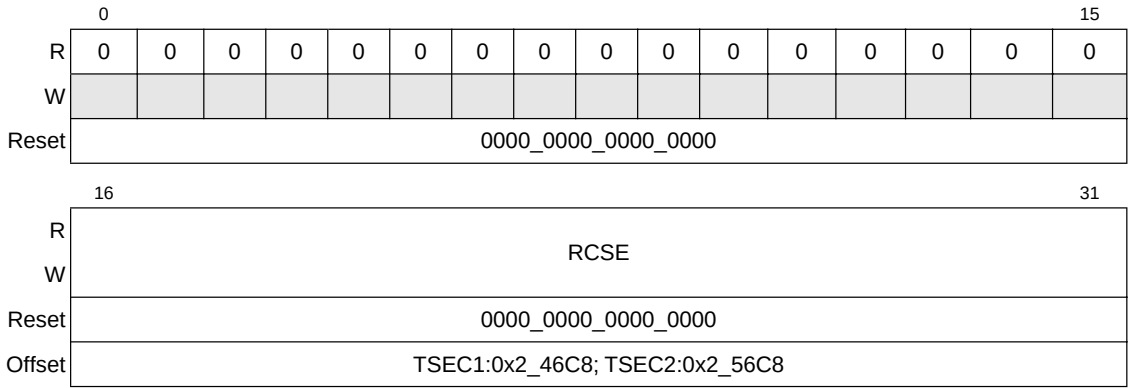


Figure 14-63. Receive Carrier Sense Error Counter Register Definition

Table 14-61 describes the fields of the RCSE register.

Table 14-61. RCSE Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RCSE	Receive false carrier counter. Increments each time a false carrier is detected during idle, as defined by a 1 on TSECn_RX_ER and an 0xE on TSECn_RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.

14.5.3.7.20 Receive Undersize Packet Counter (RUND)

Figure 14-64 describes the definition for the RUND register.

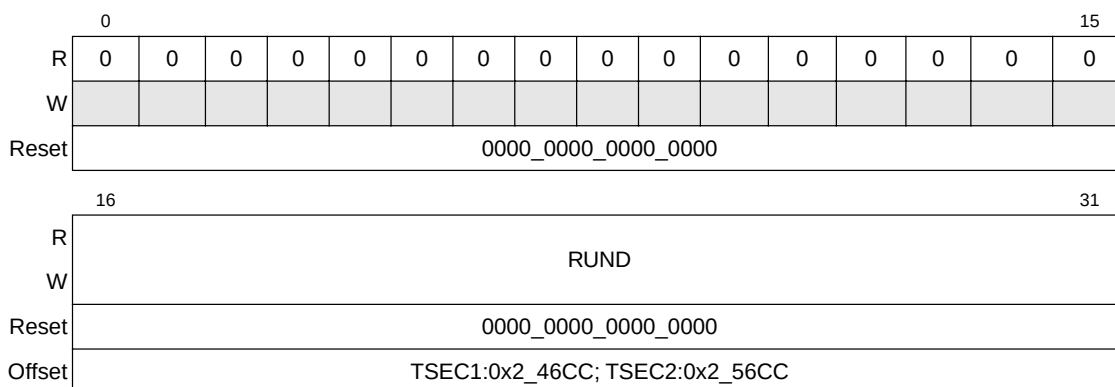
**Figure 14-64. Receive Undersize Packet Counter Register Definition**

Table 14-62 describes the fields of the RUND register.

Table 14-62. 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.

14.5.3.7.21 Receive Oversize Packet Counter (ROVR)

Figure 14-65 describes the definition for the ROVR register.

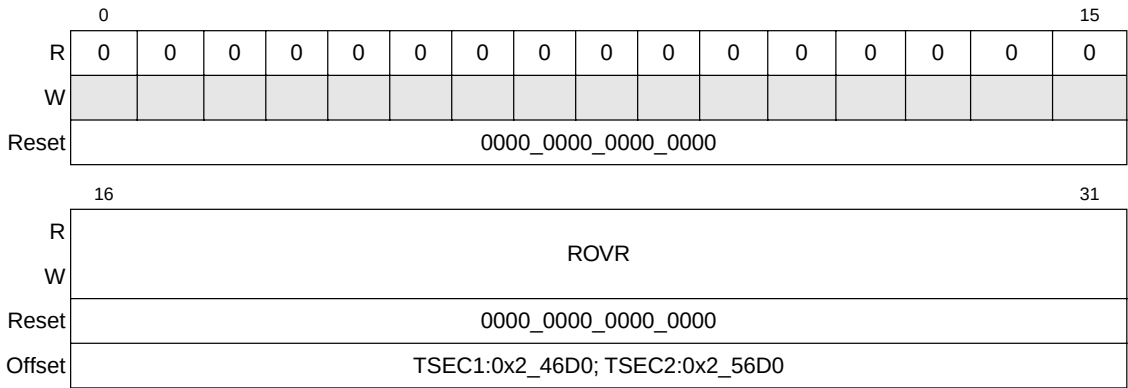


Figure 14-65. Receive Oversize Packet Counter Register Definition

Table 14-63 describes the fields of the ROVR register.

Table 14-63. 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. This count does not include range length errors.

14.5.3.7.22 Receive Fragments Counter (RFRG)

Figure 14-66 describes the definition for the RFRG register.

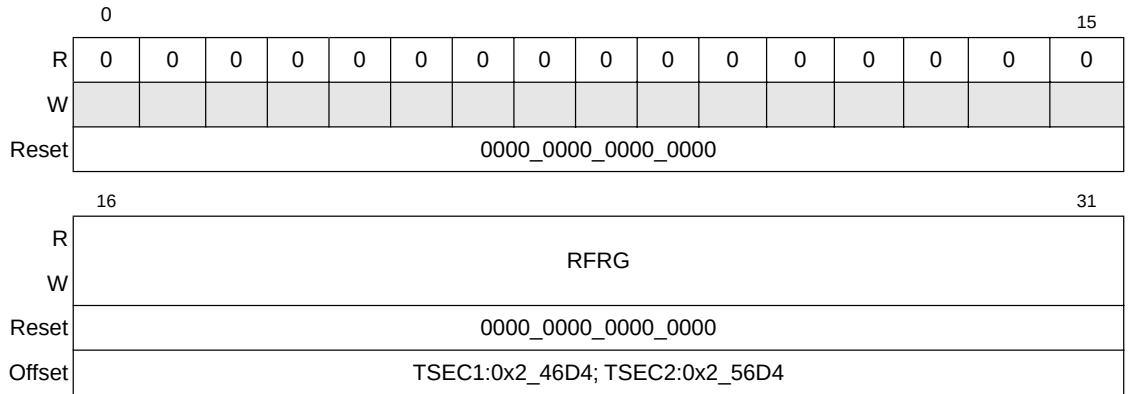


Figure 14-66. Receive Fragments Counter Register Definition

Table 14-64 describes the fields of the RFRG register.

Table 14-64. 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.

14.5.3.7.23 Receive Jabber Counter (RJBR)

Figure 14-67 describes the definition for the RJBR register.

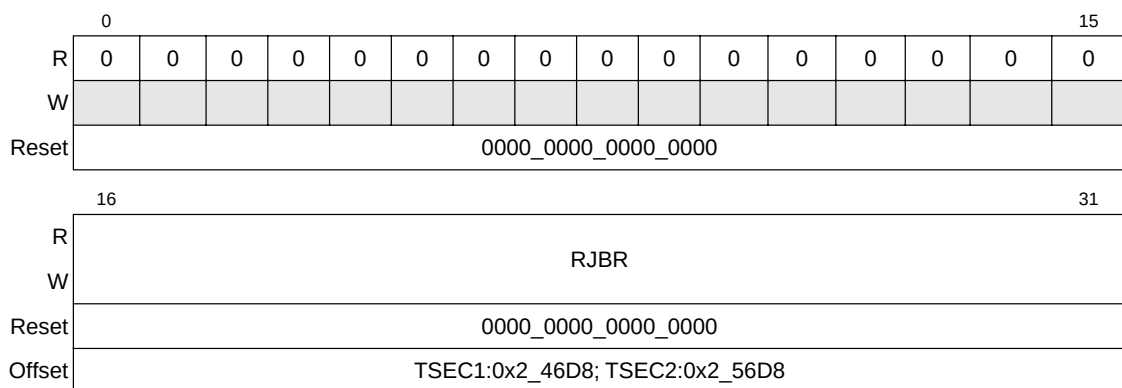
**Figure 14-67. Receive Jabber Counter Register Definition**

Table 14-65 describes the fields of the RJBR register.

Table 14-65. 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.

14.5.3.7.24 Receive Dropped Packet Counter (RDRP)

Figure 14-68 describes the definition for the RDRP register.

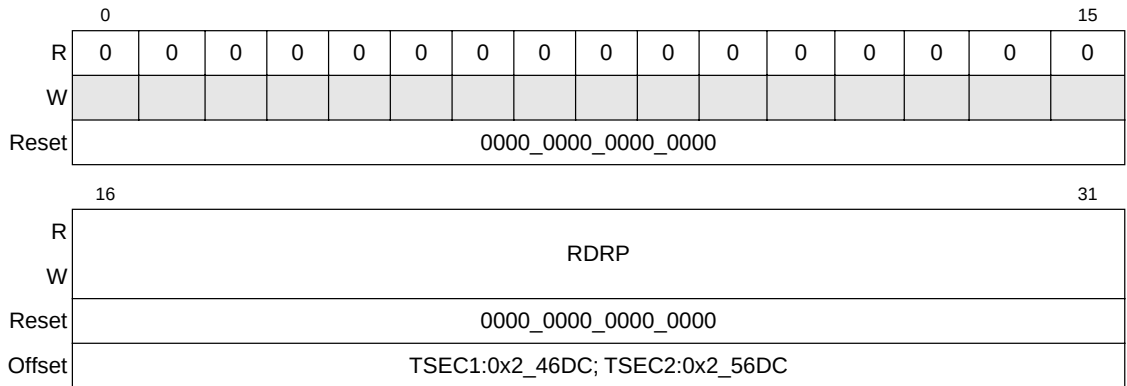


Figure 14-68. Receive Dropped Packet Counter Register Definition

Table 14-66 describes the fields of the RDRP register.

Table 14-66. 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.

14.5.3.7.25 Transmit Byte Counter (TBYT)

Figure 14-69 describes the definition for the TBYT register.

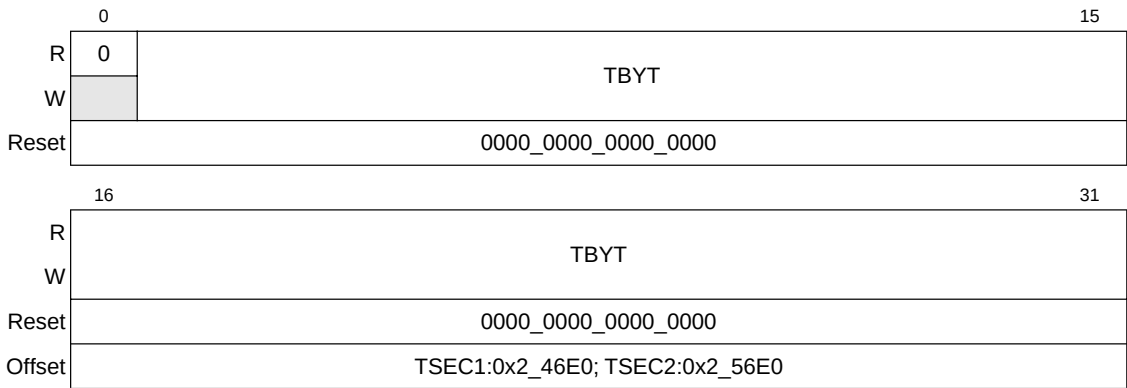


Figure 14-69. Transmit Byte Counter Register Definition

Table 14-67 describes the fields of the TBYT register.

Table 14-67. TBYT Field Descriptions

Bits	Name	Description
0	—	Reserved
1–31	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.

14.5.3.7.26 Transmit Packet Counter (TPKT)

Figure 14-70 describes the definition for the TPKT register.

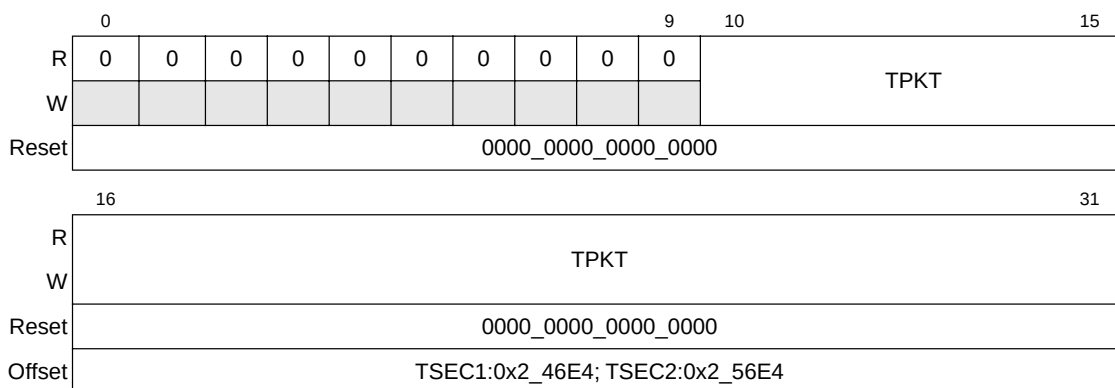
**Figure 14-70. Transmit Packet Counter Register Definition**

Table 14-68 describes the fields of the TPKT register.

Table 14-68. 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).

14.5.3.7.27 Transmit Multicast Packet Counter (TMCA)

Figure 14-71 describes the definition for the TMCA register.

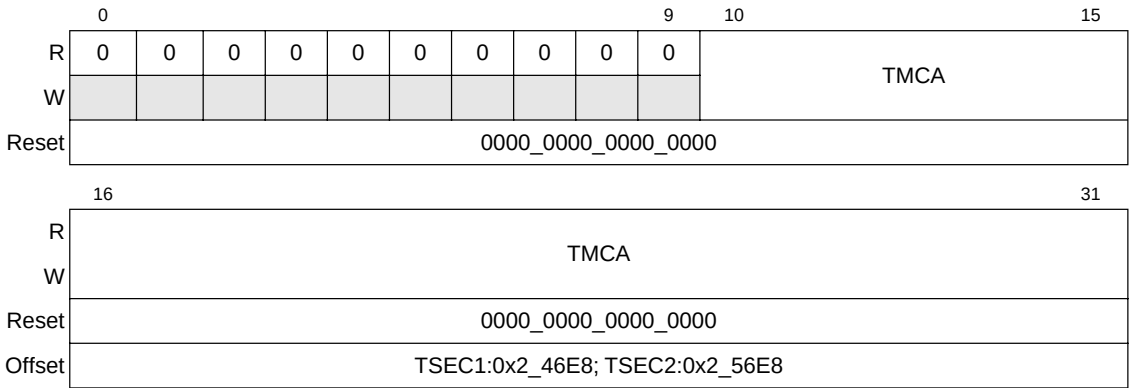


Figure 14-71. Transmit Multicast Packet Counter Register Definition

Table 14-69 describes the fields of the TMCA register.

Table 14-69. 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).

14.5.3.7.28 Transmit Broadcast Packet Counter (TBCA)

Figure 14-72 describes the definition for the TBCA register.

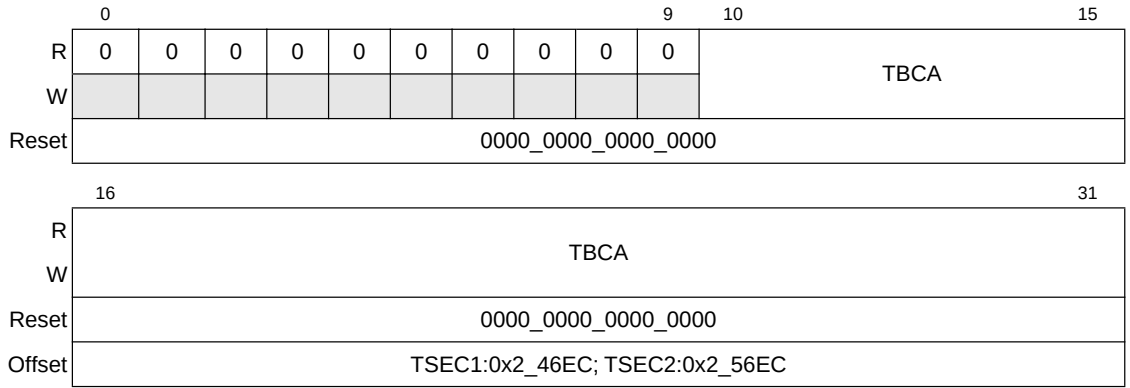


Figure 14-72. Transmit Broadcast Packet Counter Register Definition

Table 14-70 describes the fields of the TBCA register.

Table 14-70. 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).

14.5.3.7.29 Transmit Pause Control Frame Counter (TXPF)

Figure 14-73 describes the definition for the TXPF register.

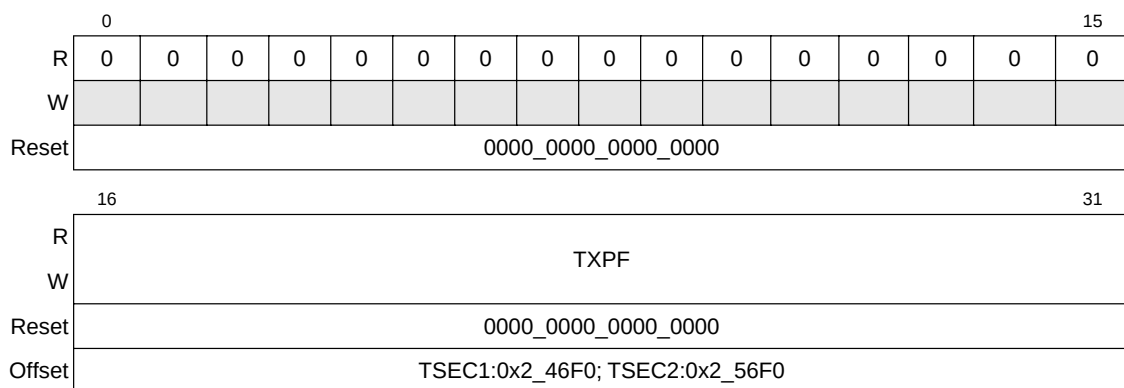
**Figure 14-73. Transmit Pause Control Frame Counter Register Definition**

Table 14-71 describes the fields of the TXPF register.

Table 14-71. 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.

14.5.3.7.30 Transmit Deferral Packet Counter (TDFR)

Figure 14-74 describes the definition for the TDFR register.

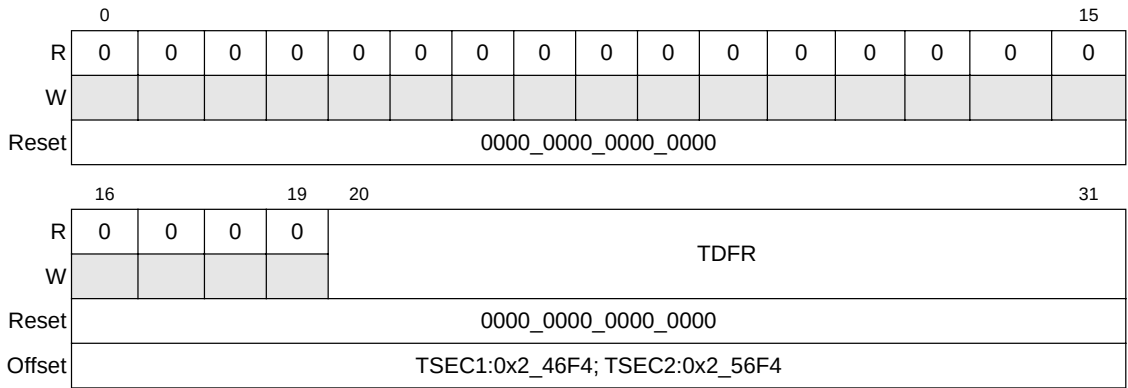


Figure 14-74. Transmit Deferral Packet Counter Register Definition

Table 14-72 describes the fields of the TDFR register.

Table 14-72. 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.

14.5.3.7.31 Transmit Excessive Deferral Packet Counter (TEDF)

Figure 14-75 describes the definition for the TEDF register.

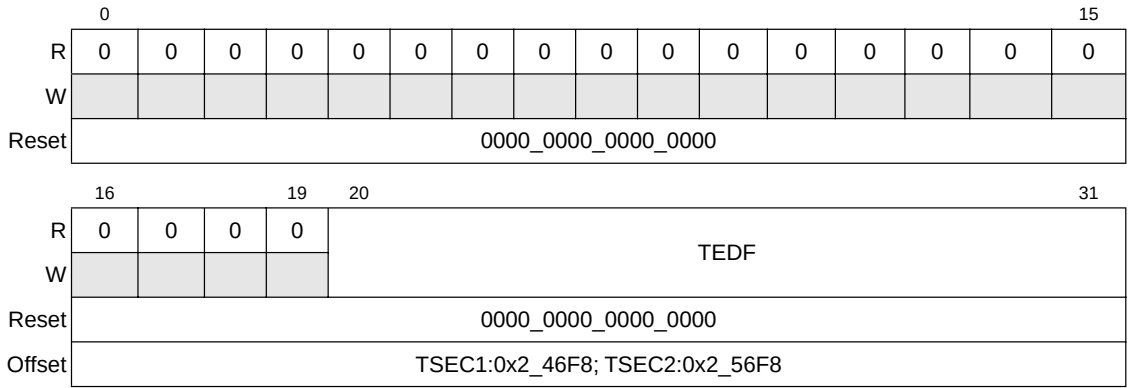


Figure 14-75. Transmit Excessive Deferral Packet Counter Register Definition

Table 14-73 describes the fields of the TEDF register.

Table 14-73. 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).

14.5.3.7.32 Transmit Single Collision Packet Counter (TSCL)

Figure 14-76 describes the definition for the TSCL register.

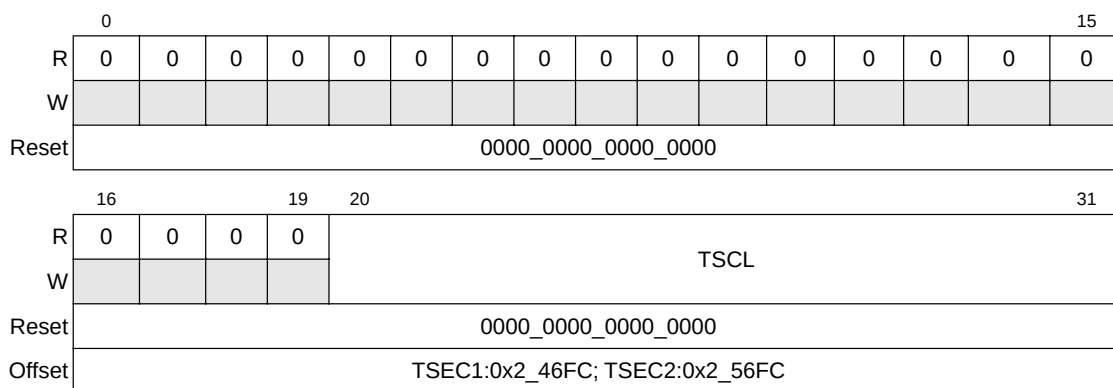
**Figure 14-76. Transmit Single Collision Packet Counter Register Definition**

Table 14-74 describes the fields of the TSCL register.

Table 14-74. 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.

14.5.3.7.33 Transmit Multiple Collision Packet Counter (TMCL)

Figure 14-77 describes the definition for the TMCL register.

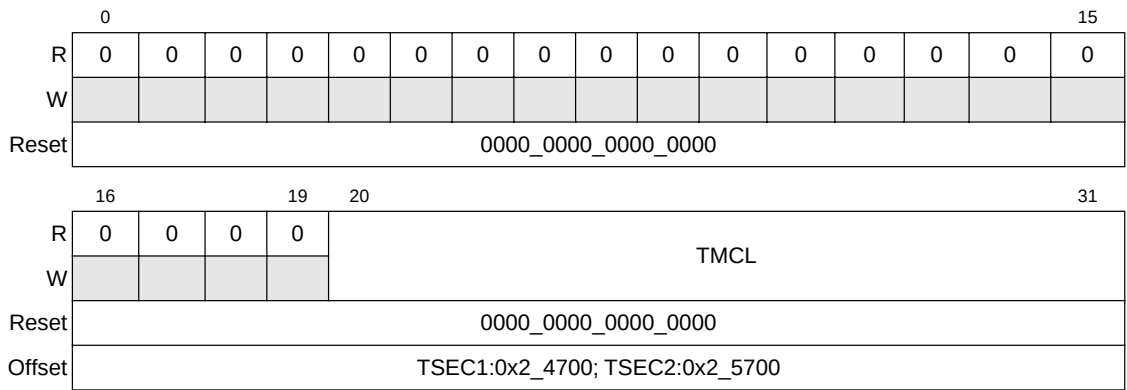


Figure 14-77. Transmit Multiple Collision Packet Counter Register Definition

Table 14-75 describes the fields of the TMCL register.

Table 14-75. 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 theHalf_Duplex[RETRANSMISSION MAXIMUM] field.

14.5.3.7.34 Transmit Late Collision Packet Counter (TLCL)

Figure 14-78 describes the definition for the TLCL register.

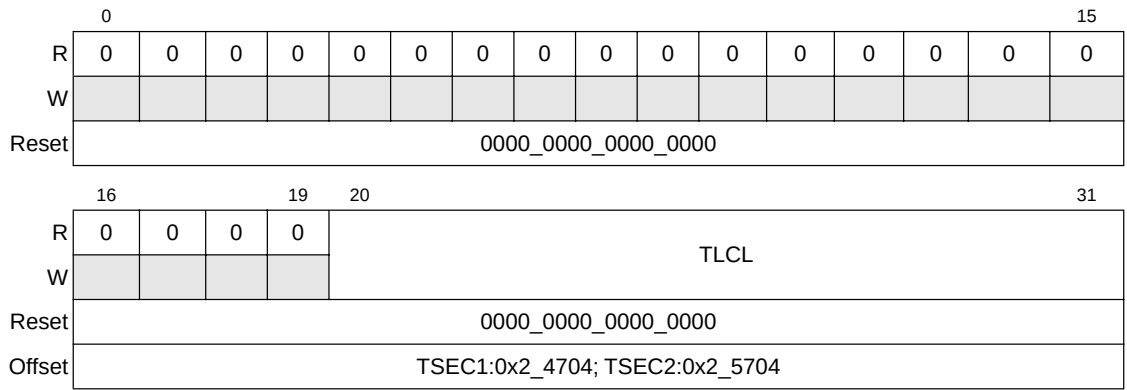


Figure 14-78. Transmit Late Collision Packet Counter Register Definition

Table 14-76 describes the fields of the TLCL register.

Table 14-76. 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.

14.5.3.7.35 Transmit Excessive Collision Packet Counter (TXCL)

Figure 14-79 describes the definition for the TXCL register.

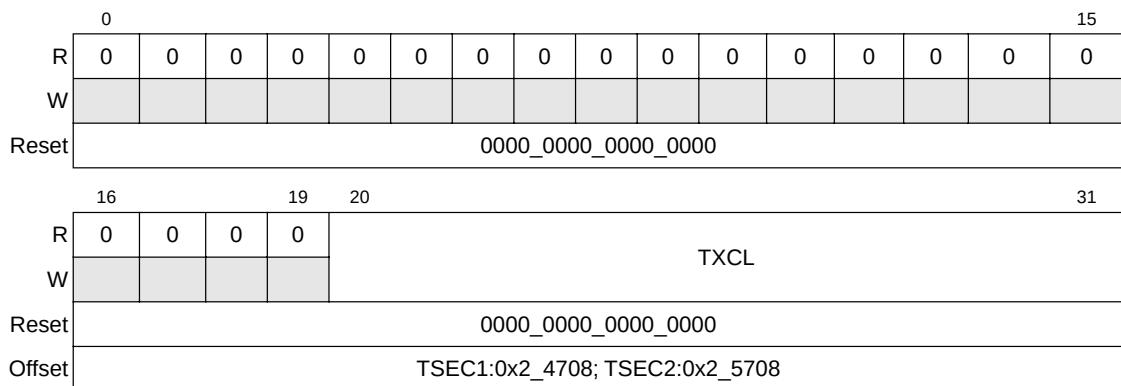
**Figure 14-79. Transmit Excessive Collision Packet Counter Register Definition**

Table 14-77 describes the fields of the TXCL register.

Table 14-77. 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.

14.5.3.7.36 Transmit Total Collision Counter (TNCL)

Figure 14-80 describes the definition for the TNCL register.

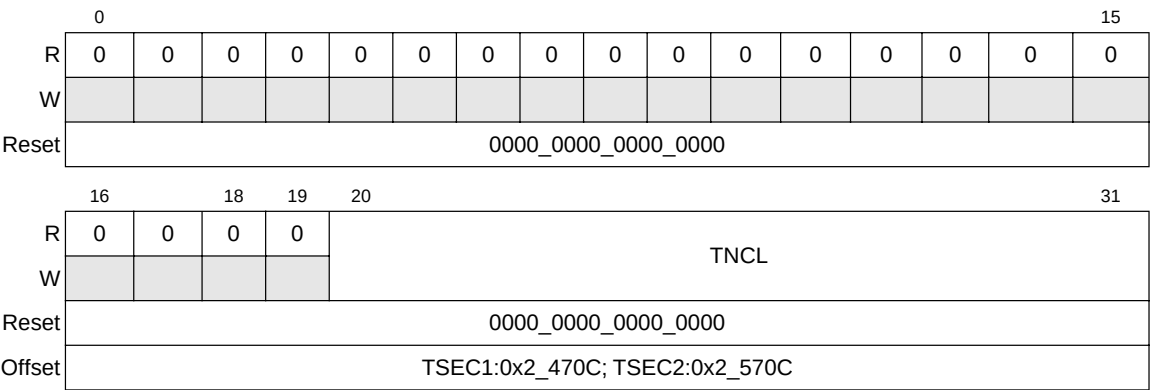


Figure 14-80. Transmit Total Collision Counter Register Definition

Table 14-78 describes the fields of the TNCL register.

Table 14-78. 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.

14.5.3.7.37 Transmit Drop Frame Counter (TDRP)

Figure 14-81 describes the definition for the TDRP register.

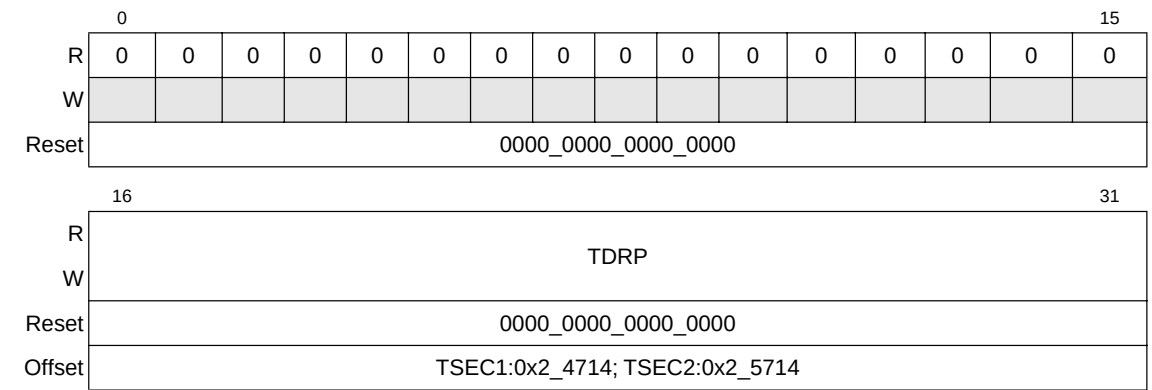


Figure 14-81. Transmit Drop Frame Counter Register Definition

Table 14-79 describes the fields of the TDRP register.

Table 14-79. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time input PFH is asserted.

14.5.3.7.38 Transmit Jabber Frame Counter (TJBR)

Figure 14-82 describes the definition for the TJBR register.

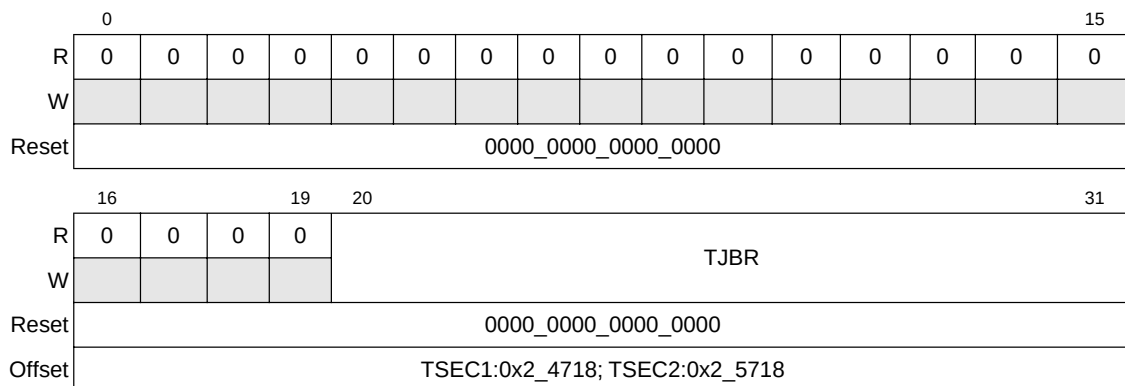
**Figure 14-82. Transmit Jabber Frame Counter Register Definition**

Table 14-80 describes the fields of the TJBR register.

Table 14-80. 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.

14.5.3.7.39 Transmit FCS Error Counter (TFCS)

Figure 14-83 describes the definition for the TFCS register.

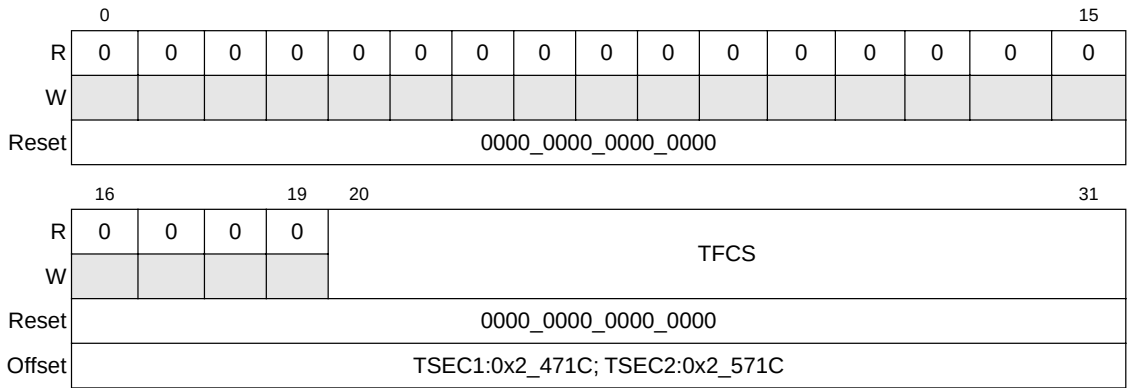


Figure 14-83. Transmit FCS Error Counter Register Definition

Table 14-81 describes the fields of the TFCS register.

Table 14-81. 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.

14.5.3.7.40 Transmit Control Frame Counter (TXCF)

Figure 14-84 describes the definition for the TXCF register.

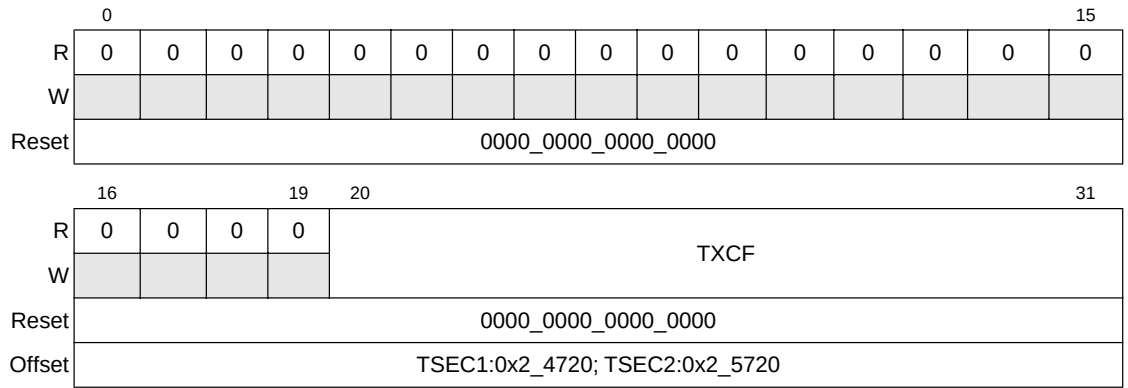


Figure 14-84. Transmit Control Frame Counter Register Definition

Table 14-82 describes the fields of the TXCF register.

Table 14-82. TXCF Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TXCF	Transmit control frame counter. Increments for every valid size frame with a type field signifying a control frame.

14.5.3.7.41 Transmit Oversize Frame Counter (TOVR)

Figure 14-85 describes the definition for the TOVR register.

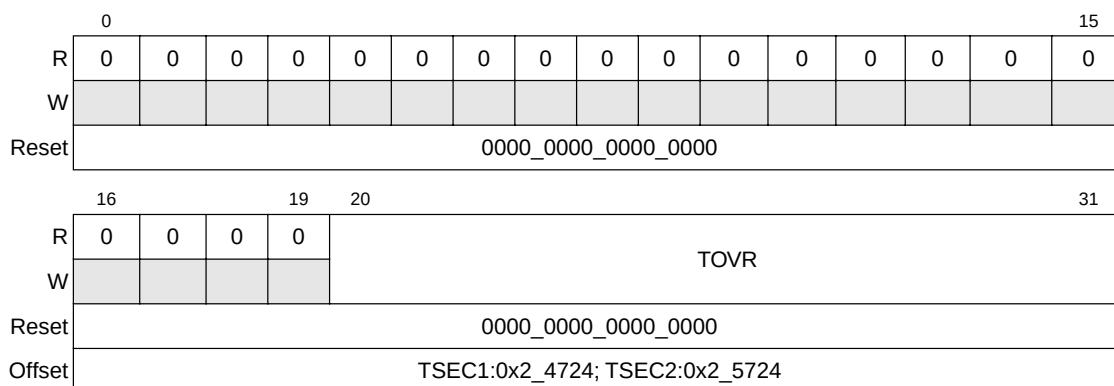
**Figure 14-85. Transmit Oversized Frame Counter Register Definition**

Table 14-83 describes the fields of the TOVR register.

Table 14-83. 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.

14.5.3.7.42 Transmit Undersize Frame Counter (TUND)

Figure 14-86 describes the definition for the TUND register.

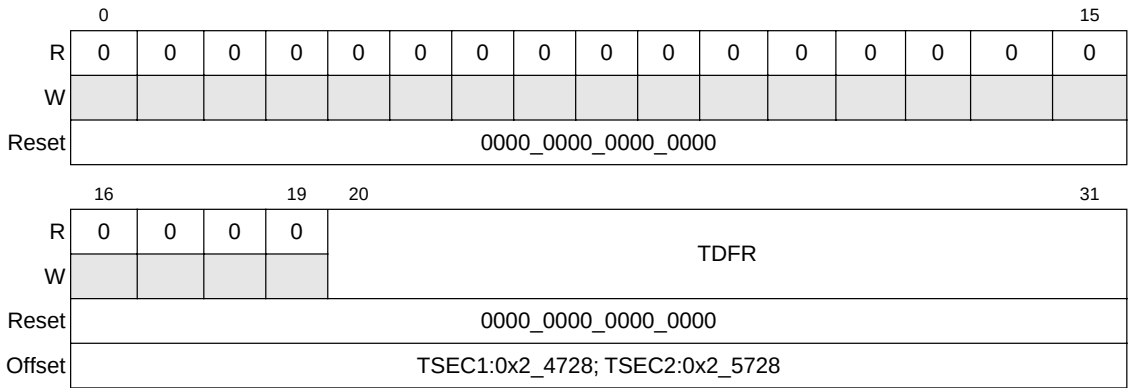


Figure 14-86. Transmit Undersize Frame Counter Register Definition

Table 14-84 describes the fields of the TUND register.

Table 14-84. TUND Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TDFR	Transmit undersize frame counter. Increments for every frame less then 64 bytes, with a correct FCS value.

14.5.3.7.43 Transmit Fragment Counter (TFRG)

Figure 14-87 describes the definition for the TFRG register.

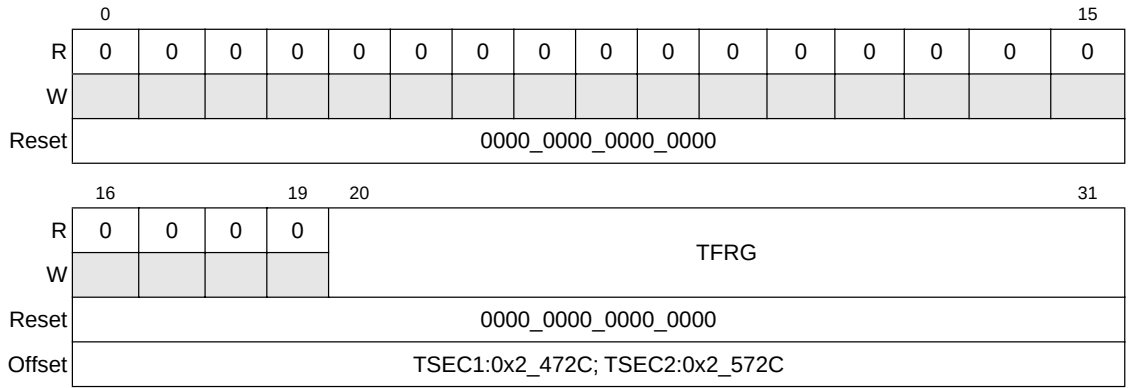


Figure 14-87. Transmit Fragment Counter Register Definition

Table 14-85 describes the fields of the TFRG register.

Table 14-85. TFRG Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less than 64 bytes, with an incorrect FCS value.

14.5.3.7.44 Carry Register One (CAR1)

Carry register bits are cleared on carry register writes when the respective bits are set. Figure 14-88 describes the definition for the CAR1 register.

	0	1	2	3	4	5	6	7							14	15
R	C1	C1	C1	C1	C1	C1	C1	0	0	0	0	0	0	0	0	C1
W	64	127	255	511	1K	MAX	MGV									RBY
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1
W	RPK	RFC	RMC	RBC	RXC	RXP	R XU	RAL	RFL	RCD	RCS	RUN	ROV	RFR	RJB	RDR
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4730; TSEC2:0x2_5730															

Figure 14-88. Carry Register One (CAR1) Register Definition

Table 14-86 describes the fields of the CAR1 register.

Table 14-86. CAR1 Field Descriptions

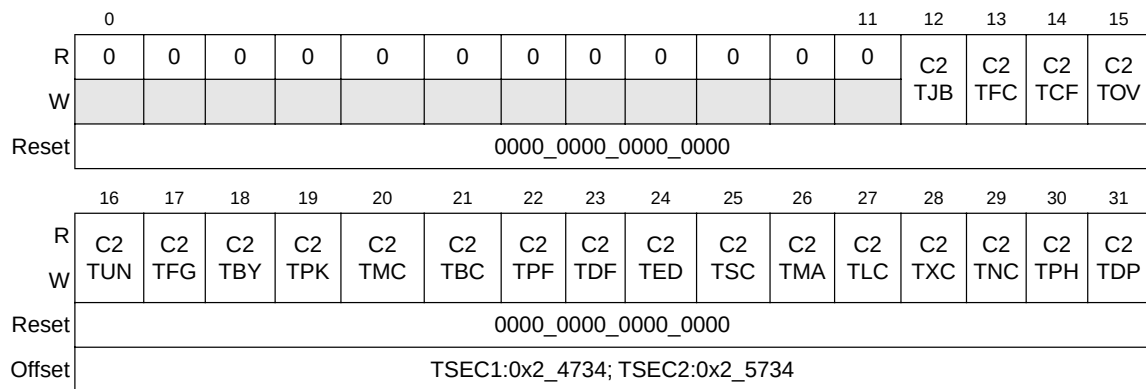
Bits	Name	Description
0	C164	Carry register 1 TR64 counter carry bit
1	C1127	Carry register 1 TR127 counter carry bit
2	C1255	Carry register 1 TR255 counter carry bit
3	C1511	Carry register 1 TR511 counter carry bit
4	C11k	Carry register 1 TR1K counter carry bit
5	C1MAX	Carry register 1 TRMAX counter carry bit
6	C1MGV	Carry register 1 TRMGV counter carry bit
7–14	—	Reserved
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

Table 14-86. CAR1 Field Descriptions (continued)

Bits	Name	Description
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
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

14.5.3.7.45 Carry Register Two (CAR2)

Figure 14-89 describes the definition for the CAR2 register.

**Figure 14-89. Carry Register Two (CAR2) Register Definition**

Carry register bits are cleared on carry register write when the respective bits are set. Table 14-87 describes the fields of the CAR2 register.

Table 14-87. CAR2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared
12	C2TJB	Carry register 2 TJBR Counter Carry bit
13	C2TFC	Carry register 2 TFCS Counter Carry bit

Table 14-87. CAR2 Field Descriptions (continued)

Bits	Name	Description
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
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

14.5.3.7.46 Carry Mask Register One (CAM1)

While one of the below mask bits are cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits all default to a set state. Figure 14-90 describes the definition for the CAM1 register.

	0	1	2	3	4	5	6	7							14	15
R	M1	M1	M1	M1	M1	M1	M1	0	0	0	0	0	0	0	0	M1
W	64	127	255	511	1K	MAX	MGV									RBY
Reset	1111_1110_0000_0001															
	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	1111_1111_1111_1111															
Offset	TSEC1:0x2_4738; TSEC2:0x2_5738															

Figure 14-90. Carry Mask Register One (CAM1) Register Definition

Table 14-88 describes the fields of the CAM1 register.

Table 14-88. 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
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
7–14	—	Reserved, should be cleared
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

14.5.3.7.47 Carry Mask Register Two (CAM2)

While one of the below mask bits are cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits default to a set state. Figure 14-91 describes the definition for the CAM2 register.

Memory Map/Register Definition

	0											11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	M2	M2	M2	M2
W													TJB	TFC	TCF	TOV
Reset	0000_0000_0000_1111															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2	M2
W	TUN	TFG	TBY	TPK	TMC	TBC	TPF	TDF	TED	TSC	TMA	TLC	TXC	TNC	TPH	TDP
Reset	1111_1111_1111_1111															
Offset	TSEC1:0x2_473C; TSEC2:0x2_573C															

Figure 14-91. Carry Mask Register Two (CAM2) Register Definition

Table 14-89 describes the fields of the CAM2 register.

Table 14-89. CAM2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared
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
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved, should be cleared
31	M2TDP	Mask register 2 TDRP counter carry bit mask

14.5.3.8 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 should further filter the address. See Section 14.6.2.6.2, “Hash Table Algorithm” for more information on the hash algorithm.

14.5.3.8.1 Individual Address Registers 0–7 (IADDRn)

The IADDRn register is written by the user. Together these registers represent 256 entries of the individual (unicast) 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 14-92 describes the definition for the IADDRn register.

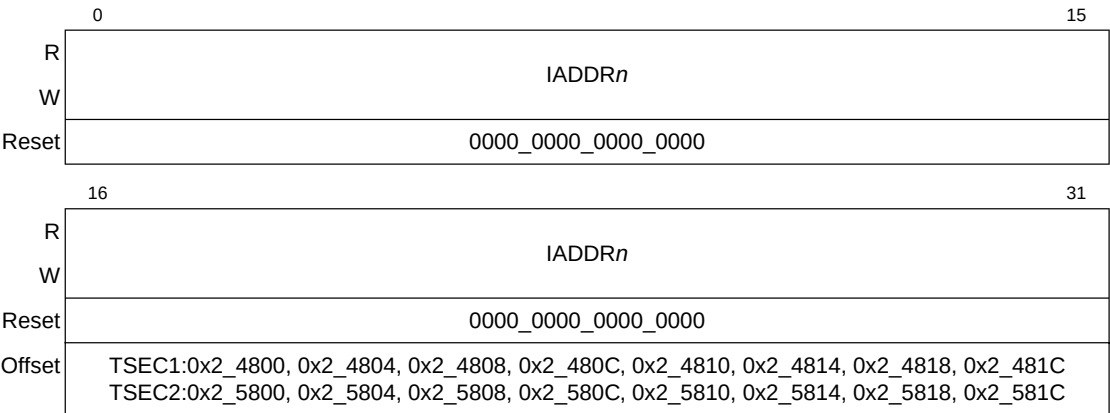


Figure 14-92. IADDRn Register Definition

Table 14-90 describes the fields of the IADDRn register.

Table 14-90. IADDRn Field Description

Bits	Name	Description
0–31	IADDRn	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.

14.5.3.8.2 Group Address Registers 0–7 (GADDRn)

The GADDRn register is 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 14-93 describes the definition for the $GADDR_n$ register.

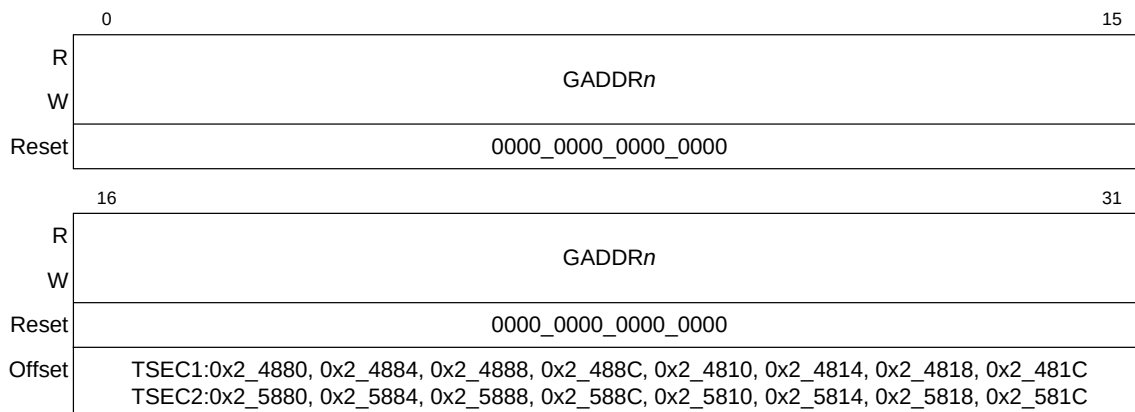


Figure 14-93. $GADDR_n$ Register Definition

Table 14-91 describes the fields of the $GADDR_n$ register.

Table 14-91. $GADDR_n$ Field Description

Bits	Name	Description
0–31	$GADDR_n$	Represents the 32-bit value associated with the corresponding register. $GADDR_0$ contains the high-order 32 bits of the 256 entry group hash table and $GADDR_7$ represents the low-order 32 bits.

14.5.3.8.3 Attribute Register (ATTR)

A frame is processed using the ATTR register, which is written by the user to determine where to file, when to extract, and how to store the received frames and their associated buffer descriptors. The reset conditions of this register are: no extraction (ATTR[EL] = 0), ATTR[BDLWT] (Buffer descriptor L2 cache write type), and ATTR[ELCWT] (Extracted L2 cache write type) are set to no allocation.

Figure 14-94 describes the definition for the ATTR register.

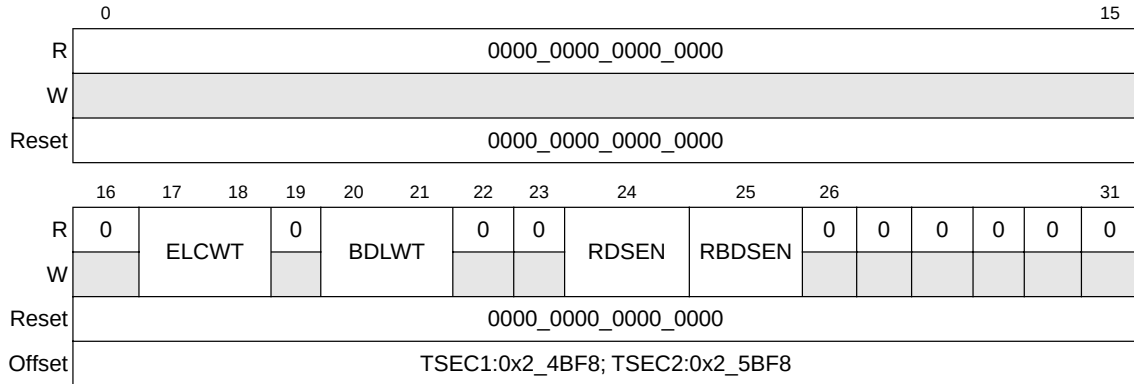
**Figure 14-94. ATTR Register Definition**

Table 14-92 describes the fields of the ATTR register.

Table 14-92. 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 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 Allocate and lock L2 cache line.
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. 01 Reserved 10 Allocate L2 cache line. 11 Allocate and lock L2 cache line.
22-23	—	Reserved
24	RDSEN	Rx data snoop enable. 0 Disables snooping of all receive frames data to memory. 1 Enables snooping of all receive frames data to memory.
25	RBDSEN	RxBD snoop enable. 0 Disables snooping of all receive BD memory accesses. 1 Enables snooping of all receive BD memory accesses.
26–31	—	Reserved

14.5.3.8.4 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 14-95 describes the definition for the ATTRELI register.

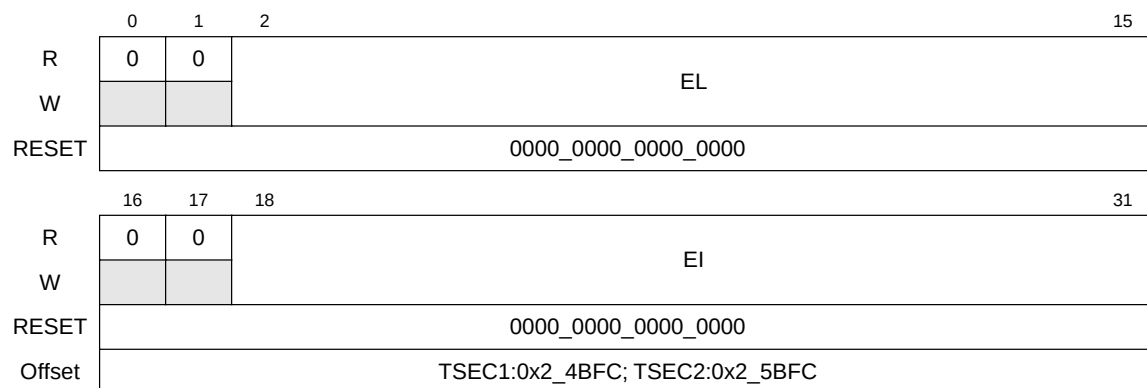
**Figure 14-95. ATTRELI Register Definition**

Table 14-93 describes the fields of the ATTRELI register.

Table 14-93. 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.

14.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

14.5.4.1 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 TSEC'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 14-94), 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 14.5.3.1, "TSEC General Control and Status Registers," and the TBI MII register set in Table 14-94. Notice that jitter diagnostics and TBI control are not IEEE 802.3 required registers and are only used for test and control of the TSEC TBI block. The TBI's TBI control register (TBI) is for configuring the TSEC 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 14-94. TBI MII Register Set

Offset Address	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				14.5.4/14-98
0x00	Control (CR)	R/W ¹	16 bits	14.5.4.2/14-99
0x01	Status (SR)	R, LH, LL	16 bits	14.5.4.3/14-100
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	14.5.4.3/14-100
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	14.5.4.5/14-103
0x06	AN expansion (ANEX)	R, LH	16 bits	14.5.4.6/14-104
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	14.5.4.7/14-105
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	14.5.4.8/14-106
0x0F	Extended status (EXST)	R	16 bits	14.5.4.9/14-107
0x10	Jitter diagnostics (JD)	R/W	16 bits	14.5.4.10/14-108
0x11	TBI control (TBICON)	R/W	16 bits	14.5.4.11/14-109

¹ R = means Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

14.5.4.2 Control Register (CR)

Figure 14-96 describes the definition for the CR register.

	0	1	2	3	4	5	6	7	8	9	10					15
R	PHY	Loop	Speed	AN	0	0	Reset	Full	0	Speed	0	0	0	0	0	
W	Reset	Back	[0]	Enable			AN	Duplex		[1]						
Reset	0001_0001_0100_0000															
Offset	0x00															

Figure 14-96. Control Register Definition

Table 14-95 describes the fields of the CR register.

Table 14-95. 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 control and status registers are returned to their default value. This in turn may change the internal state of the TBI and its link partner.															
1	Loop Back	Loop back. This bit is cleared by default. 0 Normal operation. 1 The transmit outputs of the TBI are connected to the receive inputs.															
2	Speed[0]	Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed.															
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															
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 Half-duplex operation. 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. <table border="1" data-bbox="586 1044 1115 1279"> <thead> <tr> <th>Maximum Operating Speed</th><th>Bit 2</th><th>Bit 9</th></tr> </thead> <tbody> <tr> <td>Reserved</td><td>0</td><td>0</td></tr> <tr> <td>Reserved</td><td>1</td><td>0</td></tr> <tr> <td>1000 Mbps</td><td>0</td><td>1</td></tr> <tr> <td>Reserved</td><td>1</td><td>1</td></tr> </tbody> </table>	Maximum Operating Speed	Bit 2	Bit 9	Reserved	0	0	Reserved	1	0	1000 Mbps	0	1	Reserved	1	1
Maximum Operating Speed	Bit 2	Bit 9															
Reserved	0	0															
Reserved	1	0															
1000 Mbps	0	1															
Reserved	1	1															
10–15	—	Reserved															

14.5.4.3 Status Register (SR)

Figure 14-97 describes the definition for the SR register.

	0					6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	Extend Status	0	No Pre	AN Done	Remote Fault	AN Ability	Link Status	0	Extend Ability
W															
Reset	0000_0001_0100_1001														
Offset	0x01														

Figure 14-97. Status Register Definition

Table 14-96 describes the fields of the SR register.

Table 14-96. 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 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.
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.

14.5.4.4 AN Advertisement Register (ANA)

Figure 14-98 describes the definition for the ANA register.

Memory Map/Register Definition

	0	1	2	3	4	6	7	8	9	10	11		15	
R	Next	Ack	Remote Fault	0	0	0	Pause		Half	Full	0	0	0	0
W	Page								Duplex	Duplex				
Reset	0000_0000_0000_0000													
Offset	0x04													

Figure 14-98. AN Advertisement Register Definition

Table 14-97 describes the fields of the ANA register.

Table 14-97. 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	Ack	Acknowledge. Write 0, ignore on read. This bit is read-only. (Reserved)															
2–3	Remote Fault	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 border="1"> <thead> <tr> <th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table>	RF1 bit[3]	RF2 bit[2]	Description	0	0	No error, link OK	0	1	Offline	1	0	Link_Failure	1	1	Auto-Negotiation_Error
RF1 bit[3]	RF2 bit[2]	Description															
0	0	No error, link OK															
0	1	Offline															
1	0	Link_Failure															
1	1	Auto-Negotiation_Error															
4–6	—	Reserved, should be cleared.															
7–8	Pause	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 14-98. <table border="1"> <thead> <tr> <th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr> </thead> <tbody> <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> </tbody> </table>	PAUSE bit[8]	ASM_DIR bit[7]	Capability	0	0	No PAUSE	0	1	Asymmetric PAUSE toward link partner	1	0	Symmetric PAUSE	1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device
PAUSE bit[8]	ASM_DIR bit[7]	Capability															
0	0	No PAUSE															
0	1	Asymmetric PAUSE toward link partner															
1	0	Symmetric PAUSE															
1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device															
9	Half Duplex	Half-duplex capability. 0 Designates local device as not capable of half-duplex operation. 1 Designates local device as capable of half-duplex operation.															

Table 14-97. ANA Field Descriptions (continued)

Bits	Name	Description
10	Full Duplex	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.
11–15	—	Reserved, should be cleared.

Table 14-98 describes the resolution of pause priority.

Table 14-98. 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
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

14.5.4.5 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 14-99 describes the definition for the ANLPBPA register.

	0	1	2	3	4	6	7	8	9	10	11				15
R	Next Page	Ack	Remote Fault	0	0	0	Pause		Half Duplex	Full Duplex	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
Offset	0x05														

Figure 14-99. AN Link Partner Base Page Ability Register Definition

Table 14-99 describes the fields of the ANLPBPA register.

Table 14-99. 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	Ack	Acknowledge. Ignore on read. This bit is read-only															
2–3	Remote Fault	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 border="1"> <thead> <tr> <th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr> </thead> <tbody> <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> </tbody> </table>	RF1 bit[3]	RF2 bit[2]	Description	0	0	No error, link OK	0	1	Offline	1	0	Link_Failure	1	1	Auto-Negotiation_Error
RF1 bit[3]	RF2 bit[2]	Description															
0	0	No error, link OK															
0	1	Offline															
1	0	Link_Failure															
1	1	Auto-Negotiation_Error															
4–6	—	Reserved, should be cleared.															
7–8	Pause	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 border="1"> <thead> <tr> <th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr> </thead> <tbody> <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> </tbody> </table>	PAUSE bit[8]	ASM_DIR bit[7]	Capability	0	0	No PAUSE	0	1	Asymmetric PAUSE toward link partner	1	0	Symmetric PAUSE	1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device
PAUSE bit[8]	ASM_DIR bit[7]	Capability															
0	0	No PAUSE															
0	1	Asymmetric PAUSE toward link partner															
1	0	Symmetric PAUSE															
1	1	Both symmetric PAUSE and Asymmetric PAUSE toward local device															
9	Half Duplex	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.															
10	Full Duplex	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.															
11–15	—	Reserved, should be cleared.															

14.5.4.6 AN Expansion Register (ANEX)

Figure 14-100 describes the definition for the ANEX register.

	0											12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	NP Able	Page Rx'd	0
W															
Reset	0000_0000_0000_0110														
Offset	0x06														

Figure 14-100. AN Expansion Register Definition

Table 14-100 describes the fields of the ANEX register.

Table 14-100. 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.

14.5.4.7 AN Next Page Transmit Register (ANNPT)

Figure 14-101 describes the definition for the ANNPT register.

	0	1	2	3	4	5		15							
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message / Un-formatted Code Field									
W															
Reset	0000_0000_0000_0000														
Offset	0x07														

Figure 14-101. AN Next Page Transmit Register Definition

Table 14-101 describes the fields of the ANNPT register.

Table 14-101. 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	Ack	Write 0, ignore on read. [Reference MII bit 7.14] This bit is read-only.
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.
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]

14.5.4.8 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 14-102 describes the definition for the ANLPANP register.

	0	1	2	3	4	5										15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message / Un-formatted Code Field										
W																
Reset	0000_0000_0000_0000															
Offset	0x08															

Figure 14-102. AN Link Partner Ability Next Page Register Definition

Table 14-102 describes the fields of the ANLPANP register.

Table 14-102. 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	Ack	Ignore on read. [Reference MII bit 8.14] This bit is read-only.
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.

14.5.4.9 Extended Status Register (EXST)

Figure 14-103 describes the definition for the EXST register.

	0	1	2	3	4										15
R	1000X Full	1000X Half	1000T Full	1000T Half	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	1010_0000_0000_0000														
Offset	0x0F														

Figure 14-103. Extended Status Register Definition

Table 14-103 describes the fields of the EXST register.

Table 14-103. 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

14.5.4.10 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. It may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns. Figure 14-104 describes the definition for the JD register.

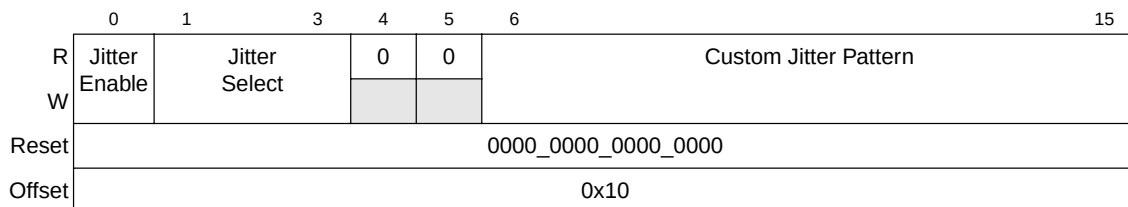
**Figure 14-104. Jitter Diagnostics Register Definition**

Table 14-104 describes the fields of the JD register.

Table 14-104. JD Field Description

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.																																				
		<table><tr><th>Jitter Pattern Select</th><th>bit[1]</th><th>bit[2]</th><th>bit[3]</th></tr><tr><td>User defined uses custom jitter pattern, bits 6–15</td><td>0</td><td>0</td><td>0</td></tr><tr><td>High frequency (+/- D21.5) 101010101010101010101010101010101010...</td><td>0</td><td>0</td><td>1</td></tr><tr><td>Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100...</td><td>0</td><td>1</td><td>0</td></tr><tr><td>Low frequency 1111100000111110000011111000001111100000...</td><td>0</td><td>1</td><td>1</td></tr><tr><td>Complex pattern (10'h17c,10'h0c9,10'h0e5,10'h2a3, 10'h17c,...)</td><td>1</td><td>0</td><td>0</td></tr><tr><td>Square Wave (- K28.7) 0011111000001111100000111110000011111000...</td><td>1</td><td>0</td><td>1</td></tr><tr><td>Reserved</td><td>1</td><td>1</td><td>0</td></tr><tr><td>Reserved</td><td>1</td><td>1</td><td>1</td></tr></table>	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) 101010101010101010101010101010101010...	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
		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) 101010101010101010101010101010101010...	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.																																				

14.5.4.11 TBI Control Register (TBICON)

Figure 14-105 describes the definition for the TBICON register.

	0	1	2	3	4	6	7	8	9	10	11	12	13	14	15	
R	Soft_Reset	0	Disable Rx Dis	Disable Tx Dis	0	0	0	AN Sense	0	0	Clock Select	MII Mode	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x10															

Figure 14-105. TBI Control Register Definition

Table 14-105 describes the fields of the TBICON register.

Table 14-105. TBICON Field Description

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation. 1 Resets the functional modules in theTBI.
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'.
11	MII 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.
12–13	—	Reserved
14–15	—	Reserved

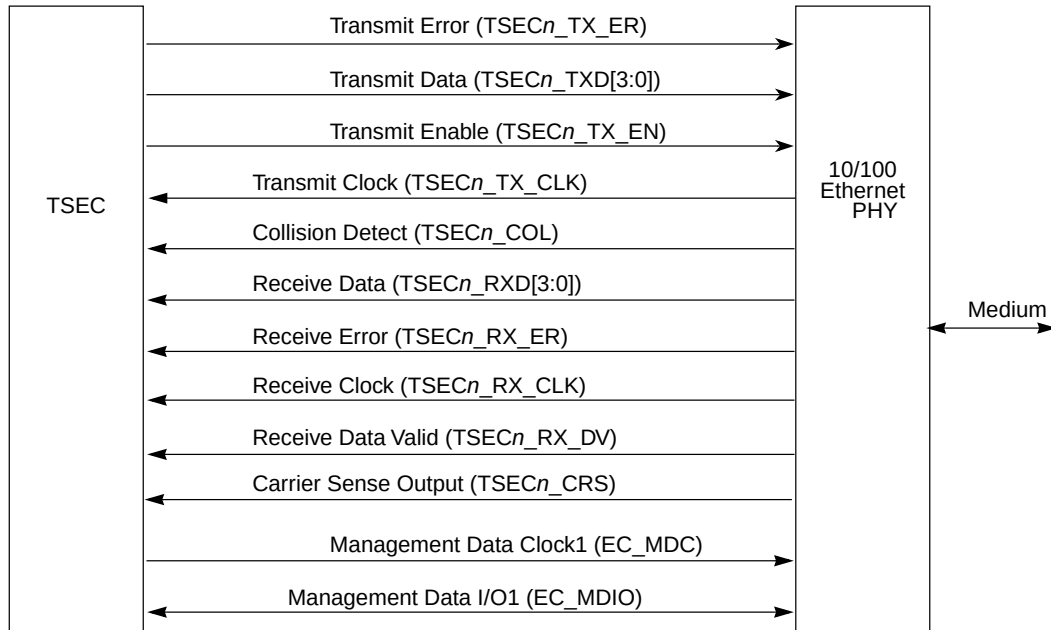
14.6 Functional Description

14.6.1 Connecting to Physical Interfaces

This section describes how to connect the TSEC to various interfaces: MII, GMII, RGMII,RTBI and TBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification because the PHY's 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).

14.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the TSEC. Figure 14-106 depicts the basic components of the MII including the signals required to establish TSEC module connection with a PHY.



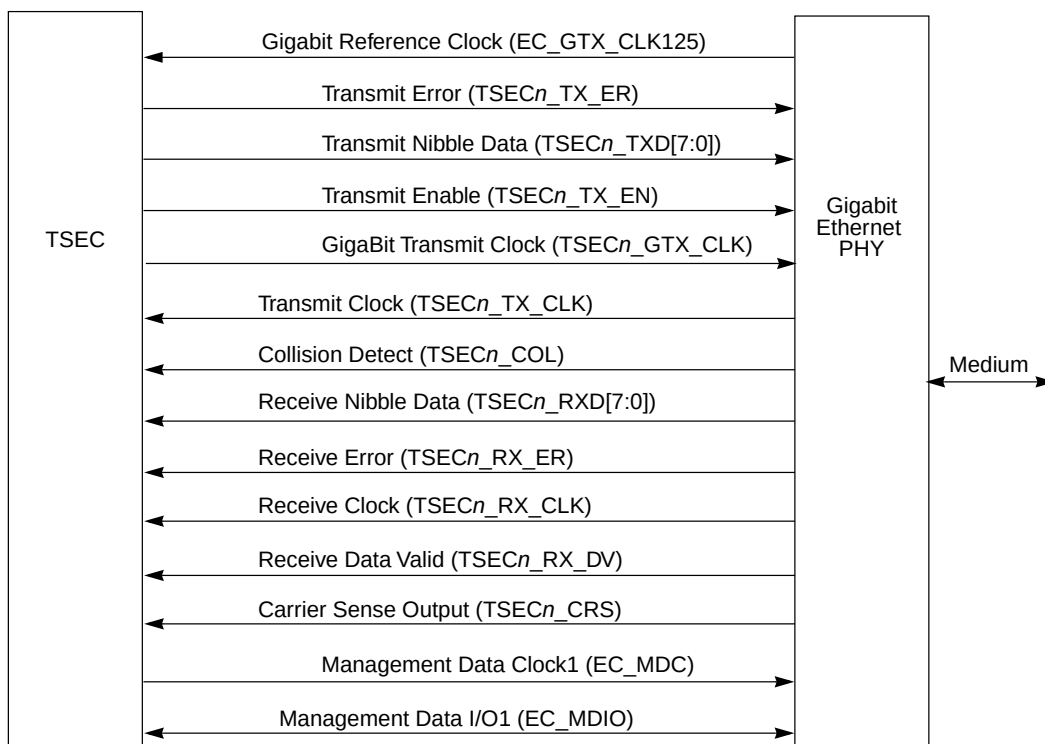
¹ The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-106. TSEC-MII Connection

An MII interface has 18 signals (including the EC_MDC and EC_MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

14.6.1.2 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the TSEC. Figure 14-107 depicts the basic components of the GMII including the signals required to establish the TSEC module connection with a PHY.



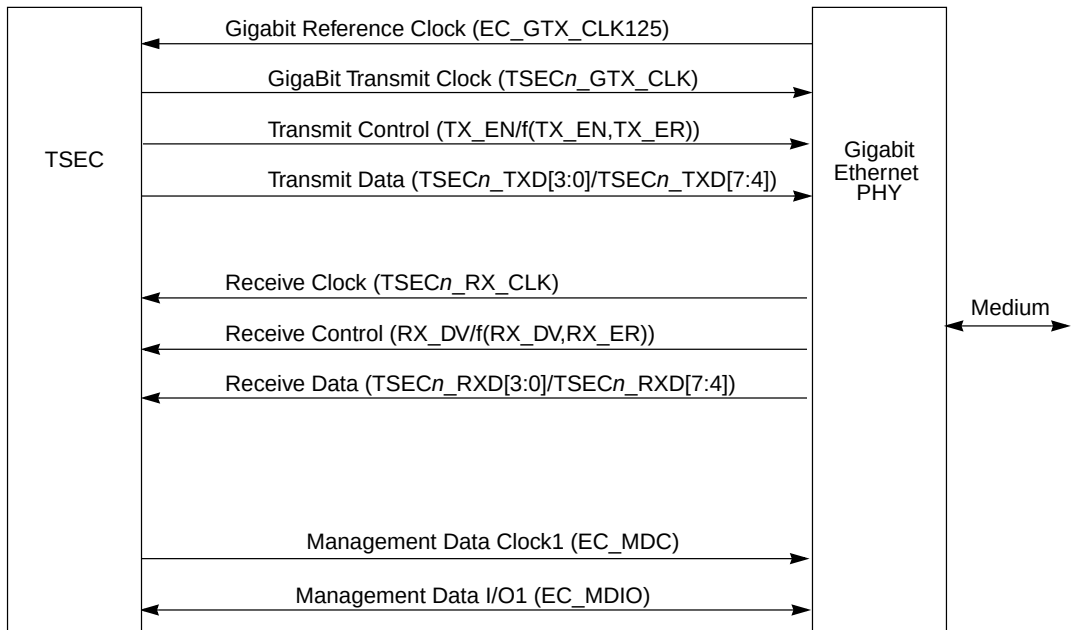
¹ The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-107. TSEC-GMII Connection

A GMII interface has 28 signals (TSECn_GTX_CLK + EC_GTX_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

14.6.1.3 Reduced Gigabit Media-Independent Interface (RGMI)

This section describes the reduced gigabit media-independent interface (RGMI) intended to be used between the PHYs and the GMII MAC. The RGMI is an alternative to the IEEE802.3u MII, the IEEE802.3z GMII and the TBI. The RGMI reduces the number of pins required to interconnect the MAC and the PHY from a maximum of 28 pins (GMII) to 15 pins (EC_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. Figure 14-108 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 RGMI is implemented as defined by the RGMI specification Version 1.2a 9/22/00.

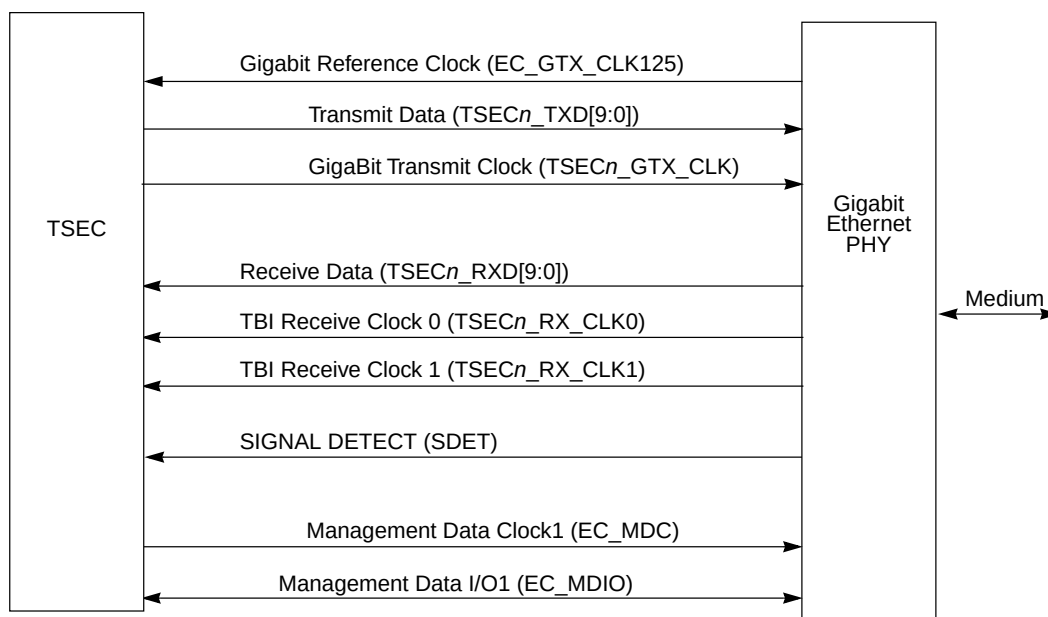


¹ The management signals (EC_MDC and EC_MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 14-108. TSEC-RGMII Connection

14.6.1.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the TSEC to implement a standard SERDES interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-109 depicts the basic components of the TBI including the signals required to establish TSEC module connection with a PHY. RBC0 and RBC1 are differential 62.5 MHz receive clocks.



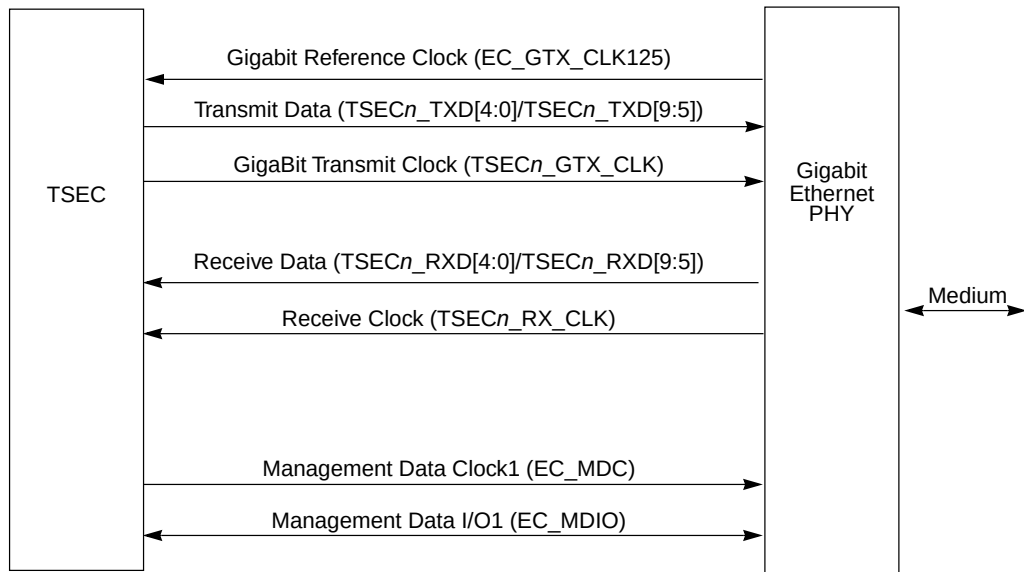
¹ The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-109. TSEC-TBI Connection

A TBI interface has 26 signals (EC_GTX_CLK125 included) for connecting to an Ethernet PHY, as defined by IEEE 802.3z GMII and TBI standards.

14.6.1.5 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the TSEC to implement a reduced pin count version of a SERDES interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-110 depicts the basic components of the RTBI including the signals required to establish TSEC module connection with a PHY.



¹ The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-110. TSEC-RTBI Connection

A RTBI interface has 15 signals (EC_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.

Table 14-106 describes the pin multiplexing for GMII, MII and TBI interfaces.

Table 14-106. GMII, MII, and TBI Pins Multiplexing

TSEC Pins			GMII Interface			MII Interface			TBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 25			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 3.3			Voltage[V] 3.3			Voltage[V] 3.3		
Signals (TSECn_)	I/O	No. of Pins	Signals (TSECn_)	I/O	No. of Pins	Signals (TSECn_)	I/O	No. of Pins	Signals (TSECn_)	I/O	No. of Pins
GTX_CLK	O	1	GTX_CLK	O	1				GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1	TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TxD[0]	O	1	TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TxD[1]	O	1	TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TxD[2]	O	1	TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TxD[3]	O	1	TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TxD[4]	O	1				TCG[4]	O	1
TxD[5]	O	1	TxD[5]	O	1				TCG[5]	O	1

Table 14-106. GMII, MII, and TBI Pins Multiplexing (continued)

TSEC Pins			GMII Interface			MII Interface			TBI Interface		
Frequency [MHz] 125			Frequency [MHz] 125			Frequency [MHz] 25			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 3.3			Voltage[V] 3.3			Voltage[V] 3.3		
TxD[6]	O	1	TxD[6]	O	1				TCG[6]	O	1
TxD[7]	O	1	TxD[7]	O	1				TCG[7]	O	1
TX_EN	O	1	TX_EN	O	1	TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TX_ER	O	1	TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK	I	1	RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RxD[0]	I	1	RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RxD[1]	I	1	RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RxD[2]	I	1	RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RxD[3]	I	1	RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RxD[4]	I	1				RCG[4]	I	1
RxD[5]	I	1	RxD[5]	I	1				RCG[5]	I	1
RxD[6]	I	1	RxD[6]	I	1				RCG[6]	I	1
RxD[7]	I	1	RxD[7]	I	1				RCG[7]	I	1
RX_DV	I	1	RX_DV	I	1	RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RX_ER	I	1	RX_ER	I	1	RCG[9]	I	1
COL	I	1				COL	I	1			
CRS	I	1				CRS	I	1	SDET	I	1
Sum		25	Sum		23	Sum		16	Sum		24

Table 14-107 describes the pin multiplexing for RGMII and RTBI interfaces.

Table 14-107. RGMII and RTBI Pins Multiplexing

TSEC Pins			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 Pins	Signals (TSEC _n)	I/O	No. of Pins	Signals (TSEC _n)	I/O	No. of Pins
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

Table 14-107. RGMII and RTBI Pins Multiplexing (continued)

TSEC Pins			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 (TSECn_)	I/O	No. of Pins	Signals (TSECn_)	I/O	No. of Pins	Signals (TSECn_)	I/O	No. of Pins
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 14-108 describes the pins shared by all interfaces.

Table 14-108. Shared Pins

Signals	I/O	# of pins	Function
EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	Management interface clock
EC_GTX_CLK125	I	1	Reference clock
Sum		3	

14.6.2 Gigabit Ethernet Channel Operation

This section describes the operation of the TSEC. 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.

14.6.2.1 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 TSEC.

14.6.2.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All TSEC's registers and control logic are reset to their default states after a hard reset has occurred.

14.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic TSEC 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 14-2 for the register list. Table 14-109 describes the minimum steps for register initialization.

Table 14-109. Steps of 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

Table 14-109. Steps of Minimum Register Initialization (continued)

Description
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT
7. Initialize IMASK
8. Initialize IADDR _n
9. Initialize GADDR _n
10. Initialize RCTRL
11. Initialize DMACTRL

After the initialization of registers is performed, the user must execute the following steps in the order described below to bring the TSEC 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 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.
3. 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.
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 14.6.3.1, “Data Buffer Descriptor,” for more information.

14.6.2.2 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 will be used, user must load the pointers into the TBASE register. Likewise if a new set of Rx buffer descriptors will be used, the RBASE register must be written with the new pointer.

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 till both set
3. Set SOFT_RESET bit in MACCFG1 register
4. Clear SOFT_RESET bit in MACCFG1 register
5. Load TBASE with new Tx BD pointer
6. Load RBASE with new Rx BD pointer
7. Setup other MAC registers (MACCFG2, MAXFRM, etc)
8. Setup WWP, WOP, TOD bits in DMACTRL register
9. Clear THLT bit in RSTAT register by writing 1 to the bit
10. Clear GRS/GTS bits in DMACTRL (do not change other bits)
11. Enable Tx_EN/Rx_EN in MACCFG1 register

14.6.2.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the TSEC begins to poll the first transmit buffer descriptor (TxBD) in the TxBD ring every 512 transmit clocks. If TxBD[R] is set, TSEC 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 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 begins 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 preamble sequence, start frame delimiter, and frame information in that order; see Figure 14-3. 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.

There is one internal buffer for out-of-sequence flow control frames. While the TSEC is between frames, this buffer is polled. The buffer must contain the whole frame. Once the TSEC 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. 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 Tx BD
- TxBD[TC] is set in first Tx BD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC] 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 be useful for transmitting expedited data ahead of previously-linked buffers or for error situations. If this bit is set, the TSEC 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 the DMACTRL[GTS] is cleared, the TSEC resumes transmission with the next frame.

While the TSEC 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.

14.6.2.4 Gigabit Ethernet Frame Reception

The TSEC 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 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 TSEC controller begins to perform the frame recognition function through destination address (DA) recognition (see frame recognition section 3.3.5-pg. 24). Based on this match the frame can be accepted or rejected. Once accepted, the TSEC 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), TSEC 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 TSEC controller 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 RxBD's 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. 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 Enable] 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 TSEC 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 TSEC hardware without fear of conflict. After DMACTRL[GRS] is cleared, the TSEC 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.

14.6.2.5 RMON Support

By using promiscuous mode, the TSEC can automatically gather network statistics required for RMON without the need to receive all addresses. The RMON MIB group1, RMON MIB group2, 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.

14.6.2.6 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

14.6.2.6.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods.

Figure 14-111 depicts 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.

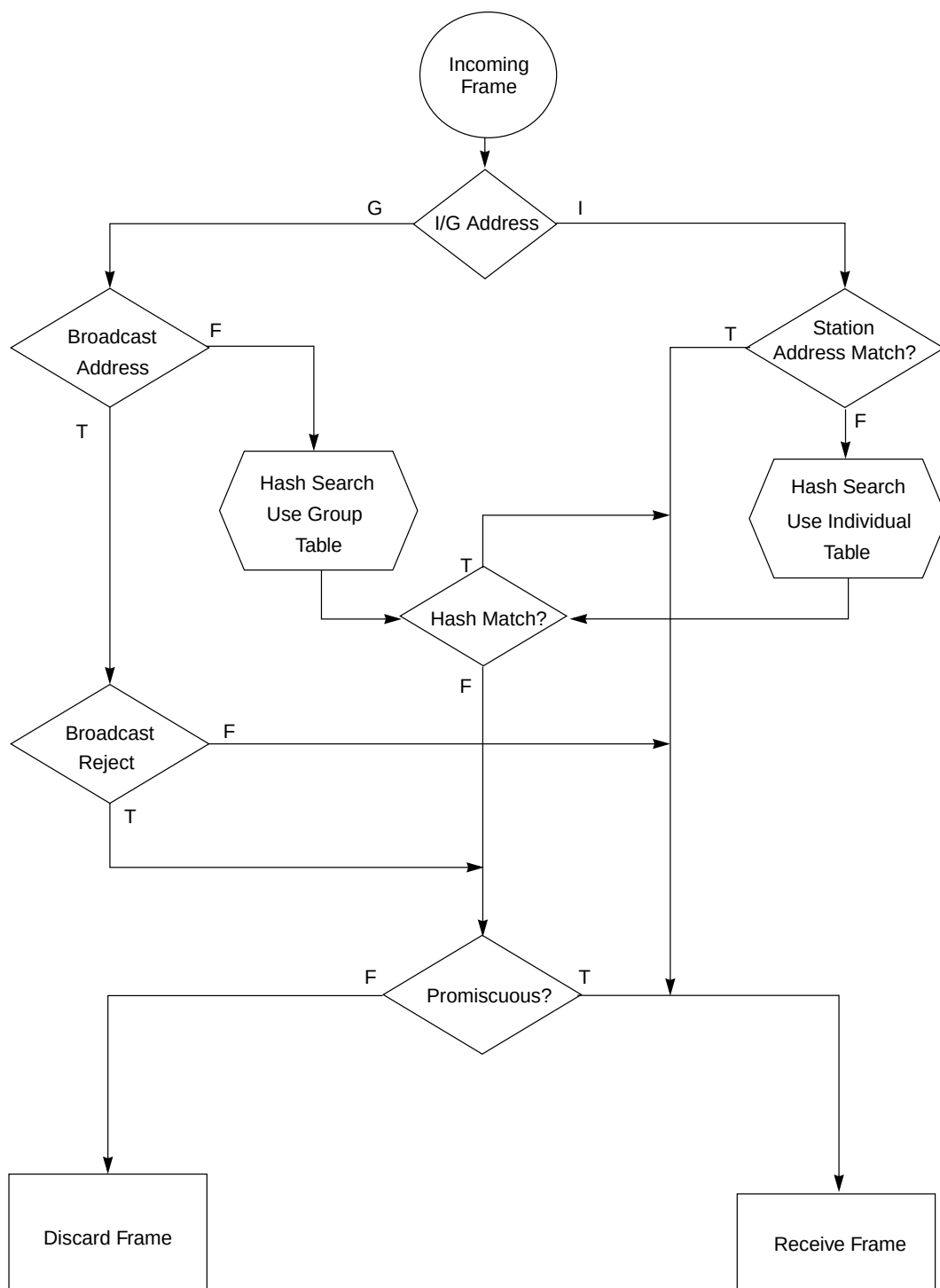


Figure 14-111. Ethernet Address Recognition Flowchart

The Ethernet controller compares the destination address field of the received frame 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, 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 Section 14.5.3.4.1, “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.

14.6.2.6.2 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.

14.6.2.7 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 14-110 lists the flow-control frame structure.

Table 14-110. 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 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[RFCE] 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.

14.6.2.8 Interrupt Handling

The following describes what usually occurs within a TSEC 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 TXE] 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 TSEC; 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 14-111.

- Obtain data from the RxB0 if IEVENT[RXC,RXB0 or RXF0] is set. If the receive speed is fast or the interrupt delay is long, the TSEC may have received more than one RxB0; thus, it is important to check more than just one RxB0 during interrupt handling. Typically, all RxB0s in the interrupt handler are processed until one is found with E set. Because the TSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 14-112.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 14-111. Non-Error Transmit Interrupts

Interrupt	Description	Action taken by TSEC
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.	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 14-112. Non-Error Receive Interrupts

Interrupt	Description	Action taken by TSEC
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.	Programmable "write with response" RxB0 to memory before setting IEVENT[RXB0].
RXF	Receive frame: A frame was received and the last receive buffer descriptor (RxB0) of that frame was updated.	Programmable "write with response" to memory on the last RxB0 before setting IEVENT[RXF0].

14.6.2.9 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.

14.6.2.10 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. In TBI mode, setting TBI CR[Loop Back] causes the transmit outputs of the TBI to be connected to the receive inputs. Clearing this bit results in normal operation. It is cleared by default.

14.6.2.11 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in Table 14-113.

Table 14-113. 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, 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).
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 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[HFE] is a 0. The controller sets IEVENT[BABT] and continues without interruption.

Reception errors are described in Table 14-114.

Table 14-114. 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 RxB[OV], sets RxB[L], closes the buffer, and sets IEVENT[RXF0]. 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[RHLT] bit is set. The halted queue resumes reception once the RSTAT[RHLT] bit is cleared.
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 (RxB[NO]) error is reported, IEVENT[RXF0] is set, and the alignment error counter increments. The TSEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.
CRC error	If a CRC error occurs, the controller sets RxB[CR], closes the buffer, and sets IEVENT[RXF0]. This TSEC 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[RHLT] bit is set. The halted queue resumes reception once the RSTAT[RHLT] 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

14.6.3 Buffer Descriptors

The TSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse.

Drawing from the MPC8260 FEC BD programming model, the TSEC descriptor base registers point to the beginning of BD rings. The TSEC BD also expands upon the MPC8260 BD model to accommodate TSEC's unique features, and to provide an expansion path for future requirements.

There is an 8-byte data BD format designed to be similar to the existing MPC8260 BD model.

The TSEC is capable of placing extracted 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. This results in substantial improvement in throughput and hence reduction in latency.

14.6.3.1 Data Buffer Descriptor

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 14-112). Data BDs encapsulate all information necessary for the TSEC 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

Functional Description

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 TSEC and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see Figure 14-113), all BDs must reside sequentially in memory. The TSEC 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 TSEC 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 TSEC.

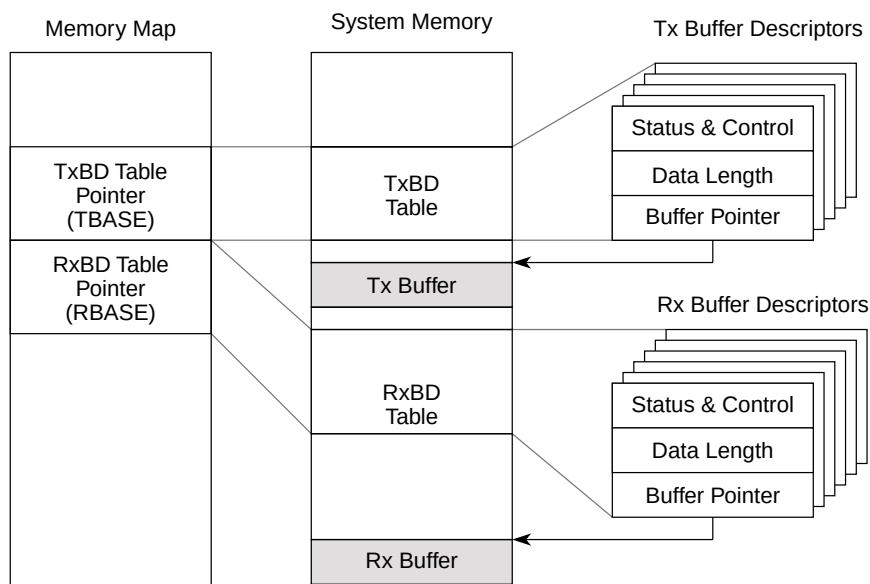


Figure 14-112. Example of TSEC Memory Structure for BD

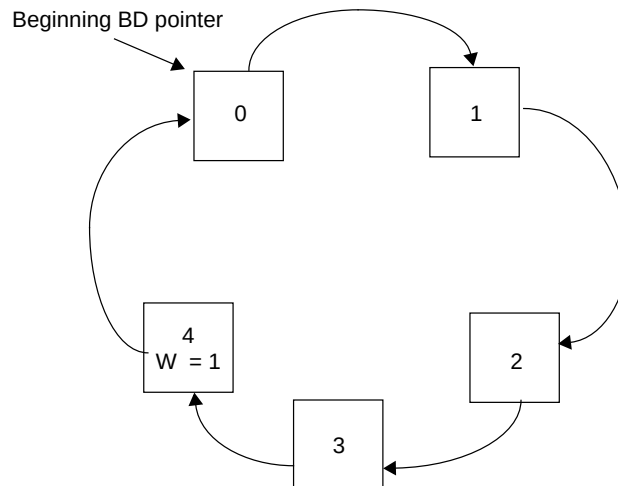


Figure 14-113. Buffer Descriptor Ring

14.6.3.2 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the TSEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The TSEC 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 14-114 describes the definition for 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	DEF	0	HFE/ LC	RL	RC				UN	0
Offset + 2	DATA LENGTH															
Offset + 4	TX DATA BUFFER POINTER															
Offset + 6																

Figure 14-114. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 14-115.

Table 14-115. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	R	Ready, written by TSEC 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 TSEC 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.

Table 14-115. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
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 enable] is cleared). If MACCFG2[PAD/CRC enable] 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. Unlike the MPC8260 which PADs up to MINFLR value, TSEC 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 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. 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.
Offset + 0	6	DEF	Hardware updates this bit after transmitting a frame (TxBD[L] is set). DEF. Defer indication. Written by hardware. 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.
Offset + 0	7	—	Reserved
Offset + 0	8	HFE/LC	Huge Frame Enable (written by user)/Late collision (written by TSEC) Huge Frame Enable (written by user). Valid only while it is set in first BD and the MACCFG2[Huge Frame Enable] is cleared. If MACCFG2[Huge Frame Enable] 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 TSEC). 0 No late collision. 1 A collision occurred after 64 bytes are sent. The TSEC terminates the transmission and updates LC.

Table 14-115. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	9	RL	Retransmission Limit. Written by TSEC. 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 TSEC terminates the transmission and updates RL.
Offset + 0	10–13	RC	Retry Count. Written by TSEC. 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.
Offset + 0	14	UN	Underrun. Written by TSEC. 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 TSEC terminates the transmission and updates UN.
Offset + 0	15	—	Reserved
Offset + 2	0–15	Data Length	Data length is the number of octets the TSEC should transmit from this BD's data buffer. It is never modified by the TSEC. 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.

14.6.3.3 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, TSEC 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 TSEC if the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed below. Figure 14-115 describes the definition for 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 14-115. Receive Buffer Descriptor

Table 14-116 describes the fields of the RxBD.

Table 14-116. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty, written by the TSEC (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.
Offset + 0	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.
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[RXB0] or IEVENT[RXF0] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN0] or IMASK[RXFEN0]). If the user wants to be interrupted only if RXF0 occurs, then the user must disable RXB0 (IMASK[RXBEN0] is cleared) and enable RXF0 (IMASK[RXFEN0] is set).
Offset + 0	4	L	Last in frame, written by the TSEC. 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 the TSEC. 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 the TSEC. (This bit is valid only if the L-bit is set and TSEC is in promiscuous mode.) This bit is set by the TSEC 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 TSEC. (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 TSEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.
Offset + 0	10	LG	Rx frame length violation, written by the TSEC (only valid if L is set). A frame length greater than maximum frame length was recognized. If MACCFG2[HUGE FRAME ENABLE] 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.
Offset + 0	11	NO	Rx non-octet aligned frame, written by the TSEC (only valid if L is set). A frame that contained a number of bits not divisible by eight was received.

Table 14-116. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	12	SH	Short frame, written by the TSEC (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 the TSEC (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 the TSEC (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, CR, and CL lose their normal meaning and are zero.
Offset + 0	15	TR	Truncation, written by the TSEC (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 Enable] 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 the TSEC. Data length is the number of octets written by the TSEC 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 the 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 TSEC.

14.7 Application Information

14.7.1 Interface Mode Configuration

This section describes how to configure the TSEC in different supported interface modes. These include MII, GMII, TBI, RGMII, RTBI. The pinout, the data registers that must be initialized, as well as speed selection options are described. The ECNTRL[TBIM] and ECNTRL[RPM] bits are written, assuming the part was not pin configured at initialization to the correct mode.

14.7.1.1 MII Interface Mode

Table 14-117 describes the pin configurations required for MII interface mode.

Table 14-117. MII Interlace Mode Pin Configuration

TSEC PINS			MII INTERFACE		
Frequency [MHz] 125			Frequency [MHz] 25		
Voltage[V] 3.3/2.5			Voltage[V] 3.3		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
GTX_CLK	Ot	1	Leave Unconnected	Ot	
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	Ot	1	TxD[0]	Ot	1
TxD[1]	Ot	1	TxD[1]	Ot	1
TxD[2]	Ot	1	TxD[2]	Ot	1
TxD[3]	Ot	1	TxD[3]	Ot	1
TxD[4]	Ot	1	Leave Unconnected	Ot	
TxD[5]	Ot	1	Leave Unconnected	Ot	
TxD[6]	Ot	1	Leave Unconnected	Ot	
TxD[7]	Ot	1	Leave Unconnected	Ot	
TX_EN	Ot	1	TX_EN	Ot	1
TX_ER	Ot	1	TX_ER	Ot	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	I	
RxD[5]	I	1	not used	I	
RxD[6]	I	1	not used	I	
RxD[7]	I	1	not used	I	
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 14-118 describes the shared pins of the MII interface.

Table 14-118. Shared MII Pins

TSEC Signals	I/O	No. of pins	II Signals	I/O	No. of pins	Function
EC_MDIO	I/Ot	1	EC_MDIO	I/Ot	1	Management interface I/O
EC_MDC	Ot	1	EC_MDC	Ot	1	Management interface Clock
EC_GTX_CLK125	I	1	not used	I	0	Reference Clock
Sum		3	Sum		2	

Table 14-119 describes the register initializations required to configure the TSEC in MII mode.

Table 14-119. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0101]
Initialize MACCFG2, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (I/F Mode = 1, Full Duplex = 0)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (Used to setup Reduced Pin mode = 0, 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
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16 for the example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] set source clock divide by 28 for example to insure that EC_MDC clock speed is not greater than 2.5 MHz
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

Table 14-119. 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 indicate that the write cycle was completed
Perform an MII Mgmt read cycle (Optional) 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_000_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_000_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_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 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 (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]

Table 14-119. 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_0101]

14.7.1.2 GMII Interface Mode

Table 14-120 describes the pin configurations required for GMII interface mode.

Table 14-120. GMII Interface Mode Pin Configuration

TSEC PINS			GMII INTERFACE		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 3.3		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
GTX_CLK	Ot	1	GTX_CLK	O	1
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	Ot	1	TxD[0]	Ot	1
TxD[1]	Ot	1	TxD[1]	Ot	1
TxD[2]	Ot	1	TxD[2]	Ot	1
TxD[3]	Ot	1	TxD[3]	Ot	1
TxD[4]	Ot	1	TxD[4]	Ot	1
TxD[5]	Ot	1	TxD[5]	Ot	1
TxD[6]	Ot	1	TxD[6]	Ot	1
TxD[7]	Ot	1	TxD[7]	Ot	1
TX_EN	Ot	1	TX_EN	Ot	1
TX_ER	Ot	1	TX_ER	Ot	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

Table 14-120. GMII Interlace Mode Pin Configuration (continued)

TSEC PINS			GMII INTERFACE		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 3.3		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1			
CRS	I	1			
Sum		25	sum		23

Table 14-121 describes the shared pins of the GMII interface.

Table 14-121. Shared GMII Pins

TSEC Signals	I/O	No. of Pins	GMII Signals	I/O	No. of Pins	Function
EC_MDIO	I/Ot	1	EC_MDIO	I/Ot	1	Management interface I/O
EC_MDC	Ot	1	EC_MDC	Ot	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 14-122 describes the register initializations required to configure the TSEC in GMII mode.

Table 14-122. 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_0101]
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] (Used to setup Reduced Pin mode = 0, 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

Table 14-122. GMII Mode Register Initialization Steps (continued)

Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0001_0000] set to 16 for the example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] set source clock divide by 28 for example to insure that EC_MDC clock speed is not greater than 2.5 MHz
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
Perform an MII Mgmt read cycle (Optional) 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_000_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_000_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]

Table 14-122. GMII Mode Register Initialization Steps (continued)

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 (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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]

14.7.1.3 TBI Interface Mode

Table 14-123 describes the pin configurations required for TBI interface mode.

Table 14-123. TBI Interlace Mode Pin Configuration

TSEC Pins			TBI Interface		
Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 3.3		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
GTX_CLK	Ot	1	GTX_CLK	Ot	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	Ot	1	TCG[0]	Ot	1
TxD[1]	Ot	1	TCG[1]	Ot	1
TxD[2]	Ot	1	TCG[2]	Ot	1
TxD[3]	Ot	1	TCG[3]	Ot	1
TxD[4]	Ot	1	TCG[4]	Ot	1
TxD[5]	Ot	1	TCG[5]	Ot	1
TxD[6]	Ot	1	TCG[6]	Ot	1
TxD[7]	Ot	1	TCG[7]	Ot	1
TX_EN	Ot	1	TCG[8]	Ot	1
TX_ER	Ot	1	TCG[9]	Ot	1
RX_CLK	I	1	RX_CLK0	I	1

Table 14-123. TBI Interlace Mode Pin Configuration (continued)

TSEC Pins			TBI Interface		
Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 3.3		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
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		I	
CRS	I	1	SDET	I	1
Sum		25	sum		24

Table 14-124 describes the shared pins for the TBI interface.

Table 14-124. Shared TBI Pins

TSEC Signals	I/O	No. of Pins	GMII Signals	I/O	No. of Pins	Function
EC_MDIO	I/Ot	1	EC_MDIO	I/Ot	1	Management interface I/O
EC_MDC	Ot	1	EC_MDC	Ot	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 14-125 describes the register initializations required to configure the TSEC in TBI mode.

Table 14-125. 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_0101]

Table 14-125. TBI Mode Register Initialization Steps (continued)

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] (Used to setup Reduced Pin mode = 0, and TBIM = 1, 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 the example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] set source clock divide by 28 for example to insure that EC_MDC clock speed is not greater than 2.5 MHz
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_0000_0000] This configures the TBI control to TBI 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
Perform an MII Mgmt read cycle (Optional) 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_0000_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_000_0010_0000] other information about the link is also returned (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)

Table 14-125. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Expansion Register MIIMADD[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_000_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 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 (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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]

14.7.1.4 RGMII Interface Mode

Table 14-126 shows the pins configurations required for RGMII interface mode.

Table 14-126. RGMII Interlace Mode Pin Configuration

TSEC PINS			RGMII INRTERFACE		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
GTX_CLK	Ot	1	GTX_CLK	Ot	1
TX_CLK	I	1			
TxD[0]	Ot	1	TxD[0]/TxD[4]	Ot	1
TxD[1]	Ot	1	TxD[1]/TxD[5]	Ot	1
TxD[2]	Ot	1	TxD[2]/TxD[6]	Ot	1
TxD[3]	Ot	1	TxD[3]/TxD[7]	Ot	1
TxD[4]	Ot	1			
TxD[5]	Ot	1			
TxD[6]	Ot	1			
TxD[7]	Ot	1			
TX_EN	Ot	1	TX_CTL (TX_EN/TX_ERR)	Ot	1
TX_ER	Ot	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			

Table 14-126. RGMII Interlace Mode Pin Configuration (continued)

TSEC PINS			RGMII INRTERFACE		
Frequency [MHz] 125			Frequency [MHz] 125		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals	I/O	No. of Pins	Signals	I/O	No. of Pins
COL	I	1			
CRS	I	1			
Sum		25	sum		12

Table 14-127 describes the shared pins for the RGMII interface.

Table 14-127. Shared RGMII Pins

TSEC Signals	I/O	No. of Pins	GMII Signals	I/O	No. of Pins	Function
EC_MDIO	I/Ot	1	EC_MDIO	I/Ot	1	Management interface I/O
EC_MDC	Ot	1	EC_MDC	Ot	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 14-128 describes the register initializations required to configure the TSEC in RGMII mode.

Table 14-128. RGMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0101]
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

Table 14-128. RGMII Mode Register Initialization Steps (continued)

Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0001_0000] set to 16 for the example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] set source clock divide by 28 for example to insure that EC_MDC clock speed is not greater than 2.5 MHz
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
Perform an MII Mgmt read cycle (Optional) 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_000_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_000_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]

Table 14-128. RGMII Mode Register Initialization Steps (continued)

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 (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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]

14.7.1.5 RTBI Interface Mode

Table 14-129 describes the pin configurations required for RTBI interface mode.

Table 14-129. RTBI Interlace Mode Pin Configuration

TSEC PINS			RTBI INRTERFACE		
Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals	I/O	No. of pins	Signals	I/O	No. of pins
GTX_CLK	Ot	1	GTX_CLK	Ot	1
TX_CLK	I	1			
TxD[0]	Ot	1	TCG[0]/TCG[5]	Ot	1
TxD[1]	Ot	1	TCG[1]/TCG[6]	Ot	1
TxD[2]	Ot	1	TCG[2]/TCG[7]	Ot	1
TxD[3]	Ot	1	TCG[3]/TCG[8]	Ot	1
TxD[4]	Ot	1			
TxD[5]	Ot	1			
TxD[6]	Ot	1			
TxD[7]	Ot	1			

Table 14-129. RTBI Interlace Mode Pin Configuration (continued)

TSEC PINS			RTBI INRTERFACE		
Frequency [MHz] 125			Frequency [MHz] 62.5		
Voltage[V] 3.3/2.5			Voltage[V] 2.5		
Signals	I/O	No. of pins	Signals	I/O	No. of pins
TX_EN	Ot	1	TCG[4]/TCG[9]	Ot	1
TX_ER	Ot	1			
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			
RxD[5]	I	1			
RxD[6]	I	1			
RxD[7]	I	1			
RX_DV	I	1	RCG[4]/RCG[9]	I	1
RX_ER	I	1			
COL	I	1			
CRS	I	1		I	
Sum		25	sum		12

Table 14-130 describes the shared pins for the RTBI interface.

Table 14-130. Shared RTBI Pins

TSEC Signals	I/O	No. of Pins	GMII Signals	I/O	No. of Pins	Function
EC_MDIO	I/Ot	1	EC_MDIO	I/Ot	1	Management interface I/O
EC_MDC	Ot	1	EC_MDC	Ot	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 14-131 describes the register initializations required to configure the TSEC in RTBI mode.

Table 14-131. 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_0101]
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 = 1, 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 the example
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0111] set source clock divide by 28 for example to insure that EC_MDC clock speed is not greater than 2.5 MHz
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_0000_0000] This configures the TBI control to TBI 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
Perform an MII Mgmt read cycle (Optional) 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_0000_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

Table 14-131. RTBI Mode Register Initialization Steps (continued)

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_000_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_000_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 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 (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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 15

DMA Controller

15.1 Introduction

The DMA controller transfers blocks of data between RapidIO and the local address space, independent of the e500 core or external hosts.

15.1.1 Block Diagram

Figure 15-1 depicts the block diagram of the DMA controller.

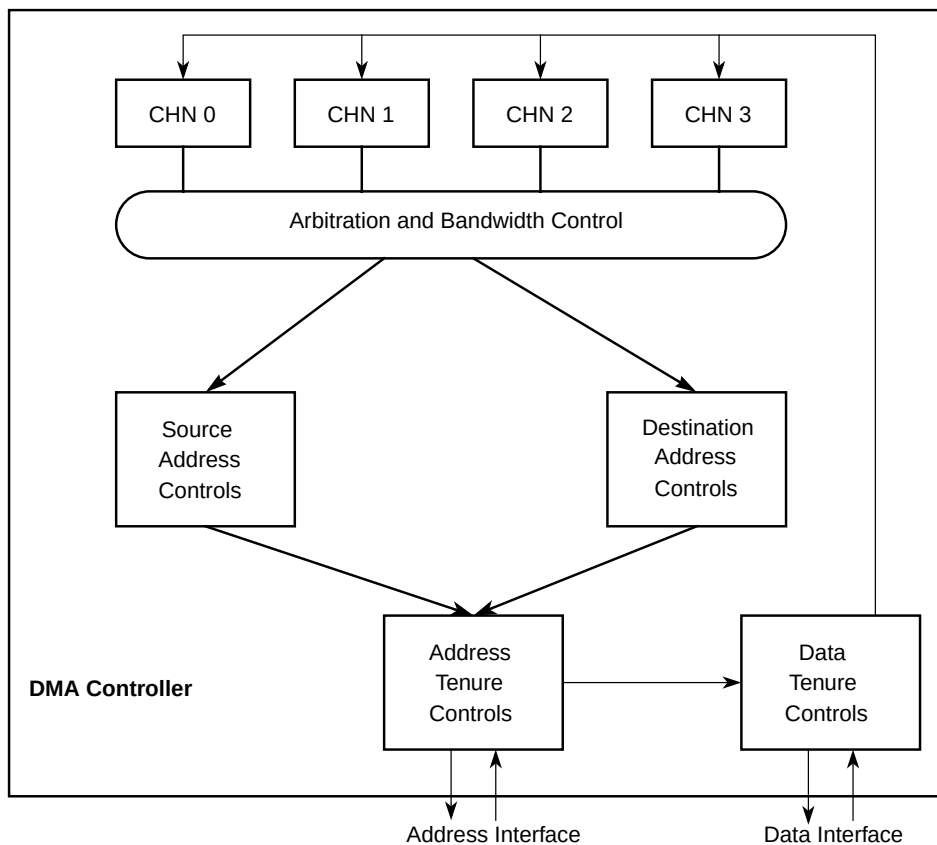


Figure 15-1. DMA Block Diagram

15.1.2 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 15-1 depicts 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 reads 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.

15.1.3 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 over the RapidIO interface
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using `DMA_DREQ`, `DMA_DACK`, and `DMA_DDONE`

15.1.4 Modes of Operation

The MPC8540 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 15-1 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 15-2 before starting a transfer.

- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 15-2 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 $MRn[CA]$. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 15-1. Relationship of Modes and Features

Mode	Mode with 1 Additional Feature	Mode with 2 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 15-2 describes bit settings required for each DMA mode of operation

Table 15-2. DMA Mode Bit Settings

Modes with Features	$MRn[XFE]$	$MRn[CTM]$	$MRn[SRW]$	$MRn[EM]$	$MRn[CDSM/SWSM]$	$MRn[EMS\_EN]$
Basic Direct Modes						
Basic direct	0	1	0	0	0	0
Basic direct external control	0	1	0	0	0	1

Table 15-2. DMA Mode Bit Settings (continued)

Modes with Features	MRn[XFE]	MRn[CTM]	MRn[SRW]	MRn[EM]	MRn[CDSM/SWSM]	MRn[EMS_EN]
Basic direct single-write start	0	1	1	0	1 or 0	0
Basic Chaining Modes						
Basic chaining	0	0	Reserved	0	0	0
Basic chaining external control	0	0	Reserved	0	0	1
Basic chaining single-write start	0	0	Reserved	0	1	0
Extended Direct Modes						
Extended direct	1	1	0	0	0	0
Extended direct external control	1	1	0	0	0	1
Extended direct single-write start	1	1	1	0	1 or 0	0
Extended Chaining Modes						
Extended chaining	1	0	Reserved	0	0	0
Extended chaining external control	1	0	Reserved	0	0	1
Extended chaining single-write start	1	0	Reserved	0	1	0

Refer to Section 15.4, “Functional Description,” for details on these modes.

Figure 15-2 depicts the general DMA operational flow chart.

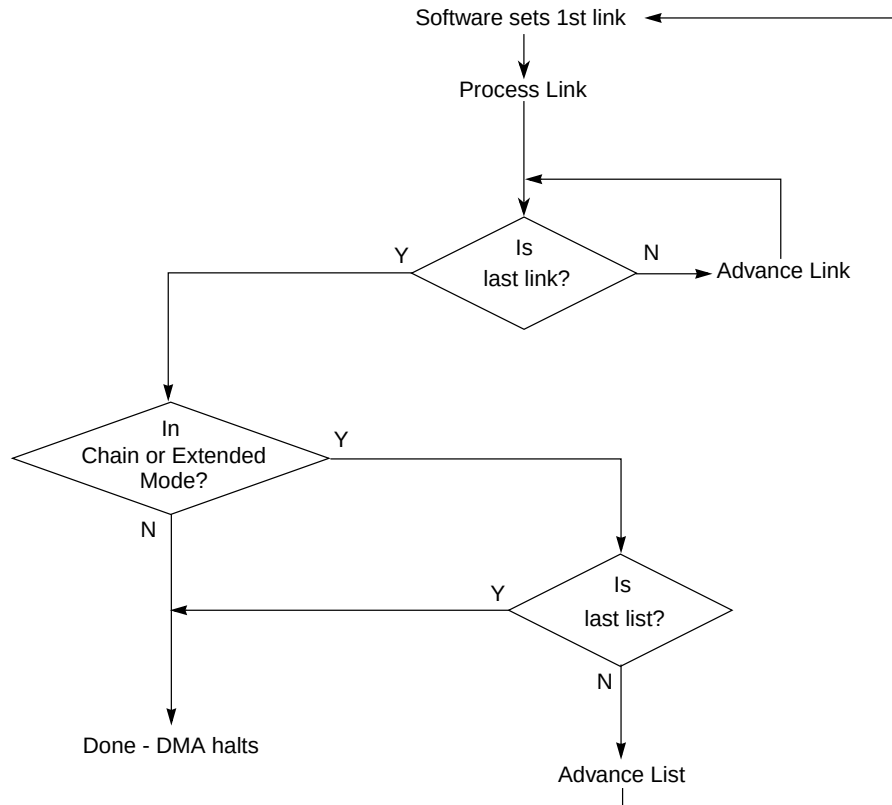


Figure 15-2. DMA Operational Flow Chart

15.2 External Signal Description

This section describes the DMA signals.

15.2.1 Signal Overview

Figure 15-3 summarizes the DMA controller signals.

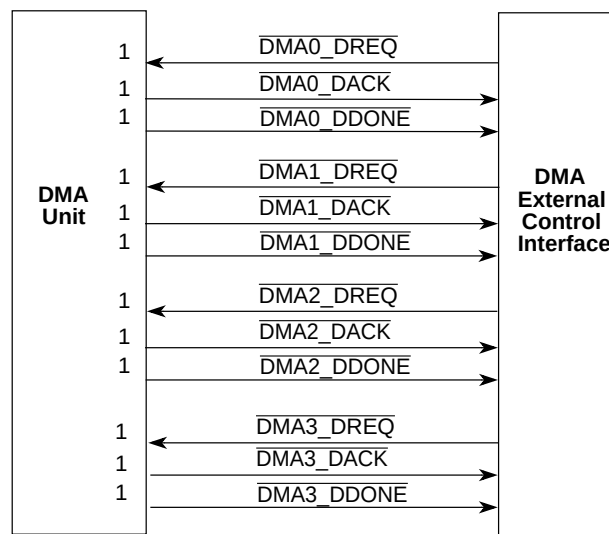


Figure 15-3. DMA Signal Summary

Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the MPC8540 device. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in Section 18.3.1.10, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

15.2.2 Detailed Signal Descriptions

The DMA signals are described in Table 15-3.

Table 15-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description
DMA _n _DREQ 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 DMA _n _DREQ causes MR _n [CS] to be set, thereby activating the corresponding DMA channel.
		State Meaning Asserted—Assertion of DMA _n _DREQ while DMA _n _DACK is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion of DMA _n _DREQ while DMA _n _DACK is asserted results in an illegal condition Negated—Negation of DMA _n _DREQ while DMA _n _DACK is asserted has no effect. Negation of DMA _n _DREQ before the assertion of DMA _n _DACK results in an illegal condition
		Timing Assertion—Can be asserted asynchronously. Negation— Must remain asserted at least until the assertion of the corresponding DMA _n _DACK.

Table 15-3. DMA Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
$\overline{\text{DMA}}_n\text{\_DACK}$	O	DMA acknowledge. The DMA acknowledge signal indicates that a DMA transfer is currently in progress.
		State Meaning Asserted—Indicates that a DMA transfer is currently in progress. Asserted some time after the assertion of $\overline{\text{DMA}}_n\text{\_DREQ}$ to indicate the start of a transfer. Negated—Negated after finishing a complete transfer or after entering a paused state if $\text{MR}_n[\text{EMP\_EN}]$ is set.
		Timing Assertion/Negation—
$\overline{\text{DMA}}_n\text{\_DDONE}$	O	DMA done. The DMA done signal indicates that a DMA transfer is complete.
		State Meaning Asserted—Indicates transfer completion. Negated—Indicates that the current transfer is in process.
		Timing Assertion—Always asserts after the negation of the final $\overline{\text{DMA}}_n\text{\_DACK}$ to indicate completion of a transfer. For a paused transfer, $\overline{\text{DMA}}_n\text{\_DDONE}$ is asserted after the negation of the final $\overline{\text{DMA}}_n\text{\_DACK}$. Negation—This signal is negated after the assertion of $\overline{\text{DMA}}_n\text{\_DREQ}$ for the next transfer.

15.3 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.

15.3.1 Module Memory Map

Figure 15-4 depicts the address map of DMA memory.

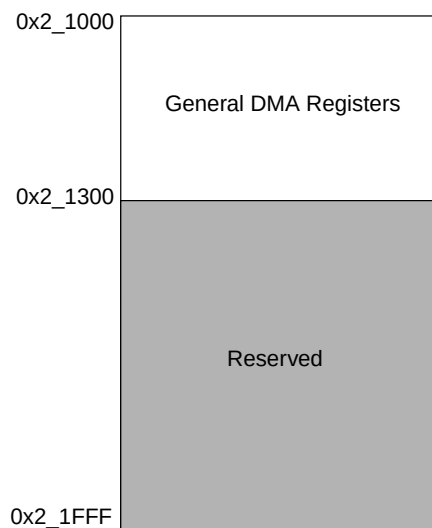
**Figure 15-4. DMA Memory Map**

Table 15-4 lists the classes and types of DMA registers.

Table 15-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
General Registers				
0x2_1100	DMA 0 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1104	DMA 0 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1108	Reserved	—	—	—
0x2_110C	DMA 0 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1110	DMA 0 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1114	DMA 0 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1118	DMA 0 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_111C	DMA 0 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1120	DMA 0 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1124	Reserved	—	—	—
0x2_1128	DMA 0 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_1130	Reserved	—	—	—
0x2_1134	DMA 0 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1138	Reserved	—	—	—
0x2_113C	DMA 0 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1140	DMA 0 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_1144	DMA 0 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1180	DMA 1 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1184	DMA 1 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1188	Reserved	—	—	—
0x2_118C	DMA 1 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1190	DMA 1 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1194	DMA 1 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1198	DMA 1 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_119C	DMA 1 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_11A0	DMA 1 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	DMA 1 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_11B0	Reserved	—	—	—
0x2_11B4	DMA 1 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	DMA 1 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_11C0	DMA 1 source stride register	R/W	0x0000_0000	15.3.2.12/15-26

Table 15-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_11C4	DMA 1 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1200	DMA 2 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1204	DMA 2 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1208	Reserved	—	—	—
0x2_120C	DMA 2 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1210	DMA 2 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1214	DMA 2 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1218	DMA 2 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_121C	DMA 2 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1220	DMA 2 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1224	Reserved	—	—	—
0x2_1228	DMA 2 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_1230	Reserved	—	—	—
0x2_1234	DMA 2 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1238	Reserved	—	—	—
0x2_123C	DMA 2 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1240	DMA 2 source stride register	R/W	0x0000_0000	15.3.2.12/15-26
0x2_1244	DMA 2 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1280	DMA 3 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1284	DMA 3 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1288	Reserved	—	—	—
0x2_128C	DMA 3 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1290	DMA 3 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1294	DMA 3 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1298	DMA 3 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_129C	DMA 3 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_12A0	DMA 3 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	DMA 3 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_12B0	Reserved	—	—	—
0x2_12B4	DMA 3 current list descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	DMA 3 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_12C0	DMA 3 source stride register	R/W	0x0000_0000	15.3.2.12/15-26

Table 15-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_12C4	DMA 3 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1300	DMA general status register	Read	0x0000_0000	15.3.2.14/15-27

15.3.2 DMA Register Descriptions

The following sections provide descriptions of the DMA registers. The majority of the DMA controller registers are channel-specific and can be identified by one of the four offsets of Figure 15-5 that describes the register.

15.3.2.1 Mode Registers (MR_n)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics.

	0	1	2	3	4	7	8	9	10	11	12	13	14	15		
R	0	0	0	0	BWC			0	0	EMP_EN	0	0	EMS_EN	DAHTS		
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SAHTS	DAHE	SAHE	0	SRW	EOSIE	EOL NIE	EOL SIE	PEIE	XFE	CDSM SWSM	CA	CTM	CC	CS	
W																
Reset	0000_0000_0000_0000															
Offset	0x2_1100, 0x2_1180, 0x2_1200, 0x2_1280															

Figure 15-5. DMA Mode Registers (MR_n)

Table 15-5 describes the bits of the mode registers.

Table 15-5. MR_n Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. This field is only defined if multiple channels are executing transfers concurrently or if MR _n [EMP_EN] is set. The field determines how many bytes a given channel is allowed to transfer before the DMA engine switches to the next channel. The DMA channel pauses during this switch if MR _n [EMP_EN] is set for this channel. 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 channel from being paused by an external master. 1 Enable the channel to be paused by an external master.
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. 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. 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.
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.

Table 15-5. MR_n Field Descriptions (continued)

Bits	Name	Description
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 MR _n [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. 1 Generate an interrupt at the completion of a data transfer (That is, SR _n [EOSI] is set). This bit overrides the CNLDR[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	PEIE	Programming error interrupt enable. 0 Do not generate an interrupt if a programming error is detected. 1 Generate an interrupt if a programming 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 15-5. 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. 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 (SR _n [CB] is cleared). If the channel is busy and this bit is set, then the DMA channel restarts from a previous halt condition.

15.3.2.2 Status Registers (SR_n)

The status registers report various DMA conditions during and after a DMA transfer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TE	0	CH	PEI	EOLNI	CB	EOSI	EOLSI
W																
Reset	0000_0000_0000_0000															
Offset	0x2_1104, 0x2_1184, 0x2_1204, 0x2_1284															

Figure 15-6. Status Registers (SR_n)

Table 15-6 describes the bits of the status registers.

Table 15-6. 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.
25	—	Reserved
26	CH	Channel halted. 0 The halted transfer has completed and cannot be resumed if SR _n [CB] also cleared. 1 DMA transfer was halted if SR _n [CB] also set. This bit is cleared automatically by hardware if MR _n [CS] is set again for resuming a halted transfer.

Table 15-6. SR_n Field Descriptions (continued)

Bits	Name	Description
27	PEI	Programming error interrupt . (Bit reset, write 1 to clear) 0 No programming error detected. 1 A programming error is detected that prevents the DMA transfer from occurring while MR _n [PEIE] is set; an interrupt is generated.
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)
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 MR _n [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)

15.3.2.3 Current Link Descriptor Address Registers (CLNDAR_n)

The current link descriptor address registers contain the address of the current link descriptor.

In basic chaining mode, software must initialize this register to point to the first link descriptor in memory. After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address register and NLNDAR_n[EOLND] in the next link descriptor address register is examined. If NLNDAR_n[EOLND] is clear, the DMA controller reads in the new current link descriptor for processing. If NLNDAR_n[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 then examines the state of NLSDAR_n[EOLSD] in the next list descriptor address register. If NLSDAR_n[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 NLSDAR_n[EOLSD] is set all DMA transfers are complete and the DMA controller halts. Figure 15-7 depicts the process of Basic Chaining Mode DMA operation.

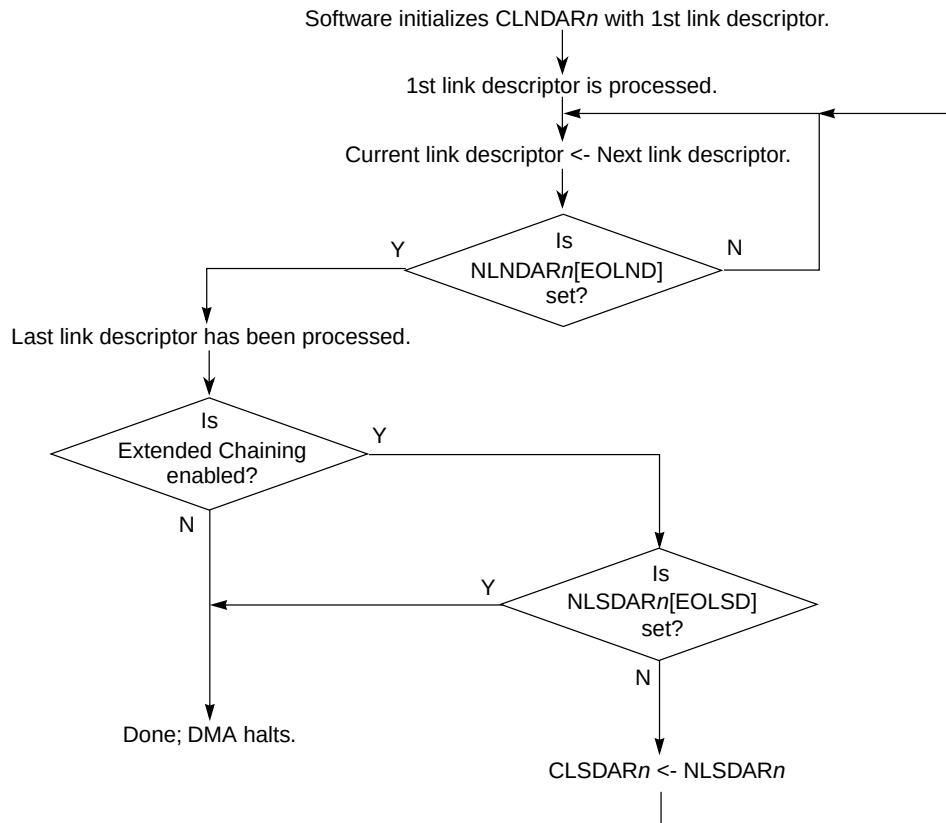


Figure 15-7. Basic Chaining Mode Flow Chart

Figure 15-8 describes the bits of the current link descriptor address registers.

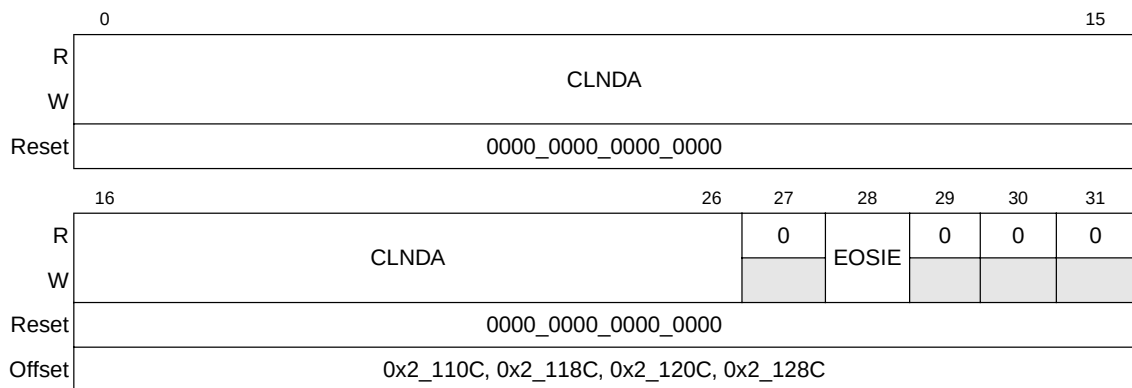


Figure 15-8. Current Link Descriptor Address Registers (CLNDARn)

Table 15-7 describes the bits of the current link descriptor address registers.

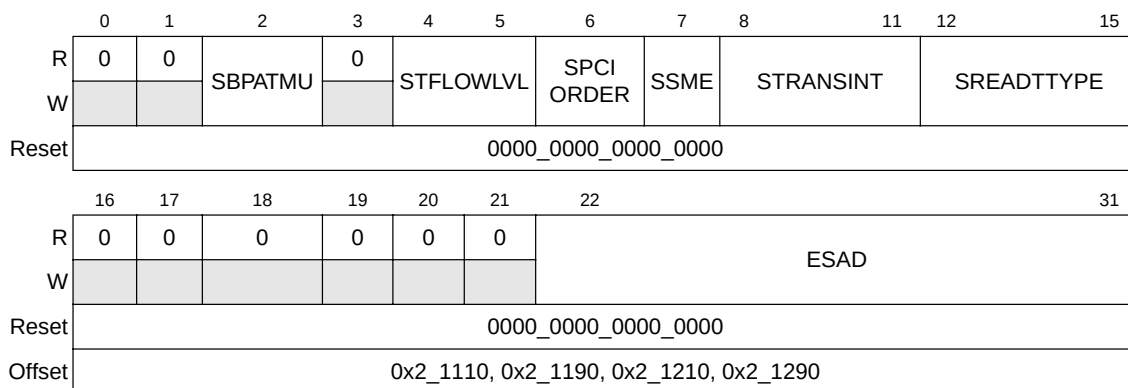
Table 15-7. CLNDAR_n Field Descriptions

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.
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

15.3.2.4 Source Attributes Registers (SATR_n)

The source attributes registers contain the transaction attributes to be used for the DMA operation. If SATR_n[SBPATMU] is not set and the source address generates a hit in the ATMU the attributes specified in the ATMU are used instead of the attributes in this register and the RapidIO source address register.

If the RapidIO interface is the source of the transaction data, several options exist for the request transaction that is generated.

**Figure 15-9. Source Attributes Registers (SATR_n)****Table 15-8. SATR_n Field Definitions**

Bits	Name	Description
0–1	—	Reserved
2	SBPATMU	Bypass ATMU for this DMA operation. 0 Route the operation through the ATMU outbound windows. SATR _n [SREADTTYPE] should specify a local address space transaction type. 1 Bypass ATMU. Never generate an address match. Always use the attributes in the source attribute registers. Route the transaction to the interface specified in SATR _n [STRANSINT]. Applicable only to RapidIO interface.

Table 15-8. SATR_n Field Definitions (continued)

Bits	Name	Description
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.
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 source stride register(SSR _n).
8–11	STRANSINT	DMA source transaction interface. 0000 0001–0011 Reserved 0100 0101–1011 Reserved 1100 RapidIO interface 1101 Reserved 1110 Reserved 1111 Reserved Applicable only to RapidIO interface, while SATR _n [SBPATMU] is set.
12–15	SREADTTYPE	DMA source transaction type to run on RapidIO interface. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read ... 1111 Reserved Transaction type to run on local address space 0000 - Reserved 0001 - Reserved 0011 - Reserved 0100 - Read, don't snoop local processor 0101 - Read, snoop local processor 0111 - Read, unlock L2 cache line 1000 - Reserved ... 1111 - Reserved

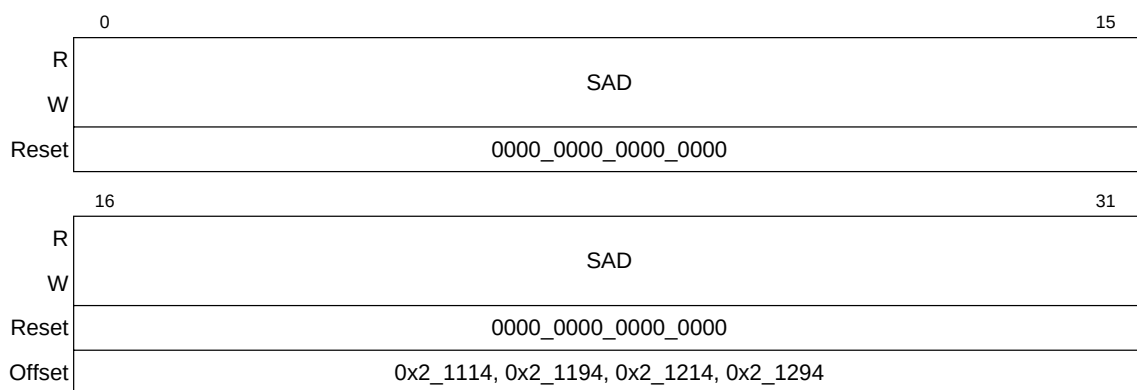
Table 15-8. SATR_n Field Definitions (continued)

Bits	Name	Description
16–21	—	Reserved
22–31	ESAD	Extended source address. ESAD[0:7] represent the target ID. ESAD[8:9] represent the two high-order bits of the local device offset over the RapidIO interface. Applicable only to RapidIO interface, while SATR _n [SBPATMU] is set.

15.3.2.5 Source Address Registers (SAR_n)

The source address registers 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.

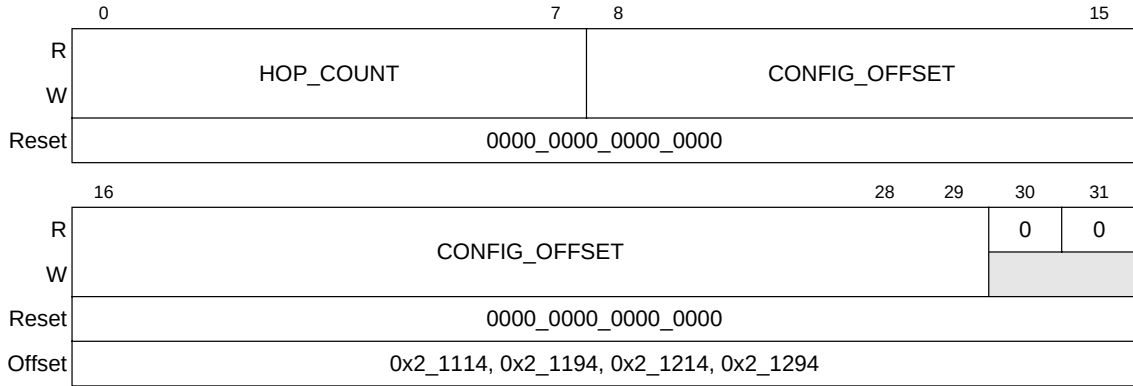
**Figure 15-10. Source Address Registers (SAR_n)****Table 15-9. SAR_n Field Descriptions**

Bit	Name	Description
0–31	SAD	Source address. This register contains the source address of the DMA transfer. The contents are updated after every DMA read operation.

15.3.2.5.1 Source Address Registers For RapidIO Maintenance Reads (SARM_n)

If RapidIO is the destination of the transaction, 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. The definition of the source address register for maintenance packets is shown below.

**Figure 15-11. Source Address Registers for RapidIO Maintenance Reads (SARM_n)****Table 15-10. SARM_n Field Descriptions**

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop_count as defined by the RapidIO 1.2 Specifications.
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the RapidIO 1.2 Specifications.
30–31	—	Reserved

15.3.2.6 Destination Attributes Registers (DATR_n)

The destination attributes registers contain the transaction attributes to be used for the DMA operation. If DATR_n[DBPATMU] is not set and the destination address generates a hit in the ATMU, the attributes specified in the ATMU are used instead of the attributes in this DATR_n register and the destination address RapidIO register.

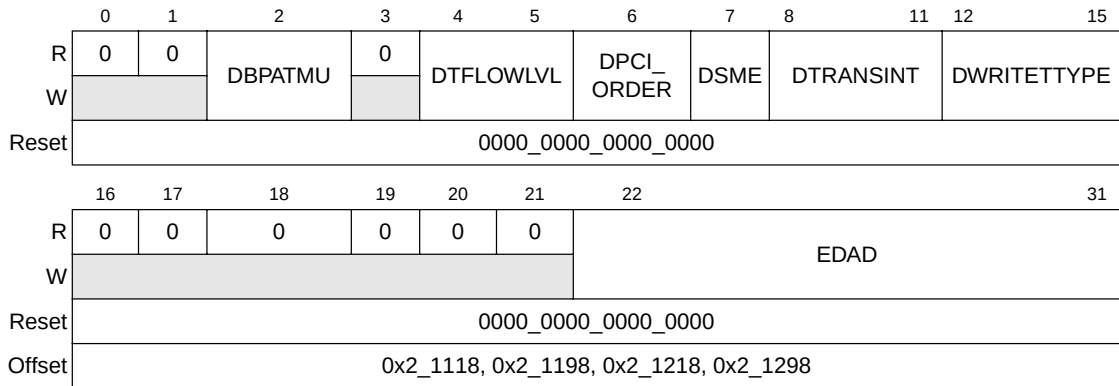
**Figure 15-12. Destination Attributes Registers (DATR_n)**

Table 15-11. DATR_n Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	DBPATMU	Bypass ATMU for this DMA operation. 0 Route the operation through the ATMU outbound windows. DATR _n [DWRITETYPE] should specify a local address space transaction type. 1 Bypass ATMU. Never generate an address match. Always use the attributes in the destination attribute registers. Route the transaction to the interface specified in the DATR _n [DTRANSINT] field. Applicable only to RapidIO interface.
3	—	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	Follow PCI transaction ordering rules on RapidIO (elevate write priority one level over reads). Applicable only while DATR _n [DBPATMU] is set.
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 enabling DATR _n [DSME] and setting the desired stride size and distance in the destination stride register(DSR _n).
8–11	DTRANSINT	DMA Destination transaction interface. 0000 0001 - Reserved ... 0100 0101 - Reserved ... 1100 - RapidIO interface 1101 - Reserved 1110 - Reserved 1111 - Reserved Applicable only while DATR _n [DBPATMU] is set.

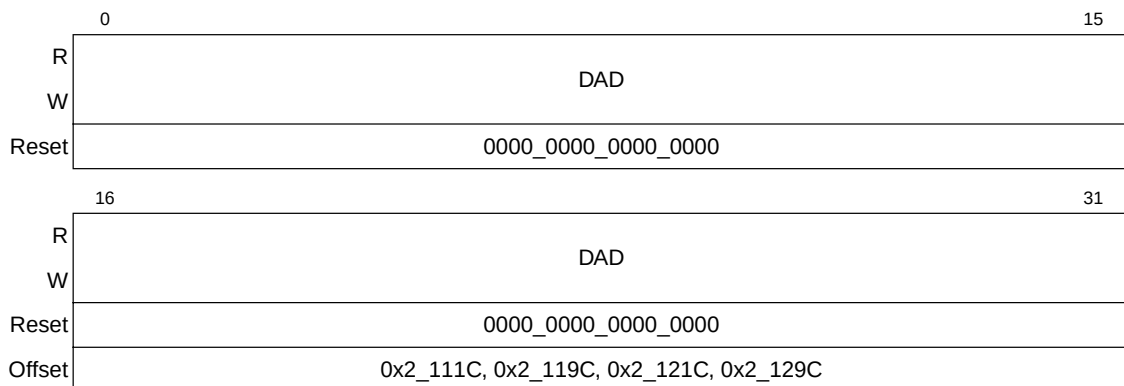
Table 15-11. DATR_n Field Descriptions (continued)

Bits	Name	Description
12–15	DWRITETYPE	DMA Destination Transaction type to run on RapidIO interface. 0000 - Reserved 0001 - Flush 0010 - Reserved 0011 - SWRITE for all but last, NWRITE_R for last transaction 0100 - NWRITE for all but last, NWRITE_R for last transaction 0101 - NWRITE_R 0110 - Message (single packet, 256 bytes max, double-word aligned) 0111 - MAINTENANCE write 1000 - Reserved ... 1111 - Reserved Transaction type to run on local address space 0000 - 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 - Reserved ... 1111 - Reserved
16–21	—	Reserved
22–31	EDAD	Extended Destination Address. For RapidIO interface, EDAD[0:7] represents the target ID and EDAD[8:9] represents different things depending on the transaction: Message: value for “mbox” field in message packet Other: two high-order bits of the local device offset Applicable only for RapidIO interface, while DATR_n[DBPATMU] is set.

15.3.2.7 Destination Address Registers (DAR_n)

The destination address registers 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 15-13. Destination Address Registers (DAR_n)****Table 15-12. DAR_n Field Definitions**

Bits	Name	Description
0–31	DAD	Destination address. This register contains the destination address of the DMA transfer. The contents are updated after every DMA write operation.

15.3.2.7.1 Destination Address Registers for RapidIO Maintenance Writes (DARM_n)

If RapidIO is the destination of a transaction, several options exist for the transaction type that can be specified. There exist a number of noncoherent write and flush types for address-based write transactions, and message type for port-based address-less write transactions.

Maintenance packets have an offset instead of an address. The definition of the destination address register for maintenance packets is shown below.

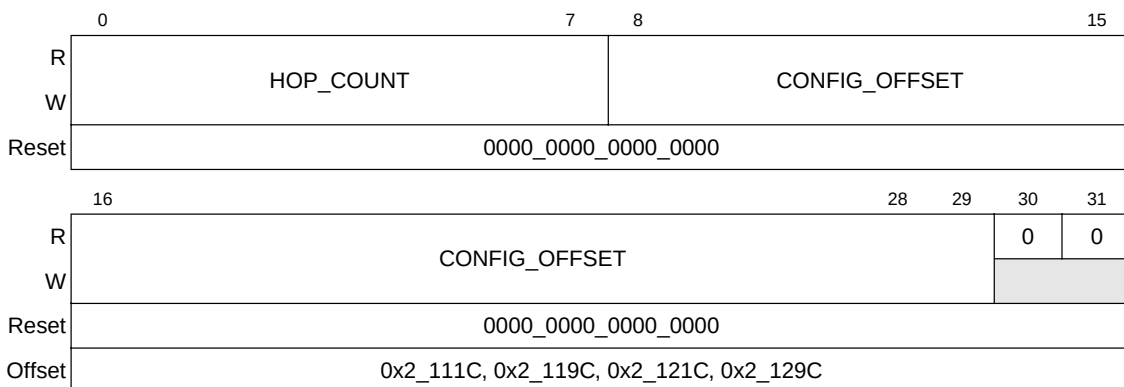
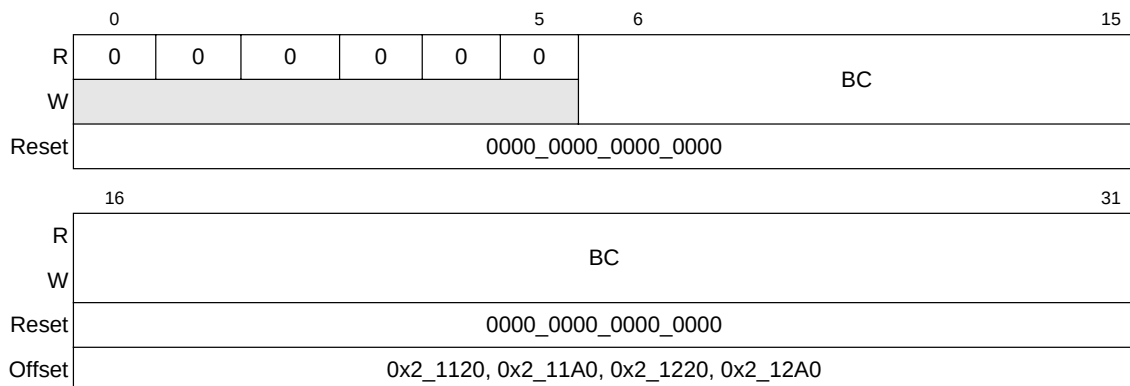
**Figure 15-14. Destination Address Registers For RapidIO Maintenance Writes (DARM_n)**

Table 15-13. DARM_n Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop count as defined by the RapidIO 1.2 Specifications.
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the RapidIO 1.2 Specifications.
30–31	—	Reserved

15.3.2.8 Byte Count Registers (BCR_n)

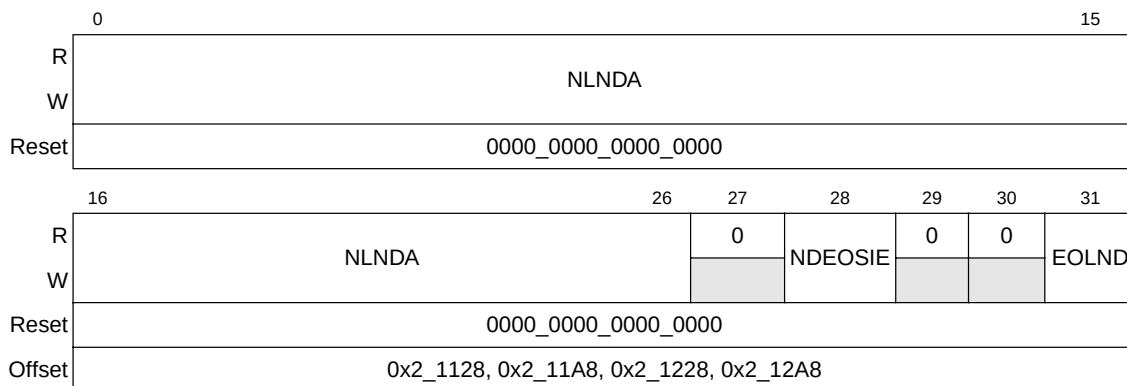
The byte count register contains the number of bytes to transfer.

**Figure 15-15. Byte Count Registers (BCR_n)****Table 15-14. 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 a byte less than 64 MB.

15.3.2.9 Next Link Descriptor Address Registers (NLNDAR_n)

The next link descriptor address registers contain the address for the next link descriptor in memory. If the contents are transferred to the current descriptor address registers they become effective for the current transfer in basic and extended chaining modes.

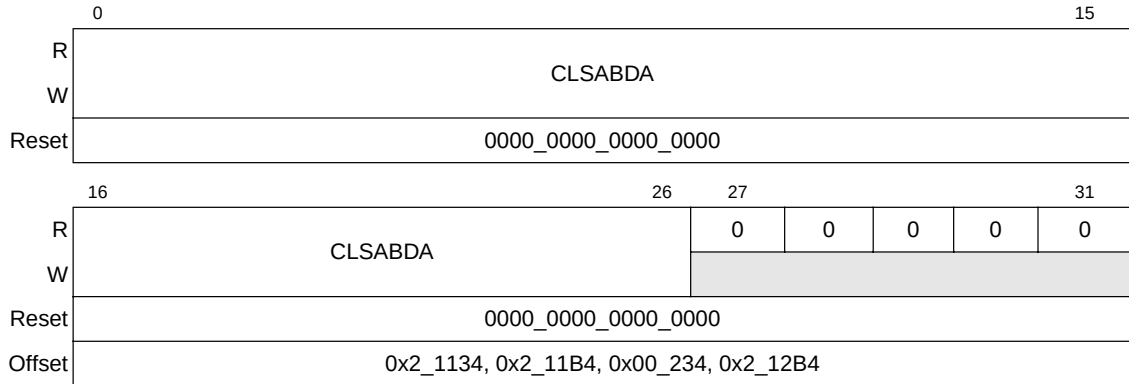
**Figure 15-16. Next Link Descriptor Address Registers (NLNDAR_n)****Table 15-16. NLNDAR_n Field Descriptions**

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.

15.3.2.10 Current List Descriptor Address Registers (CLSDAR_n)

The current list descriptor address registers contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize the CLSDAR_n register 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 NLSDAR_n[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 15-17. Current list Descriptor Address Registers (CLSDAR_n)****Table 15-17. CLSDAR_n Field Descriptions**

Bits	Name	Description
0–26	CLSABDA	Current list descriptor address. Contains the 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	—	Reserved
28–31	—	Reserved

15.3.2.11 Next List Descriptor Address Registers (NLSDAR_n)

The next list descriptor address register contains 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.

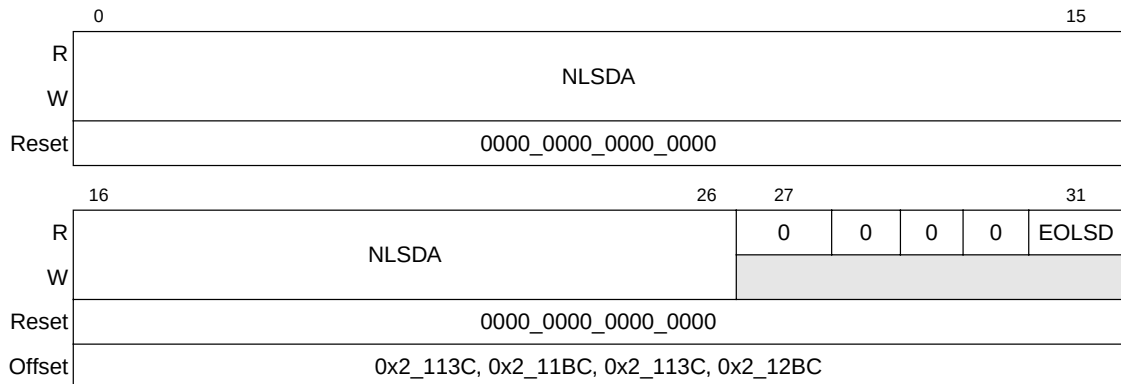
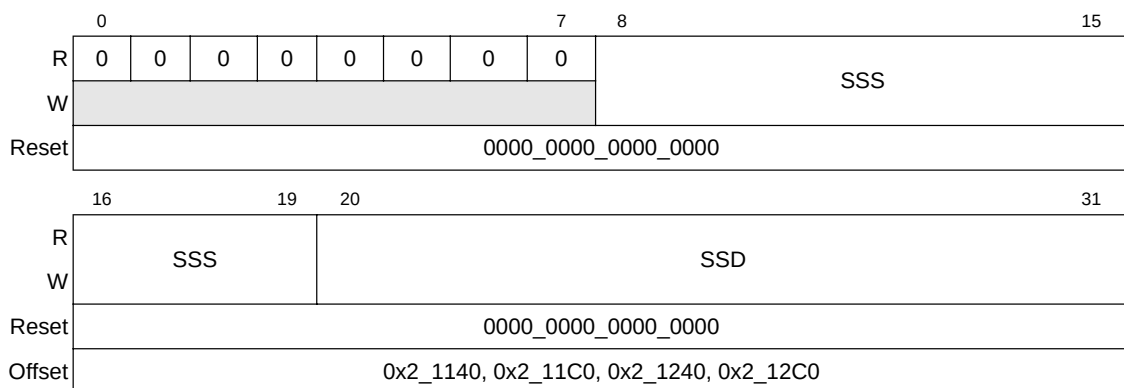
**Figure 15-18. Next List Descriptor Address Registers (NLSDAR_n)**

Table 15-18. NLSDAR_n Field Descriptions

Bits	Name	Description
0–26	NLSDA	Next list descriptor address. Contains the next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27	—	Reserved
28–30	—	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.

15.3.2.12 Source Stride Registers (SSR_n)

The source stride register contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

**Figure 15-19. Source Stride Registers (SSR_n)****Table 15-19. 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.

15.3.2.13 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.

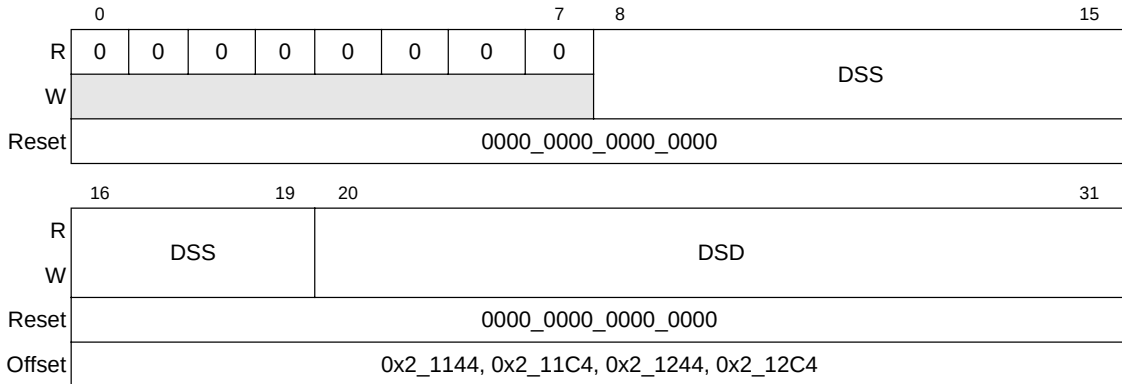


Figure 15-20. Destination Stride Registers (DSRn)

Table 15-20. DSRn 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.

15.3.2.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register, including the enhanced DMA channel. This register is read-only.

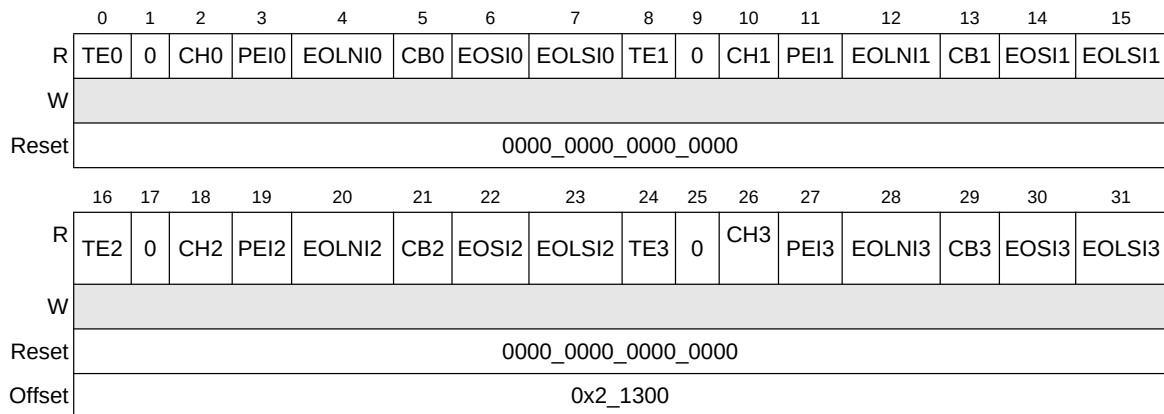


Figure 15-21. DMA General Status Register (DGSR)

Table 15-21. 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
3	PEI0	Programming error interrupt, 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	PEI1	Programming error interrupt, 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	PEI2	Programming error interrupt, 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	PEI3	Programming error interrupt, Channel 3

Table 15-21. DGSR Field Descriptions (continued)

Bits	Name	Description
28	EOLNI3	End-of-links interrupt, Channel 3
29	CB3	Channel busy, Channel 3
30	EOSI3	End-of-segment interrupt, Channel 3
31	EOLSI3	End-of-lists/direct interrupt, Channel 3

15.4 Functional Description

15.4.1 DMA Channel Operation

All DMA channels support two different modes of operation; a basic mode ($MRn[XFE]$ is cleared) and an extended mode ($MRn[XFE]$ is set). In both modes, a channel can be activated by clearing and setting $MRn[CS]$, or via the single-write start mode using $MRn[CDSM/SWSM]$ and $MRn[SRW]$, or via an external control mode using $MRn[ECS\_EN]$. 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 over the basic mode 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 mode register(MRn), the channel enters a paused state after transferring the data described in the bandwidth control. The channel can be restarted from a paused state by the external control.

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 halted condition by setting the channel continue bit, MR_n[CC], in the mode register.

15.4.1.1 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.

15.4.1.1.1 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 the source address(SAR_n), source attributes(SATR_n), destination address(DAR_n), destination attributes(DATR_n), and byte count(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 status register channel busy bit, SR_n[CB], to confirm that the specific DMA channel is idle.(SR_n[CB] is cleared)
2. Initialize the source address(SAR_n), source attributes(SATR_n), destination address(DAR_n), destination attributes(DATR_n) and byte count(BCR_n) registers.
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.

15.4.1.1.2 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 source attributes(SATR_n), destination attributes(DATR_n), and byte count(BCR_n) registers. Setting MR_n[SRW] configures the DMA controller to begin the DMA transfer either when the source address register(SAR_n) is written or when the destination address register(DAR_n) is written, determined by the state of MR_n[CDSM/SWSM]. Writing to the source address register(SAR_n) initiates the DMA transfer if MR_n[CDSM/SWSM] is set. Writing to the destination address register(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 status register channel busy bit, SR_n[CB], to confirm that the specific DMA channel is idle.(SR_n[CB] is cleared)
2. Initialize the source attributes (SATR_n), destination attributes (DATR_n) and byte count (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 the source address register(SAR_n). Clear MR_n[CDSM/SWSM] for transfers started using the destination address register(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.

15.4.1.1.3 Basic Chaining Mode

In basic chaining mode, the 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 status register channel busy bit, $SRn[CB]$, to confirm that the specific DMA channel is idle. ($SRn[CB]$ is cleared)
3. Initialize the current link descriptor address register($CLNDARn$) to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, $MRn[CTM]$, as well as $MRn[XFE]$, 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, $MRn[CS]$, to start the DMA transfer.
6. $SRn[CB]$ is set by the DMA controller to indicate the DMA transfer is in progress.
7. $SRn[CB]$ is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted ($MRn[CA]$ transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.1.4 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($CLNDARn$). Setting $MRn[CDSM/SWSM]$ in the mode register causes $MRn[CS]$ 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, $MRn[CDSM/SWSM]$, and the extended features enable bit $MRn[XFE]$. Also, clear the channel transfer mode bit, $MRn[CTM]$. 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 status register channel busy bit, $SRn[CB]$, to confirm that the specific DMA channel is idle. ($SRn[CB]$ is cleared)
4. Initialize the current link descriptor address register($CLNDARn$) to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set $MRn[CS]$.
5. $SRn[CB]$ is set by the DMA controller to indicate the DMA transfer is in progress.
6. $SRn[CB]$ is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted ($MRn[CA]$ transitions from a 0 to 1), or if an error occurs during any of the transfers.

15.4.1.2 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 $MRn[XFE]$.

15.4.1.2.1 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 $MRn[XFE]$ being set. Striding on the source address can be accomplished by setting $SATRn[SSME]$ and setting the desired stride size and distance in the source stride register($SSRn$). Striding on the destination address can be accomplished by setting $DATRn[DSME]$ and setting the desired stride size and distance in the destination stride register($DSRn$).

15.4.1.2.2 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 $MRn[XFE]$ being set. Striding on the source address can be accomplished by setting $SATRn[SSME]$ and setting the desired stride size and distance in the source stride register($SSRn$). Striding on the destination address can be accomplished by setting $DATRn[DSME]$ and setting the desired stride size and distance in the destination stride register($DSRn$).

15.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then the current list descriptor address register($CLSDARn$) 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 status register channel busy bit, $SRn[CB]$, to confirm that the specific DMA channel is idle.($SRn[CB]$ is cleared)

3. Initialize the current list descriptor address register($CLSDAR_n$) 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.

15.4.1.2.4 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 the current list descriptor address register($CLSDAR_n$) is written. 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 status register channel busy bit, $SR_n[CB]$, to confirm that the DMA channel is idle. ($SR_n[CB]$ is cleared)
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.

15.4.1.3 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

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or pause a 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 $MRn[EMP\_EN]$. $MRn[BWC]$ specifies how much data to allow a specific channel to transfer before entering a paused state by clearing $MRn[CS]$. The channel can be restarted from a paused state by the 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:

- $\overline{DMA\_DREQ}$, Indicates a DMA transfer start or restart from a pause request. Sets $MRn[CS]$.
- $\overline{DMA\_DACK}$, Indicates a DMA transfer currently in progress. $SRn[CB]$ is set.
- $\overline{DMA\_DDONE}$, Indicates the completion of the DMA transfer. $SRn[CB]$ is clear.

Detailed descriptions of the external control interface are in Table 15-4. The timing diagram of the external control interface is shown in Figure 15-22.

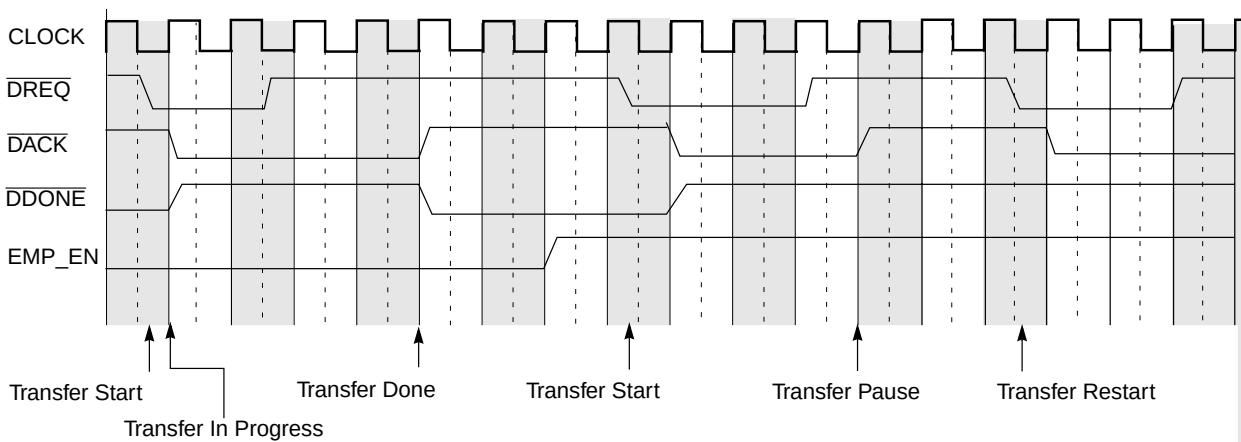


Figure 15-22. External Control Interface Timing

15.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode(enabled when $MRn[CC]$ is set) offers software the flexibility of having the DMA controller 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 stopped state while software continues to build other descriptors in memory. Software can then set $MRn[CC]$ to force hardware to continue where it left off. channel continue is only meaningful for chaining modes, not direct mode.

If $MRn[CC]$ is set by software while the channel is currently 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 the EOLND or EOLSD are 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.

If $MRn[CC]$ 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 the EOLND or EOLSD bit are 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.

15.4.1.4.1 Basic Mode

On a channel continue, the descriptor at the current link descriptor address register ($CLNDARn$) is refetched to get the next link descriptor address field as updated by software. The channel halts if $NLNDARn[EOLND]$ is still set. If $NLNDARn[EOLND]$ is cleared, the next link descriptor address is copied into the $CLNDARn$ register 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.

15.4.1.4.2 Extended Mode

On a channel continue, the descriptor at the current list descriptor ($CLSDARn$) address register is refetched to get the next list descriptor address field as updated by software. The channel halts if $NLSDARn[EOLSD]$ is still set. If not, the next list descriptor address is copied into the $CLSDARn$ register 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.

15.4.1.5 Channel Abort

Software can abort a previously initiated transfer by setting $MRn[CA]$. Once the DMA channel controller detects a 0 to one transition of $MRn[CA]$, 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 $SRn[CB]$. Successful completion of a software initiated abort request can be recognized by $MRn[CA]$ being set and $SRn[CB]$ being cleared. Obviously, if the controller was already halted because of an error condition ($SRn[TE]$ is set), or the channel has completed all transfers, then $SRn[CB]$ being cleared may not signify that the controller entered a halt state due to the abort request.

15.4.1.6 Bandwidth Control

MRn[BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware. This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 kB at a time when no other channel is active.

15.4.1.7 Channel State

Table 15-22 defines the state of a channel based on the values of the channel start (MRn[CS]), channel busy (SRn[CB]), transfer error (SRn[TE]), and channel continue (MRn[CC]) bits.

Table 15-22. Channel State Table

MRn[CS]	SRn[CB]	SRn[TE]	MRn[CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle.
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state.
0	1	0	0	Software halted channel. The channel was busy and software cleared MRn[CS].
0	1	0	1	Channel remains in halt state.
-	1	1	-	The Channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed.
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In Direct mode, the Channel Continue has no effect.
1	0	1	0	Error occurred during transfer.
1	0	1	1	Channel remains in error halt state.
1	1	0	0	Transfer in progress.
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue.

15.4.1.8 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 15-23 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 is repeated until the amount of data transferred equals the transfer size.

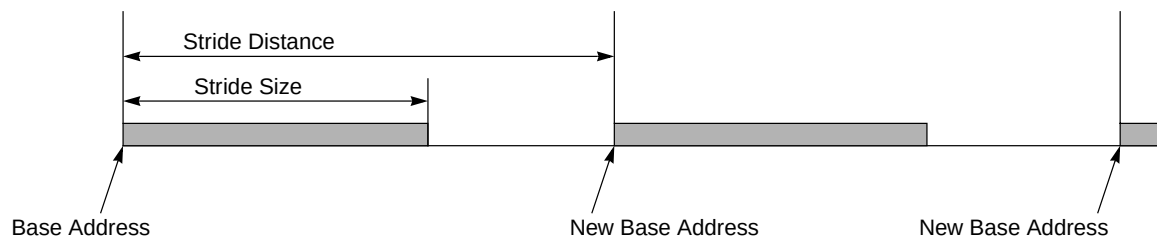


Figure 15-23. Stride Size and Stride Distance

15.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

15.4.3 DMA Errors

On a transfer error, the DMA halts by setting $SRn[TE]$. On a programming error, the DMA sets $SRn[PE]$ and generates an interrupt if enabled ($SRn[PEI]$ 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 types used for the transfer.

15.4.4 DMA Descriptors

List descriptors and link descriptors are the two types of descriptors recognized by the DMA engine. List descriptors are used to 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 aligned on a 32-byte boundary. The last link descriptor in the last list in memory sets the EOTD bit in the next link descriptor and 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.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

The last link descriptor in the last list in memory sets NLNDAR n [EOLND] 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 as shown in Figure 15-26. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor.

Table 15-23. List DMA Descriptor Summary

Descriptor Field	Description
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 address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list.
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list.

Table 15-24. 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 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.

Offset

0x00	Reserved
0x04	Next list descriptor address
0x08	Reserved
0x0C	First link descriptor address
0x10	Source stride
0x14	Destination stride
0x18	Reserved
0x1C	Reserved

Figure 15-24. List Descriptor Format

Offset

0x00	Source attributes
0x04	Source address
0x08	Destination attributes
0x0C	Destination address
0x10	Reserved
0x14	Next link descriptor address
0x18	Byte count
0x1C	Reserved

Figure 15-25. Link Descriptor Format

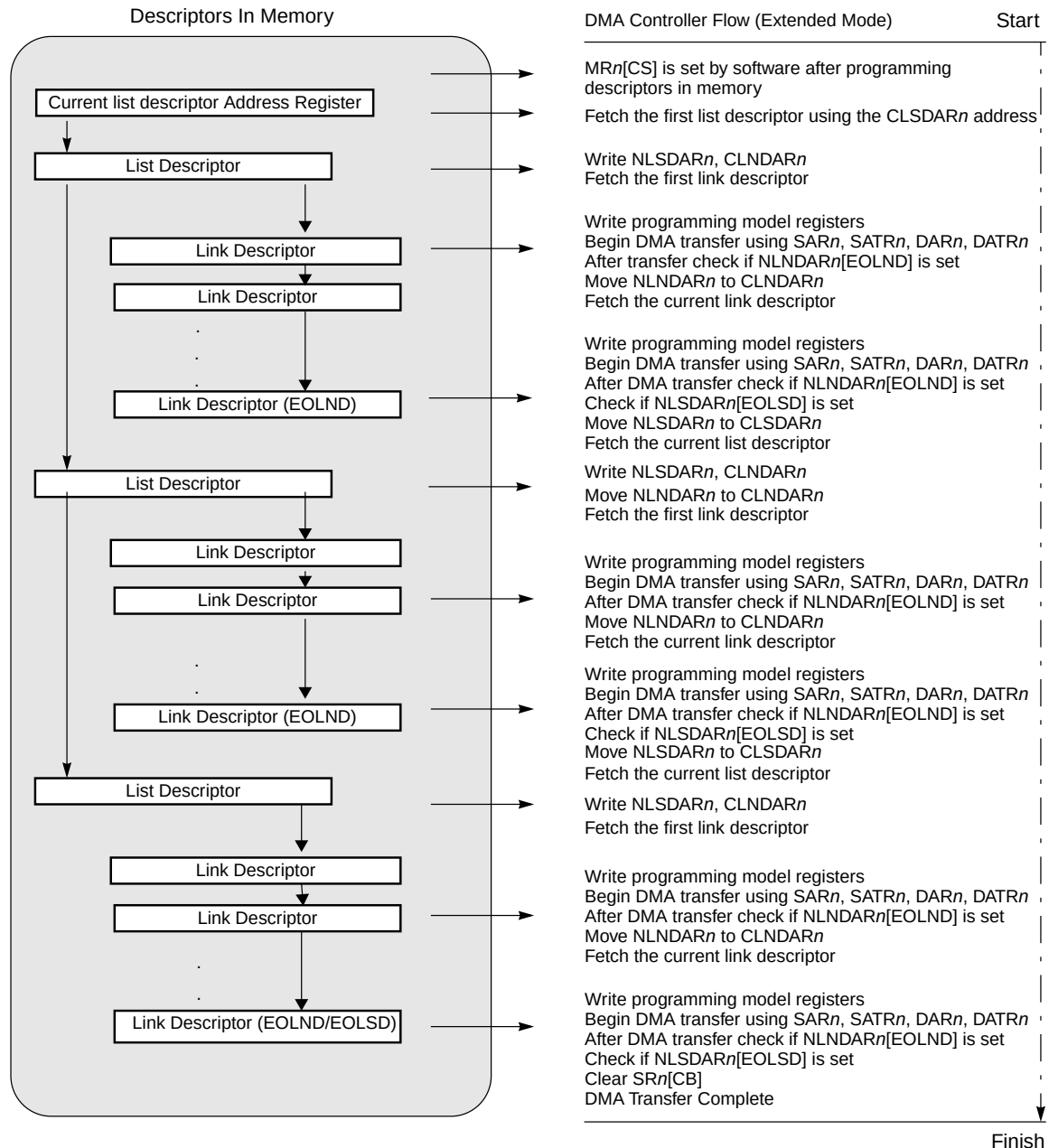


Figure 15-26. DMA Transaction Flow with DMA Descriptors

15.4.5 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:

- 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 32-bytes or greater.
- If $MRn[SAHE]$ is set, the source interface transfer size capability must be greater than or equal to $MRn[SAHTS]$. The source address must be aligned to the size specified by $MRn[SAHTS]$.
- If $MRn[DAHE]$ is set, the destination interface transfer size capability must be greater than or equal to $MRn[DAHTS]$. The destination address must be aligned to the size specified by $MRn[DAHTS]$.
- The stride mode is not supported if $MRn[DAHE]$ or $MRn[SAHE]$ are 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 destination address is double-word aligned and that the byte count is a double-word multiple.
- Striding does not work if the destination transaction type is message because messages have no memory addresses. Software is responsible for disabling striding if the destination transaction is message. Results are boundedly undefined if this restriction is violated.

15.5 DMA System Considerations

Figure 15-27 depicts the most important data paths within the MPC8540. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller. This section provides information about how to make most effective use of these DMA channels including the following topics:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources and destinations
- Transparency of the bus interfaces to DMA operations

- What is useful as opposed to what is possible. For example, it is possible to address any internal register via the internal control bus, which means configuration and control registers can be DMA source or destination targets. However, the typical use of DMA functionality is to reduce host processor loading by moving large amounts of data with minimal CPU involvement. Using a general-purpose DMA controller to load small amounts of configuration data only makes sense in special circumstances (such as, possibly, during system boot).

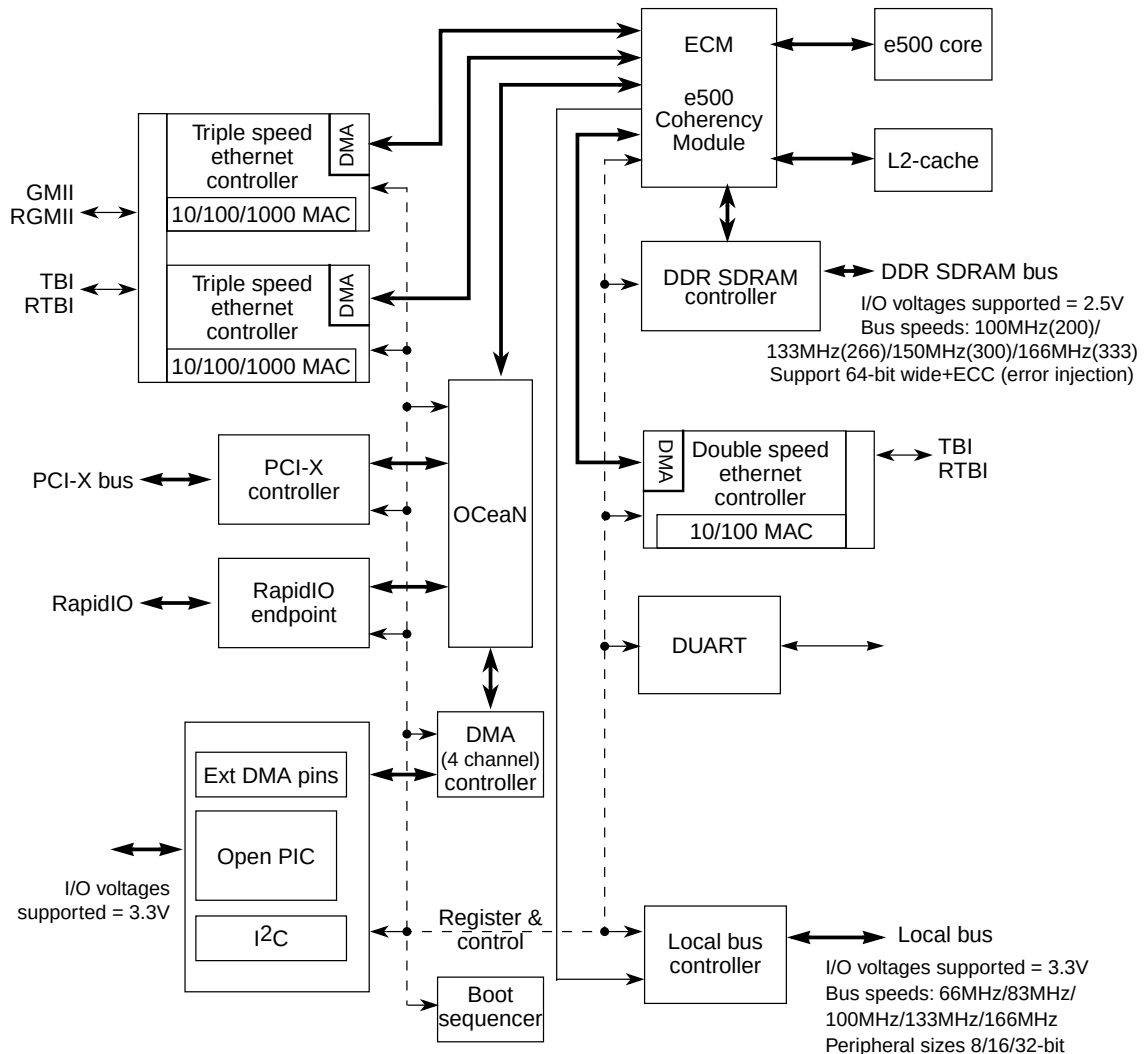


Figure 15-27. Block Diagram Showing Major Data Paths

Table 14-25 lists all the DMA controllers (both captive and general-purpose) on the device and the most likely DMA targets on and off-chip. The bus controllers themselves cannot initiate DMA transfers; rather, they operate transparently with respect to both on- and off-chip DMA controllers. The codes in the table cells have the following meaning:

- Y - Supported

DMA System Considerations

- NR - Possible, but not recommended. Inefficient use of system resources.
- NS - Possible, but not supported. Resulting system behavior is not defined.

Table 15-25. DMA Paths

DMA Controllers		On-Chip Targets		Off-Chip Targets					
		Memory Mapped L2	Config ¹ Registers	DDR	Local Bus	PCI-X	RIO	DUART FIFO	Ethernet Buffers
On-Chip	Ethernet ²	Y	NS	Y	Y	Y	Y	Y	Y
	4Channel ³	Y	NS	Y	Y	Y	Y	Y	Y
Off-Chip	PCI-X Controller	Y	NS	Y	Y	–	Y	Y	Y
	RapidIO Controller	Y	NS	Y	Y	Y	–	Y	Y

¹ Includes I²C data register.

² Captive resource. Not available to external masters.

³ Can serve external masters.

15.5.1 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:

- 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 via an internal control bus, which means configuration and control registers can be DMA targets.

15.5.1.1 DMA to e500 Core

The L1 cache is the only large block of memory inside the e500 core suitable as a typical DMA target. The core's L1 cache cannot be a direct DMA target because this memory 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.

15.5.1.2 DMA to Ethernet

The Ethernet controllers themselves 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 the Ethernet data buffers are located in RAM outside the Ethernet controllers, the general-purpose DMA engines can move data to or from these regions of memory. Additionally, because the 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. In fact, there is no need to create separate DMA transaction descriptors pointing to the same memory range because, by setting a special mode bit, the general-purpose DMA controller can use the Ethernet buffer descriptors.

15.5.1.3 DMA to Configuration and Control Registers

Because any internal register can be addressed with the four-channel DMA controller, configuration and control registers throughout the device are valid DMA targets. However, the primary purpose of DMA—to reduce processor load by moving large blocks of data—is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I²C controller or programmable interrupt controller, 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. Also, because many fields in these registers are reserved, there is a risk of catastrophic misconfiguration when loading them with a DMA controller, which, by its nature is insensitive to the initial value of the reserved fields.

NOTE

Loading configuration registers before reading them is not recommended because many fields in these registers are reserved. Even though reserved fields return 0, this should not be assumed by the programmer. Reserved bits should always be written with the value they returned when read. In other words, all configuration registers should be programmed by reading the value, modifying the appropriate fields, and writing back the value. This read-modify-write requirement is not satisfied by normal DMA transfers. Ignoring this requirement results in unpredictable system behavior.

15.5.1.4 DMA to I²C

The I²C controller is not transparent to DMA transfers. Observe the caveats listed in Section 15.5.1.3, “DMA to Configuration and Control Registers,” when accessing any I²C register, including the data register (I2CDR).

15.5.1.5 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in Chapter 12, “DUART,” specifically, Section 12.4.5, “FIFO Mode.”

Chapter 16

PCI Bus Interface

16.1 Introduction

The MPC8540 PCI/PCI-X interface complies with the *PCI Local Bus Specification*, Rev. 2.2 and the *PCI-X Addendum to the PCI Local Bus Specification*, Rev. 1.0a. It is beyond the scope of this manual to document the intricacies of the PCI and PCI-X buses. This chapter provides a basic description of the PCI and PCI-X bus operations. The specific emphasis is directed at how the MPC8540 implements the PCI and PCI-X buses. Designers of systems incorporating PCI/PCI-X devices should refer to the respective specification for a thorough description of the PCI/PCI-X buses.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Because this is inconsistent with the terminology in this manual, the terms ‘word’ and ‘double word’ are not used in this chapter. Instead, the number of bits or bytes indicates the exact quantity.

This chapter describes the PCI/PCI-X (PCI/X) controller of this device. The controller acts as a bridge between the PCI/X interface and the OCeaN switch fabric. Figure 16-1 is a high-level block diagram of the PCI/X controller.

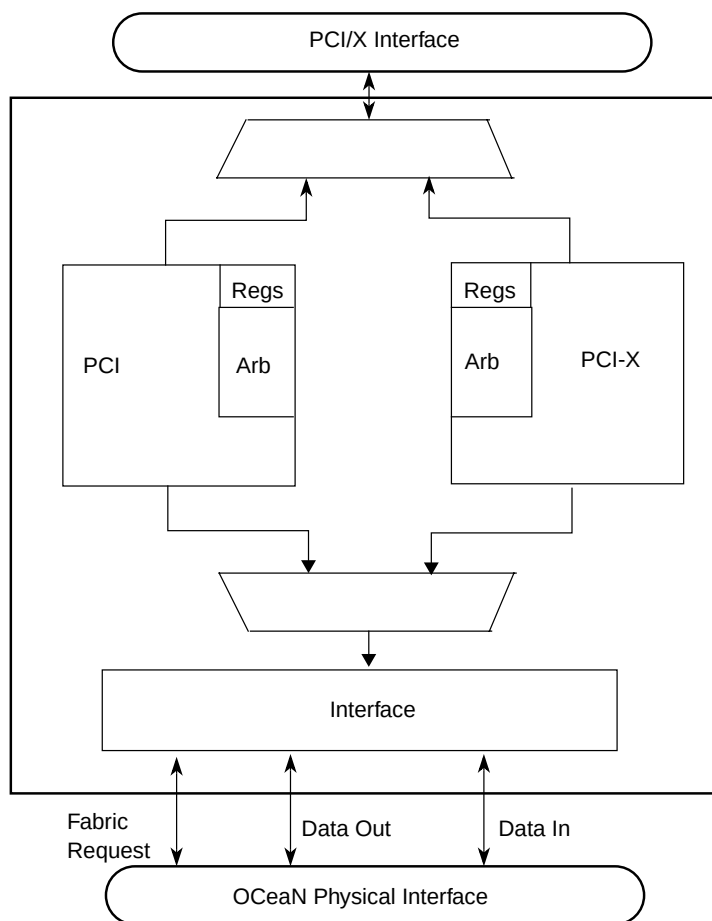


Figure 16-1. PCI/X Controller Block Diagram

16.1.1 Overview

The PCI interface connects the OCeaN to the PCI/PCI-X bus, to which I/O components are connected. The PCI bus uses a 32- or 64-bit multiplexed address/data bus, plus various control and error signals. The PCI interface supports address and data parity with error checking and reporting.

The PCI interface of the MPC8540 functions both as a master (initiator) and a target device. Internally, the design is divided into the following:

- Data path blocks
- Control logic blocks
- Memory

The data path blocks contain the queues, tables for transaction tracking, and ordering. The control blocks contain control logic and state-machines for buffer control, bus protocol, tag

generation, and transaction resizing. The memory blocks are used solely for inbound and outbound data storage.

This allows the MPC8540 to handle separate PCI transactions simultaneously. For example, if the MPC8540, as an initiator, is trying to run a burst-write to a PCI device, it may get disconnected before finishing the transaction. If another PCI device is granted the PCI bus and requests a burst-read from local memory, the MPC8540, as a target, can accept the burst-read transfer. When the MPC8540 is granted mastership of the PCI bus, the burst-write transaction continues.

There are two blocks of memory in the design:

- The inbound buffers.
- The outbound read buffers combined with the outbound write buffers.

There are many blocks of control logic in the block. On the PCI/X side there are machines for PCI/X controller initiated address and data tenures for inbound and outbound data, respectively. On the OCeaN side there are machines for fabric arbitration, outbound data and inbound data.

As an initiator, the MPC8540 supports read and write operations to the PCI memory space, the PCI I/O space, and the 256-byte PCI configuration space. As an initiator, the MPC8540 also supports generating PCI special-cycle and interrupt-acknowledge transactions. As a target, the MPC8540 supports read and write operations to local memory, and, when configured in agent mode, read and write operations to the internal PCI configuration registers.

The MPC8540 can function as either a PCI host bridge referred to as PCI host mode or a peripheral device on the PCI bus referred to as PCI agent mode. See Section 16.1.3.1, “Host/Agent/Agent Lock Mode,” for more information.

In PCI agent mode, all of the PCI configuration registers in the MPC8540 can be programmed from the PCI bus. See Section 16.3.2.2, “PCI Agent Mode Configuration Register Access,” for more information.

The PCI interface provides bus arbitration for the MPC8540 and up to five other PCI bus masters. The arbitration algorithm is a programmable two-level, round-robin priority selector. The on-chip PCI arbiter can operate in both host and agent modes or it can be disabled to allow for an external PCI arbiter.

The MPC8540 also provides an address translation mechanism to map inbound PCI to OCeaN accesses and outbound OCeaN to PCI accesses. The interface can be programmed for either little-endian or big-endian formatted data, and provides data lane redirection in hardware.

16.1.1.1 MPC8540 as a PCI Initiator

Upon detecting an OCeaN-to-PCI transaction, the MPC8540 requests the use of the PCI bus. For OCeaN-to-PCI bus write operations, the MPC8540 requests mastership of the PCI bus when the source completes the write operation to the OCeaN. For OCeaN-to-PCI read operations, the MPC8540 requests mastership of the PCI bus when it decodes that the access is for PCI address space.

Once granted, the MPC8540 drives the PCI address (PCI_AD[63:0]) and the bus command (PCI_C/BE[7:0]) signals.

The master part of the interface can initiate master-abort cycles, recognizes target-abort, target-retry, and target-disconnect cycles, and supports various device selection timings. The master interface does not run fast back-to-back or exclusive accesses.

16.1.1.2 MPC8540 as a PCI Target

As a target, on detection of a PCI address phase, the MPC8540 decodes the address and bus command to determine if the transaction is within the local memory access boundaries. If the transaction is destined for local memory, the target interface latches the address, decodes the PCI bus command, and forwards the transaction to the OCeaN control unit. On writes to local memory, data is forwarded along with the byte enables to the internal control unit. On reads, the data is driven on the PCI bus and the byte enables determine which byte lanes contain meaningful data.

The target interface of the MPC8540 can issue target-abort, target-retry, and target-disconnect cycles. The target interface supports fast back-to-back transactions. The target interface uses the fastest device selection timing.

The MPC8540 supports data streaming to and from local memory. This means that the MPC8540 can accept or provide data to or from local memory as long as the internal buffers are not filled.

16.1.2 Features

The following is a list of PCI features that is supported:

- PCI interface 2.2 compatible
- 66 MHz and 33 MHz support
- 64- and 32-bit PCI interface support on primary PCI port
- Host and agent mode support
- 64-bit dual address cycle (DAC) support
- On-chip arbitration with support for 5 high priority request and grant signal pairs
- Support for accesses to all PCI memory and I/O address spaces

- Support for PCI to memory and memory to PCI streaming
- Memory prefetching of PCI read accesses
- Support posting of processor to PCI and PCI to memory writes
- Support selectable snoop for Inbound accesses
- PCI configuration registers
- PCI 3.3V compatible

The following is a list of PCI-X features supported:

- PCI-X rev 1.0a compatible
- Support for up to 133 MHz point to point connection
- Support for 32-bit and 64-bit interface
- 64-bit dual address cycle (DAC) support
- On-chip arbitration with support for 5 high priority request and grant signal pairs
- Support for accesses to all PCI-X memory and I/O address spaces
- Support for 4 split transactions
- Complete allowable disconnect boundary (ADB) support
- All PCI-X ordering rules enforced
- Implementation of relaxed-ordering
- PCI-X 3.3V compatible

16.1.3 Modes of Operation

16.1.3.1 Host/Agent/Agent Lock Mode

The MPC8540 can function as either a PCI host bridge (referred to as PCI host mode) or a peripheral device on the PCI bus (referred to as PCI agent mode). The $\overline{\text{LWE2}}$ configuration signal, sampled at reset, configures the MPC8540 for PCI host or PCI agent mode as described in Chapter 4, “Reset, Clocking, and Initialization.”

Note that in PCI agent mode, the MPC8540 ignores all PCI memory accesses (except to the memory mapped registers) until inbound address translation is enabled. See Section 16.5.1, “Power-On Reset Configuration Modes,” for more information.

- Host mode. When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). See Section 16.5.1.1, “Host Mode,” for more information.
- Agent mode. When the device powers up in agent mode, it acknowledges inbound configuration accesses. See Section 16.5.1.2, “Agent Mode,” for more information.

- Agent config lock mode. When the device powers up in agent config lock mode, it retries inbound configuration accesses until the ACL bit is cleared to 0. See Section 16.5.1.3, “Agent Config Lock Mode,” for more information.

16.1.3.1.1 PCI/PCI-X Signal Output Hold Timing

To meet minimum output hold specifications relative to SYSCLK for both PCI and PCI-X systems, the MPC8540 has a programmable output hold delay for PCI signals. The initial value of the output hold delay is determined by the values on the $\overline{\text{LWE}}[0:1]$ power-on reset configuration signals. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8540 Hardware Specifications*, for more information on these values and signal timing.

16.2 External Signal Descriptions

Figure 16-2 shows the external PCI/PCI-X signals.

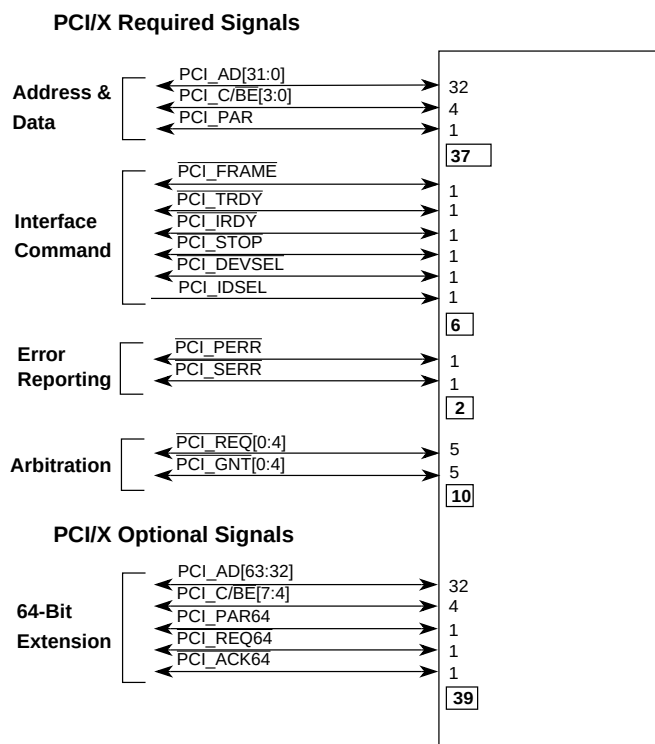


Figure 16-2. PCI/X Interface External Signals

Table 16-1 contains the detailed descriptions of the external PCI/PCI-X interface signals.

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
PCI_ACK64	I/O	64-bit transaction acknowledge. The 64-bit transaction acknowledge (PCI_ACK64) signal is both an input and output signal. It indicates that the current target supports 64-bit transfers during the data phase of the current transaction.	
	O	As outputs for the bi-directional 64-bit transaction acknowledge, these signals operate as follows:	
		State Meaning	Asserted—Indicates that this PCI controller, as the target of a PCI transaction, may use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that this PCI controller, as the target of a PCI transaction, may use only 32 bits of the data bus in servicing a data transfer.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional 64-bit transaction acknowledge, these signals operate as follows:	
		State Meaning	Asserted—Indicates that the target of a PCI transaction may use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that the target of a PCI transaction may only use 32 bits of the data bus during the data phase of the transaction.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_AD[63:0]	I/O	PCI address/data bus. The PCI address/data bus (PCI_AD[63:0]) consists of 64 signals that are both input and output signals on this PCI controller.	
	O	As outputs for the bi-directional PCI address/data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the physical address during the address phase of a PCI transaction. During the data phase(s) of a PCI transaction, the PCI address/data bus contain the data being written. The PCI_AD[7:0] signals define the LSB and PC_AD[31:24] the MSB.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional PCI address/data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the address to be decoded as a check for device select during the address phase of a PCI transaction or the data being received during the data phase(s) of a PCI transaction.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
PCI_C/BE[7:0]	I/O	Command/byte enable. The command/byte enable (PCI_C/BE[7:0]) signals are both input and output signals on this PCI controller. The command encodings are as follows: PCI_C/BE[3:0] Command 0b0000 Interrupt acknowledge 0b0001 Special cycle 0b0010 I/O read 0b0011 I/O write 0b0100 Reserved 0b0101 Reserved 0b0110 Memory read 0b0111 Memory write 0b1000 Reserved 0b1001 Reserved 0b1010 Configuration read 0b1011 Configuration write 0b1100 Memory read multiple 0b1101 Dual access cycle 0b1110 Memory read line 0b1111 Memory write and invalidate
		O As outputs for the bi-directional command/byte enable, these signals operate as described below.
		State Meaning Asserted/Negated—During the address phase, PCI_C/BE[7:0] define the bus command. Byte enables determine which byte lanes carry meaningful data. The PCI_C/BE0 signal applies to the LSB.
		Timing Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
		I As inputs for the bi-directional command/byte enable, these signals operate as described below.
		State Meaning Asserted/Negated—During the address phase, PCI_C/BE[7:0] indicate the command that another master is sending. During the data phase, PCI_C/BE[7:0] indicate which byte lanes are valid.
		Timing Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_DEVSEL	I/O	Device select. The device select (PCI_DEVSEL) signal is both an input and output signal on this PCI controller.
		O As outputs for the bi-directional device select, these signals operate as described below.
		State Meaning Asserted—Indicates that this PCI controller has decoded the address and is the target of the current access. Negated—Indicates that this PCI controller has decoded the address and is not the target of the current access.
		Timing Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
		I As inputs for the bi-directional device select, these signals operate as described below.
		State Meaning Asserted—Indicates that some PCI agent (other than this PCI controller) has decoded its address as the target of the current access. Negated—Indicates that no PCI agent has been selected.
		Timing Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_FRAME	I/O	Frame. The frame (PCI_FRAME) signal is both an input and output signal on this PCI controller.	
		O As outputs for the bi-directional frame, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI master, is initiating a bus transaction. While PCI_FRAME is asserted, data transfers may continue. Negated—If PCI_IRDY is asserted, indicates that the PCI transaction is in the final data phase; if PCI_IRDY is negated, indicates that the PCI bus is idle.
	I	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
		As inputs for the bi-directional frame, these signals operate as described below.	
		State Meaning	Asserted—Indicates that another PCI master is initiating a bus transaction. Negated—Indicates that the transaction is in the final data phase or that the bus is idle.
PCI_GNT[4:0]	O	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
		PCI bus grant. Output signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled PCI_GNT[0] is an input signal. Note that PCI_GNT[n] is a point-to-point signal. Every master has its own bus grant signal.	
		State Meaning	Asserted—Indicates that this PCI controller has granted control of the PCI bus to agent <i>n</i> . Negated—Indicates that this PCI controller has not granted control of the PCI bus to agent <i>n</i> .
PCI_IDSEL	I	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
		Initialization device select. The initialization device select (PCI_IDSEL) signal is an input signal on this PCI controller. It is used as a chip select during configuration read and write transactions.	
		State Meaning	Asserted—Indicates this PCI controller is being selected as a target of a configuration read or write transactions. Negated—Indicates this PCI controller is not being selected as a target of configuration read or write transactions.

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
PCI_IRDY	I/O	Initiator ready. The initiator ready (PCI_IRDY) signal is both an input and output signal on this PCI controller.
	O	As outputs for the bi-directional initiator ready, these signals operate as described below.
	State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI master, can complete the current data phase of a PCI transaction. During a write, this PCI controller asserts PCI_IRDY to indicate that valid data is present on PCI_AD[63:0]. During a read, this PCI controller asserts $\overline{\text{PCI\_IRDY}}$ to indicate that it is prepared to accept data. Negated—Indicates that the PCI target needs to wait before this PCI controller, acting as a PCI master, can complete the current data phase. During a write, this PCI controller negates $\overline{\text{PCI\_IRDY}}$ to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates $\overline{\text{PCI\_IRDY}}$ to insert a wait cycle when it cannot accept data from the target.
	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional initiator ready, these signals operate as described below.
	State Meaning	Asserted—Indicates another PCI master is able to complete the current data phase of a transaction. Negated—If $\overline{\text{PCI\_FRAME}}$ is asserted, indicates a wait cycle from another master. If $\overline{\text{PCI\_FRAME}}$ is negated, indicates the PCI bus is idle.
PCI_PAR	I/O	PCI parity. The PCI parity (PCI_PAR) signal is both an input and output signal on this PCI controller.
	O	As outputs for the bi-directional PCI parity, these signals operate as described below.
	State Meaning	Asserted—Indicates odd parity across the PCI_AD[31:0] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals during address and data phases. Negated—Indicates even parity across the PCI_AD[31:0] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals during address and data phases.
	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional PCI parity, these signals operate as described below.
	State Meaning	Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases. Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.
PCI_PAR	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_PAR64	I/O	Upper DWORD parity. The PCI parity (PCI_PAR64) signal is both an input and output signal on this PCI controller. PCI_PAR64 is the even parity bit that protects the upper 32 bits of data and upper 4 bits of command/byte enable.	
	O	As outputs for the bi-directional Upper DWORD Parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity across the PCI_AD[63:32] and PCI_C/BE[7:4] signals during address and data phases. Negated—Indicates even parity across the PCI_AD[63:32] and PCI_C/BE[7:4] signals during address and data phases.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional Upper DWORD Parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity driven by another PCI master or the PCI target during read data phases. Negated—Indicates even parity driven by another PCI master or the PCI target during read data phases.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_PERR	I/O	PCI parity error. The PCI parity error (PCI_PERR) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional PCI parity error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI agent, detected a data parity error. (The PCI initiator drives PCI_PERR on read operations; the PCI target drives PCI_PERR on write operations.) Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional PCI parity error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that another PCI agent detected a data parity error while this PCI controller was sourcing data (this PCI controller was acting as the PCI initiator during a write, or was acting as the PCI target during a read). Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_REQ[4:0]	I	PCI bus request. Input signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled, PCI_REQ[0] is an output. Note that PCI_REQ[n] is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the PCI_REQ[n] input.	
		State Meaning	Asserted—Indicates that agent <i>n</i> is requesting control of the PCI bus to perform a transaction. Negated—Indicates that agent <i>n</i> does not require use of the PCI bus.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_REQ64	I/O	64-bit transaction request. Both an input and output signal on this PCI controller. It indicates that the current master desires to transfer data using 64-bit transfers.	
	O	As outputs for the bi-directional 64-bit transaction request, these signals operate as follows:	
		State Meaning	Asserted—Indicates that this PCI controller, as the master of a PCI transaction, desires to use all 64 bits. Negated—Indicates that this PCI controller, as the master of a PCI transaction, uses only 32 bits of the data bus in servicing a data transfer.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional 64-bit transaction request, these signals operate as described below.	
		State Meaning	Asserted—Indicates that the master of a PCI transaction is requesting to use the full 64-bit data bus for the data phase of the transaction. Negated—Indicates that the master of a PCI transaction uses only 32 bits of the data bus during the data phase of the transaction.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_SERR	I/O	PCI system error. The PCI system error ($\overline{\text{PCI\_SERR}}$) signal is both an input and output signal on this PCI controller.	
	O	As outputs for the bi-directional PCI system error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that an address parity error, a target-abort (when this PCI controller is acting as the initiator), or some other system error (where the result is a catastrophic error) was detected. Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional PCI system error, these signals operate as described below.	
		State Meaning	Asserted—Indicates that a target (other than this PCI controller) has detected a catastrophic error. Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 16-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
PCI_STOP	I/O	Stop. The stop (PCI_STOP) signal is both an input and output signal on this PCI controller.
	O	As outputs for the bi-directional stop, these signals operate as described below.
	State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI target, is requesting that the initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional stop, these signals operate as described below.
	State Meaning	Asserted—Indicates that a target is requesting that the PCI initiator stop the current transaction. Negated—Indicates that the current transaction can continue.
PCI_TRDY	I/O	Target ready. Both an input and output signal on this PCI controller.
	O	As outputs for the bi-directional target ready, these signals operate as described below.
	State Meaning	Asserted—Indicates that this PCI controller, acting as a PCI target, can complete the current data phase of a PCI transaction. During a read, this PCI controller asserts PCI_TRDY to indicate that valid data is present on PCI_AD[31:0]. During a write, this PCI controller asserts $\overline{\text{PCI\_TRDY}}$ to indicate that it is prepared to accept data. Negated—Indicates that the PCI initiator needs to wait before this PCI controller, acting as a PCI target, can complete the current data phase. During a read, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates $\overline{\text{PCI\_TRDY}}$ to insert a wait cycle when it cannot accept data from the initiator.
	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional target ready, these signals operate as described below.
	State Meaning	Asserted—Another PCI target is able to complete the current data phase of a transaction. Negated—Indicates a wait cycle from another target.
PCI_TRDY	Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

16.3 Memory Map/Register Definitions

The PCI/PCI-X block of the MPC8540 supports the following two types of registers:

- Memory-mapped registers—these registers control PCI address translation, PCI error management, and PCI configuration register access on the MPC8540. These registers are described in Section 16.3.1, “PCI/PCI-X Memory Mapped Registers,” and its subsections.
- PCI/X configuration registers contained within the PCI/X configuration header—these registers are specified by the PCI bus specification for every PCI

device. These registers are described in Section 16.3.2, “PCI/PCI-X Configuration Header,” and its subsections.

16.3.1 PCI/PCI-X Memory Mapped Registers

The PCI/PCI-X 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 PCSRBAR on the PCI/PCI-X side) plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except PCI Configuration Data Access register, PCI_CFG_DATA) must only be accessed as 32-bit quantities.

Table 16-2 lists the memory-mapped registers.

Table 16-2. PCI/X Memory-Mapped Register Map

Offset	Register	Access	Reset	Section/Page
PCI/X Configuration Access Registers				
0x0_8000	PCI Configuration address access (PCI_CFG_ADDR)	R/W	0x0000_0000	16.3.2.1/16-34
0x0_8004	PCI Configuration data access (PCI_CFG_DATA)	R/W	0x0000_0000	16.3.2.1/16-34
0x0_8008	PCI/X Int_Ack access	R	0x0000_0000	16.3.1.4/16-33
0x0_800C - 0x0_8BFC	Reserved	—	—	—
PCI/X ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C—Outbound Window 0 (default)				
0x0_8C00	PCI/X Outbound window 0 (default) translation address register	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8C04	PCI/X Outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-18
0x0_8C08	PCI/X outbound window 0 (default) base address register	R/W	0x0000_0000	16.3.1.1.3/16-19
0x0_8C0C	Reserved	—	—	
0x0_8C10	PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.1.4/16-19
0x0_8C14	Reserved	—	—	
0x0_8C18	Reserved	—	—	
0x0_8C1C	Reserved	—	—	
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	PCI/X Outbound window 1 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8C24	PCI/X Outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-18
0x0_8C28	PCI/X Outbound window 1 base address register	R/W	0x0000_0000	16.3.1.1.3/16-19
0x0_8C2C	Reserved	—	—	

Table 16-2. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C30	PCI/X Outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.1.4/16-19
0x0_8C34	Reserved	—	—	
0x0_8C38	Reserved	—	—	
0x0_8C3C	Reserved	—	—	
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	PCI/X Outbound window 2 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8C44	PCI/X Outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-18
0x0_8C48	PCI/X Outbound window 2 base address register	R/W	0x0000_0000	16.3.1.1.3/16-19
0x0_8C4C	Reserved	—	—	
0x0_8C50	PCI/X Outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.1.4/16-19
0x0_8C54	Reserved	—	—	
0x0_8C58	Reserved	—	—	
0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	PCI/X Outbound window 3 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8C64	PCI/X Outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-18
0x0_8C68	PCI/X Outbound window 3 base address register	R/W	0x0000_0000	16.3.1.1.3/16-19
0x0_8C6C	Reserved	—	—	
0x0_8C70	PCI/X Outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.1.4/16-19
0x0_8C74	Reserved	—	—	
0x0_8C78	Reserved	—	—	
0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	PCI/X Outbound window 4 translation address register	R/W	0x0000_0000	16.3.1.1.1/16-17
0x0_8C84	PCI/X Outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.1.2/16-18
0x0_8C88	PCI/X Outbound window 4 base address register	R/W	0x0000_0000	16.3.1.1.3/16-19
0x0_8C8C	Reserved	—	—	
0x0_8C90	PCI/X Outbound window 4 attributes register	R/W	0x0000_0000	16.3.1.1.4/16-19
0x0_8C94	Reserved	—	—	
0x0_8C98	Reserved	—	—	
0x0_8C9C	Reserved	—	—	

Table 16-2. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8CA0 — 0x0_8D9C	Reserved	—	—	
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PCI/X Inbound window 3 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-22
0x0_8DA4	PCI/X Inbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-37
0x0_8DA8	PCI/X Inbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.2/16-23
0x0_8DAC	Reserved	—	—	
0x0_8DB0	PCI/X Inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-24
0x0_8DB4	Reserved	—	—	
0x0_8DB8	Reserved	—	—	
0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PCI/X Inbound window 2 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-22
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PCI/X Inbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.2/16-23
0x0_8DCC	PCI/X Inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.2.3/16-23
0x0_8DD0	PCI/X inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-24
0x0_8DD4	Reserved	—	—	
0x0_8DD8	Reserved	—	—	
0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PCI/X inbound window 1 translation address register	R/W	0x0000_0000	16.3.1.2.1/16-22
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PCI/X Inbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.2/16-23
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PCI/X Inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-24
0x0_8DF4	Reserved	—	—	
0x0_8DF8	Reserved	—	—	
0x0_8DFC	Reserved	—	—	
PCI/X Error Management Registers				
0x0_8E00	PCI/X error detect register	Special	0x0000_0000	16.3.1.3/16-26
0x0_8E04	PCI/X error capture disabled register	R/W	0x0000_0000	16.3.1.3/16-26

Table 16-2. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8E08	PCI/X error enable register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E0C	PCI/X error attributes capture register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E10	PCI/X error address capture register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E14	PCI/X error extended address capture register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E18	PCI/X error data low capture register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E1C	PCI/X error data high capture register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E20	PCI/X gasket timer register	R/W	0x0000_0000	16.3.1.3/16-26
0x0_8E24– 0x0_8EFC	Reserved	—	—	
0x0_8F00– 0x0_8FFC	Reserved for debug	—	—	

16.3.1.1 PCI/X ATMU Outbound Registers

The outbound address translation and mapping unit controls the mapping of transactions from the internal 32-bit address space of the MPC8540 to the external PCI address space. The outbound ATMU consists of four translation windows plus a default translation for transactions that do not hit in one of the four windows.

Each window contains a base address that points to the beginning of the window in the local address map, a translation address that specifies the high-order bits of the transaction in the external PCI address space, and a set of attributes including window size and external transaction type.

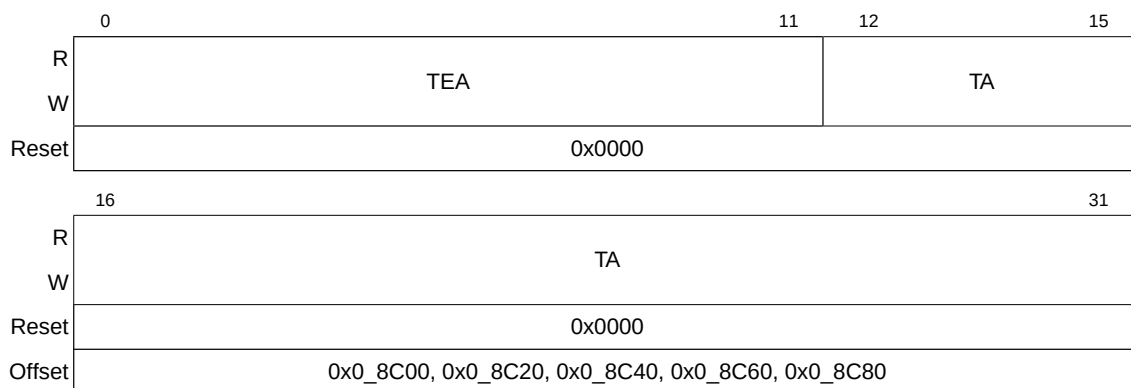
Each window must be aligned to the size of the window. If two outbound ATMU windows overlap in the local address space, the mapping of the lower number window are applied.

Window 0 is the default window and is initially enabled upon reset.

The outbound translation windows must be aligned based on the granularity selected by the size fields. Outbound window misses either generate an addressing error or use the default outbound register set. Overlapping outbound windows resolve to the lowest numbered window.

16.3.1.1.1 PCI/X Outbound Translation Address Registers (POTAR)

The PCI/X outbound translation address registers select the starting addresses in the PCI address space for window hits within the PCI/X outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address. Table 16-3 shows the field descriptions of the PCI/X outbound translation address registers.

**Figure 16-3. PCI/X Outbound Translation Address Register****Table 16-3. Bit Settings for POTAR**

Bits	Name	Description
0–11	TEA	Translation extended address. Represents bits 43–32 of a 64-bit PCI address (bit 0 is lsb).
12–31	TA	Translation address. Represents bits 31–12 of the PCI address. Based on the size of the window, the low-order bits of this field may be ignored.

16.3.1.1.2 PCI/X Outbound Translation Extended Address Registers (POTEAR)

The PCI/X outbound translation extended address registers contain the most significant bits of a 64-bit translation address. Table 16-4 shows the field descriptions of the PCI/X outbound translation extended address registers.

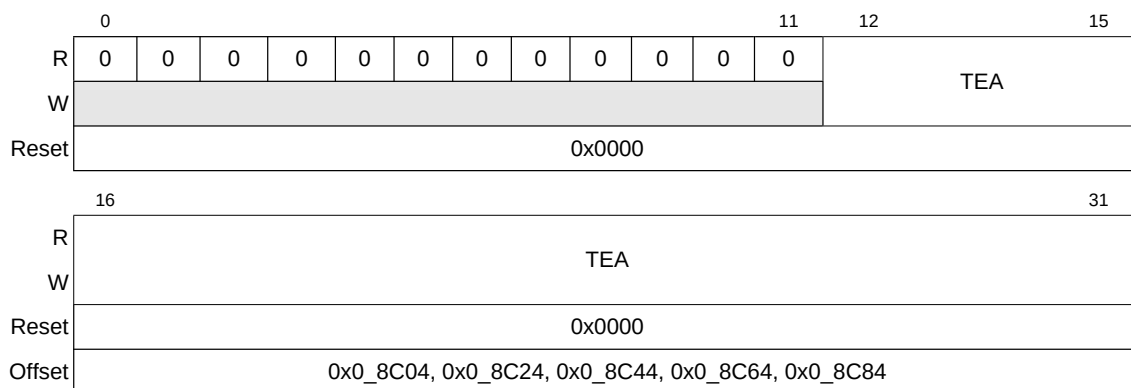
**Figure 16-4. PCI/X Outbound Translation Extended Address Register**

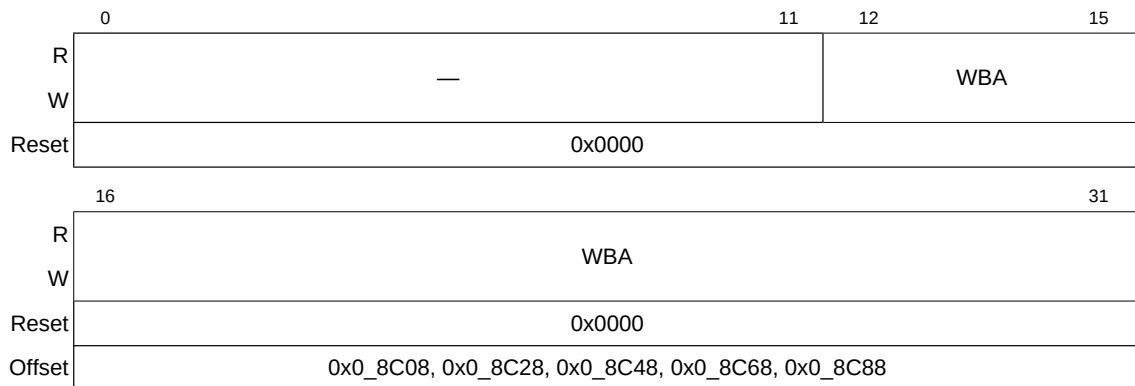
Table 16-4. Bit Settings for POTEAR

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. Comprise bits [44–63] of the translation address register

16.3.1.1.3 PCI/X Outbound Window Base Address Registers (POWBAR)

The PCI/X outbound window base address registers point to the beginning of each translation window in the local 32-bit address space. Addresses for outbound transactions are compared to the appropriate bits in these registers, according to the sizes of the windows. If a transaction does not fall within one of these windows, the default translation and mapping is used. The default window is always used when the other windows miss. It cannot be disabled, and no errors are generated.

Note that POWBAR for outbound ATMU window 0 is not used, because window 0 is the default window used when no other windows match. The registers may be read and written, but the value has no effect.

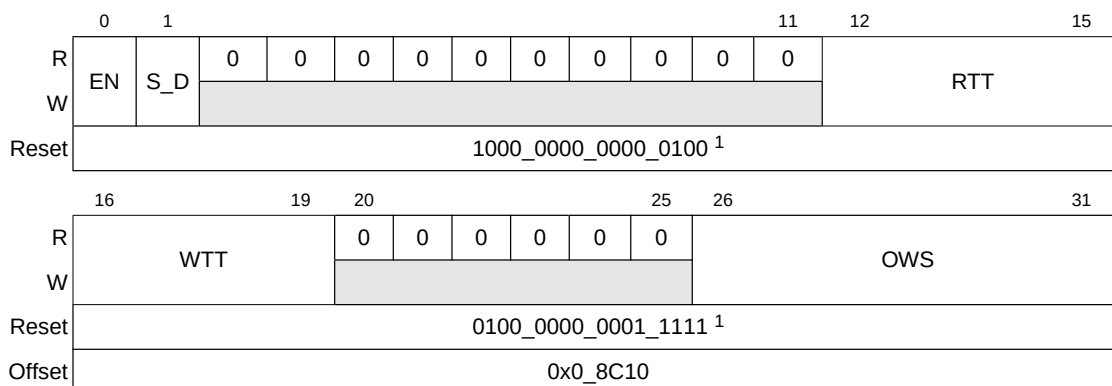
**Figure 16-5. PCI/X Outbound Window Base Address Register****Table 16-5. Bit Settings for POWBAR**

Bits	Name	Description
0–11	—	Reserved, should be cleared.
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.

16.3.1.1.4 PCI/X Outbound Window Attributes Registers (POWAR)

The PCI/X outbound window attributes registers define the window sizes to translate and other attributes for the translations. The minimum window size is 4 Kbytes. The maximum window size is 4 Gbytes. Figure 16-6 shows the reset values for POWAR0 (note that the fields for all of the POWAR registers are the same, only the reset values are different).

Memory Map/Register Definitions



¹ The default window is enabled, configured for memory read and memory write, and set to an OWS size of 4 Gbytes.

Figure 16-6. PCI/X Outbound Window Attributes Register 0 (Default)

Figure 16-7 shows the reset values for POWAR1–POWAR4.

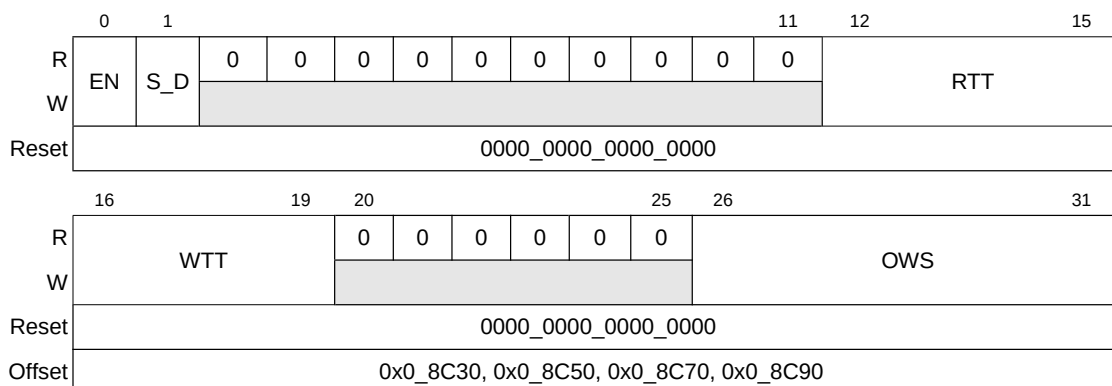


Figure 16-7. PCI/X Outbound Window Attributes Registers 1–4

Table 16-6. POWAR_n Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation
1	S_D	ByteSwap_Dis. This bit disables byte lane redirection on the data bus
2–11	—	Reserved

Table 16-6. POWARn Field Descriptions (continued)

Bits	Name	Description
12–15	RTT	Read. Transaction type to run on PCI 0000 - reserved ... 0011 - reserved 0100 - Memory Read 0101 - reserved ... 0111 - reserved 1000 - I/O Read 1001 - reserved ... 1111 - reserved
16–19	WTT	WriteTType. Transaction type to run on PCI 0000 - reserved ... 0011 - reserved 0100 - Memory Write 0101 - reserved ... 0111 - reserved 1000 - I/O Write 1001 - reserved ... 1111 - reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound 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: Reserved 100001: Reserved 100010: Reserved ... 111111: Reserved The default POWAR register (0x010) has an OWS value of 011111.

16.3.1.2 PCI/X ATMU Inbound Registers

The inbound address translation and mapping unit controls the mapping of transactions from the external PCI address space to the local address space of the MPC8540. The inbound ATMU comprises four windows—a configuration window and three general translation windows.

Each window contains the following:

- A base address, which points to the beginning of the window in the external PCI address map. The base address of each window is also accessible by PCI configuration transactions as base address registers within the PCI configuration header, as shown in Figure 16-21]. The registers may be read or updated equivalently through the ATMU memory map or through PCI configuration transactions to the PCI configuration header.
- A translation address, which specifies the upper order bits of the transaction in the local address space
- A set of attributes including window size and internal transaction attributes

Each window must be aligned to the size of the window. If two general inbound ATMU windows overlap in the external PCI address space, the mappings of the lower number window are applied; however, it is illegal for an inbound window to overlap the PCSRBAR window.

Note that the PCSRBAR register in the PCI configuration header acts as a fourth inbound window that translates a 1-Mbyte region of PCI space to the local configuration space pointed to by CCSRBAR. PCSRBAR can be accessed by PCI configuration cycles or by accessing the PCI configuration header through the CFG_ADDR and CFG_DATA registers. See Section 16.3.2.1, “Accessing PCI/X Configuration Header Registers,” and Section 16.3.2.9, “PCI Base Address Registers.”

16.3.1.2.1 PCI/X Inbound Translation Address Registers (PITAR)

The PCI/X inbound translation address registers points to the beginning of the local address space for the inbound window. The new translated address is created by concatenating the transaction offset to this translation address.

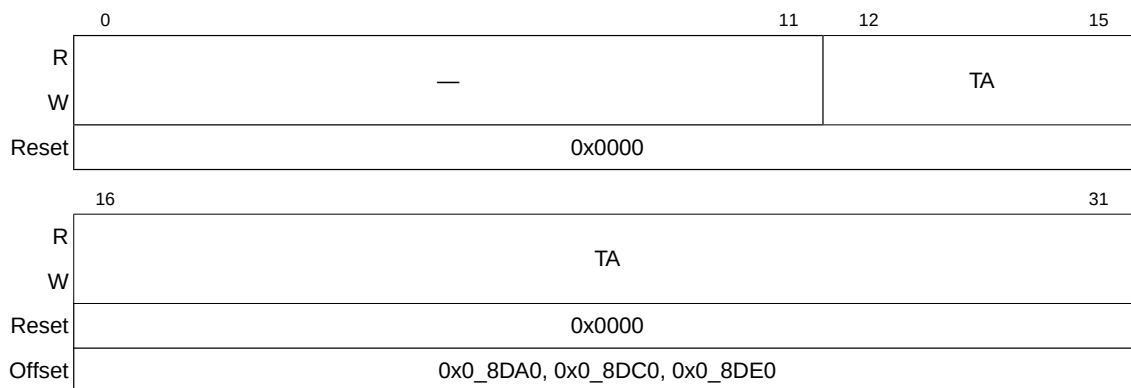


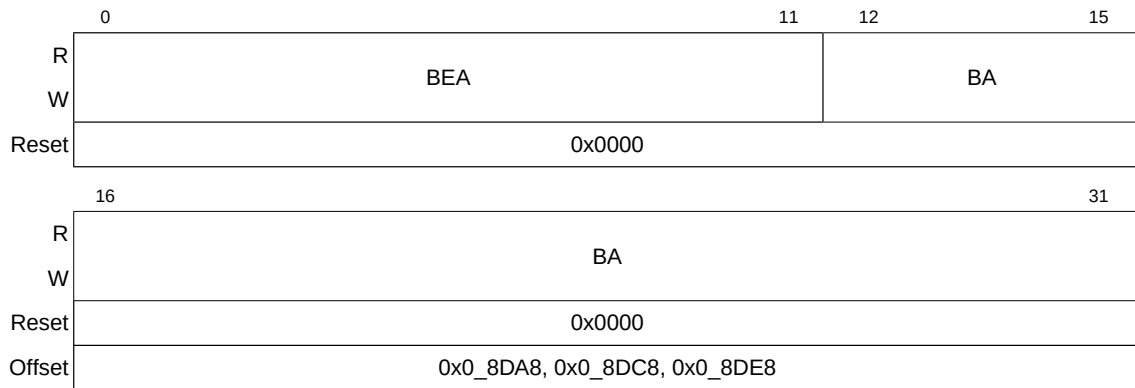
Figure 16-8. PCI/X Inbound Translation Address Register

Table 16-7. Bit Settings for PITAR

Bits	Name	Description
0–11	—	Reserved, should be cleared.
12–31	TA	Target address. Indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. TA corresponds to the high-order 20 bits of a 32-bit local address.

16.3.1.2.2 PCI/X Inbound Window Base Address Registers (PIWBAR)

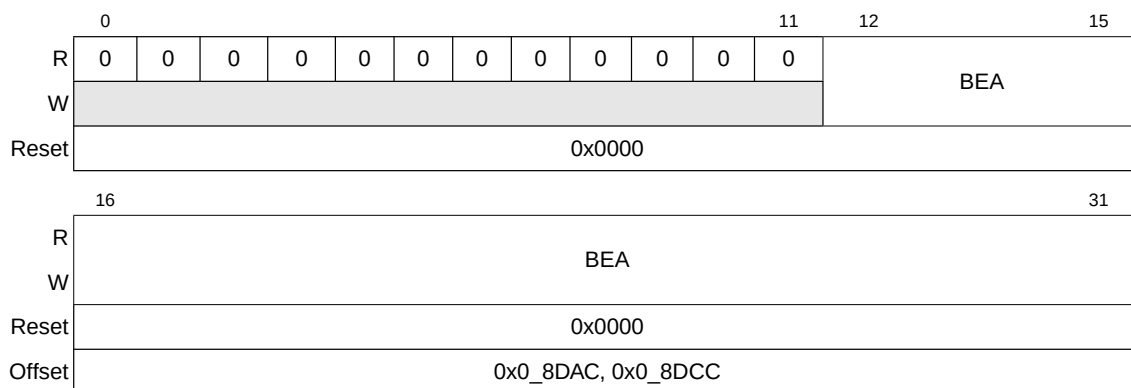
The PCI/X inbound window base address registers select the base address for the windows that are translated to the local address space of MPC8540. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the PCI/X interface does not assert DEVSEL.

**Figure 16-9. PCI/X Inbound Window Base Address Register****Table 16-8. Bit Settings for PIWBAR**

Bits	Name	Description
0–11	BEA	Base extended address. Corresponds to bits 43-32 of a 64-bit PCI base address.
12–31	BA	Base Address. Corresponds to bits 31-12 of a PCI base address.

16.3.1.2.3 PCI/X Inbound Window Base Extended Address Registers (PIWBEAR)

The PCI/X inbound window base extended address registers contain the most significant bits of a 64-bit base address. Note that inbound window 1 supports only a 32-bit base address and does not define an inbound window base extended address register.

**Figure 16-10. PCI/X Inbound Window Base Extended Address Register****Table 16-9. Bit Settings for PIWBEAR**

Bits	Name	Description
0–11	—	Reserved
12–31	BEA	Base extended address. Corresponds to bits 63:44 of a 64-bit PCI base address.

16.3.1.2.4 PCI/X Inbound Window Attributes Registers (PIWAR)

The PCI/X inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed.

**Figure 16-11. PCI/X Inbound Window Attributes Register**

Table 16-10. Bit Settings for PIWAR

Bits	Name	Description
0	EN	Enable. This bit enables this address translation
1	S_D	Byte swap_disable. Disables byte lane redirection on the internal data bus. Set this bit to access little endian structures in the target memory. By default, bytes lanes are swapped between the little-endian PCI bus and the internal data buses to access memory using a big-endian convention.
2	PF	Prefetchable. Indicates that the address space is prefetchable so that prefetching and streaming are attempted. 0 not prefetchable 1 prefetchable
3–7	—	Reserved
8–11	TGI	Target Interface. 0000 - 1011 = Reserved 1100 = RapidIO 1101 - 1110 = Reserved 1111 = Local Memory (DDR SDRAM, Local Bus, SRAM) Note: If this field is set to an I/O port rather than local memory space, attributes for the external I/O transaction are assigned in an outbound ATMU of that I/O controller.
12–15	RTT	Read transaction type. Transaction type to run if access is a read. The field description differs subject to the transaction being targeted to I/O interface or to local memory. Following are the transaction type settings for reads to the RapidIO interface: 0000- 0011 Reserved 0100 READ 0101- 1111 Reserved Following are the transaction type settings for reads to local memory: 0000- 0011 = Reserved 0100 = READ, don't snoop local processor 0101 = READ, snoop local processor 0110 = Reserved 0111 = READ, unlock L2 cache line 1000- 1111 = Reserved
16–19	WTT	Write transaction type. Transaction type to run if access is a write. The field description differs subject to the transaction being targeted to an I/O interface or to local memory. Following are the transaction type settings for writes to the RapidIO interface: 0000- 0011 Reserved 0100 WRITE 0101- 1111 Reserved Following are the transaction type settings for writes to local memory: 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 16-10. Bit Settings for PIWAR

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 4K bytes. 000000: Reserved ... 001011: 4K window size 001100: 8K window size ... 011111: 4G window size 100000: 8G window size 100001: 16B window size 100010: reserved ... 111111: reserved For configuration and run-time registers, the window size is fixed at 010011: 1M window size For register set 0, the window size is limited to 4G or smaller.

16.3.1.3 PCI/X Error Management Registers

The detected error bits are bit-reset 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 25 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 set the appropriate error bits in the error detection registers, but relevant information is captured only for the first error of a particular unit. The error bit is set regardless of the state of the enable bit. The enable bits are used to send or block the error reporting to the interrupt mechanism. The interrupt can be cleared by writing 0xFFFF_FFFF to the PCI/X error detect register.

A master-abort condition during a configuration cycle is not necessarily an error. In this case, if relevant, the master abort error enable can be disabled to prevent the reporting of master-aborts during outbound configuration cycles. Master-aborts during configuration reads return 0xFFFF.

16.3.1.3.1 PCI/X Error Detect Register

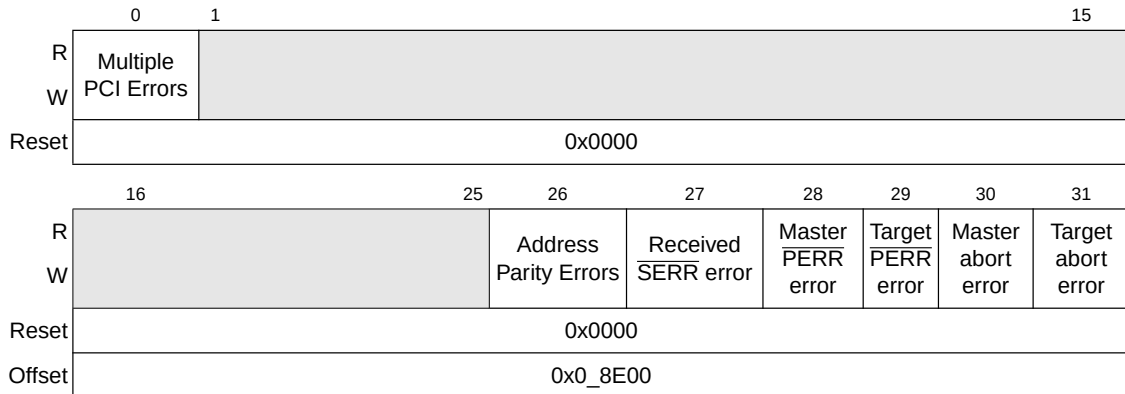


Figure 16-12. PCI/X Error Detect Register

Table 16-11. Bit Settings for PCI/X Error Detect Register

Bits	Name	Description
0	Multiple PCI Errors	0 multiple PCI errors of the same type were not detected (bit-reset) 1 multiple PCI errors of the same type were detected
1–25	—	Reserved (bit-reset)
26	Address parity ($\overline{\text{SERR}}$) error	Address parity error (bit-reset)
27	Received $\overline{\text{SERR}}$ error	Received $\overline{\text{SERR}}$ error (bit-reset)
28	Master $\overline{\text{PERR}}$ error	Master $\overline{\text{PERR}}$ error (bit-reset)
29	Target $\overline{\text{PERR}}$ error	Target $\overline{\text{PERR}}$ error (bit-reset)
30	Master abort error	Master abort error (bit-reset)
31	Target abort error	Target abort error (bit-reset)

16.3.1.3.2 PCI/X Error Capture Disable Register

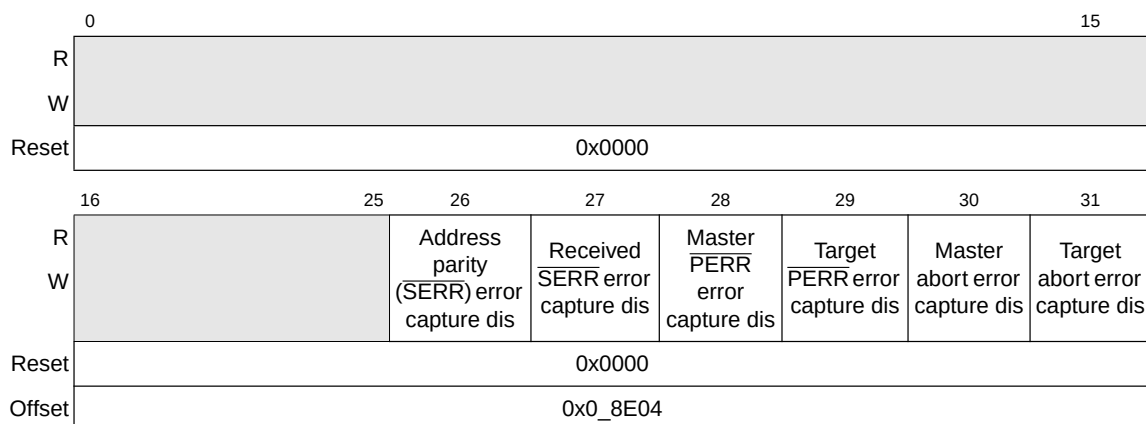


Figure 16-13. PCI/X Error Capture Disable Register

Table 16-12. Bit Settings for PCI/X Error Capture Disable Register

Bits	Name	Description
0–25	—	Reserved
26	Address parity ($\overline{\text{SERR}}$) error capture disable	Disable capture for address parity errors
27	Received $\overline{\text{SERR}}$ error capture disable	Disable capture for received $\overline{\text{SERR}}$ errors
28	Master $\overline{\text{PERR}}$ error capture disable	Disable capture for master $\overline{\text{PERR}}$ errors
29	Target $\overline{\text{PERR}}$ error capture disable	Disable capture for target $\overline{\text{PERR}}$ errors
30	Master abort error capture disable	Disable capture for master abort errors
31	Target abort error capture disable	Disable capture for target abort errors

16.3.1.3.3 PCI/X Error Enable Register

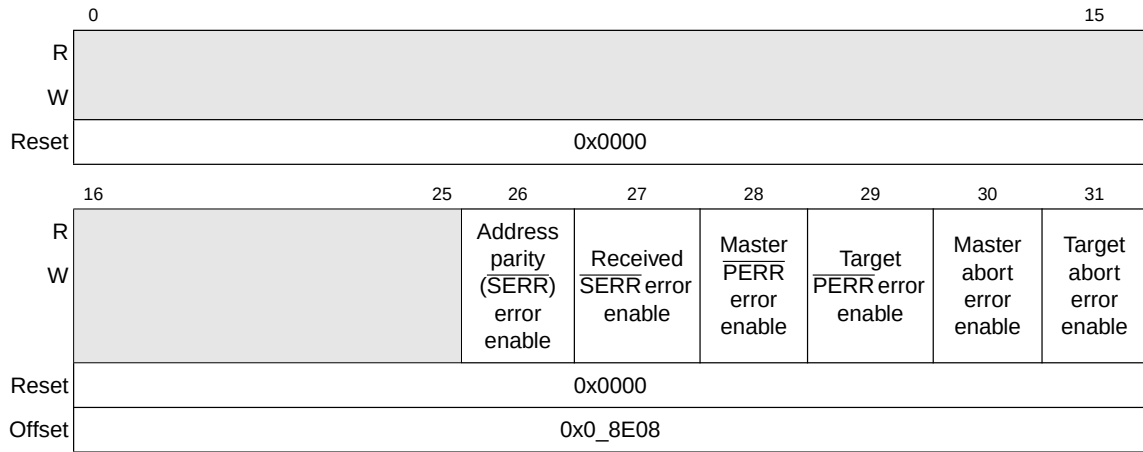


Figure 16-14. PCI/X Error Enable Register

Table 16-13. Bit Settings for PCI/X Error Enable Register

Bits	Name	Description
0–25	—	Reserved
26	Address parity (SERR) error enable	Enable reporting address parity errors
27	Received SERR error enable	Enable reporting received SERR errors
28	Master PERR error enable	Enable reporting master PERR errors
29	Target PERR error enable	Enable reporting target PERR errors
30	Master abort error enable	Enable reporting master abort errors
31	Target abort error enable	Enable reporting target abort errors

16.3.1.3.4 PCI/X Error Attributes Capture Register

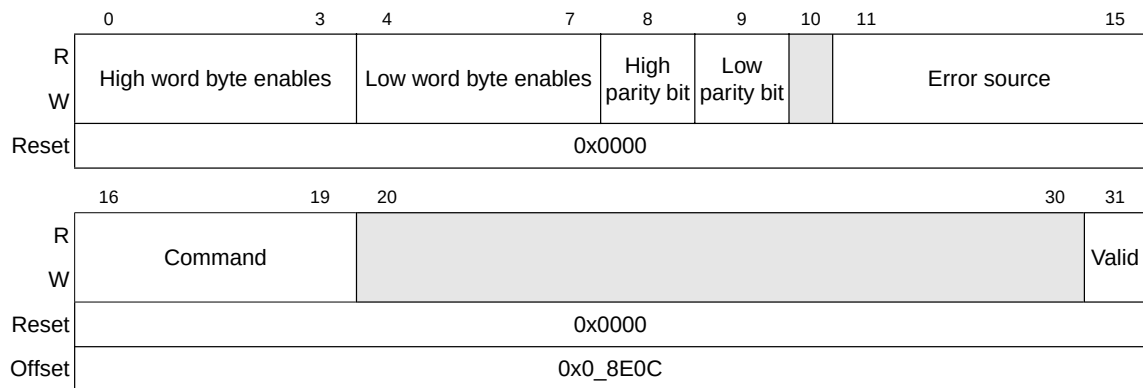


Figure 16-15. PCI/X Error Attributes Capture Register

Table 16-14. Bit Settings for PCI/X Error Attributes Capture Register

Bits	Name	Description
0–3	High word byte enables	PCI byte enables for most significant word of the doubleword
4–7	Low word byte enables	PCI byte enables for least significant word of the doubleword
8	High parity bit	Parity bit for most significant PCI bus data word (only valid for 64-bit PCI bus)
9	Low parity bit	Parity bit for least significant PCI bus data word
10	—	Reserved
11–15	Error source	The source of the PCI transaction 00000 - PCI/PCI-X interface 00001 - reserved 00010 - reserved 00011 - reserved 00100 - Local Bus interface 00101 - reserved 00110 - reserved 00111 - reserved 01000 - reserved 01001 - reserved 01010 - reserved 01011 - reserved 01100 - RapidIO interface 01101 - reserved 01110 - reserved 01111 - reserved 10000 - e500 core (instruction) 10001 - e500 core (data) 10010 - reserved 10011 - reserved 10100 - reserved 10101 - DMA 10110 - RDC 10111 - SAP 11000 - Ethernet interface 1 11001 - Ethernet interface 2 11010 - Ethernet interface 3 11011 - reserved 11100 - RapidIO Message Units 11101 - RapidIO Doorbell Units 11110 - RapidIO Port-write Units 11111 - reserved
16–19	Command	PCI command
20–30	—	Reserved
31	Valid info	The PCI bus capture registers contain valid information

16.3.1.3.5 PCI/X Error Address Capture Register

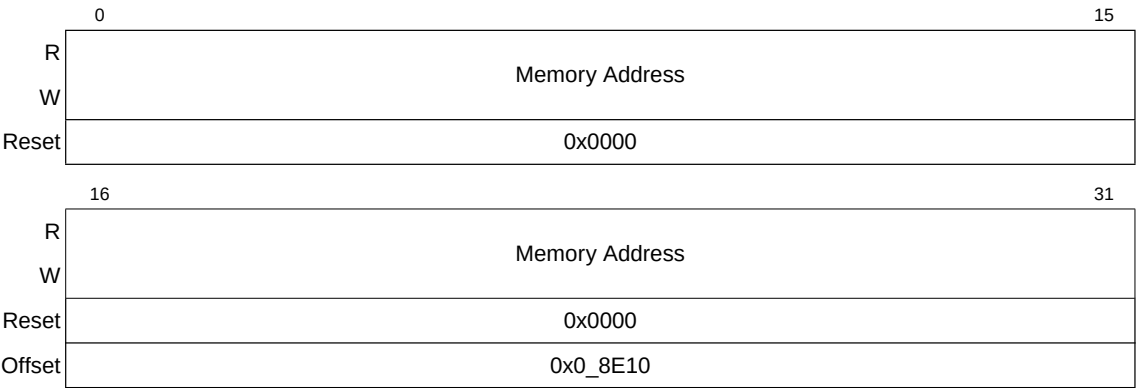


Figure 16-16. PCI/X Error Address Capture Register

Table 16-15. Bit Settings for PCI/X Error Address Capture Register

Bits	Name	Description
0–31	Memory address	Memory transaction address

16.3.1.3.6 PCI/X Error Extended Address Capture Register

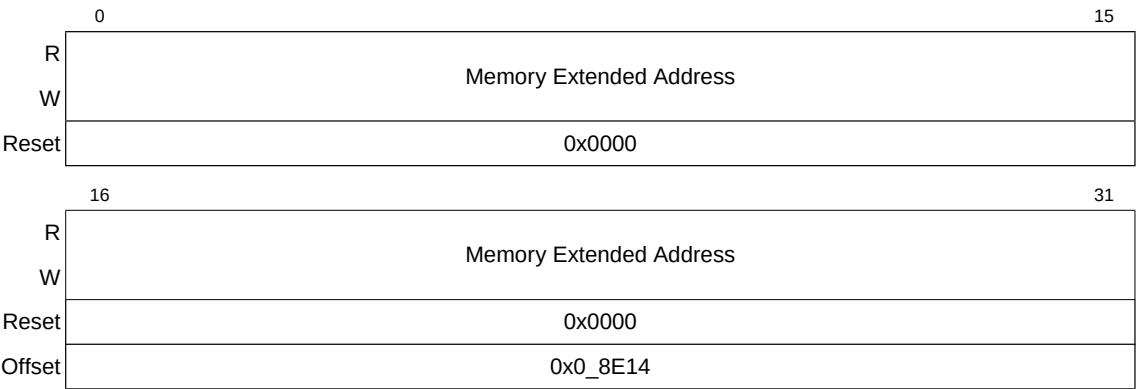


Figure 16-17. PCI/X Error Extended Address Capture Register

Table 16-16. Bit Settings for PCI/X Error Extended Address Capture Register

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

16.3.1.3.7 PCI/X Error Data Low Capture Register

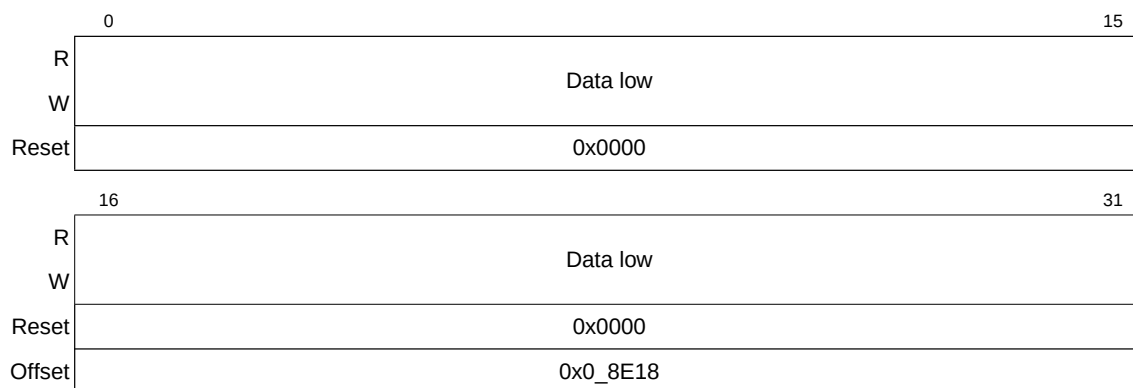


Figure 16-18. PCI/X Error Data Low Capture Register

Table 16-17. Bit Settings for PCI/X Error Data Low Capture Register

Bits	Name	Description
0–31	Data low	Least significant PCI bus data word

16.3.1.3.8 PCI/X Error Data High Capture Register

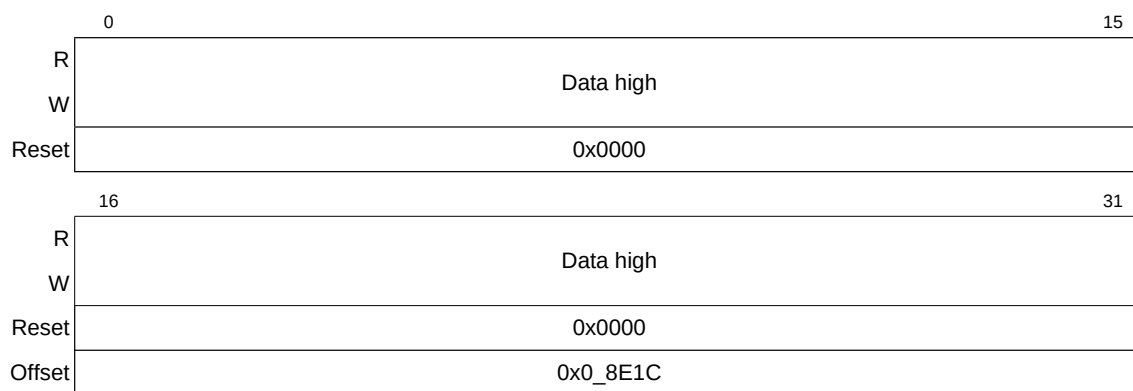


Figure 16-19. PCI/X Error Data High Capture Register

Table 16-18. Bit Settings for PCI/X Error Data High Capture Register

Bits	Name	Description
0–31	Data high	Most significant PCI bus data word (only valid with 64-bit PCI bus)

16.3.1.4 PCI/X Interrupt Acknowledge Register

An external PCI/X interrupt acknowledge transaction is generated by reading the INT_ACK register at offset 0x0_8008. INT_ACK is read-only and a write to that address results in nothing.

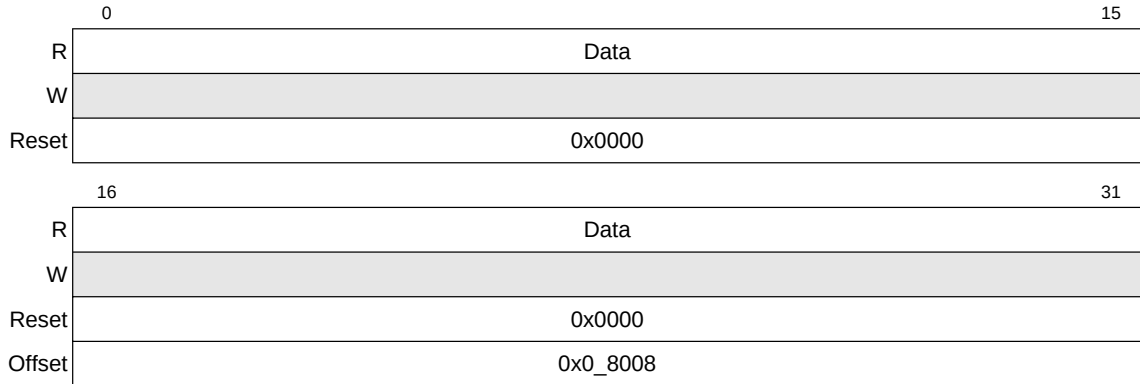


Figure 16-20. PCI Interrupt Acknowledge Register

16.3.2 PCI/PCI-X Configuration Header

Figure 16-21 lists the common PCI/PCI-X configuration header as implemented by the MPC8540.

☐ Reserved				Address Offset (Hex)
Device ID (0x0003)		Vendor ID (0x1057)		00
PCI Bus Status		PCI Bus Command		04
Bus Base Class Code	Subclass Code	Bus Programming Interface	Revision ID	08
BIST Control	Header Type	Bus Latency Timer	Bus Cache Line Size	0C
PCI Configuration and Status Register Base Address Register (PCSRBAR)				10
32-bit Memory Base Address Register				14
64-bit Low Memory Base Address Register				18
64-bit High Memory Base Address Register				1C
64-bit Low Memory Base Address Register				20
64-bit High Memory Base Address Register				24
Subsystem ID				2C
Subsystem Vendor ID				2C
PCI Bus Capability Pointer				34
////////				38
PCI Bus MAX LAT	PCI Bus MAX GNT	PCI Bus Interrupt Pin	PCI Bus Interrupt Line	3C
////////				40
PCI Bus Arbiter Configuration		PCI Bus Function		44

Figure 16-21. MPC8540 Common PCI and PCI-X Configuration Header

Figure 16-22 lists the additional PCI-X configuration registers.

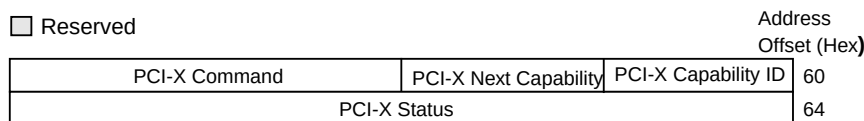


Figure 16-22. PCI-X Additional Configuration Registers

16.3.2.1 Accessing PCI/X Configuration Header Registers

The PCI/X configuration registers, shown in Figure 16-21 and Figure 16-22, are accessed via an indirect method utilizing a pair of 32-bit memory-mapped access registers, CFG_ADDR at offset 0x0_8000 and CFG_DATA at offset 0x0_8004. The CFG_ADDR register, shown in Figure 16-23, is described in Table 16-19. Note that the PCI/PCI-X configuration header is a little-endian data structure. The CFG_DATA register is a window into this data structure; therefore byte addressing within the CFG_DATA register uses little-endian convention. PowerPC processor accesses to the CFG_DATA register should use the load/store with byte-reversed instructions. External PCI masters that use the local address map to access configuration space do not need to reverse bytes since proper byte swapping from the little-endian PCI bus is done internally.

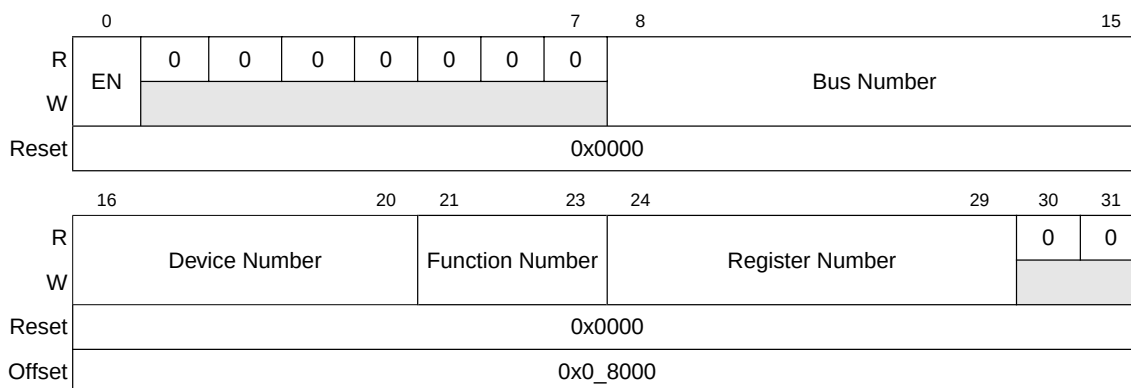


Figure 16-23. PCI/PCI-X Configuration Header Register

Table 16-19. Bit Settings for CFG_ADDR

Bits	Name	Description
0	Enable	Allow a PCI configuration access when CFG_DATA is accessed
1–7	—	Reserved
8–15	Bus Number	PCI bus number to access
16–20	Device Number	Device number to access on specified bus
21–23	Function Number	Function to access within specified device
24–29	Register Number	32-bit register to access within specified device
30–31	—	Reserved, hardwired to logic 00

Bus number 0xb00 and device number 0b0_0000 are used to configure the internal PCI/X configuration registers of the MPC8540 itself.

The CFG_DATA register, shown in Figure 16-23, is a 32-bit, data-only register and is described in Table 16-20.

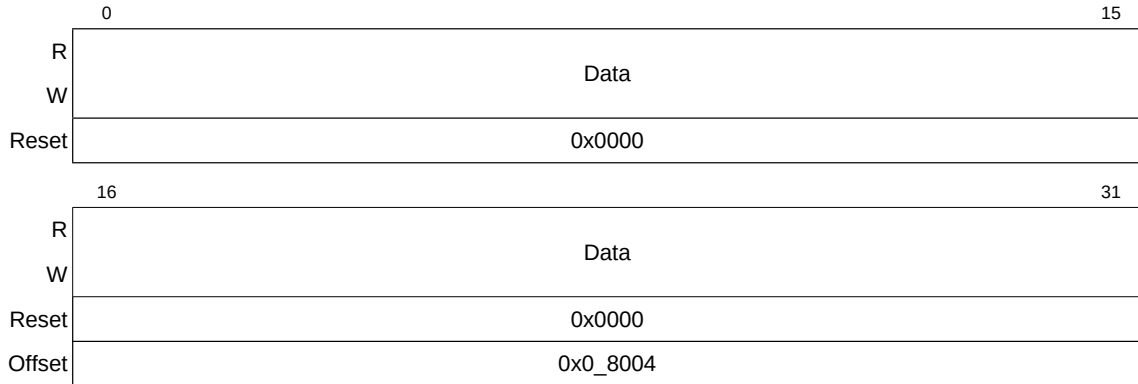


Figure 16-24. PCI CFG_DATA Register

Table 16-20. Bit Settings for CFG_DATA

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI configuration cycle if the CFG_ADDR enable bit is set. If the enable bit is not set, a PCI I/O transaction is generated.

16.3.2.2 PCI Agent Mode Configuration Register Access

When this device is configured as an agent device it responds to a remote host-generated PCI configuration cycle. This is indicated by decoding the configuration command along with PCI/X's IDSEL being asserted. A remote host can access the 256-byte PCI/X configuration area. The memory-mapped configuration registers and the memory-mapped runtime registers that are described in this chapter are accessed through read and write transactions using one of the PCI/X base address registers.

16.3.2.3 PCI Bus Command Register—Offset 0x04

Table 16-21. Bit Settings for INT_ACK

Bits	Name	Reset Value	Description
0–31	Data	0	A read to this register generates a PCI interrupt acknowledge cycle.

The 2-byte PCI bus command register provides control over the ability to generate and respond to PCI cycles. Table 16-22 describes the bits of the PCI bus command register.

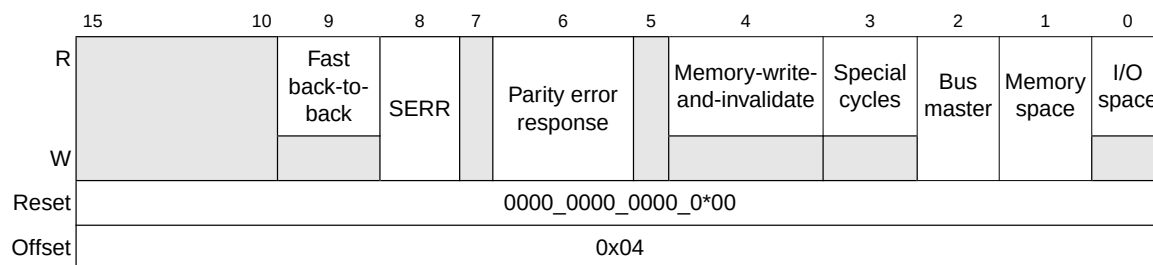


Figure 16-25. PCI Bus Command Register

Table 16-22. Bit Settings for PCI Bus Command Register—0x04

Bits	Name	Reset Value	Description
15–10	—	000000	Reserved
9	Fast back-to-back	0	This bit is hard-wired to 0, indicating that this PCI controller (as a master) does not run fast back-to-back transactions.
8	SERR	0	Controls the $\overline{\text{PCI\_SERR}}$ driver of this PCI controller. This bit (and bit 6) must be set to report address parity errors. 0 Disables the $\overline{\text{PCI\_SERR}}$ driver 1 Enables the $\overline{\text{PCI\_SERR}}$ driver
7	—	0	Reserved
6	Parity error response	0	This bit controls whether this PCI controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Action is taken on a parity error.
5	—	0	Reserved
4	Memory-write-and-invalidate	0	This bit is hard-wired to 0, indicating that this PCI controller, acting as a master, can not generate the memory-write-and-invalidate command.
3	Special-cycles	0	This bit is hard-wired to 0, indicating that this PCI controller (as a target) ignores all special-cycle commands.
2	Bus master	1	This bit indicates whether this PCI controller is configured as a master. This indicates the setting of the host/agent configuration input ($\overline{\text{LWE2}}$) at power-on reset. 0 Disables the ability to generate PCI accesses 1 Enables this PCI controller to behave as a PCI bus master (Host)
1	Memory space	0	This bit controls whether this PCI controller (as a target) responds to memory accesses. This indicates the setting of the host/agent configuration input ($\overline{\text{LWE2}}$) at power-on reset. 0 This PCI controller does not respond to PCI memory space accesses. 1 This PCI controller responds to PCI memory space accesses. (Agent)
0	I/O space	0	This bit is hard-wired to 0, indicating that this PCI controller (as a target) does not respond to PCI I/O space accesses.

16.3.2.4 PCI Bus Status Register—Offset 0x06

The 2-byte PCI bus status register is used to record status information for PCI bus bus-related events. The definition of each bit is given in Table 16-23. Only 2-byte accesses to address offset 0x06 are allowed.

Reads to this register behave normally. Writes are slightly different in that bits can be cleared, but not set. 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 14 and not affect any other bits in the register, write the value 0b0100_0000_0000_0000 to the register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	0
R	Detected parity error	Signaled system error	Received master abort	Received target abort	Signaled target abort	DEVSEL timing		Data parity detected	Fast back-to-back capable		66-MHz capable	Capabilities list		
W														
Reset	0000_0000_1010_0000													
Offset	0x06													

Figure 16-26. PCI Bus Status Register

Table 16-23. Bit Settings for PCI Bus Status Register—0x06

Bits	Name	Description
15	Detected parity error	This bit is set whenever this PCI controller detects a PCI parity error, even if parity error handling is disabled (as controlled by bit 6 in the PCI bus command register).
14	Signaled system error	This bit is set whenever this PCI controller asserts <code>PCI_SERR</code> .
13	Received master-abort	This bit is set whenever this PCI controller, acting as the PCI master, terminates a transaction (except for a special-cycle) using master-abort.
12	Received target-abort	This bit is set whenever a PCI transaction initiated by this PCI controller (excluding a special-cycle) is terminated by a target-abort.
11	Signaled target-abort	This bit is set whenever this PCI controller, acting as the PCI target, issues a target-abort to a PCI master.
10–9	DEVSEL timing	These bits are hard-wired to 0b00, indicating that this PCI controller uses fast device select timing.
8	Data parity detected	This bit is set upon detecting a data parity error. Three conditions must be met for this bit to be set: This PCI controller detected a parity error. This PCI controller was acting as the bus master for the operation in which the error occurred. Bit 6 in the PCI bus command register was set.
7	Fast back-to-back capable	This bit is hard-wired to 1, indicating that this PCI controller (as a target) is capable of accepting fast back-to-back transactions.
6	—	Reserved

Table 16-23. Bit Settings for PCI Bus Status Register—0x06

Bits	Name	Description
5	66-MHz capable	This bit is read-only and indicates that this PCI controller is capable of 66 MHz PCI bus operation.
4	Capabilities List	hard-wired to 0
3–0	—	Reserved

16.3.2.5 PCI Bus Programming Interface Register—Offset 0x09

Table 16-24 describes the PCI Bus programming interface register (PIR).

	7	0
R	Programming Interface	
W		
Reset	0000_0000	
Offset	0x09	

Figure 16-27. PCI Bus Programming Interface Register**Table 16-24. Bit Settings for PCI Bus Programming Interface Register—0x09**

Bits	Name	Description
7–0	Programming Interface	0x00 When the PCI controller is configured as host bridge 0x01 When the PCI controller is configured as an agent device to indicate the programming model

16.3.2.6 PCI Bus Base Class Code Register—Offset 0x0B

Table 16-25 describes the PCI bus base code register (PBCCR).

	7	0
R	Base Class Code	
W		
Reset	0000_0000	
Offset	0x0B	

Figure 16-28. PCI Bus Base Class Code Register

Table 16-25. Bit Settings for PCI Bus Base Class Code Register—0x0B

Bit	Name	Reset Value	Description
7–0	Base Class Code	0x0n	0x06 When this PCI controller is configured as a host bridge to indicate host bridge. 0x0E When this PCI controller is configured as a target device to indicate the device is an agent.

16.3.2.7 PCI Bus Cache Line Size Register—Offset 0x0C

Table 16-26 describes the PCI bus cache line size register (PCLSR).

	7	0
R	Cache Line Size	
W		
Reset	0000_1000	
Offset	0x0C	

Figure 16-29. PCI Bus Cache Line Size Register**Table 16-26. Bit Settings for PCI Bus Cache Line Size Register—0x0C**

Bit	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). This register is read-write, however for PCI operation an attempt to program this register to any value other than 0x8 results in clearing it; for PCI-X operation, this register is hardwired to 0x8.

16.3.2.8 PCI Bus Latency Timer Register—0x0D

Table 16-27 describes the PCI latency timer register (PLTR).

	7	3	2	0
R	Latency Timer			Latency Timer
W				
Reset	0000_0000			
Offset	0x0D			

Figure 16-30. PCI Bus Latency Timer Register

Table 16-27. Bit Settings for PCI Bus Latency Timer Register—0x0D

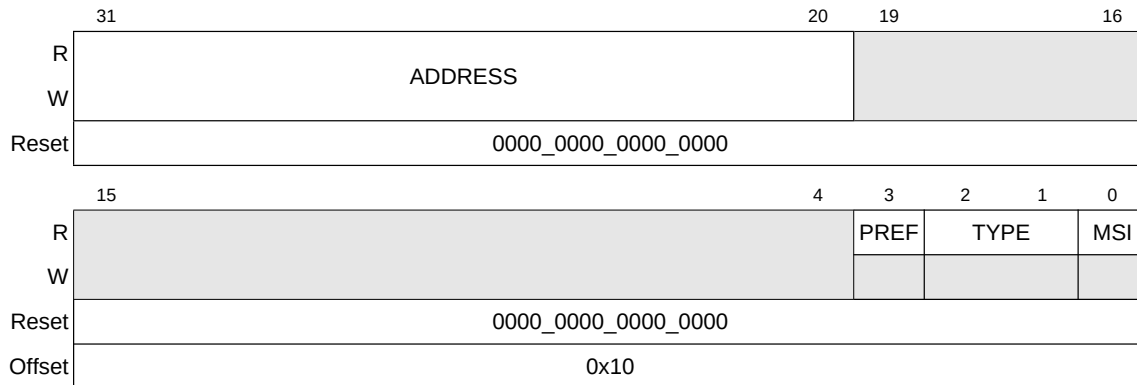
Bits	Name	Description
7–3	Latency Timer	The maximum number of PCI clocks that the device, when mastering a transaction, holds the bus after PCI bus grant has been negated. The value is in PCI clocks. The PCI 2.2 specification gives rules by which the PCI bus interface unit completes transactions when the timer has expired.
2–0	Latency Timer	Read only bits. The minimum latency timer value when set is 8 PCI clocks.

16.3.2.9 PCI Base Address Registers

The PCI base address registers point to the beginnings of each of the address ranges to which the device responds by asserting DEVSEL_b. The BAR at offset 0x10 is a fixed 1-Mbyte window that is automatically translated to the local configuration, control, and status registers address space.

The other base address registers are aliases (with differing format) of the PCI inbound ATMU windows, see Section 16.3.1.2, “PCI/X ATMU Inbound Registers.” The 32-bit base address register at offset 0x14 corresponds in inbound ATMU window 1; the 64-bit base address registers at offsets 0x18 and 0x20 correspond to inbound ATMU windows 2 and 3. If one of these registers is written, the corresponding ATMU register is also updated; if a PCI inbound ATMU register is written, the corresponding BAR is also updated.

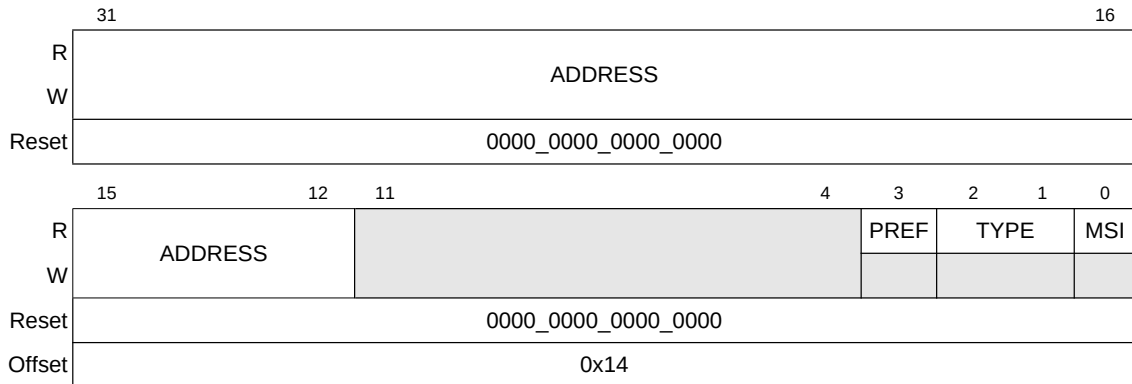
Note that PCSRBAR cannot be updated through the inbound ATMU registers.

**Figure 16-31. PCI Configuration and Status Register Base Address Register (PCSRBAR)****Table 16-28. Bit Settings for PCSRBAR**

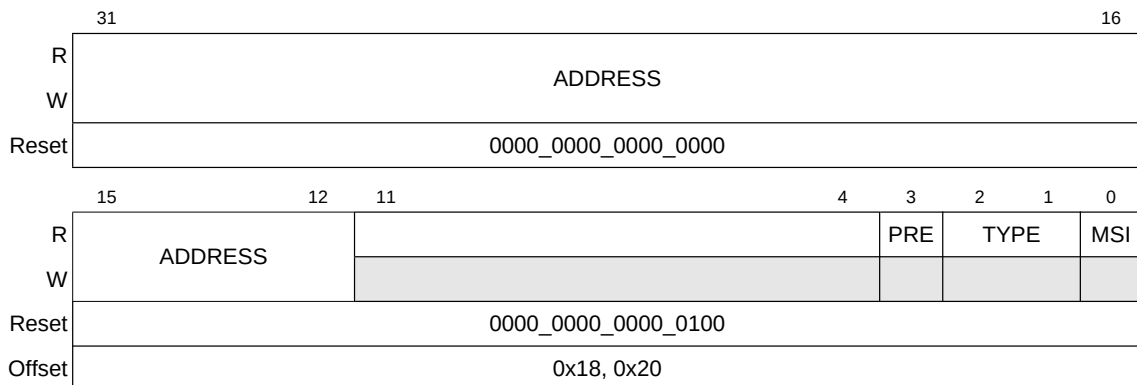
Bits	Name	Description
31–20	ADDRESS	Indicates the base address that the inbound configuration/run-time window resides at. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable

Table 16-28. Bit Settings for PCSRBAR

Bits	Name	Description
2–1	TYPE	Type. Locate anywhere in 32-bit address space.
0	MSI	Memory space indicator

**Figure 16-32. 32-Bit Memory Base Address Register****Table 16-29. Bit Settings for 32-Bit Memory Base Address Register**

Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows. If no access to local memory is to be permitted by external masters then all bits are programmed.
11–4	—	Reserved. The device allows a 4kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type: 00 - Locate anywhere in 32-bit address space.
0	MSI	Memory Space Indicator.

**Figure 16-33. 64-bit Low Memory Base Address Register**

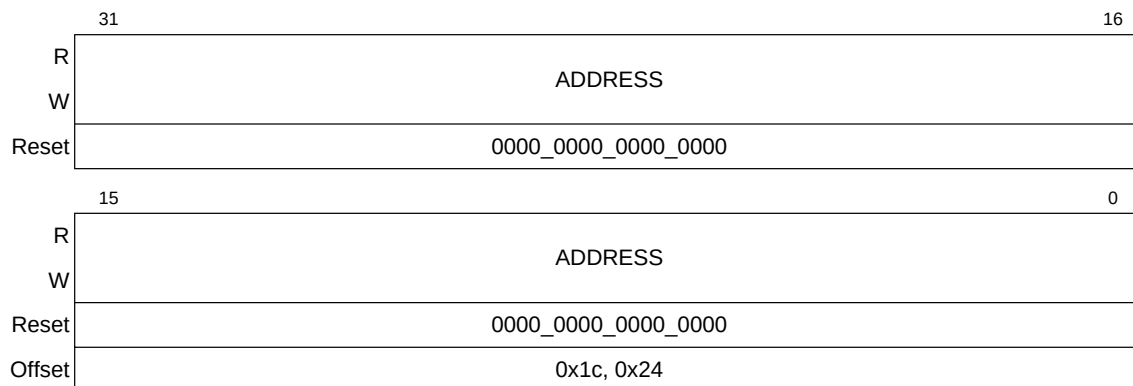


Figure 16-34. 64-bit High Memory Base Address Register

Table 16-30. Bit Settings for 64-Bit Low Memory Base Address Register

Bits	Name	Reset Value	Description
31–12	ADDRESS	0x0000 0	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows. If no access to local memory is to be permitted by external masters then all bits are programmed.
11–4	—	0x00	The device allows a 4kbyte window minimum.
3	PREF	0	Prefetchable
2–1	TYPE	10	Type: 10 - Locate anywhere in 64-bit address space.
0	MSI	0	Memory space indicator

Table 16-31. Bit Setting for 64-Bit High Memory Base Address Register

Bits	Name	Reset Value	Description
31–0	ADDRESS	0x0000_0000	Indicates the base address that the inbound memory window resides at. The number of upper bits that the device allows to be writable is selected through the inbound translation windows. If no access to local memory is to be permitted by external masters then all bits are programmed.

16.3.2.10 PCI Bus Capabilities Pointer Register

The PCI bus capabilities pointer identifies additional functionality supported by the device. This field is set to point to the PCI-X configuration registers. The definition of each bit is given in Table 16-32.

	7	0
R	Capabilities Pointer	
W		
Reset	0110_0000	
Offset	0x34	

Figure 16-35. PCI Bus Capabilities Pointer Register**Table 16-32. Bit Settings for PCI Bus Capabilities Pointer Register—0x34**

Bits	Name	Description
7–0	Capabilities Pointer	Specifies the byte offset in the configuration space containing the first item in the capabilities list. The default value points to the PCI-X configuration registers.

16.3.2.11 PCI Bus Interrupt Line Register

	7	0
R	Interrupt Line	
W		
Reset	0000_0000	
Offset	0x3C	

Figure 16-36. PCI Bus Interrupt Line Register**Table 16-33. Bit Settings for PCI Bus Interrupt Line Register—0x3C**

Bits	Name	Description
7–0	Interrupt Line	Reserved

16.3.2.12 PCI Bus Interrupt Pin Register

The MPC8540 does not generate specific PCI_INTA, PCI_INTB, etc. outputs. Nor does this device respond to INTACK or special cycle commands on the PCI/PCI-X interfaces. The MPC8540 does have 12 general purpose interrupt request lines (IRQ[0:11]) and an interrupt output, $\overline{\text{IRQ\_OUT}}$ (active low, level sensitive), to which all external and most internal interrupt sources (including PCI/PCI-X) can be routed.

	7	0
R	Interrupt Pin	
W		
Reset	0000_0000	
Offset	0x3D	

Figure 16-37. PCI Bus Interrupt Pin Register

Table 16-34. Bit Settings for PCI Bus Interrupt Pin Register—0x3D

Bits	Name	Description
7–0	Interrupt Pin	No PCI_INT pin selected

16.3.2.13 PCI Bus Maximum Grant (MAX GNT) Register

	7	0
R	MAXGNT	
W		
Reset	0000_0000	
Offset	0x3E	

Figure 16-38. PCI Bus Maximum Grant Register

Table 16-35. Bit Settings for PCI Bus Maximum Grant Register—0x3E

Bits	Name	Description
7–0	MAXGNT	Specifies the length of the device's burst period (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers)

16.3.2.14 PCI Bus Maximum Latency (MAX LAT) Register

	7	0
R	MAXLAT	
W		
Reset	0000_0000	
Offset	0x3F	

Figure 16-39. PCI Bus Maximum Latency Register

Table 16-36. Bit Settings for PCI Bus Maximum Latency Register—0x3F

Bits	Name	Description
7–0	MAXLAT	Specifies how often the device needs to gain access to the PCI bus (0x00 indicates that this PCI controller has no major requirements for the settings of latency timers)

16.3.2.15 PCI Bus Function Register (PBFR)

The 2-byte PCI bus function register is used to determine different features how the PCI interface in bus 0 is configured. This register is at PCI configuration space at offset 0x44.

	15								6		5	4		3		2		1		0
R	0	0	0	0	0	0	0	0	0	0	ACL	0	P64	0	0	0	0	PAH		
W																				
Reset	0000_0000_00*0_*00*																			
Offset	0x44																			

Figure 16-40. PCI Bus Function Register**Table 16-37. Bit Settings for PCI Bus Function Register—0x44**

Bits	Name	Description
15–6	—	Reserved
5	ACL	Agent_Cfg_Lock. Indicates to an external host whether the local processor is doing internal configuration and must be explicitly set and cleared by the local processor during this time. ACL is set during reset if the CPU_BOOT input selects the CPU as the configuration owner. This bit is only meaningful in agent mode. 0 PCI interface allows incoming PCI configuration cycles. 1 PCI interface retries all incoming PCI configuration cycles.
4	—	Reserved
3	P64	PCI_64. PCI 64-bit configuration. Read only. This bit indicates the value of the PCI 64-bit configuration signal at power-on reset. 0 64-bit interface functions as a 32-bit interface. 1 64-bit interface functions as a 64-bit interface.
2–1	—	Reserved
0	PAH	PCI_AH. PCI agent/host. Read only. PAH indicates the value of the PCI host/agent configuration signal at power-on reset. 0 PCI interface is in host mode 1 PCI interface is in agent mode

16.3.2.16 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

	15	14	13	12	11	10	9	8	7	6		1	0
R	PAD	PM	0	PBMD	0	0	0	0	0				
W													
Reset	0000_0000_0000_0000												
Offset	0x46												

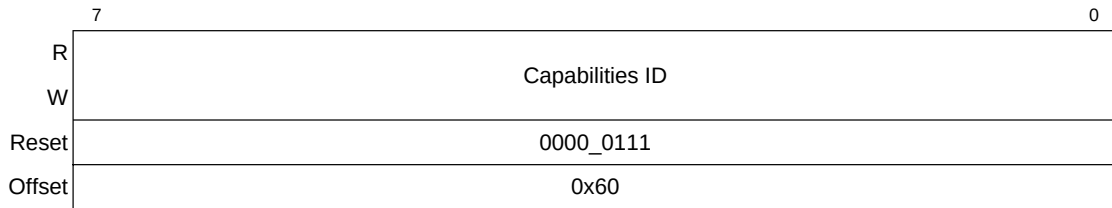
Figure 16-41. PCI Bus Arbiter Configuration Register

Table 16-38. Bit Settings for PCI Bus Arbiter Configuration Register—0x46

Bits	Name	Description
15	PAD	PCI_Arb_Dis. Determines if the device is the PCI arbiter on the PCI bus or not. The reset state of this bit is determined by the inverse of the PCI_GNT2 configuration input signal when reset is released. 0 Device is the PCI arbiter. 1 Device is not the PCI arbiter. Device presents its request on $\overline{\text{PCI\_REQ0}}$ to the external arbiter and receives its grant on $\overline{\text{PCI\_GNT0}}$.
14	PM	Parking mode. controls which device receives the bus grant when there are no outstanding bus requests and the bus is idle. 0 The bus is parked on the last device to use the bus. 1 The bus is parked on the device.
13	—	Reserved
12	PBMD	PCI broken master disable. This bit determines if the device ignores the bus requests of an initiator that requests the bus for an excessive period without using the bus. 0 An initiator that requests the bus and receives the grant must begin using the bus within 16 PCI clock periods after the bus becomes idle or else its request is subsequently ignored. 1 No requests are ignored.
11–7	—	Reserved
6–2	PBMP	PCI bus master priorities. These bits determine the arbitration priority given to the different masters on the PCI bus. Bit 6 corresponds to the priority of the master sourcing $\overline{\text{PCI\_REQ0}}$, and bit 2 corresponds to the priority of the master sourcing $\overline{\text{PCI\_REQ4}}$. 0 Master <i>n</i> is low priority. 1 Master <i>n</i> is high priority.
1	—	Reserved
0	DP	Device priority. This bit determines the device's arbitration priority. 0 Device is low priority. 1 Device is high priority.

16.3.2.17 PCI-X Next Capabilities ID Register—0x60

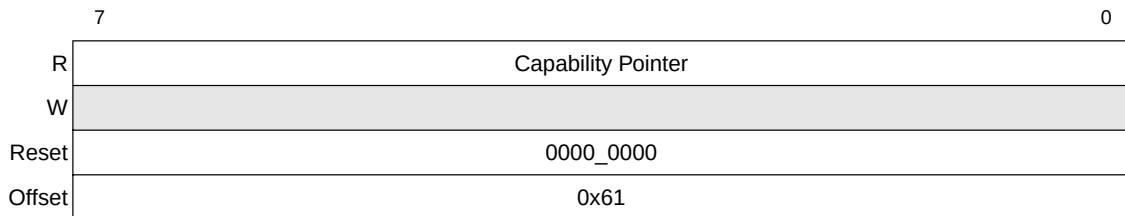
This register is an additional standard register specified by the *PCI-X Addendum to the PCI Local Bus Specification*.

**Figure 16-42. PCI-X Next Capabilities ID Register****Table 16-39. Bit Settings for PCI-X Next Capabilities ID Register—0x60**

Bits	Name	Description
7–0	Capabilities ID	Read-Only. Specifies the PCI-X Capability ID

16.3.2.18 PCI-X Capability Pointer Register—0x61

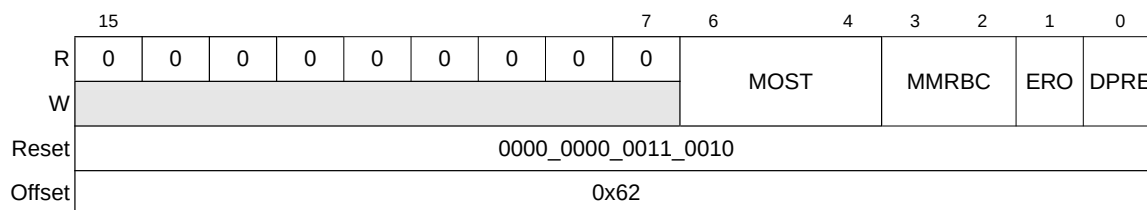
The PCI-X next capabilities pointer register is an additional standard register specified by the *PCI-X Addendum to the PCI Local Bus Specification*. This register identifies additional functionality supported by the device. The definition of each bit is given in Table 16-40.

**Figure 16-43. PCI-X Capability Pointer Register****Table 16-40. Bit Settings for PCI-X Capability Pointer Register—0x61**

Bits	Name	Description
7–0	Capability Pointer	Specifies the byte offset in the configuration space containing the first item in the capabilities list.

16.3.2.19 PCI-X Command Register—0x62

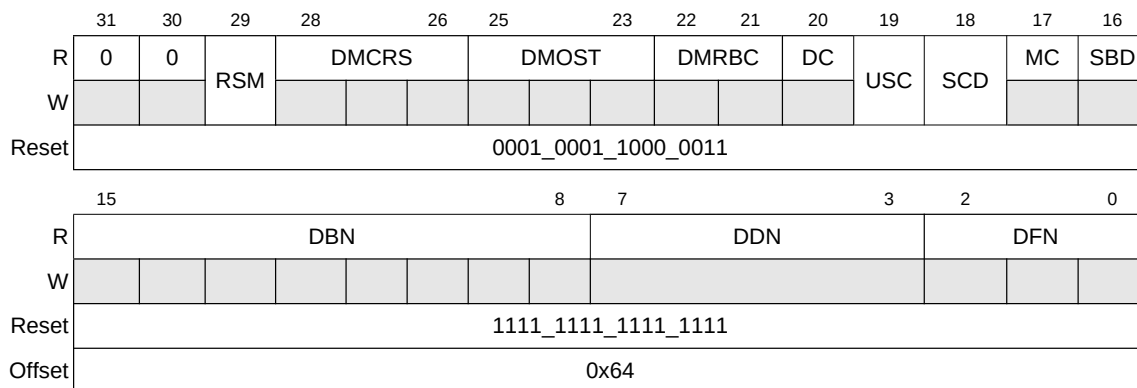
The 2-byte PCI-X command register is an additional standard register specified by the *PCI-X Addendum to the PCI Local Bus Specification*. This register provides control over the ability to generate and respond to PCI-X cycles. Table 16-41 describes the bits of the PCI-X command register.

**Figure 16-44. PCI-X Command Register****Table 16-41. Bit Settings for PCI-X Command Register—0x62**

Bits	Name	Description
15–7	—	Reserved
6–4	MOST	Maximum outstanding split transactions (4 supported)
3–2	MMRBC	Maximum memory read byte count (128 supported)
1	ERO	Enable relax ordering (supported)
0	DPRE	Data parity error recovery enable (supported)

16.3.2.20 PCI-X Status Register—0x64

The 4-byte PCI-X status register is an additional standard register specified by the *PCI-X Addendum to the PCI Local Bus Specification*. This register is used to record additional status information for PCI-X bus-related events. The definition of each bit is given in Table 16-42. Only 4-byte accesses to address offset 0x64 are allowed.

**Figure 16-45. PCI-X Status Register****Table 16-42. Bit Settings for PCI-X Status Register—0x64**

Bits	Name	Description
31–30	—	Reserved
29	RSM	Receive split completion error message (Write 1 to clear)

Table 16-42. Bit Settings for PCI-X Status Register—0x64 (continued)

Bits	Name	Description
28–26	DMCRS	Designed maximum cumulative read size (16 KBytes)
25–23	DMOST	Designed maximum outstanding split transactions (4 outstanding)
22–21	DMRBC	Designed maximum memory read byte count (128 Bytes)
20	DC	Device complexity (simple device, can retry writes)
19	USC	Unexpected split completion (Write 1 to clear)
18	SCD	Split completion discarded (Write 1 to clear)
17	MC	133-MHz capability
16	SBD	64-bit device
15-8	DBN	Bus number (used for diagnostic purposes)
7–3	DDN	Device number (used for diagnostic purposes)
2–0	DFN	Function number (used for diagnostic purposes)

16.4 Functional Description

This section describes the functionality of the PCI/PCI-X interface.

16.4.1 PCI/PCI-X Bus Arbitration

PCI bus arbitration is access-based. Bus masters must arbitrate for each access performed on the bus. The PCI bus uses a central arbitration scheme where each master has its own unique request ($\overline{\text{REQ}}$) output and grant ($\overline{\text{GNT}}$) input signal. A simple request/grant handshake is used to gain access to the bus. Arbitration for the bus occurs during the previous access so that no PCI bus cycles are consumed due to arbitration (except when the bus is idle).

The MPC8540 provides bus arbitration logic for its master interface and up to five other external PCI bus masters. The on-chip PCI arbiter is independent of host or agent mode. The on-chip PCI arbiter functions in both host and agent modes, or it can be disabled to allow for an external PCI arbiter.

A configuration signal ($\overline{\text{PCI\_GNT2}}$) sampled at the negation of the reset signal ($\overline{\text{HRESET}}$) determines if the on-chip PCI arbiter is enabled (high) or disabled (low). The on-chip PCI arbiter can also be enabled or disabled by programming bit 15 of the PCI bus arbitration control register (PBACR[PAD]). Note that the sense of PBACR[PAD] corresponds to the inverse of the configuration signal (that is, when PAD = 0b0 the arbiter is enabled, and when PA = 0b1 the arbiter is disabled). See Chapter 4, “Reset, Clocking, and Initialization,” for more information on the reset configuration signals.

If the on-chip PCI arbiter is enabled, a request-grant pair of signals is provided for each external master ($\overline{\text{PCI\_REQ}}[0:4]$ and $\overline{\text{PCI\_GNT}}[0:4]$). In addition, there is an internal request/grant pair for the internal master state machine of the MPC8540 that governs processor, DMA, or RapidIO accesses to the PCI interface. If the on-chip PCI arbiter is disabled, the MPC8540 uses the $\overline{\text{PCI\_REQ0}}$ signal as an output to issue its request to the external arbiter and uses the $\overline{\text{PCI\_GNT0}}$ signal as an input to receive its grant from the external arbiter.

The following sections describe the operation of the on-chip PCI arbiter that arbitrates between external PCI masters and the internal PCI bus master of the MPC8540.

16.4.1.1 PCI Bus Arbiter Operation

The on-chip PCI arbiter uses a programmable two-level, round-robin arbitration algorithm. Each of the five external masters, plus the MPC8540, can be programmed for two priority levels, high or low, using the appropriate bits in the PBACR. Within each priority group, the PCI bus grant is asserted to the next requesting device in numerical order, with the MPC8540 positioned before device 0.

Conceptually, the lowest priority device is the master that is currently using the bus, and the highest priority device is the device that follows the current master in numerical order and group priority. This is considered to be a fair algorithm, since a single device cannot prevent other devices from having access to the bus; it automatically becomes the lowest priority device as soon as it begins to use the bus. If a master is not requesting the bus, then its transaction slot is given to the next requesting device within its priority group.

A grant is awarded to the highest priority requesting device as soon as the current master begins a transaction; however, the granted device must wait until the bus is relinquished by the current master before initiating a transaction.

The grant given to a particular device may be removed and awarded to another higher priority device, whenever the higher priority device asserts its request. If the bus is idle when a device requests the bus, then the arbiter withholds the grant for one clock cycle. The arbiter re-evaluates the priorities of all requesting devices and grants the bus to the highest priority device in the following clock cycle. This allows a turnaround clock when a higher priority device is using address stepping or when the bus is parked.

The low-priority group collectively has one bus transaction request slot in the high-priority group. For N high-priority devices and M low-priority devices, each high-priority device is guaranteed at least 1 of $N+1$ bus transactions and each low-priority device is guaranteed at least 1 of $(N+1) \times M$ bus transactions, with one low-priority device receiving the grant in 1 of $N+1$ bus transactions. If all devices are programmed to the same priority level, or if the low-priority group has only one device, the algorithm defaults to give each device an equal number of bus grants in round-robin sequence.

For the example in Figure 16-46, assume that several devices are requesting the bus. If two masters are in the high-priority group and three are in the low-priority group, each high-priority master is guaranteed at least one out of three transaction slots and each low-priority master is guaranteed one out of nine transaction slots.

In Figure 16-46, the grant sequence (with all devices, except device 4 requesting the bus and device 3 being the current master) is 0, 2, MPC8540, 0, 2, 1, 0, 2, 3, ..., and repeating. If device 2 is not requesting the bus, the grant sequence is 0, MPC8540, 0, 1, 0, 3, ..., and repeating. If device 2 requests the bus when device 0 is conducting a transaction and the MPC8540 has the next grant, the MPC8540 has its grant removed and device 2 is awarded the grant since device 2 is higher priority than the MPC8540 when device 0 has the bus.

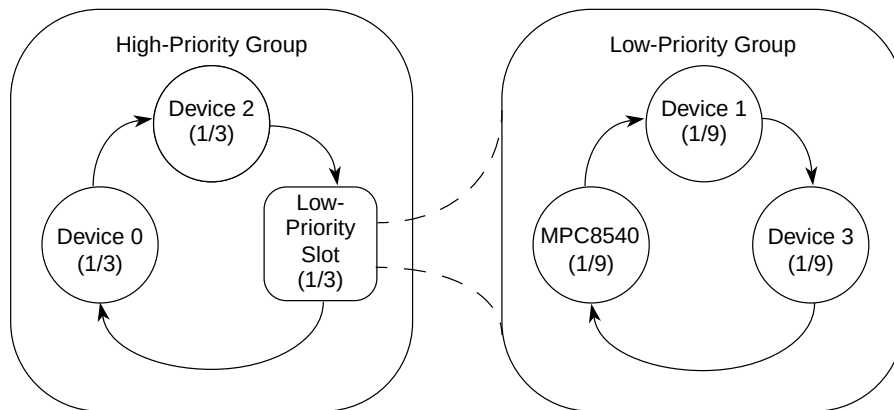


Figure 16-46. PCI Arbitration Example

16.4.1.2 PCI Bus Parking

When no device is using or requesting the bus, the PCI arbiter grants the bus to a selected device. This is known as parking the bus on the selected device. The selected device is required to drive the PCI_AD[31:0], PCI_C/BE[0:3], and the PCI parity signals to a stable value, preventing these signals from floating.

The parking mode parameter (PBACR[PM]) determines which device the arbiter selects for parking the PCI bus. If PBACR[PM] = 0b0 (or if the bus is not idle), then the bus is parked on the last master to use the bus. If the bus is idle and PBACR[PM] = 0b1, the bus is parked on the MPC8540.

16.4.1.3 Broken Master Lock-Out

The PCI bus arbiter on the MPC8540 has a feature that allows it to lock out any masters that are broken or ill-behaved. The broken master feature is controlled by programming bit 12 of the PCI bus arbitration control register (0b0 = enabled, 0b1 = disabled).

When the broken master feature is enabled, a granted device that does not assert PCI_FRAME within 16 PCI clock cycles after the bus is idle, has its grant removed and

subsequent requests are ignored until its $\overline{\text{REQ}}$ is negated for at least one clock cycle. This prevents ill-behaved masters from monopolizing the bus. When the broken master feature is disabled, a device that requests the bus and receives a grant never loses its grant until and unless it begins a transaction or negates its $\overline{\text{REQ}}$ signal. Note that disabling the broken master feature is not recommended.

16.4.1.4 Power-Saving Modes and the PCI Arbiter

In the sleep power-saving mode, the clock signal driving SYSCLK can be disabled. If the clock is disabled, the arbitration logic is not able to perform its function. System programmers must park the bus with a device that can sustain the PCI_AD[31:0], PCI_C/ $\overline{\text{BE}}$ [3:0], and parity signals prior to disabling the SYSCLK signal. If the bus is parked on the MPC8540 when its clocks are stopped, the MPC8540 sustains the PCI_AD[31:0], PCI_C/ $\overline{\text{BE}}$ [3:0], and parity signals in their prior states. In this situation, the only way for another agent to use the PCI bus is by waking the MPC8540. In nap and doze power-saving modes, the arbiter continues to operate allowing other PCI devices to run transactions.

16.4.2 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in Section 16.4.2.7, “PCI Bus Transactions.” Refer to Figure 16-47, Figure 16-48, Figure 16-49, and Figure 16-50 for examples of the transfer-control mechanisms described in this section.

All signals are sampled on the rising edge of the PCI bus clock (SYSCLK). Each signal has a setup and hold aperture with respect to the rising clock edge in which transitions are not allowed. Outside this aperture, signal values or transitions have no significance. See the *MPC8540 Hardware Specifications* for specific setup and hold times.

16.4.2.1 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst. A burst is composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals— $\overline{\text{PCI\_FRAME}}$ (frame), $\overline{\text{PCI\_IRDY}}$ (initiator ready), and $\overline{\text{PCI\_TRDY}}$ (target ready). An initiator asserts $\overline{\text{PCI\_FRAME}}$ to indicate the beginning of a PCI bus transaction and negates $\overline{\text{PCI\_FRAME}}$ to indicate the end of a PCI bus transaction. An initiator negates $\overline{\text{PCI\_IRDY}}$ to force wait cycles. A target negates $\overline{\text{PCI\_TRDY}}$ to force wait cycles.

The PCI bus is considered idle when both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated. The first clock cycle in which $\overline{\text{PCI\_FRAME}}$ is asserted indicates the beginning of the address phase. The address and bus command code are transferred in that first cycle. The next cycle begins the first of one or more data phases. Data is transferred between initiator and target

in each cycle that both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted. Wait cycles may be inserted in a data phase by the initiator (by negating $\overline{\text{PCI\_IRDY}}$) or by the target (by negating $\overline{\text{PCI\_TRDY}}$).

Once an initiator has asserted $\overline{\text{PCI\_IRDY}}$, it cannot change $\overline{\text{PCI\_IRDY}}$ or $\overline{\text{PCI\_FRAME}}$ until the current data phase completes regardless of the state of $\overline{\text{PCI\_TRDY}}$. Once a target has asserted $\overline{\text{PCI\_TRDY}}$ or $\overline{\text{PCI\_STOP}}$, it cannot change $\overline{\text{PCI\_DEVSEL}}$, $\overline{\text{PCI\_TRDY}}$, or $\overline{\text{PCI\_STOP}}$ until the current data phase completes. In simpler terms, once an initiator or target has committed to the data transfer, it cannot change its mind.

When the initiator intends to complete only one more data transfer (which could be immediately after the address phase), $\overline{\text{PCI\_FRAME}}$ is negated and $\overline{\text{PCI\_IRDY}}$ is asserted (or kept asserted), indicating the initiator is ready. After the target indicates the final data transfer (by asserting $\overline{\text{PCI\_TRDY}}$), the PCI bus may return to the idle state (both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated) unless a fast back-to-back transaction is in progress. In the case of a fast back-to-back transaction, an address phase immediately follows the last data phase.

16.4.2.2 PCI Bus Commands

A PCI bus command is encoded in the $\text{PCI\_C}/\overline{\text{BE}}[3:0]$ signals during the address phase of a PCI transaction. The bus command indicates to the target the type of transaction the initiator is requesting. Table 16-43 describes the PCI bus commands implemented by the MPC8540.

Table 16-43. PCI Bus Commands

$\text{PCI\_C}/\overline{\text{BE}}[3:0]$	PCI Bus Command	MPC8540 Supports as an Initiator	MPC8540 Supports as a Target	Definition
0000	Interrupt-acknowledge	Yes	No	The interrupt-acknowledge command is a read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to the interrupt-acknowledge command. Other devices ignore the interrupt-acknowledge command. See Section 16.4.2.12.1, "Interrupt-Acknowledge Transactions," for more information.
0001	Special cycle	Yes	No	The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. See Section 16.4.2.12.2, "Special-Cycle Transactions," for more information.
0010	I/O-read	Yes	No	The I/O-read command accesses agents mapped into the PCI I/O space.
0011	I/O-write	Yes	No	The I/O-write command accesses agents mapped into the PCI I/O space.
0100	Reserved ¹	No	No	—
0101	Reserved ¹	No	No	—

Table 16-43. PCI Bus Commands (continued)

PCI C/ BE[3:0]	PCI Bus Command	MPC8540 Supports as an Initiator	MPC8540 Supports as a Target	Definition
0110	Memory-read	Yes	Yes	The memory-read command accesses either local memory or agents mapped into PCI memory space, depending on the address. When a PCI master issues a memory-read command to local memory, the MPC8540 (the target) fetches data from the requested address to the end of the cache line (32 bytes) from local memory, even though all of the data may not be requested by (or sent to) the initiator.
0111	Memory-write	Yes	Yes	The memory-write command accesses either local memory or agents mapped into PCI memory space, depending on the address.
1000	Reserved ¹	No	No	—
1001	Reserved ¹	No	No	—
1010	Configuration-read	Yes	Agent mode only	The configuration-read command accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See Section 16.4.2.11, “Configuration Cycles,” for more detail on PCI configuration cycles.
1011	Configuration-write	Yes	Agent mode only	The configuration-write command accesses the 256-byte configuration space of a PCI agent. A specific agent is selected when its IDSEL signal is asserted during the address phase. See Section 16.4.2.11, “Configuration Cycles,” for more detail on PCI configuration accesses.
1100	Memory-read-multiple	Yes (for DMA cycles)	Yes	The memory-read-multiple command functions similarly to the memory-read command, but it also causes a prefetch of the next cache line (32 bytes).
1101	Dual-address-cycle	Yes	Yes	The dual-address-cycle command is used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices.
1110	Memory-read-line	Yes	Yes	The memory-read-line command indicates that an initiator is requesting the transfer of an entire cache line (32 bytes). This only occurs when the processor is performing a burst read. Note that these processors only perform burst reads when the appropriate cache is enabled and the transaction is not cache-inhibited.
1111	Memory-write-and-invalidate	Yes (for DMA cycles)	Yes	The memory-write-and-invalidate command indicates that an initiator is transferring an entire cache line (32 bytes); if this data is in any cacheable memory, that cache line needs to be invalidated.

¹ Reserved command encodings are reserved for future use. The MPC8540 does not respond to these commands.

16.4.2.3 Addressing

PCI defines three physical address spaces—PCI memory space, PCI I/O space, and PCI configuration space. Access to the PCI memory and I/O space is straightforward, although

one must take into account the local memory access window and address translation being used. The address translation registers are described in Section 16.3.1, “PCI/PCI-X Memory Mapped Registers.” Access to the PCI configuration space is described in Section 16.4.2.11, “Configuration Cycles.”

Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding—positive decoding and subtractive decoding. For positive decoding, each device is looking for accesses in the address range that the device has been assigned. For subtractive decoding, one device on the bus is looking for accesses that no other device has claimed. See Section 16.4.2.4, “Device Selection,” for information about claiming transactions.

The information contained in the two low-order address bits (PCI_AD[1:0]) varies by the address space (memory, I/O, or configuration). Regardless of the encoding scheme, the two low-order address bits are always included in parity calculations.

16.4.2.3.1 Memory Space Addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address—linear incrementing (PCI_AD[1:0] = 0b00) and cache wrap mode (PCI_AD[1:0] = 0b10), as shown in Table 16-44. The other two PCI_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the MPC8540 always encodes PCI_AD[1:0] = 0b00 for PCI memory space accesses. As a target, the MPC8540 executes a target disconnect after the first data phase completes if PCI_AD[1:0] = 0b01 or PCI_AD[1:0] = 0b11 during the address phase of a local memory access. See Section 16.4.2.8.2, “Target-Initiated Termination,” for more information on target disconnect conditions.

Table 16-44. Supported Combinations of PCI_AD[1:0]

PCI_AD[1:0]		MPC8540 as Target		MPC8540 as Initiator	
		Read	Write	Read	Write
00	Linear	√	√	√	√
01	Reserved	TD	TD	—	—
10	Cache Wrap	√	TD	—	—
11	Reserved	TD	TD	—	—

For linear incrementing mode, the memory address is encoded/decoded using PCI_AD[63:2]. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4-byte data width per data phase is implied). Note that the two low-order bits on the address bus are included in all parity calculations.

For cache wrap mode ($\text{PCI\_AD}[1:0] = 0b10$) reads, the critical memory address is decoded using $\text{PCI\_AD}[63:2]$. The address is incremented by 4 bytes after each data phase completes until the end of the cache line is reached. For cache-wrap reads, the address wraps to the beginning of the current cache line and continues incrementing until the entire cache line (32 bytes) is read. The MPC8540 does not support cache-wrap write operations and executes a target disconnect after the data phase for the end of the cache line completes for writes with $\text{PCI\_AD}[1:0] = 0b10$. That is, the MPC8540 does not wrap back to the beginning of the cache line. Note that the two low-order bits on the address bus are included in all parity calculations.

16.4.2.3.2 I/O Space Addressing

For PCI I/O accesses, 32 address signals ($\text{PCI\_AD}[31:0]$) are used to provide a byte address. After a target has claimed an I/O access, it must determine if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range of the target, the entire access cannot complete. In this case, the target does not transfer any data and terminates the transaction with a target-abort error. See Section 16.4.2.8.2, “Target-Initiated Termination,” for more information.

16.4.2.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses that use different formats for the $\text{PCI\_AD}[31:0]$ signals during the address phase. The two low-order bits of the address indicate the format used for the configuration address phase—type 0 ($\text{PCI\_AD}[1:0] = 0b00$) or type 1 ($\text{PCI\_AD}[1:0] = 0b01$). Both address formats identify a specific device and a specific configuration register for that device. See Section 16.4.2.11, “Configuration Cycles,” for descriptions of the two formats.

16.4.2.4 Device Selection

The $\overline{\text{PCI\_DEVSEL}}$ signal is driven by the target of the current transaction. $\overline{\text{PCI\_DEVSEL}}$ indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction. $\overline{\text{PCI\_DEVSEL}}$ may be driven one, two, or three clock cycles (fast, medium, or slow device select timing) following the address phase. Device select timing is encoded into the device’s PCI status register. If no agent asserts $\overline{\text{PCI\_DEVSEL}}$ within three clock cycles of $\overline{\text{PCI\_FRAME}}$, the agent responsible for subtractive decoding may claim the transaction by asserting $\overline{\text{PCI\_DEVSEL}}$.

A target must assert $\overline{\text{PCI\_DEVSEL}}$ (claim the transaction) before or coincident with any other target response (assert $\overline{\text{PCI\_TRDY}}$, $\overline{\text{PCI\_STOP}}$, or data signals). In all cases except target-abort, once a target asserts $\overline{\text{PCI\_DEVSEL}}$, it must not negate $\overline{\text{PCI\_DEVSEL}}$ until $\overline{\text{PCI\_FRAME}}$ is negated (with $\overline{\text{PCI\_IRDY}}$ asserted) and the last data phase has completed. For normal termination, negation of $\overline{\text{PCI\_DEVSEL}}$ coincides with the negation of $\overline{\text{PCI\_TRDY}}$ or $\overline{\text{PCI\_STOP}}$.

If the first access maps into a target's address range, that target asserts $\overline{\text{PCI\_DEVSEL}}$ to claim the access. However, if the initiator attempts to continue the burst access across the resource boundary, then the target must issue a target disconnect.

The MPC8540 is hardwired for fast device select timing (PCI status register [10–9] = 0b00). Therefore, when the MPC8540 is the target of a transaction (local memory access or configuration register access), it asserts $\overline{\text{PCI\_DEVSEL}}$ one clock cycle following the address phase.

As an initiator, if the MPC8540 does not detect the assertion of $\overline{\text{PCI\_DEVSEL}}$ within four clock cycles after the address phase (that is, five clock cycles after it asserts $\overline{\text{PCI\_FRAME}}$), it terminates the transaction with a master-abort termination; see Section 16.4.2.8.1, “Master-Initiated Termination.”

16.4.2.5 Byte Alignment

The byte enable signals of the PCI bus ($\text{PCI\_C/BE}[7:0]$, during a data phase) are used to determine which byte lanes carry meaningful data. The byte enable signals may enable different bytes for each of the data phases. The byte enables are valid on the edge of the clock that starts each data phase and stay valid for the entire data phase. Note that parity is calculated for all bytes regardless of the state of the byte enable signals. See Section 16.4.2.13.1, “PCI Parity,” for more information.

If the MPC8540, as a target, detects no byte enables asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the MPC8540 expects that the data is not changed, and on a write transaction, the data is not stored.

16.4.2.6 Bus Driving and Turnaround

To avoid contention, a turnaround cycle is required on all signals that may be driven by more than one agent. The turnaround cycle occurs at different times for different signals. The $\overline{\text{PCI\_IRDY}}$, $\overline{\text{PCI\_TRDY}}$, $\overline{\text{PCI\_DEVSEL}}$, and $\overline{\text{PCI\_STOP}}$ signals use the address phase as their turnaround cycle. $\overline{\text{PCI\_FRAME}}$, $\text{PCI\_C/BE}[7:0]$, and $\text{PCI\_AD}[63:0]$ signals use the idle cycle between transactions (when both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated) as their turnaround cycle. The $\overline{\text{PCI\_PERR}}$ signal has a turnaround cycle on the fourth clock after the last data phase.

The PCI address/data signals, $\text{PCI\_AD}[63:0]$, are driven to a stable condition during every address/data phase. Even when the byte enables indicate that byte lanes carry meaningless data, the signals carry stable values. Parity is calculated on all bytes regardless of the byte enables. See Section 16.4.2.13.1, “PCI Parity,” for more information.

16.4.2.7 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in Section 16.4.2, “PCI Bus Protocol.” Read and write

transactions are similar for the memory and I/O spaces, so they are described as generic read transactions and generic write transactions.

The timing diagrams in this section show the relationship of significant signals involved in bus transactions. When a signal is drawn as a solid line, it is actively being driven by the current master or target. When a signal is drawn as a dashed line, no agent is actively driving it. High-impedance signals are indicated to have indeterminate values when the dashed line is between the two rails.

The terms ‘edge’ and ‘clock edge’ always refer to the rising edge of the clock. The terms ‘asserted’ and ‘negated’ always refer to the globally visible state of the signal on the clock edge, and not to signal transitions. ‘’ represents a turnaround cycle in the timing diagrams.

16.4.2.7.1 PCI Read Transactions

This section describes PCI single-beat read transactions and PCI burst read transactions.

A read transaction starts with the address phase, occurring when an initiator asserts $\overline{\text{PCI\_FRAME}}$. During the address phase, $\text{PCI\_AD}[63:0]$ contains a valid address and $\text{PCI\_C}/\overline{\text{BE}}[7:0]$ contains a valid bus command.

The first data phase of a read transaction requires a turnaround cycle. This allows the transition from the initiator driving $\text{PCI\_AD}[63:0]$ as address signals to the target driving $\text{PCI\_AD}[63:0]$ as data signals. The turnaround cycle is enforced by the target with the $\overline{\text{TRDY}}$ signal. The target provides valid data at the earliest one cycle after the turnaround cycle. The target must drive the $\text{PCI\_AD}[63:0]$ signals when $\overline{\text{PCI\_DEVSEL}}$ is asserted.

During the data phase, the $\text{PCI\_C}/\overline{\text{BE}}[7:0]$ signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The $\text{PCI\_C}/\overline{\text{BE}}[7:0]$ signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted on the same clock edge. When either $\overline{\text{PCI\_IRDY}}$ or $\overline{\text{PCI\_TRDY}}$ is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating $\overline{\text{PCI\_FRAME}}$ when $\overline{\text{PCI\_IRDY}}$ is asserted. The transaction is considered complete when data is transferred in the last data phase.

Figure 16-47 illustrates a PCI single-beat read transaction. Figure 16-48 illustrates a PCI burst read transaction.

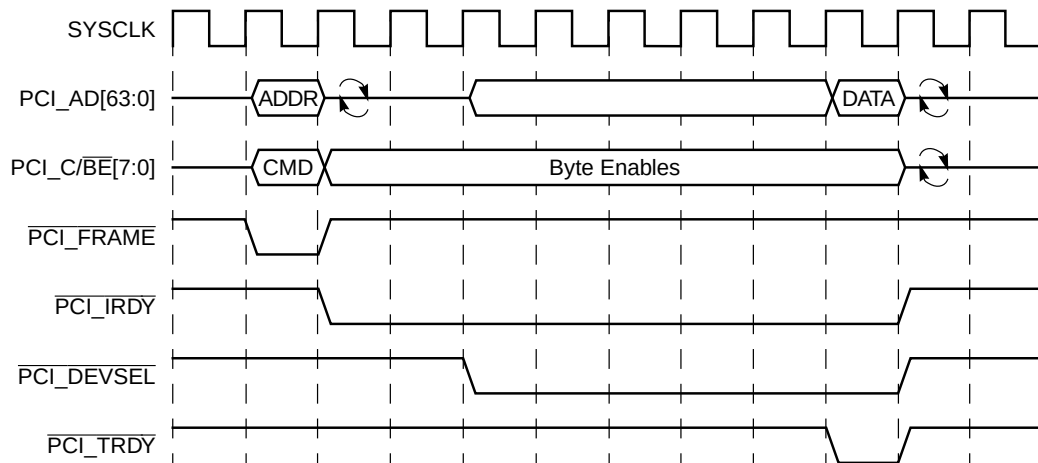


Figure 16-47. PCI Single-Beat Read Transaction

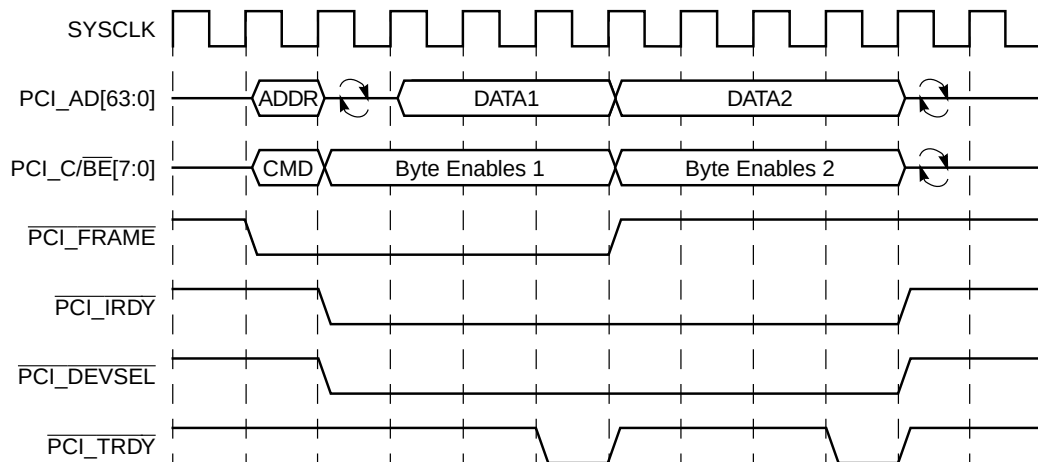


Figure 16-48. PCI Burst Read Transaction

16.4.2.7.2 PCI Write Transactions

This section describes PCI single-beat write transactions, and PCI burst write transactions. A PCI write transaction starts with the address phase, occurring when an initiator asserts `PCI_FRAME`. A write transaction is similar to a read transaction except no turnaround cycle is needed following the address phase because the initiator provides both address and data. The data phases are the same for both read and write transactions. Although not shown in the figures, the initiator must drive the `PCI_C/BE[7:0]` signals, even if the initiator is not ready to provide valid data (`PCI_IRDY` negated).

Figure 16-49 illustrates a PCI single-beat write transaction. Figure 16-50 illustrates a PCI burst write transaction.

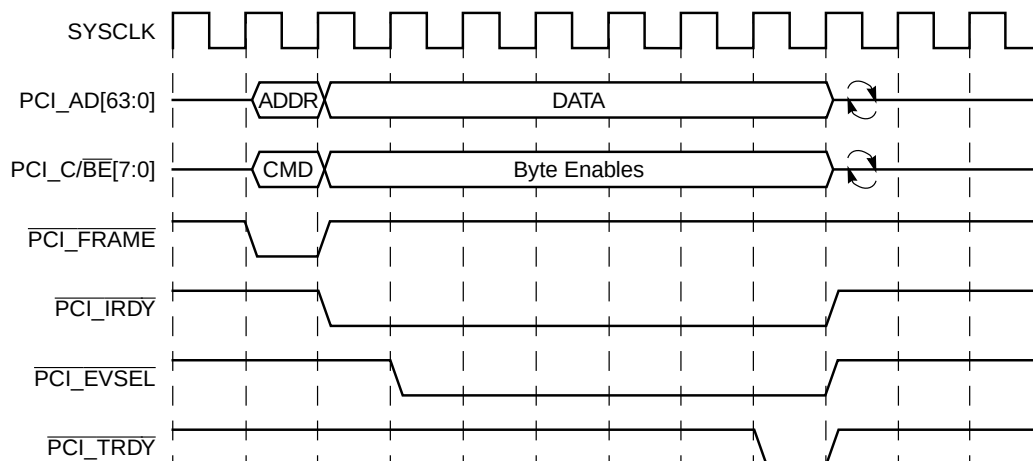


Figure 16-49. PCI Single-Beat Write Transaction

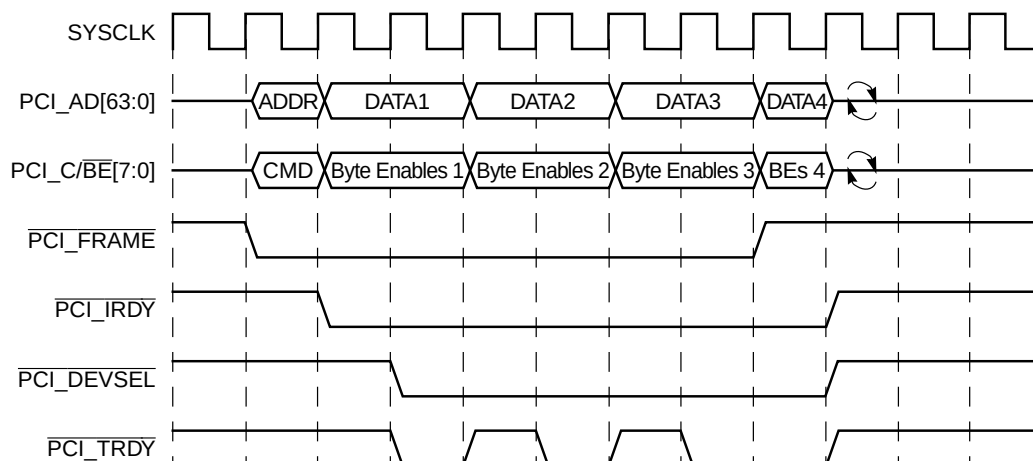


Figure 16-50. PCI Burst Write Transaction

16.4.2.8 Transaction Termination

A PCI transaction may be terminated by either the initiator or the target. The initiator is ultimately responsible for concluding all transactions, regardless of the cause of the termination. All transactions are concluded when $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are both negated, indicating the bus is idle.

16.4.2.8.1 Master-Initiated Termination

Normally, a master initiates termination by negating $\overline{\text{PCI\_FRAME}}$ and asserting $\overline{\text{PCI\_IRDY}}$. This indicates to the target that the final data phase is in progress. The final data transfer occurs when both $\overline{\text{PCI\_TRDY}}$ and $\overline{\text{PCI\_IRDY}}$ are asserted. The transaction is

considered complete when data is transferred in the last data phase. After the final data phase, both $\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ are negated (the bus becomes idle).

There are three types of master-initiated termination:

- **Completion**—Refers to termination when the initiator has concluded its intended transaction. This is the most common reason for termination.
- **Timeout**—Refers to termination when the initiator loses its bus grant ($\overline{\text{GNTn}}$ is negated), and its internal latency timer has expired. The intended transaction is not necessarily concluded.
- **Master-abort**—An abnormal case of master-initiated termination. If no device (including the subtractive decoding agent) asserts $\overline{\text{PCI\_DEVSEL}}$ to claim a transaction, the initiator terminates the transaction with a master-abort. For a master-abort termination, the initiator negates $\overline{\text{PCI\_FRAME}}$ and then negates $\overline{\text{PCI\_IRDY}}$ on the next clock. If a transaction is terminated by master-abort (except for a special-cycle command), the received master-abort bit (bit 13) of the PCI status register is set.

As an initiator, if the MPC8540 does not detect the assertion of $\overline{\text{PCI\_DEVSEL}}$ within four clock cycles following the address phase (five clock cycles after asserting $\overline{\text{PCI\_FRAME}}$), it terminates the transaction with a master-abort. On reads that are master-aborted, the MPC8540 sends an error message back through the OCeAN to the internal requestor. On writes that are master-aborted, the data is discarded.

16.4.2.8.2 Target-Initiated Termination

By asserting the $\overline{\text{PCI\_STOP}}$ signal, a target may request that the initiator terminate the current transaction. Once asserted, the target holds $\overline{\text{PCI\_STOP}}$ asserted until the initiator negates $\overline{\text{PCI\_FRAME}}$. Data may or may not be transferred during the request for termination. If $\overline{\text{PCI\_TRDY}}$ and $\overline{\text{PCI\_IRDY}}$ are asserted during the assertion of $\overline{\text{PCI\_STOP}}$, data is transferred. However, if $\overline{\text{PCI\_TRDY}}$ is negated when $\overline{\text{PCI\_STOP}}$ is asserted, it indicates that the target will not transfer any more data; therefore, the initiator does not wait for a final data transfer as it would in a completion termination.

When a transaction is terminated by $\overline{\text{PCI\_STOP}}$, the initiator must negate its $\overline{\text{REQn}}$ signal for a minimum of two PCI clock cycles, (one corresponding to when the bus goes to the idle state ($\overline{\text{PCI\_FRAME}}$ and $\overline{\text{PCI\_IRDY}}$ negated)). If the initiator intends to complete the transaction, it can reassert its $\overline{\text{REQn}}$ immediately following the two clock cycles. If the initiator does not intend to complete the transaction, it can assert $\overline{\text{REQn}}$ whenever it needs to use the PCI bus again.

There are three types of target-initiated termination:

- **Disconnect**—Disconnect refers to termination requested because the target is temporarily unable to continue bursting. Disconnect implies that some data has been transferred. The initiator may restart the transaction at a later time starting with the

Functional Description

address of the next untransferred data. (That is, data transfer may resume where it left off.)

- **Retry**—Retry refers to termination requested because the target is currently in a state where it is unable to process the transaction. Retry implies that no data was transferred. The initiator may start the entire transaction over again at a later time. Note that the *PCI Local Bus Specification*, Rev. 2.2 requires that all retried transactions must be completed.
- **Target-Abort**—Target-abort is an abnormal case of target-initiated termination. Target-abort is used when a fatal error has occurred or when a target can never respond.

As a target, the MPC8540 terminates a transaction with a target disconnect due to the following:

- It is unable to respond within eight PCI clock cycles (not including the first data phase).
- The transaction is attempting to cross a 4-Kbyte boundary.
- A single beat of data has been transferred and the inbound ATMU is marked non-prefetchable.
- The end of a cache line has been transferred for a cache-wrap mode write transaction. See Section 16.4.2.3.1, “Memory Space Addressing,” for more information.

As a target, the MPC8540 responds to a transaction with a retry due to the following:

- The 16-clock latency timer has expired, and the first data phase has not begun.
- There is no more internal buffer space available for an inbound transaction.

Target-abort is indicated by asserting $\overline{\text{PCI\_STOP}}$ and negating $\overline{\text{PCI\_DEVSEL}}$. This indicates that the target requires termination of the transaction and does not want the transaction retried. If a transaction is terminated by target-abort, the received target-abort bit (bit 12) of the initiator’s status register and the signaled target-abort bit (bit 11) of the target’s status register are set. Note that any data transferred in a target-aborted transaction may be corrupt.

For PCI writes to local memory, if an address parity error or data parity error occurs, the MPC8540 aborts the transaction internally, but continues the transaction on the PCI bus.

Figure 16-51 shows several target-initiated terminations.

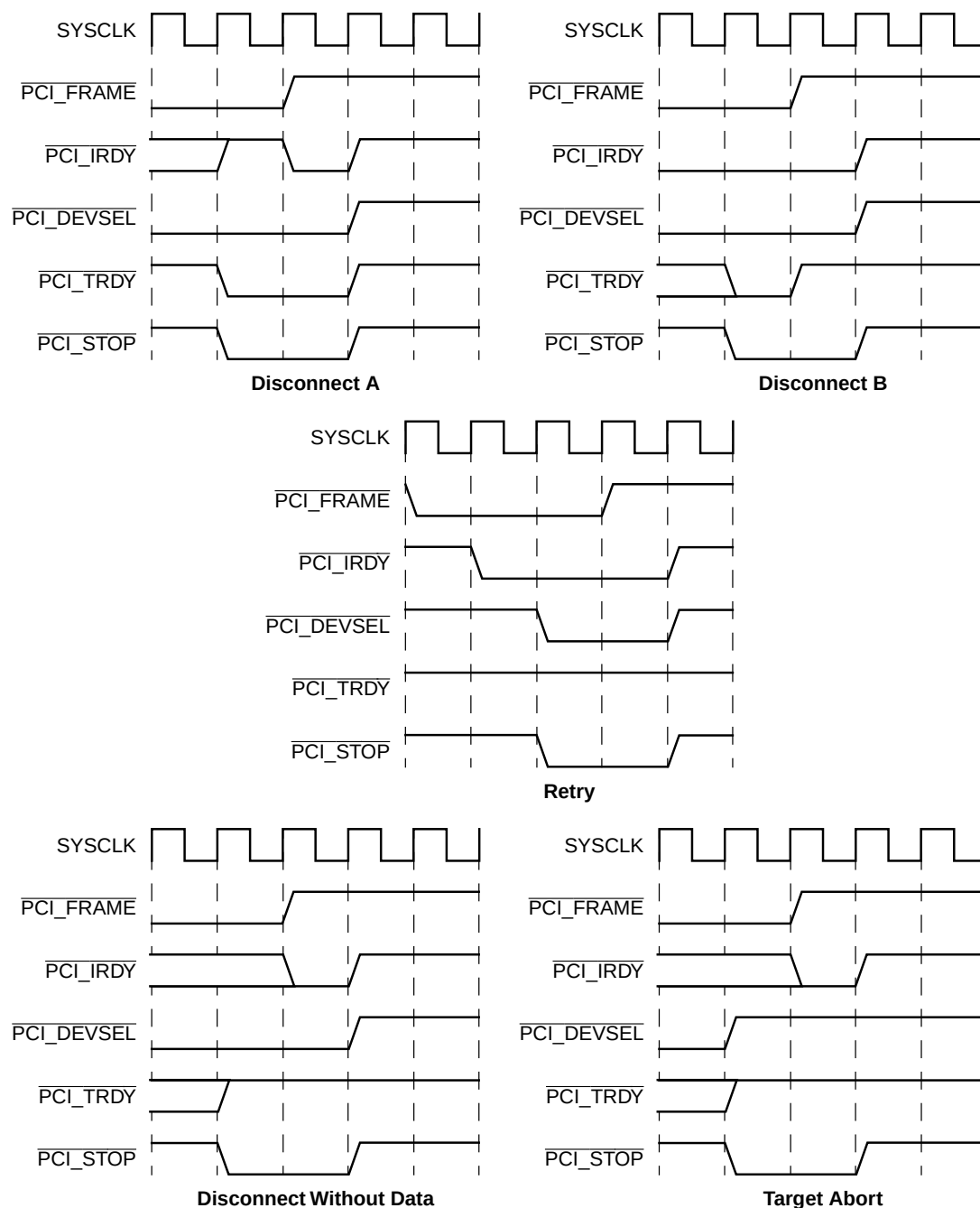


Figure 16-51. PCI Target-Initiated Terminations

The three disconnect terminations are unique in the data transferred at the end of the transaction. For disconnect A, the initiator is negating $\overline{\text{PCI\_IRDY}}$ when the target asserts $\overline{\text{PCI\_STOP}}$ and data is transferred only at the end of the current data phase. For disconnect B, the target negates $\overline{\text{PCI\_TRDY}}$ one clock after it asserts $\overline{\text{PCI\_STOP}}$, indicating that the

target can accept the current data, but no more data can be transferred. For disconnect-without-data, the target asserts $\overline{\text{PCI\_STOP}}$ when $\overline{\text{PCI\_TRDY}}$ is negated indicating that the target cannot accept any more data.

16.4.2.9 Fast Back-to-Back Transactions

The PCI bus allows fast back-to-back transactions by the same master. During a fast back-to-back transaction, the initiator starts the next transaction immediately without an idle state. The last data phase completes when $\overline{\text{PCI\_FRAME}}$ is negated, and $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted. The current master starts another transaction in the clock cycle immediately following the last data transfer for the previous transaction.

Fast back-to-back transactions must avoid contention on the $\overline{\text{PCI\_TRDY}}$, $\overline{\text{PCI\_DEVSEL}}$, $\overline{\text{PCI\_PERR}}$, and $\overline{\text{PCI\_STOP}}$ signals. There are two types of fast back-to-back transactions—those that access the same target and those that access multiple targets sequentially. The first type places the burden of avoiding contention on the initiator; the second type places the burden of avoiding contention on all potential targets.

As an initiator, the MPC8540 does not perform any fast back-to-back transactions. As a target, the MPC8540 supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC8540 monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC8540 and the current transaction is directed at the MPC8540, it delays the assertion of $\overline{\text{PCI\_DEVSEL}}$ (as well as $\overline{\text{PCI\_TRDY}}$, $\overline{\text{PCI\_STOP}}$, and $\overline{\text{PCI\_PERR}}$) for one clock cycle to allow the other target to stop driving the bus.

16.4.2.10 Dual Address Cycles

The MPC8540 supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as both an initiator and a target. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The MPC8540 block supports single-beat and burst DAC transactions.

The timing sequence of the PCI signals for single-beat DAC reads is shown in Figure 16-52. The timing for a DAC burst read is shown in Figure 16-53. Figure 16-54 and Figure 16-55 show timing examples for single-beat DAC writes and burst DAC writes, respectively.

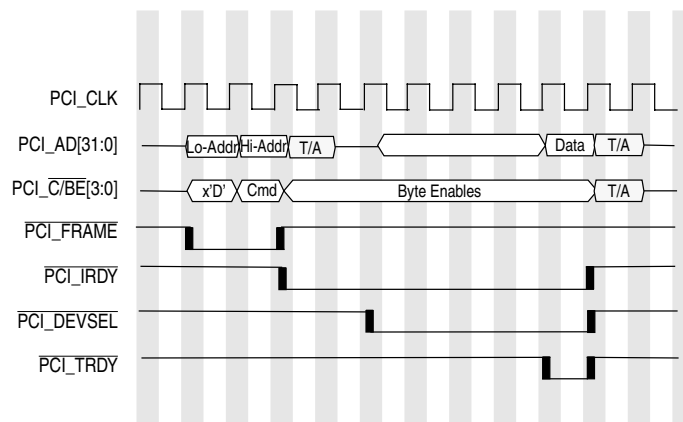


Figure 16-52. DAC Single-Beat Read Example

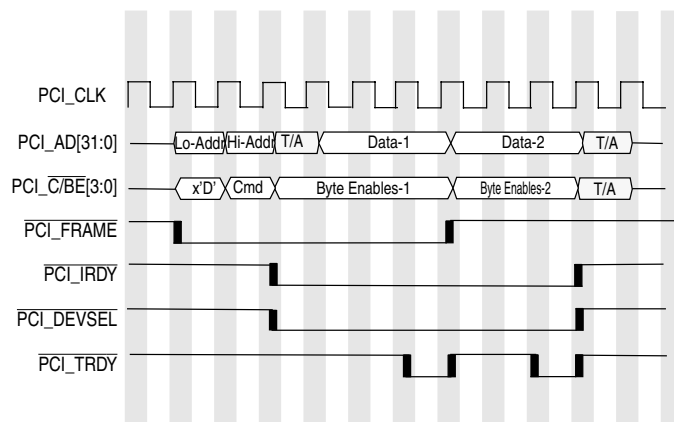


Figure 16-53. DAC Burst Read Example

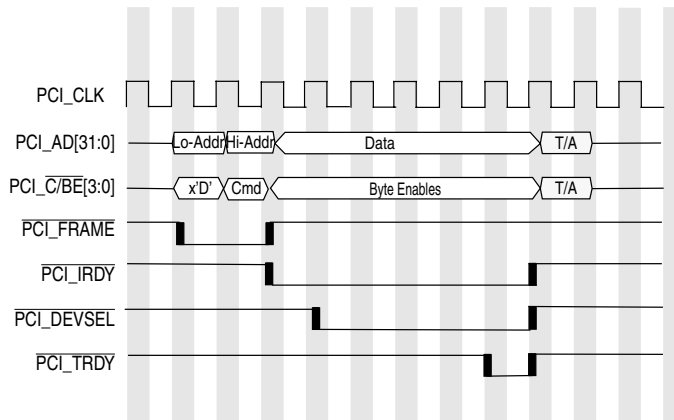


Figure 16-54. DAC Single-Beat Write Example

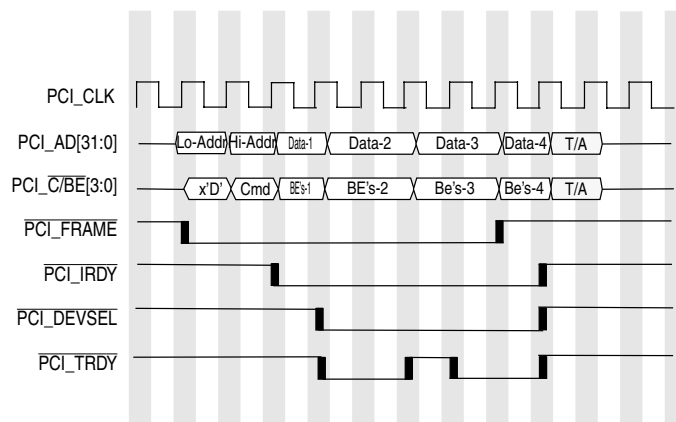


Figure 16-55. DAC Burst Write Example

16.4.2.11 Configuration Cycles

This section describes PCI configuration cycles used for configuring standard PCI devices. The PCI configuration space of any device is intended for configuration, initialization, and catastrophic error-handling functions only. Access to the PCI configuration space should be limited to initialization and error-handling software.

16.4.2.11.1 PCI Configuration Space Header

The first 64 bytes of the 256-byte configuration space consists of a predefined header that every PCI device must support. The predefined header for all PCI devices is shown in Figure 16-56. The first 16 bytes of the predefined header are defined the same for all PCI devices; the remaining 48 bytes of the header may have differing layouts depending on the function of the device. Most PCI devices use the configuration header layout shown in Figure 16-56. The rest of the 256-byte configuration space is device-specific. The PCI header specific to the MPC8540 is described in Section 16.3.2, “PCI/PCI-X Configuration Header.”

				Address Offset
Device ID		Vendor ID		0x00
Status		Command		0x04
Class Code			Revision ID	0x08
BIST	Header Type	Latency Timer	Cache Line Size	0x0C
Base Address Registers				0x10
				0x14
				0x18
				0x1C
				0x20
				0x24
Reserved				0x28
Subsystem ID		Subsystem Vendor ID		0x2C
Expansion ROM Base Address				0x30
Reserved				0x34
Reserved				0x38
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	0x3C

Figure 16-56. Standard PCI Configuration Header

Table 16-45 summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*, Rev. 2.2.

Table 16-45. PCI Configuration Space Header Summary

Address Offset (Hex)	Register Name	Description
0x00	Vendor ID	Identifies the manufacturer of the device (assigned by the PCI SIG (special-interest group) to ensure uniqueness).
0x02	Device ID	Identifies the particular device (assigned by the vendor).
0x04	Command	Provides coarse control over a device's ability to generate and respond to PCI bus cycles
0x06	Status	Records status information for PCI bus-related events
0x08	Revision ID	Specifies a device-specific revision code (assigned by vendor)
0x09	Class code	Identifies the generic function of the device and (in some cases) a specific register-level programming interface
0x0C	Cache line size	Specifies the system cache line size in 32-bit units
0x0D	Latency timer	Specifies the value of the latency timer in PCI bus clock units for the device when acting as an initiator

Table 16-45. PCI Configuration Space Header Summary (continued)

Address Offset (Hex)	Register Name	Description
0x0E	Header type	Bits 0–6 identify the layout of bytes 10–3F; bit 7 indicates a multifunction device. The most common header type (0x00) is shown in Figure 16-56 and in this table.
0x0F	BIST	Optional register for control and status of built-in self test (BIST)
0x10–0x27	Base address registers	Address mapping information for memory and I/O space
0x28	—	Reserved for future use
0x2C	Subsystem Vendor ID	Identifies the subsystem vendor ID.
0x2E	Subsystem ID	Identifies the subsystem ID.
0x30	Expansion ROM base address	Base address and size information for expansion ROM contained in an add-on board
0x34	—	Reserved for future use
0x38	—	Reserved for future use
0x3C	Interrupt line	Contains interrupt line routing information
0x3D	Interrupt pin	Indicates which interrupt pin the device (or function) uses
0x3E	Min_Gnt	Specifies the length of the device's burst period in 0.25 μ s units
0x3F	Max_Lat	Specifies how often the device needs access to the bus in 0.25 μ s units

16.4.2.11.2 Accessing the PCI Configuration Space

This section describes accessing the external PCI configuration space. See Section 16.3.2, “PCI/PCI-X Configuration Header,” for information on accessing the internal configuration registers of the MPC8540.

To support hierarchical bridges, two types of configuration accesses are supported. The first type of configuration access, type 0, is used to select a device on the local PCI bus. Type 0 configuration accesses are not propagated beyond the local PCI bus and must be claimed by a local device or terminated with a master-abort. The second type of configuration access, type 1, is used to pass a configuration request to another PCI bus (through a PCI-to-PCI bridge). Type 1 accesses are ignored by all targets except PCI-to-PCI bridges.

To access the configuration space, a 32-bit value must be written to the CONFIG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. A read or write to the CONFIG_DATA register causes the host bridge to translate the access into a PCI configuration cycle (provided the enable bit in CONFIG_ADDR is set and the device number is not 0b1_1111).

For the MPC8540, the CONFIG_ADDR register is called CFG_ADDR and is located in the memory-mapped registers at offset 0x0_8000.

The format of CFG_ADDR is shown in Figure 16-57.

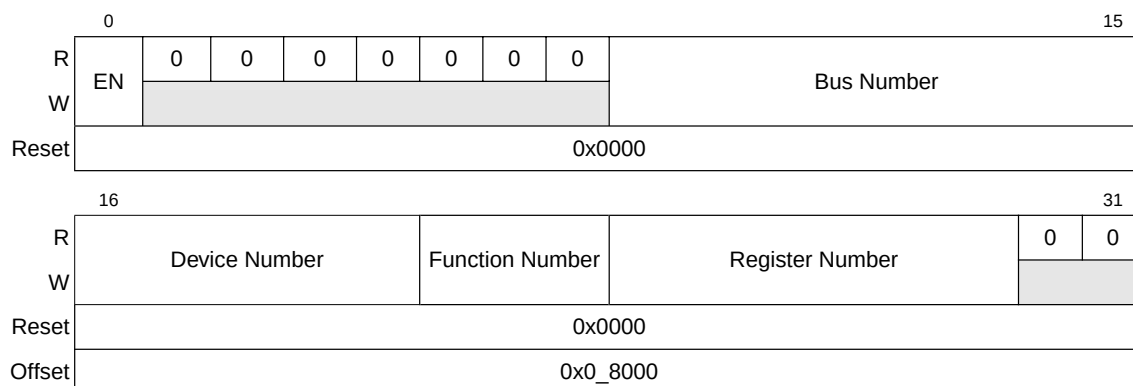


Figure 16-57. CFG ADDR Register Format

Table 16-46 describes the fields within CFG_ADDR.

Table 16-46. CFG_ADDR Register Fields

Bits	Field Name	Description
0	E(nable)	The enable flag controls whether accesses to CFG_DATA are translated into PCI configuration cycles. 1 Enabled 0 Disabled
1–7	—	Reserved (must be 0b000_0000)
8–15	Bus number	This field is an encoded value used to select the target bus of the configuration access. For target devices on the PCI bus connected to the MPC8540, this field should be set to 0x00.
16–20	Device number	This field is used to select a specific device on the target bus
21–23	Function number	This field is used to select a specific function in the requested device. Single-function devices should respond to function number 0b000.
24–29	Register number	This field is used to select the address offset in the configuration space of the target device.
30–31	—	Reserved (must be 0b00)

As with the CONFIG_ADDR register, the CONFIG_DATA register is called CFG_DATA and is located in the memory-mapped registers at offset 0x0_8004. The CFG_DATA register, is shown in Figure 16-58.

Functional Description

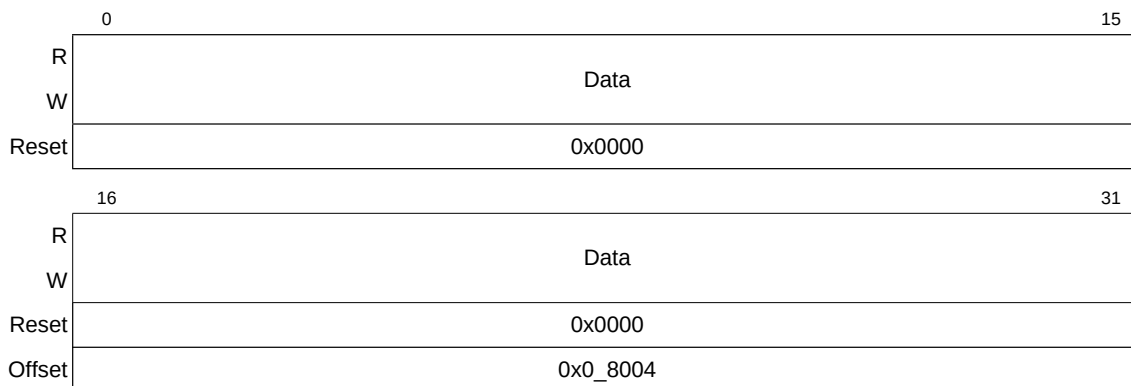


Figure 16-58. CFG_DATA Register

Note that CFG_DATA may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

When the MPC8540 detects an access to CFG_DATA, it checks the enable flag and the device number in the CFG_ADDR register. If the enable bit is set, and the device number is not 0b1_1111, the MPC8540 performs a configuration cycle translation function and runs a configuration-read or configuration-write transaction on the PCI bus. The device number 0b1_1111 is used for performing interrupt-acknowledge and special- cycle transactions. See Section 16.4.2.12, “Other Bus Transactions,” for more information. If the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC8540 performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8540 performs a type 1 configuration cycle translation.

16.4.2.11.3 Type 0 Configuration Translation

Figure 16-59 shows the type 0 translation function performed on the contents of the CFG_ADDR register to the PCI_AD[31:0] signals on the PCI bus during the address phase of the configuration cycle.

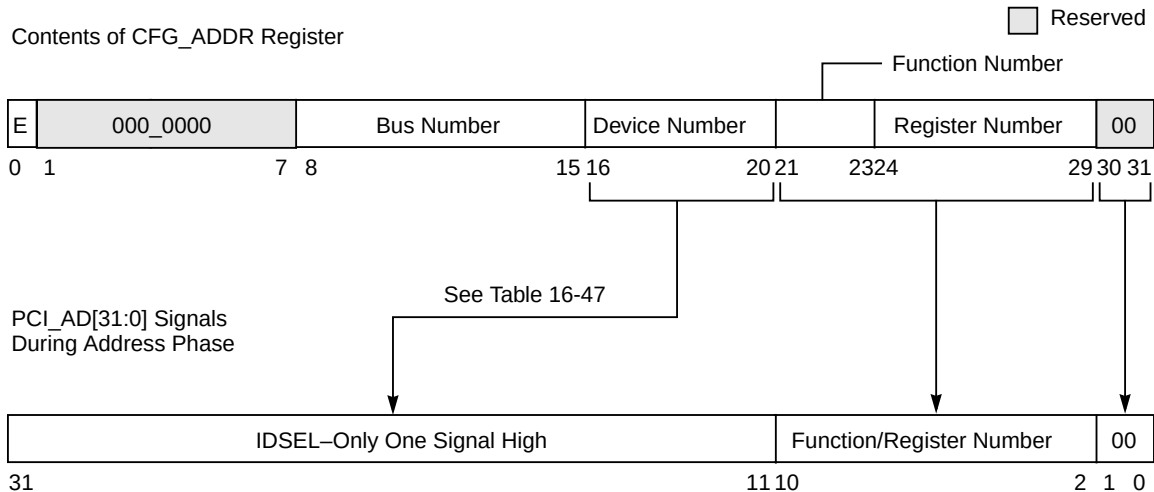


Figure 16-59. Type 0 Configuration Translation

For type 0 configuration cycles, the MPC8540 translates the device number field of the CFG_ADDR register into a unique PCI_IDSEL signal for up to 21 different devices. Each device connects its PCI_IDSEL input to one of the PCI_AD[31:11] signals. For type 0 configuration cycles, the MPC8540 translates the device number to PCI_IDSEL as shown in Table 16-47.

**Table 16-47. Type 0 Configuration—
Device Number to PCI_IDSEL Translation**

Device Number		PCI_IDSEL
Binary	Decimal	
0b0_0000–0b0_1001	0–9	—
0b0_1010	10	AD31
0b0_1011	11	AD11
0b0_1100	12	AD12
0b0_1101	13	AD13
0b0_1110	14	AD14
0b0_1111	15	AD15
0b1_0000	16	AD16
0b1_0001	17	AD17
0b1_0010	18	AD18
0b1_0011	19	AD19
0b1_0100	20	AD20
0b1_0101	21	AD21
0b1_0110	22	AD22

**Table 16-47. Type 0 Configuration—
Device Number to PCI_IDSEL Translation (continued)**

Device Number		PCI_IDSEL
Binary	Decimal	
0b1_0111	23	AD23
0b1_1000	24	AD24
0b1_1001	25	AD25
0b1_1010	26	AD26
0b1_1011	27	AD27
0b1_1100	28	AD28
0b1_1101	29	AD29
0b1_1110	30	AD30
0b1_1111 ¹	31	—

¹A device number of all ones indicates a PCI special-cycle or interrupt-acknowledge transaction.

For type 0 translations, the function number and register number fields are copied without modification onto the PCI_AD[10:2] signals during the address phase. The PCI_AD[1:0] signals are driven to 0b00 during the address phase for type 0 configuration cycles. The MPC8540 implements address stepping on configuration cycles so that the target's PCI_IDSEL, which is connected directly to one of the PCI_AD lines, reaches a stable value. This means that a valid address and command are driven on the PCI_AD[31:0] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals one clock before the assertion of $\overline{\text{PCI\_FRAME}}$.

16.4.2.11.4 Type 1 Configuration Translation

For type 1 translations, the MPC8540 copies the 30 high-order bits of the CFG_ADDR register (without modification) onto the PCI_AD[31:2] signals during the address phase. The MPC8540 automatically translates PCI_AD[1:0] into 0b01 during the address phase to indicate a type 1 configuration cycle.

16.4.2.12 Other Bus Transactions

There are two other PCI transactions that the MPC8540 supports—interrupt acknowledge and special cycles. As an initiator, the MPC8540 may initiate both interrupt acknowledge and special-cycle transactions; however, as a target, the MPC8540 ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the CFG_ADDR and CFG_DATA registers described in Section 16.4.2.11.2, “Accessing the PCI Configuration Space.”

16.4.2.12.1 Interrupt-Acknowledge Transactions

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read operation implicitly addressed to the system interrupt controller. Note that the PCI interrupt-acknowledge command does not address the MPC8540 PIC processor interrupt-acknowledge register and does not return the interrupt vector address from the PIC unit. See Chapter 10, “Programmable Interrupt Controller,” for more information about the PIC unit.

When the MPC8540 detects a read to the CFG_DATA register, it checks the enable flag and the device number in the CFG_ADDR register. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1_1111), the function number is all ones (0b111), and the register number is zero (0b00_0000), then the MPC8540 performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the MPC8540 performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

The address phase contains no valid information other than the interrupt-acknowledge command ($\text{PCI\_C}/\overline{\text{BE}}[3:0] = 0b0000$). Although there is no explicit address, $\text{PCI\_AD}[63:0]$ are driven to a stable state, and parity is generated. Only one device (the system interrupt controller) on the PCI bus should respond to the interrupt-acknowledge command by asserting $\overline{\text{PCI\_DEVSEL}}$. All other devices on the bus should ignore the interrupt-acknowledge command. As stated previously, the MPC8540 PIC unit does not respond to PCI interrupt-acknowledge commands.

During the data phase, the responding device returns the interrupt vector on $\text{PCI\_AD}[63:0]$ when $\overline{\text{PCI\_TRDY}}$ is asserted. The size of the interrupt vector returned is indicated by the value driven on the $\text{PCI\_C}/\overline{\text{BE}}[7:0]$ signals.

The MPC8540 also provides a direct way to generate PCI interrupt-acknowledge transactions. Reads from PCI/X INT_ACK at offset 0x0_8008 generate PCI interrupt-acknowledge transactions. Note that processor writes to these addresses do nothing.

16.4.2.12.2 Special-Cycle Transactions

The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. The special-cycle command contains no explicit destination address but is broadcast to all PCI agents.

When the MPC8540 detects a write to the CFG_DATA register, it checks the enable flag and the device number in the CFG_ADDR register. If the enable bit is set, the bus number corresponds to the local PCI bus (bus number = 0x00), the device number is all ones (0b1_1111), the function number is all ones (0b111), and the register number is zero (0b00_0000), then the MPC8540 performs a special-cycle transaction on the local PCI bus.

Functional Description

If the bus number indicates a nonlocal PCI bus, the MPC8540 performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

Aside from the special-cycle command ($\text{PCI\_C}/\overline{\text{BE}}[3:0] = 0b0001$) the address phase contains no other valid information. Although there is no explicit address, $\text{PCI\_AD}[63:0]$ are driven to a stable state, and parity is generated. During the data phase, $\text{PCI\_AD}[63:0]$ contain the special-cycle message and an optional data field. The special-cycle message is encoded on the 16 least-significant bits ($\text{PCI\_AD}[15:0]$); the optional data field is encoded on the most-significant 16 lines ($\text{PCI\_AD}[31:16]$). The special-cycle message encodings are assigned by the PCI SIG steering committee. The current list of defined encodings are provided in Table 16-48.

Table 16-48. Special-Cycle Message Encodings

PCI_AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 Architecture-Specific
0x0003–0xFFFF	—

Note that the MPC8540 does not automatically issue a special-cycle message when it enters any of its power-saving modes. It is the responsibility of software to issue the appropriate special-cycle message, if needed.

Each receiving agent must determine whether the special-cycle message is applicable to itself. Assertion of $\overline{\text{PCI\_DEVSEL}}$ in response to a special-cycle command is not necessary. The initiator of the special-cycle transaction can insert wait states but since there is no specific target, the special-cycle message and optional data field are valid on the first clock $\overline{\text{PCI\_IRDY}}$ is asserted. All special-cycle transactions are terminated by master-abort; however, the master-abort bit in the initiator's status register is not set for special-cycle terminations.

16.4.2.13 PCI Error Functions

PCI provides for parity and other system errors to be detected and reported. The PCI command register provides for selective enabling of specific PCI error detection. The PCI status register provides PCI error reporting. This section describes generation and detection of parity and error reporting for the PCI bus.

16.4.2.13.1 PCI Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate even parity; that is, the number of ones on the $\text{PCI\_AD}[31:0]$, $\text{PCI\_C}/\overline{\text{BE}}[3:0]$, and PCI_PAR signals all sum to an even number and

the number of ones on the PCI_AD[63:33], PCI_C/ $\overline{\text{BE}}$ [7:4], and PCI_PAR64 signals all sum to an even number.

Parity provides a way to determine, on each transaction, if the initiator successfully addressed the target and transferred valid data. The PCI_C/ $\overline{\text{BE}}$ [7:4] and PCI_C/ $\overline{\text{BE}}$ [3:0] signals are included in the parity calculation to ensure that the correct bus command is performed (during the address phase) and correct data is transferred (during the data phase). The agent responsible for driving the bus must also drive even parity on the PAR and PCI_PAR64 signals one clock cycle after a valid address phase or valid data transfer, as shown in Figure 16-60.

During the address and data phases, parity covers all 64 address/data signals and 8 command/byte enable signals, regardless of whether all lines carry meaningful information. Byte lanes not actually transferring data must contain stable (albeit meaningless) data and are included in parity calculation. During configuration, special-cycle, or interrupt-acknowledge commands; some address lines are not defined, but are driven to stable values and are included in parity calculation.

Agents that support parity checking must set the detected parity error bit in the PCI status register when a parity error is detected. Any additional response to a parity error is controlled by the parity error response bit in the PCI command register. If the parity error response bit is cleared, the agent ignores all parity errors.

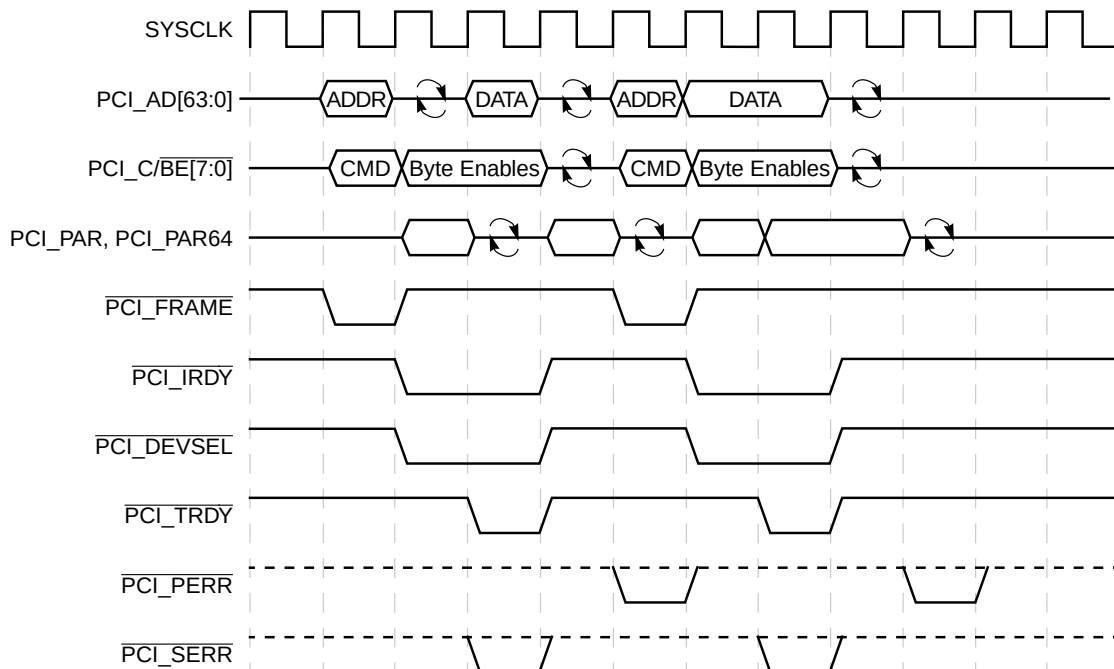


Figure 16-60. PCI Parity Operation

16.4.2.13.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors—`PCI_PERR` and `PCI_SERR`. The `PCI_PERR` signal is used exclusively to report data parity errors on all transactions except special cycles. The `PCI_SERR` signal is used for other error signaling including address parity errors and data parity errors on special-cycle transactions; it may also be used to signal other system errors.

16.5 Initialization/Application Information

16.5.1 Power-On Reset Configuration Modes

The PCI/X block can power-on in three modes: host mode, agent mode and agent config lock mode. Certain bits in the configuration registers are set differently according to the POR (power-on reset) mode. Also, certain configuration bits have different implications when compared with past Motorola parts and PCI implementations.

The affected configuration bits are defined in Table 16-49.

Table 16-49. Affected Configuration Register Bits for POR

Register Bit	Register Description	Register Location
MEM_ACC (memory access)	Controls the acknowledgement of inbound memory transactions. If set to 0, all inbound memory accesses (including accesses to PCSRBAR space) end in a master abort. This bit is independent of host or agent mode.	PCI/X configuration register[0x04] - Bit 1
BUS_MAS (bus master)	Controls whether the device can master a transaction on the PCI/X bus. If set to 0, the device can not master a transaction. This bit is independent of host or agent mode.	PCI/x configuration Register[0x04] - Bit 2
ACL (agent config lock)	Valid only in agent mode. It controls the acknowledgement of inbound configuration accesses. If set, all inbound configuration accesses are retried. If cleared, inbound configuration accesses are acknowledged. In host mode all inbound configuration accesses end in master aborts.	PCI/X bus function register[0x44] - Bit 5
PAH	This bit determines whether the device is in agent mode or host mode. If set to 0, the device is in host mode.	PCI/X bus function register[0x44] - Bit 0

The POR reset values for the above discussed bits are described in Table 16-50.

Table 16-50. Power-On Reset Values for Affected Configuration Bits

Config Bit	Setting in Host Mode	Setting in Agent Mode	Setting in Agent Config Lock Mode
MEM_ACC	0	0	0
BUS_MAS	1	0	0

Table 16-50. Power-On Reset Values for Affected Configuration Bits

Config Bit	Setting in Host Mode	Setting in Agent Mode	Setting in Agent Config Lock Mode
ACL	X	0	1
PAH	0	1	1

16.5.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). The ACL bit is a don't care. The part comes up with the ability to master transactions on the PCI bus, however in order to acknowledge memory transactions, MEM_ACC has to be set to 1.

Upon POR, the part can not be switched from host to agent mode.

16.5.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. However the part can not acknowledge inbound memory accesses or master transactions on the PCI bus until the appropriate configuration bits have been set.

Upon POR, the part can not be switched from agent to host mode.

16.5.1.3 Agent Config Lock Mode

When the device powers up in agent config lock mode, it retries inbound configuration accesses until the ACL bit is cleared to 0. Also, the part can not acknowledge inbound memory accesses or master transactions on the PCI bus until the appropriate configuration bits have been set. This setting is very similar to host mode other than the behavior caused by the ACL bit.

Upon POR, the part can not be switched from agent to host mode.

Outbound PCI or PCI-X transactions that cross a window boundary are not protected. No interrupt is generated nor is an error reported.

Inbound PCI-X read requests should not cross a window boundary. Although an error is reported when this happens, the transaction is allowed to make forward progress.

16.5.2 Nonposted Writes in PCI-X

The PCI-X block may pipeline I/O write and configuration Write accesses to a target that signals a split response. If full serialization of these non-posted writes is required, then the processor should chase each write transaction with a read of the same target, followed by proper barrier instructions to guarantee completion of the read before continuing with the

subsequent write. The method guarantees completion of the write and serialization of a series of writes.

16.5.3 Bridging between RapidIO and PCI/PCI-X

In order for bridging to work properly between RapidIO and the PCI/PCI-X block, the rules set forth in the RapidIO Interoperability Specification must be followed.

In addition, the NWRITE_R (write with response) transaction type should not be used to target the PCI/PCI-X interface on the MPC8540. The MPC8540 may return a response to the RapidIO interface before the transaction is completed to the PCI/PCI-X target. If the functionality of a RapidIO NWRITE_R is desired, the software should use an NWRITE followed by an NREAD to the same address.

Chapter 17

RapidIO Interface

17.1 Introduction

The RapidIO interface conforms to Revision 1.2 of the Parallel *RapidIO Interconnect Specification*.

Figure 17-1 depicts that the RapidIO controller is partitioned into inbound and outbound blocks, and that these blocks are further partitioned into three implementation layers: physical, protocol and logical. These implementation layers do not precisely correlate with the layers described in the *RapidIO Interconnect Specification*. The physical and protocol layers primarily address the portion of the *RapidIO Interconnect Specification* titled “Physical layer 8/16 LP-EP.” The logical layer refers primarily to the logical portions of the *RapidIO Interconnect Specification*. The physical layer operates at the RapidIO interface applied frequency and data is sampled at twice this frequency. The protocol and logical layers operate at the system frequency.

The RapidIO controller has a RapidIO interface, a system core interface, those signals that connect to the application in which RapidIO is instantiated, are divided into several groups. These groups include the RapidIO request, response, ACK done, and data transfer interfaces; the systemcore request, response and data transfer interfaces; the register access interface, which is used to access the programming model registers; the interrupt interface, the systems functions interface, and the performance monitor interface.

The RapidIO interface requires extra process-specific circuitry for receive clock recovery and for transmit clock and data alignment.

8 double data rate (DDR) data bits are received and registered through the RapidIO inbound interface. 64 bits are accumulated in the asynchronous boundary FIFO and transferred across the asynchronous boundary into the protocol layer. At the protocol layer, the 64 bits are split into packet header, packet data, and control symbol information fields. Once the packet header information arrives in the inbound packet assembly logic, packets are stripped of cyclic redundancy check (CRC) information and other transport information not required by the logical layer. CRC is checked at the end of a packet. Packet header information and packet data are then passed to the inbound logical layer. The header is interpreted by the logical layer and the transaction is stored in the inbound queue if space is available. The data is held in the inbound data queue until it is transferred out of RapidIO

and into the system. Some control symbols (ACKs and link responses, for example) are passed to the outbound block for processing.

On the outbound side, requests are received from the system at the logical layer and queued in the outbound queue. Transactions from the outbound queue are scheduled according to priority and may be steered to the misalign block or directly to the protocol layer if global arbitration is won. Transactions whose address and data payload are not aligned according to the RapidIO alignment requirements are sent to the misalign block for processing. The response queue handles responses from the system for previously-received inbound requests and schedules them as transactions out of the response queue according to priority.

Of all transactions from the outbound queue, misalign block, and the response queue, one is prioritized subject to global arbitration to be sent to the protocol layer. At the protocol layer, the packet CRC is performed. The protocol layer logic chooses to transmit either control symbols or packets on the outbound RapidIO interface. The packets and control symbols are then stored in the asynchronous boundary FIFO. The physical layer logic then reads 32-bit quantities from the FIFO and transmits them on the outbound RapidIO interface as 8-bit symbols. While there exist no packet or control symbols to send, the physical layer sends idle symbols. Idle symbols may also be inserted in a packet if a throttle request is received on the inbound interface.

The following sections provide greater detail on the organization and operation of RapidIO functionality. These sections are organized as follows:

- Section 17.2, “External Signal Description,” describes the interface signal groups, their meanings, and timing relationships.
- Section 17.3, “Memory Map/Register Definition,” describes the configuration registers, runtime registers, and debug registers that constitute the programming model.
- Section 17.4.1, “RapidIO Transaction, Packet, and Control Symbol Summary” describes the functional organization and operation of the design.
- Section 17.5, “RapidIO Functionality”

17.1.1 RapidIO Terminology

This section describes terminology used through this chapter while describing RapidIO functionality.

ACK	Acknowledge control symbol
CAR	Capability register
CRC	Cyclic redundancy check
CSR	Command and status register
Domain	A logically associated group of processing elements

Doorbell	A port on a device that generates an interrupt to a processor
Double Word	An 8-byte quantity(aligned on 8-byte boundaries)
Header	Part of a packet that contains all of the information about the packet excluding any data
ID	IDentifier, the name of a processing element on the RapidIO interconnect
Logical layer	Handles all logic transformations of packets to go between the protocol layer and the physical layer
Mailbox	Dedicated hardware that receives a message in the message passing interfacePhysical layer. The physical handling of the transport of packets over the RIO interface
Priority	The relative importance of a transaction or packet; in most systems a higher priority transaction or packet is serviced or transmitted before a lower priority transaction.
Protocol layer	The handling of packets at the protocol level on the RIO interface
Receiver	The destination of a packet on the RapidIO interface, also referred to as a destination or a target
Source	The originator of a packet on the RapidIO interconnect
Symbol	One beat of a packet appearing on the interface pins
Target	The destination of a packet on the RapidIO interface, also referred to as a destination or a receiver
Word	A 4-byte quantity(aligned on 4-byte boundaries)

17.1.2 Block Diagram

Figure 17-1 is a block diagram of the RapidIO module.

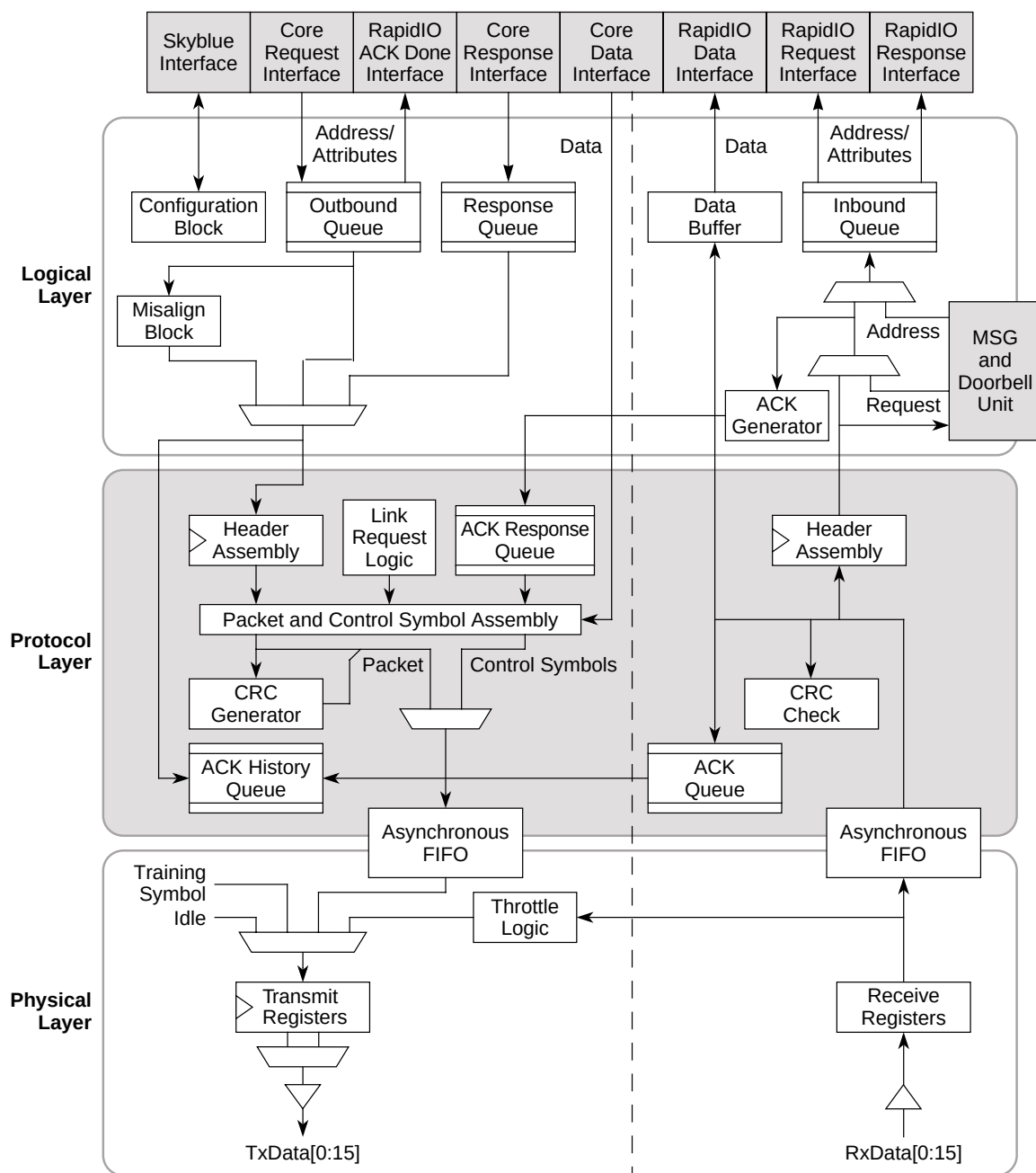


Figure 17-1. RapidIO High Level Block Diagram

17.1.3 RapidIO Interface Features

This section describes the RapidIO interface features.

17.1.3.1 Supported Features

RapidIO supports the following features of the RapidIO specification:

- 8-bit interface as defined by the 8/16 LP-EP physical layer specification
- Support for non-coherent I/O transactions
- A message-passing programming model
- CRC (CCITT-CRC16) based error detection as described in the 8/16 LP-EP physical specifications
- 256-byte data payloads
- Packet pacing/retry capability
- Only the small size transport information field (tt = 0b00)
- Link validation, error recovery, training, time-out support as required by the physical layer specification
- RapidIO error injection

17.1.3.2 Features Not Implemented

The following portions of the RapidIO specification are not supported:

- No support for large size transport information field (tt = 0b01)
- No support for tod-sync control symbols (these are treated as idle control symbols)
- No support for ATOMIC swap and ATOMIC test-and-swap transactions

17.1.4 RapidIO Modes of Operation

This section describes the RapidIO modes of operation.

17.1.4.1 Transmit Clock-Select Mode

The RapidIO transmit clock can be chosen from either an internal source or from the RapidIO receive clock. This feature is implemented externally to RapidIO, most likely in the device's clock-generation module.

17.1.4.2 CRC Checking Modes

RapidIO offers following modes for packet CRC checking as shown in Table 17-1.

Table 17-1. CRC Checking Modes

Mode	crc_enable	crc_bypass
CRC enabled with hardware-based recovery	0	0
CRC checking disabled	1	Don't Care

17.1.4.3 Error Checking Disable Mode

Error checking can be enabled or disabled by setting a bit in the port control CSR. This mode disables all RapidIO transmission error checking. Device behavior is undefined if error checking is disabled and an error condition occurs.

17.1.4.4 Output Port Enable Mode

The output port can be disabled by setting a bit in the port control CSR. In this mode, RapidIO does not issue any packets of its own, except in response to maintenance packets.

17.1.4.5 Output Port Driver Disable Mode

This mode automatically forces the output port enable signal low. The mode is controlled by a bit in the port control CSR.

17.1.4.6 Accept-All Mode

This mode is controlled by a bit in the RapidIO configuration register. While this mode is enabled, all packets are accepted without regards to the target ID.

Table 17-2. Accept All Mode Behavior

Accept All	Device Behavior for Target ID $\neq$ Device ID
0	Error
1	Accept and process the packet without regard to the target ID

17.1.4.7 Packet Response Time-Out Disable Mode

The RapidIO response time-out counters can be disabled by setting a bit in the RapidIO notification/fatal error detect register. See Section 17.3.3.3.1, “Port Notification/Fatal Error Detect Register (PNFEDR).”

17.1.4.8 Link Response Time-Out Disable Mode

The RapidIO link time-out counters can be disabled by setting a bit in the RapidIO notification/fatal error detect register. See Section 17.3.3.3.1, “Port Notification/Fatal Error Detect Register (PNFEDR).”

17.2 External Signal Description

This section describes the signal names of the RapidIO block, their meanings, and the timing relationships that exist among the signals.

17.2.1 Overview

The RapidIO block implements the functionality described in the *RapidIO Interconnect Specification* for an 8-bit parallel interface with separate transmit and receive ports as shown in Figure 17-2.

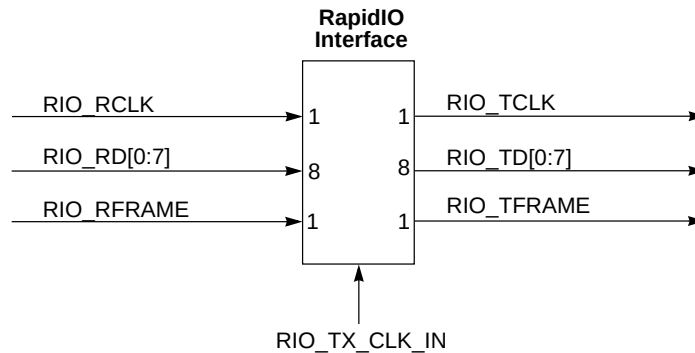


Figure 17-2. RapidIO Interface Signals

The RapidIO signals are implemented as differential pairs of signals. Table 17-3 lists the external RapidIO signals implemented on this interface.

Table 17-3. RapidIO Interface Signals

Signal Name	Function	Pins	I/O
RIO_RCLK	Port receive clock	1	I
$\overline{\text{RIO\_RCLK}}$	Port receive clock, inverted (differential pair)	1	I
RIO_RD [0:7]	Port receive data	8	I
$\overline{\text{RIO\_RD [0:7]}}$	Port receive data, inverted (differential pair)	8	I
RIO_RFRAME	Port receive framing signal	1	I
$\overline{\text{RIO\_RFRAME}}$	Port receive framing signal, inverted (differential pair)	1	I
RIO_TCLK	Port transmit clock	1	O
$\overline{\text{RIO\_TCLK}}$	Port transmit clock, inverted (differential pair)	1	O
RIO_TD[0:7]	Port transmit data	8	O
$\overline{\text{RIO\_TD[0:7]}}$	Port transmit data, inverted (differential pair)	8	O
RIO_TFRAME	Port transmit framing signal	1	O
$\overline{\text{RIO\_TFRAME}}$	Port transmit framing signal, inverted (differential pair)	1	O
RIO_TX_CLK_IN	Port transmit clock in	1	I
$\overline{\text{RIO\_TX\_CLK\_IN}}$	Port transmit clock in, inverted (differential pair)	1	I

17.2.2 Detailed Signal Descriptions

Table 17-4 describes the RapidIO interface signals in detail.

Table 17-4. RapidIO Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description
RIO_RCLK	I	Receive clock. The receive clock signal, RIO_RCLK, is a free-running input clock used to capture control and data signals. RIO_RCLK is used to capture the frame signal RIO_RFRAME and the data bus RIO_RD[0:7]. RIO_RCLK is the differential pair of RIO_RCLK.
		State Meaning Asserted/Negated—A clock event is determined as the rising or falling edge of RIO_RCLK.
		Timing Assertion/Negation—Free-running clock. See the <i>MPC8540 Hardware Specification</i> for specific timing information for this signal.
RIO_RD[0:7]	I	Receive data. The receive data bus RIO_RD[0:7] carries packet information as well as control symbols. Data is captured on both edges of the receive clock, RIO_RCLK. RIO_RD[0:7] is the differential pair of RIO_RD[0:7]
		State Meaning Asserted/Negated—Represents the data being received on the RapidIO interface.
		Timing RIO_RD[0:7] data is captured on both the rising and falling edges of the receive clock RIO_RCLK..
RIO_RFRAME	I	Receive frame. The receive frame signal RIO_RFRAME is a non return to zero (NRZ) input. It toggles at the beginning of a packet or control symbol. RIO_RFRAME is the differential pair of RIO_RFRAME.
		State Meaning Asserted/Negated—Assertion is signalled by a toggling of the signal. Toggling indicates a special packet framing event on the receive data (RIO_RD) pins.
		Timing RIO_RFRAME is sampled with respect to RIO_RCLK.
RIO_TCLK	O	Transmit clock out. The transmit clock RIO_TCLK is a free-running output clock which is launched in phase with the output data, RIO_TD, and the frame signal, RIO_TFRAME. RIO_TCLK should track the data bus RIO_TD[0:7] through all the associated routing (board, package, and so forth). RIO_TCLK is the differential pair of RIO_TCLK.
		State Meaning Asserted/Negated—A clock event in the transmit domain is determined as the rising or falling edge of this signal.
		Timing Assertion/Negation—RIO_TCLK is a free-running clock. Note that the RIO_TCLK is NOT phase aligned to the system clock.

Table 17-4. RapidIO Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
RIO_TD[0:7]	O	Transmit data. The transmit data bus RIO_TD[0:7] carries packet information as well as control symbols. The data bus is switched on both edges of RIO_TCLK. RIO_TD[0:7] is the differential pair of RIO_TD[0:7].
		State Meaning Asserted/Negated—Represents the data being transmitted on the RapidIO interface.
		Timing Assertion of RIO_TD is always performed with a fixed relationship to RIO_TCLK.
RIO_TFRAME	I	Transmit frame. The transmit frame signal RIO_TFRAME is a non return to zero (NRZ) output. It toggles at the beginning of a packet or control symbol. RIO_TFRAME_b is the differential pair of RIO_TFRAME.
		State Meaning Asserted/Negated—Assertion is signalled by a toggling of the signal. Toggling indicates a special packet framing event on the transmit data (RIO_TD[0:7]) pins.
		Timing RIO_TFRAME is generated with respect to RIO_TCLK.
RIO_TX_CLK_IN	I	Transmit clock in. The transmit clock in signal RIO_TX_CLK_IN is an externally generated reference clock for the RapidIO transmit domain. RIO_TX_CLK_IN is the differential pair of the RIO_TX_CLK_IN signal.
		State Meaning Asserted/Negated—
		Timing Assertion/Negation—

17.3 Memory Map/Register Definition

17.3.1 RapidIO Registers

The RapidIO memory map is shown in Table 17-5.

Table 17-5. RapidIO Module Memory Map

Offset	Register	Access	Reset	Section/Page
Architectural Registers				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0002_0002	17.3.2.1/17-14
0xC_0004	Device information capability register (DICAR)	R	0x8020_0010	17.3.2.2/17-14
0xC_0008	Assembly identity capability register (AIDCAR)	Special ¹	0x0000_0000	17.3.2.3/17-15
0xC_000C	Assembly information capability register (AICAR)	Special ¹	0x0000_0100	17.3.2.4/17-16
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE088_0009	17.3.2.5/17-16
0xC_0014	Switch port information capability register (SPICAR)	R	0x0000_0100	17.3.2.6/17-17
0xC_0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	17.3.2.7/17-18
0xC_001C	Destination operations capability register (DOCAR)	R	0x0600_FCF4	17.3.2.8/17-19
0xC_0040	Mailbox command and status register (MSR)	R	0x0000_0000	17.3.2.9/17-21

Table 17-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_0044	Port -Write and Doorbell Command and Status Register (PWDCSR)	R	0x0000_0020	17.3.2.10/17-22
0xC_004C	Processing Element Logical Layer Control Command and Status Register (PELLCCSR)	R	0x0000_0001	17.3.2.11/17-23
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	R/W	0x0000_0000	17.3.2.12/17-23
0xC_0060	Base Device ID Command and Status Register (BDIDCSR)	R/W	0x00[cfg]_0000	17.3.2.13/17-24
0xC_0068	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	Special ¹	0x0000_FFFF	17.3.2.14/17-25
0xC_006C	Component Tag Command and Status Register (CTCSR)	R/W	0x0000_0000	17.3.2.15/17-25
0xC_0100	8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)	R	0x0000_0002	17.3.2.16/17-26
0xC_0120	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	R/W	0xFFFF_FF00	17.3.2.17/17-27
0xC_0124	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	R/W	0xFFFF_FF00	17.3.2.18/17-27
0xC_013C	Port General Control Command and Status Register (PGCCSR)	R/W	0x[cfg]000_0000	17.3.2.19/17-28
0xC_0140	Port Link Maintenance Request Command and Status Register (PLMREQCSR)	R/W	0x0000_0000	17.3.2.19/17-28
0xC_0144	Port Link Maintenance Response Command and Status Register (PLMRESPCSR)	R	0x0000_0000	17.3.2.20/17-29
0xC_0148	Port Local Ackid Status Command and Status Register (PLASCSR)	Special ¹	0x0000_0000	17.3.2.21/17-29
0xC_0158	Port Error and Status Command and Status Register (PESCSR)	Special ¹	0x0000_0000	17.3.2.22/17-30
0xC_015C	Port Control Command and Status Register (PCCSR)	Special ¹	0x4400_0000	17.3.2.23/17-31
Implementation Registers				
0xD_0000	Configuration Register (CR)	R/W	0x0000_0001	17.3.3.1.1/17-33
0xD_0010	Port Configuration Register (PCR)	R/W	0x0000_0010	17.3.3.1.2/17-34
0xD_0014	Port Error Injection Register (PEIR)	Special ¹	0x0000_0000	17.3.3.1.3/17-35
ATMU Registers				
0xD_0C00	RapidIO Outbound Window Translation Address Register 0 (ROWTAR0)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C10	RapidIO Outbound Window Attributes Register 0 (ROWAR0)	R/W	0x0004_401F	17.3.3.2.3/17-39

Table 17-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0C20	RapidIO Outbound Window Translation Address Register 1 (ROWTAR1)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C28	RapidIO Outbound Window Base Address Register 1 (ROWBAR1)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0C30	RapidIO Outbound Window Attributes Register 1 (ROWAR1)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C40	RapidIO Outbound Window Translation Address Register 2 (ROWTAR2)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C48	RapidIO Outbound Window Base Address Register 2 (ROWBAR2)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0C50	RapidIO Outbound Window Attributes Register 2 (ROWAR2)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C60	RapidIO Outbound Window Translation Address Register 3 (ROWTAR3)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C68	RapidIO Outbound Window Base Address Register 3 (ROWBAR3)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0C70	RapidIO Outbound Window Attributes Register 3 (ROWAR3)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0C80	RapidIO Outbound Window Translation Address Register 4 (ROWTAR4)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0C88	RapidIO Outbound Window Base Address Register 4 (ROWBAR4)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0C90	RapidIO Outbound Window Attributes Register 4 (ROWAR4)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CA0	RapidIO Outbound Window Translation Address Register 5 (ROWTAR5)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CA8	RapidIO Outbound Window Base Address Register 5 (ROWBAR5)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0CB0	RapidIO Outbound Window Attributes Register 5 (ROWAR5)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CC0	RapidIO Outbound Window Translation Address Register 6 (ROWTAR6)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CC8	RapidIO Outbound Window Base Address Register 6 (ROWBAR6)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0CD0	RapidIO Outbound Window Attributes Register 6 (ROWAR6)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0CE0	RapidIO Outbound Window Translation Address Register 7 (ROWTAR7)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0CE8	RapidIO Outbound Window Base Address Register 7 (ROWBAR7)	R/W	0x0000_0000	17.3.3.2.2/17-38

Table 17-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0CF0	RapidIO Outbound Window Attributes Register 7 (ROWAR7)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0D00	RapidIO Outbound Window Translation Address Register 8 (ROWTAR8)	R/W	0x0000_0000	17.3.3.2.1/17-36
0xD_0D08	RapidIO Outbound Window Base Address Register 8 (ROWBAR8)	R/W	0x0000_0000	17.3.3.2.2/17-38
0xD_0D10	RapidIO Outbound Window Attributes Register 8 (ROWAR8)	R/W	0x0004_401F	17.3.3.2.3/17-39
0xD_0D60	RapidIO Inbound Window Translation Address Register 4 (RIWTAR4)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0D68	RapidIO Inbound Window Base Address Register 4 (RIWBAR4)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0D70	RapidIO Inbound Window Attributes Register 4 (RIWAR4)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0D80	RapidIO Inbound Window Translation Address Register 3 (RIWTAR3)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0D88	RapidIO Inbound Window Base Address Register 3 (RIWBAR3)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0D90	RapidIO Inbound Window Attributes Register 3 (RIWAR3)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DA0	RapidIO Inbound Window Translation Address Register 2 (RIWTAR2)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DA8	RapidIO Inbound Window Base Address Register 2 (RIWBAR2)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0DB0	RapidIO Inbound Window Attributes Register 2 (RIWAR2)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DC0	RapidIO Inbound Window Translation Address Register 1 (RIWTAR1)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DC8	RapidIO Inbound Window Base Address Register 1 (RIWBAR1)	R/W	0x0000_0000	17.3.3.2.5/17-42
0xD_0DD0	RapidIO Inbound Window Attributes Register1 (RIWAR1)	R/W	0x0004_401F	17.3.3.2.6/17-42
0xD_0DE0	RapidIO Inbound Window Translation Address Register 0 (RIWTAR0)	R/W	0x0000_0000	17.3.3.2.4/17-41
0xD_0DF0	RapidIO Inbound Window Attributes Register 0 (RIWAR0)	R/W	0x0004_401F	17.3.3.2.6/17-42
Error Management Registers				
0xD_0E00	Port Notification/Fatal Error Detect Register (PNFEDR)	Special ¹	0x0000_0000	17.3.3.3.1/17-44
0xD_0E04	Port Notification/Fatal Error Disable Register (PNFEDiR)	R/W	0x0000_0000	17.3.3.3.2/17-47

Table 17-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0E08	Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)	R/W	0x0000_0000	17.3.3.3.3/17-49
0xD_0E0C	Port Error Control Register (PECR)	Special ¹	0x0000_0000	17.3.3.3.4/17-50
0xD_0E10	Port Error Packet/Control Symbol Register 0 (PEPCSR0)	Special ¹	0x0000_0000	17.3.3.3.5/17-51
0xD_0E14	Port Error Packet Register 1 (PEPR1)	Special ¹	0x0000_0000	17.3.3.3.6/17-52
0xD_0E18	Port Error Packet Register 2 (PEPR2)	Special ¹	0x0000_0000	17.3.3.3.16/17-58
0xD_0E20	Port Recoverable Error Detect Register (PREDR)	Special ¹	0x0000_0000	17.3.3.3.23/17-62
0xD_0E28	Port Error Recovery Threshold Register (PERTR)	Special ¹	0x00FF_0000	17.3.3.3.24/17-65
0xD_0E2C	Port Retry Threshold Register (PRTR)	Special ¹	0x00FF_0000	17.3.3.3.25/17-66
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	Outbound Mode Register (OMR)	R/W	0x0000_0000	17.3.4.1.1/17-67
0xD_1004	Outbound Status Register (OSR)	Special	0x0000_0000	17.3.4.1.2/17-69
0xD_100C	Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)	R/W	0x0000_0000	17.3.4.1.3/17-70
0xD_1014	Outbound Source Address Register (OSAR)	R/W	0x0000_0000	17.3.4.1.4/17-71
0xD_1018	Outbound Destination Port Register (ODPR)	R/W	0x0000_0000	17.3.4.1.5/17-72
0xD_101C	Outbound Destination Attributes Register (ODATR)	R/W	0x0006_0000	17.3.4.1.6/17-72
0xD_1020	Outbound Double-word Count Register (ODCR)	R/W	0x0000_0000	17.3.4.1.7/17-73
0xD_1028	Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)	R/W	0x0000_0000	17.3.4.1.8/17-87
RapidIO Inbound Message Registers				
0xD_1060	Inbound Mailbox Mode Register (IMR)	R/W	0x0000_0000	17.3.4.2.1/17-75
0xD_1064	Inbound Mailbox Status Register (ISR)	Special	0x0000_0000	17.3.4.2.2/17-77
0xD_106C	Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)	R/W	0x0000_0000	17.3.4.2.3/17-91
0xD_1074	Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)	R/W	0x0000_0000	17.3.4.2.4/17-79
RapidIO Doorbell Registers				
0xD_1460	Doorbell Mode Register (DMR)	R/W	0x0000_0000	17.3.4.3.1/17-80
0xD_1464	Doorbell Status Register (DSR)	Special	0x0000_0000	17.3.4.3.2/17-82
0xD_146C	Doorbell Queue Dequeue Pointer Address Register (DQDPAR)	R/W	0x0000_0000	17.3.4.3.3/17-83
0xD_1474	Doorbell Queue Enqueue Pointer Address Register (DQEPAR)	R/W	0x0000_0000	17.3.4.3.4/17-99

Table 17-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
RapidIO Port-Write Registers				
0xD_14E0	Port-write Mode Register (PWMR)	R/W	0x0000_0000	17.3.4.4.1/17-85
0xD_14E4	Port-write Status Register (PWSR)	Special	0x0000_0000	17.3.4.4.2/17-86
0xD_14EC	Port-write Queue Base Address Register (PWQBAR)	R/W	0x0000_0000	17.3.4.4.3/17-87

¹ The bits of special registers are cleared by setting them.

17.3.2 Architectural Registers

Following are the architectural registers of the RapidIO module.

17.3.2.1 Device Identity Capability Register (DIDCAR)

The DIDCAR, shown in Figure 17-3, is a read only register.

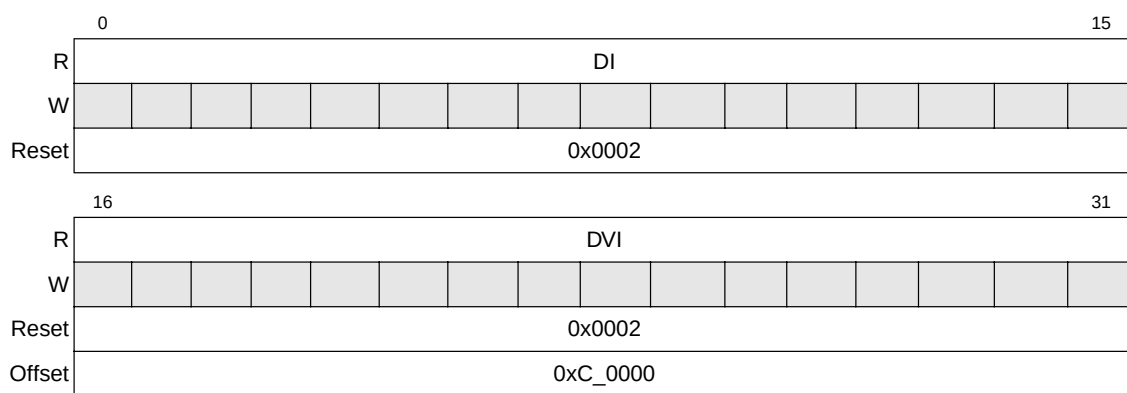
**Figure 17-3. Device Identity Capability Register (DIDCAR)**

Table 17-6 describes the bits of DIDCAR.

Table 17-6. DIDCAR Field Descriptions

Bits	Name	Description
0–15	DI	Device identity field ... 0x0002—MPC8540
16–31	DVI	Device vendor identity field 0x0002—Motorola

17.3.2.2 Device Information Capability Register (DICAR)

The DICAR, shown in Figure 17-4, is a read only register. This register identifies the revision level of the device. The value is assigned and managed by the vendor specified by

the device vendor identity DIDCAR[DVI] field. This register represents a copy of the device’s SVR.

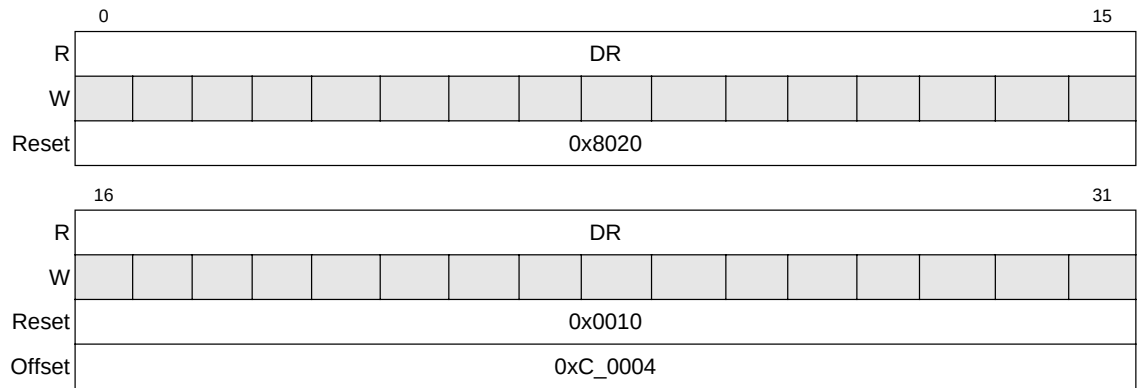


Figure 17-4. Device Information Capability Register (DICAR)

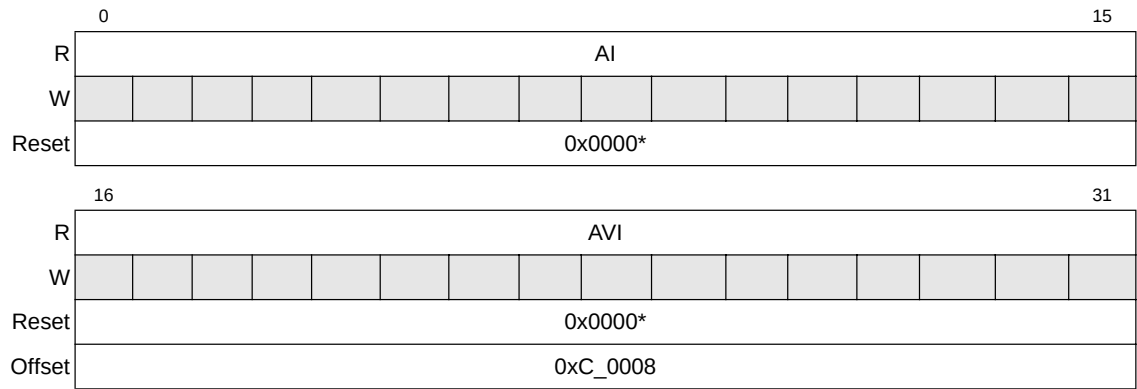
Table 17-7 describes the bits of DICAR.

Table 17-7. DICAR Field Descriptions

Bits	Name	Description
0–31	DR	Device revision

17.3.2.3 Assembly Identity Capability Register (AIDCAR)

The AIDCAR, shown in Figure 17-5, is a read only register.



* Alternate value may be read from serial ROM

Figure 17-5. Assembly Identity Capability Register (AIDCAR)

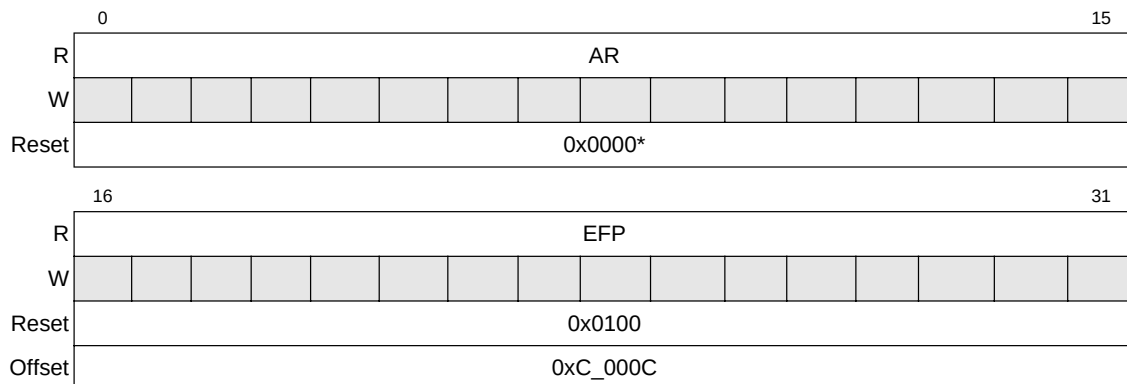
Table 17-8 describes the bits of AIDCAR.

Table 17-8. AIDCAR Field Descriptions

Bits	Name	Description
0–15	AI	Assembly identity field
16–31	AVI	Assembly vendor identity field

17.3.2.4 Assembly Information Capability Register (AICAR)

The AICAR, shown in Figure 17-6, is a read only register.



* Alternate value may be read from serial ROM

Figure 17-6. Assembly Information Capability Register (AICAR)

Table 17-9 describes the bits of AICAR.

Table 17-9. AICAR Field Descriptions

Bits	Name	Description
0–15	AR	Assembly revision field
16–31	EFP	Extended features pointer field

17.3.2.5 Processing Element Features Capability Register (PEFCAR)

The PEFCAR, shown in Figure 17-7, is a read only register.

	0	1	2	3	4		7	8	9	10	11	12	13	14	15	
R	B	M	P	S	0	0	0	0	M0	M1	M2	M3	D	0	0	0
W																
Reset	0xE088															

	16										26	27	28	29	31	
R	0	0	0	0	0	0	0	0	0	0	0	CTLS	EF	EAS		
W																
Reset	0x0009															
Offset	0xC_0010															

Figure 17-7. Processing Element Features Capability Register (PEFCAR)

Table 17-10 describes the bits of PEFCAR.

Table 17-10. PEFCAR Field Descriptions

Bits	Name	Description
0	B	Bridge. The RapidIO controller has a DRAM interface
1	M	Memory. The RapidIO controller has local address space
2	P	The RapidIO controller has a local processor
3	S	Switch. The RapidIO controller is not a switch.
4–7	—	Reserved
8	M0	Mailbox 0. The RapidIO controller supports mailbox 0
9	M1	Mailbox 1. The RapidIO controller supports mailbox 1
10	M2	Mailbox 2. The RapidIO controller supports mailbox 2
11	M3	Mailbox 3. The RapidIO controller supports mailbox 3
12	D	Doorbell. The RapidIO controller supports inbound doorbells
13–26	—	Reserved
27	CTLS	The RapidIO controller does not support the large common transport route field
28	EF	Extended features pointer is valid
29–31	EAS	Extended addressing support. The RapidIO controller supports 34-bit local addresses

17.3.2.6 Switch Port Information Capability Register (SPICAR)

The SPICAR, shown in Figure 17-8, is a read-only register.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0x0000															

	16							23	24	31						
R	PT								PN							
W																
Reset	0x01								0x00							
Offset	0xC_0014															

Figure 17-8. Switch Port Information Capability Register (SPICAR)

Table 17-11 describes the bits of SPICAR.

Table 17-11. SPICAR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–23	PT	The RapidIO controller has one RapidIO port
24–31	PN	This register was read from RapidIO port number * read value port number dependent

17.3.2.7 Source Operations Capability Register (SOCAR)

The SOCAR, shown in Figure 17-9, is a read only register. It indicates the RapidIO transaction types that this device can initiate.

	0	1	2	3	4	5	6	7	8	9	10					15
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	0	0	0	0	0	0
R	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
W																
Reset	0x0600															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NR	NW	SW	NWR	M	D	0	ATS	AI	AD	AS	AC	0	PW	0	0
W																
Reset	0xFCF0															
Offset	0xC_0018															

Figure 17-9. Source Operations Capability Register (SOCAR)

Table 17-12 describes the bits of SOCAR.

Table 17-12. SOCAR Field Descriptions

Bits	Name	Description
0	R	The RapidIO controller can initiate a Read operation
1	IR	The RapidIO controller can initiate an IRead operation
2	RO	The RapidIO controller can initiate a Read_To_Own operation
3	DCI	The RapidIO controller can initiate a Dkill operation
4	C	The RapidIO controller can initiate a Castout operation
5	F	The RapidIO controller can initiate a Flush operation
6	IOR	The RapidIO controller can initiate an I/O-Read operation
7	ICI	The RapidIO controller can initiate an IKill operation
8	TIE	The RapidIO controller can initiate a TLBIE operation
9	TIES	The RapidIO controller can initiate a TLBSYNC operation
10–15	—	Reserved
16	NR	The RapidIO controller can initiate a Nread operation
17	NW	The RapidIO controller can initiate a Nwrite operation
18	SW	The RapidIO controller can initiate an Swrite operation
19	NWR	The RapidIO controller can initiate a Nwrite_R operation
20	M	The RapidIO controller can initiate a Message operation
21	D	The RapidIO controller can initiate a Doorbell operation
22	—	Reserved
23	ATS	The RapidIO controller cannot initiate an Atomic-Test-and-Swap operation
24	AI	The RapidIO controller can initiate an Atomic-Inc operation
25	AD	The RapidIO controller can initiate an Atomic-Dec operation
26	AS	The RapidIO controller can initiate an Atomic-Set operation
27	AC	The RapidIO controller can initiate an Atomic-Clr operation
28	—	Reserved
29	PW	The RapidIO controller can initiate a Port-Write operation
30–31	—	Reserved

17.3.2.8 Destination Operations Capability Register (DOCAR)

The DOCAR, shown in Figure 17-10, is a read only register. It indicates the RapidIO transaction types that this device can service.

Memory Map/Register Definition

	0	1	2	3	4	5	6	7	8	9	10					15
	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	0	0	0	0	0	0
R	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
W																
Reset	0x0600															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	NR	NW	SW	NWR	M	D	0	ATS	AI	AD	AS	AC	0	PW	0	0
W																
Reset	0xFCF4															
Offset	0xC_001C															

Figure 17-10. Destination Operations Capability Register (DOCAR)

Table 17-13 describes the bits of DOCAR.

Table 17-13. DOCAR Field Descriptions

Bits	Name	Description
0	R	The RapidIO controller can service a Read operation
1	IR	The RapidIO controller can service an IRead operation
2	RO	The RapidIO controller can service a Read_To_Own operation
3	DCI	The RapidIO controller can service a Dkill operation
4	C	The RapidIO controller can service a Castout operation
5	F	The RapidIO controller can service a Flush operation
6	IOR	The RapidIO controller can service an I/O-Read operation
7	ICI	The RapidIO controller can service an IKill operation
8	TIE	The RapidIO controller can service a TLBIE operation
9	TIES	The RapidIO controller can service a TLBSYNC operation
10–15	—	Reserved
16	NR	The RapidIO controller can service a Nread operation
17	NW	The RapidIO controller can service a Nwrite operation
18	SW	The RapidIO controller can service an Swrite operation
19	NWR	The RapidIO controller can service a Nwrite_R operation
20	M	The RapidIO controller can service a Message operation
21	D	The RapidIO controller can service a Doorbell operation
22	—	Reserved
23	ATS	The RapidIO controller cannot service an Atomic-Test-and-Swap operation
24	AI	The RapidIO controller can service an Atomic-Inc operation

Table 17-13. DOCAR Field Descriptions (continued)

Bits	Name	Description
25	AD	The RapidIO controller can service an Atomic-Dec operation
26	AS	The RapidIO controller can service an Atomic-Set operation
27	AC	The RapidIO controller can service an Atomic-Clr operation
28	—	Reserved
29	PW	The RapidIO controller can service a Port-Write operation
30–31	—	Reserved

17.3.2.9 Mailbox Command and Status Register (MSR)

The MSR, shown in Figure 17-11, is a read-only register.

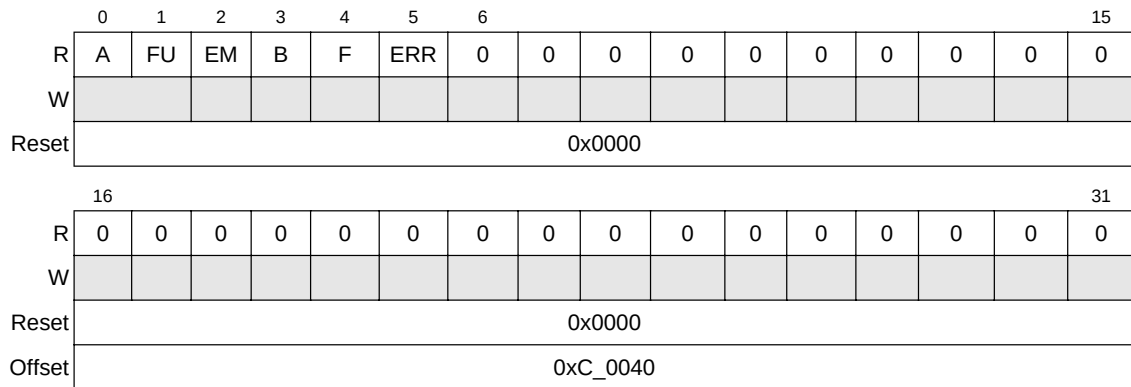
**Figure 17-11. Mailbox Status Register (MSR)**

Table 17-14 describes the bits of MSR.

Table 17-14. MSR Field Definitions

Bits	Name	Description
0	A	Available—Mailbox 0 is ready to accept messages
1	FU	Full—Mailbox 0 is full; new messages are retried
2	EM	Empty—Mailbox 0 has no outstanding messages
3	B	Busy—Mailbox 0 is busy processing a message
4	FA	Failure—Mailbox 0 has encountered an internal error
5	ERR	Erro—Mailbox 0 received an illegal message operation
6–31	—	Reserved

17.3.2.10 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 17-12, is a read only register. Details of this register can be found in the RapidIO 1.2 Specification.

	0	1	2	3	4	5	6									15
R	A	FU	EM	B	F	ERR	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															
	16							23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	PA	PFU	PEM	PB	PFA	PE	0	0
W																
Reset	0x0020															
Offset	0xC_0044															

Figure 17-12. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 17-15 describes the bits of PWDCSR.

Table 17-15. PWDCSR Field Descriptions

Bits	Name	Description
0	A	Available—Doorbell unit is ready to accept doorbells
1	FU	Full—Doorbell unit is full; new doorbells are retried
2	EM	Empty—Doorbell unit has no outstanding doorbells
3	B	Busy—Doorbell unit is busy processing a doorbell
4	FA	Failure—Doorbell unit has encountered an internal error
5	ERR	Error—Doorbell unit encountered an error condition
6–23	—	Reserved
24	PA	Write-port controller is initialized and ready to accept a write port transaction. If not available, all incoming port-write transactions are discarded.
25	PFU	Write-port hardware is full. All incoming port-write transactions are discarded.
26	PEM	Write-port hardware has no outstanding write port messages
27	PB	Write-port hardware is busy processing a write port message. Incoming port-write transactions are discarded.
28	PFA	Write-port hardware had an internal fault or error condition. This bit is cleared because no condition is identified matching this description.
29	PE	Write-port transaction is of illegal size (greater than 64 bytes)
30–31	—	Reserved

17.3.2.11 Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

The PELLCCSR, shown in Figure 17-13, is a read only register. Details of this register can be found in the RapidIO 1.2 Specification.

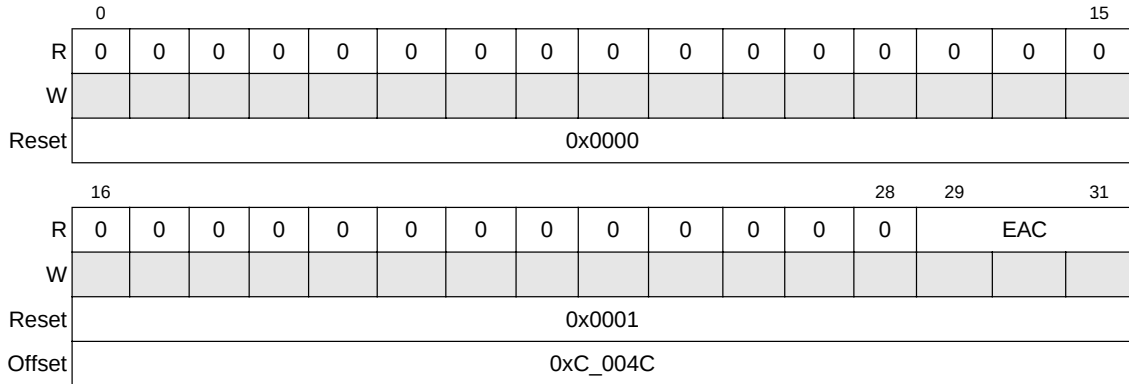


Figure 17-13. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 17-16 describes the bits of PELLCCSR.

Table 17-16. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. The RapidIO controller supports 34-bit addresses. (Read only)

17.3.2.12 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 17-14, contains the least significant 32 bits of the double-word address of the base of the device's register space. The address represents the base address for the device's register space.

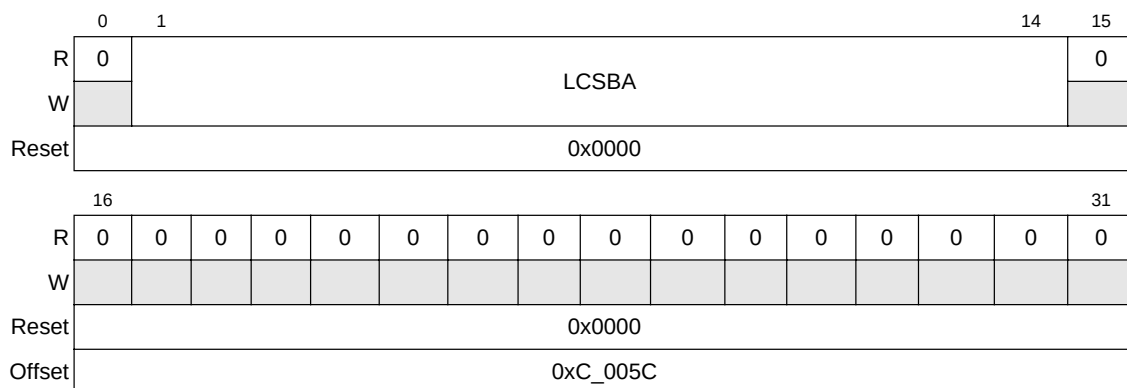


Figure 17-14. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 17-17 describes the bits of LCSBA1CSR.

Table 17-17. 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

17.3.2.13 Base Device ID Command and Status Register (BDIDCSR)

The BDIDCSR is shown in Figure 17-15.

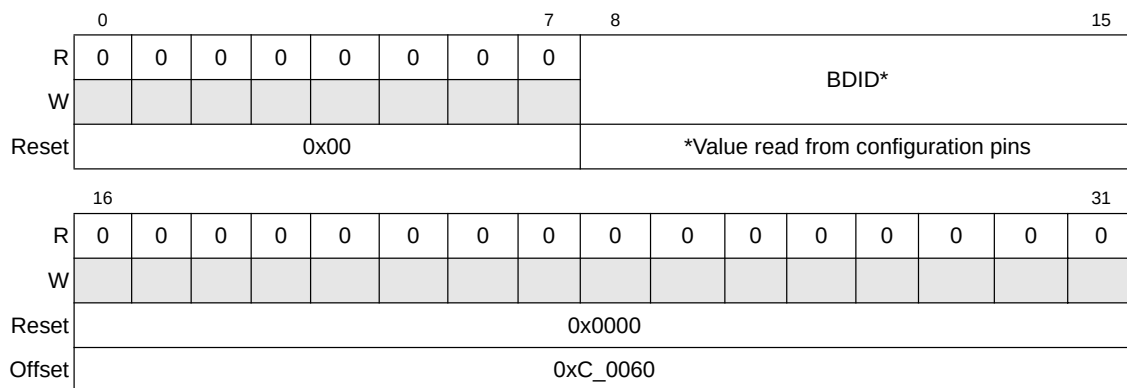


Figure 17-15. Base Device ID Command and Status Register (BDIDCSR)

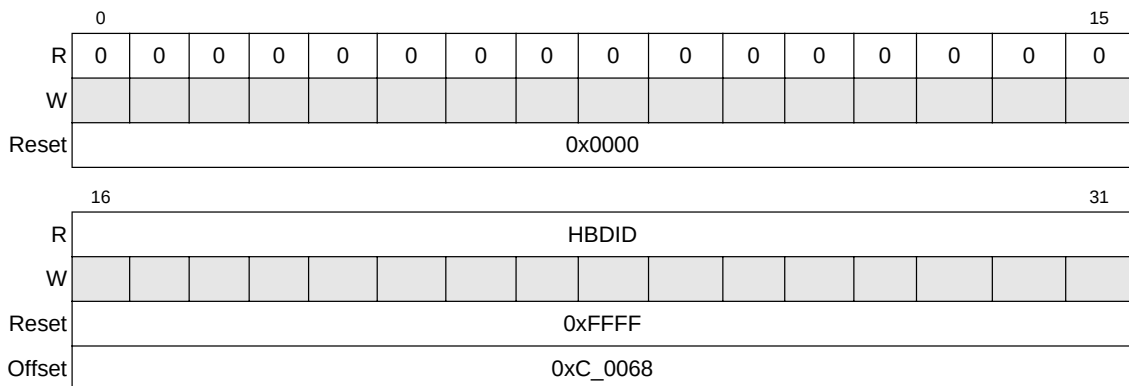
BDIDCSR fields are described in Table 17-18.

Table 17-18. BDIDCSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	BDID	Base device ID (RapidIO device ID). Value read from reset configuration pins.
16–31	—	Reserved

17.3.2.14 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

The host base device ID lock CSR contains the base device ID value for the processing element in the system that is responsible for initializing this processing element. HBDID is a write-once/resettable field which provides a lock function. After 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. The HBDIDLCSR is shown in Figure 17-16.

**Figure 17-16. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)**

HBDIDLCSR fields are described in Table 17-19.

Table 17-19. 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.

17.3.2.15 Component Tag Command and Status Register (CTCSR)

The CTCSR is shown in Figure 17-17.

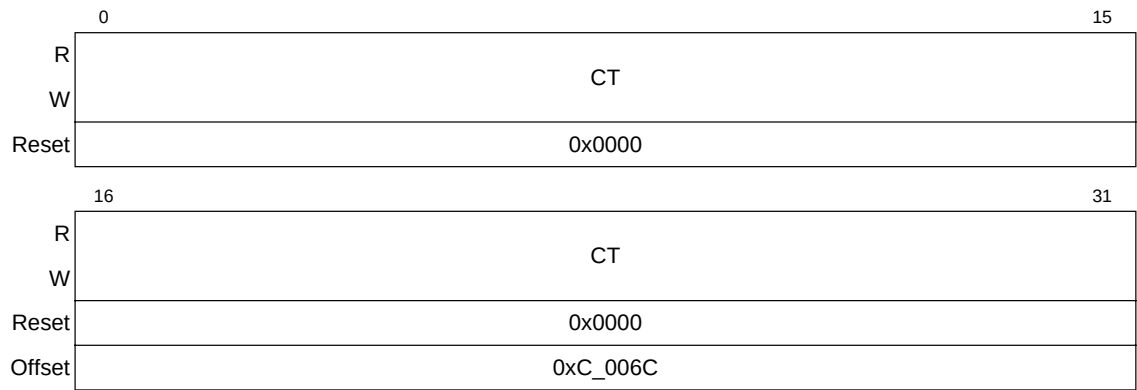


Figure 17-17. Component Tag Command and Status Register (CTCSR)

CTCSR fields are described in Table 17-20.

Table 17-20. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag

17.3.2.16 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

The PMBH0CSR is shown in Figure 17-18.

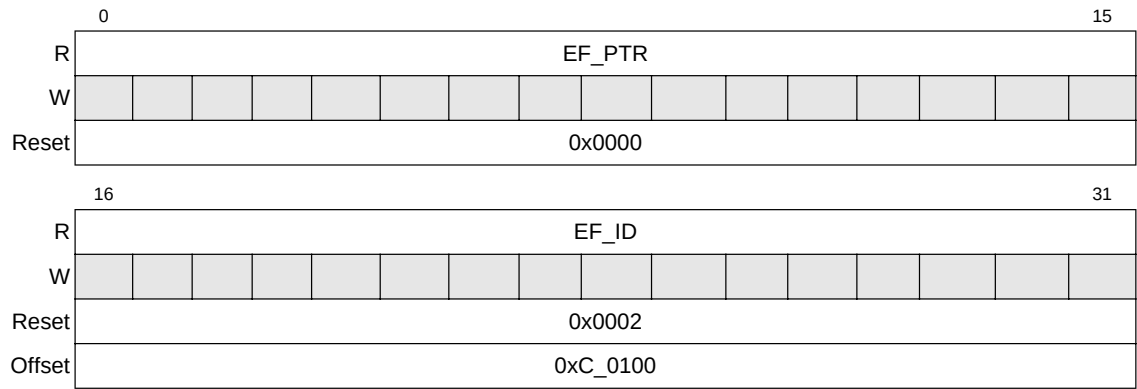


Figure 17-18. 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

PMBH0CSR fields are described in Table 17-21.

Table 17-21. PMBH0CSR Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

17.3.2.17 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The PLTOCCSR is shown in Figure 17-19.

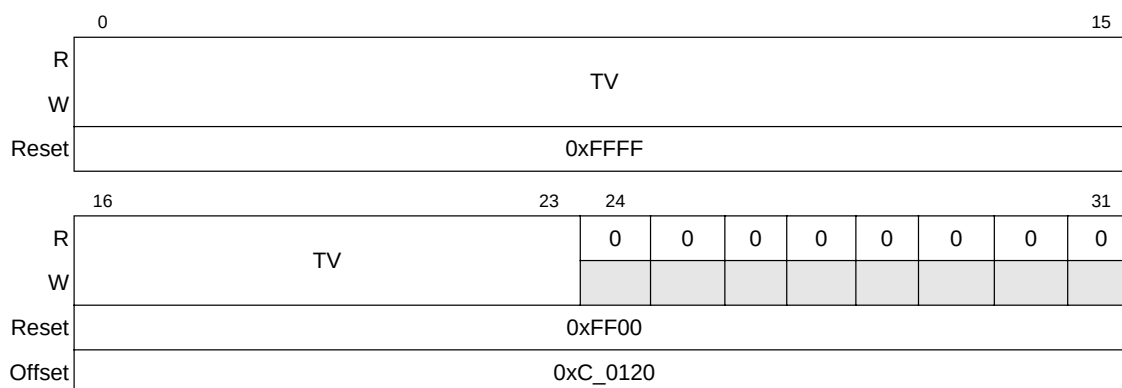


Figure 17-19. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

PLTOCCSR fields are described in Table 17-22.

Table 17-22. PLTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value
24–31	—	Reserved

17.3.2.18 Port Response Time-Out Control Command and Status Register (PRTTOCCSR)

The PRTTOCCSR is shown in Figure 17-20.

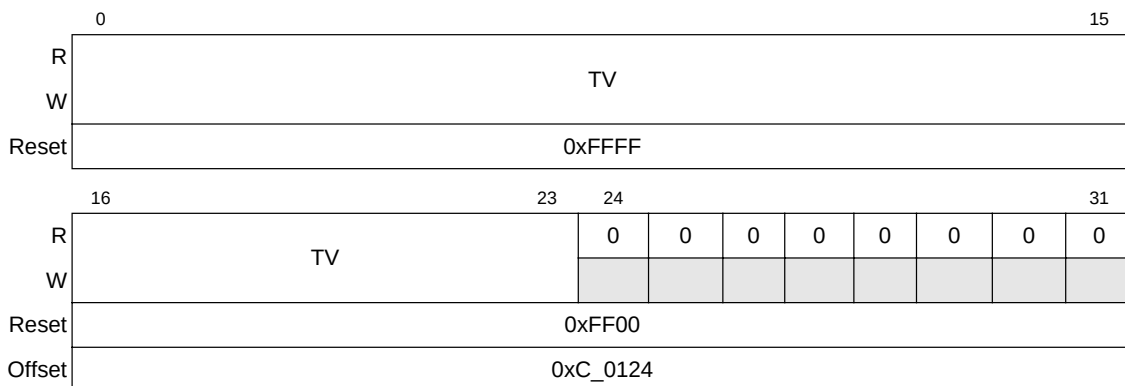


Figure 17-20. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR fields are described in Table 17-23.

Table 17-23. PRTOCCSR Field Descriptions

Bitws	Name	Description
0–23	TV	Time-out value
24–31	—	Reserved

17.3.2.19 Port General Control Command and Status Register (PGCCSR)

The PGCCSR is shown in Figure 17-21.

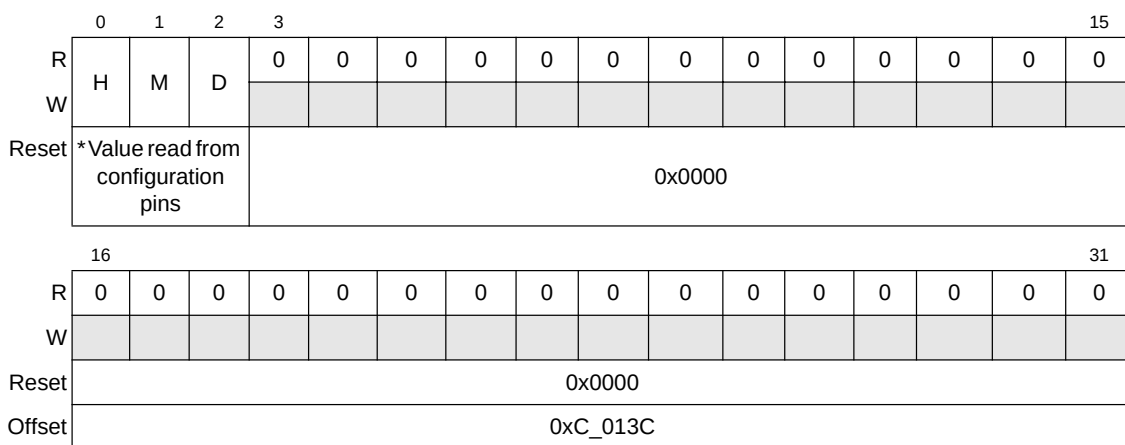


Figure 17-21. Port General Control Command and Status Register (PGCCSR)

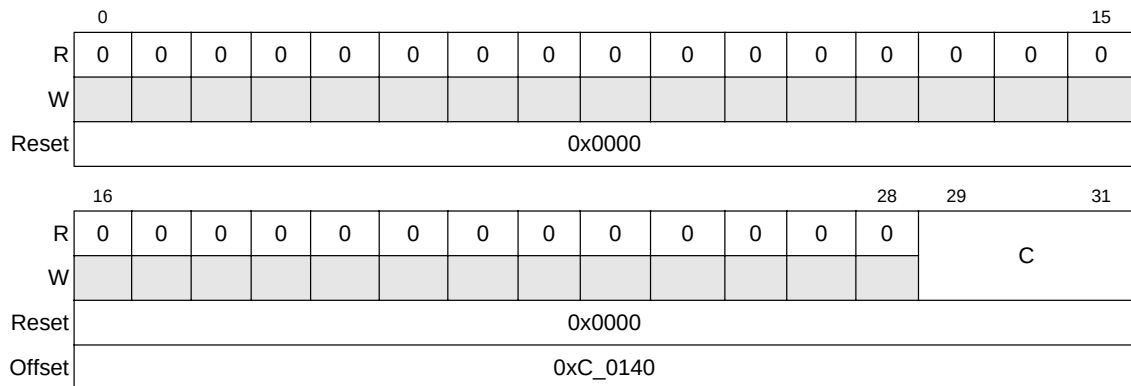
PGCCSR fields are described in Table 17-24.

Table 17-24. PGCCSR Field Descriptions

Bits	Name	Description
0	H	Contains the value read from configuration pins 0 Agent device 1 Host device
1	M	Contains the value read from configuration pins 0 Device is not enabled to issue requests into the system 1 Device is enabled to issue requests into the system
2	D	Contains value read from configuration pins 0 Device has not been discovered by system host 1 Device has been discovered by system host
3–31	—	Reserved

17.3.2.20 Port Link Maintenance Request Command and Status Register (PLMREQCSR)

The PLMREQCSR is shown in Figure 17-22.

**Figure 17-22. Port Link Maintenance Request Command and Status Register (PLMREQCSR)**

PLMREQCSR fields are described in Table 17-25.

Table 17-25. PLMREQCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	C	LINK_REQUEST command to send

17.3.2.21 Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

The PLMRESPCSR, shown in Figure 17-23, is a read only register.

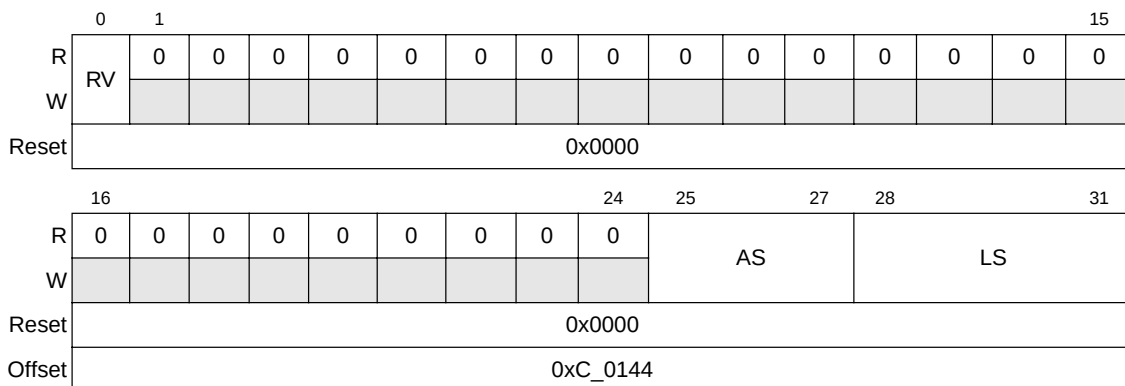


Figure 17-23. Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

PLMRESPCSR fields are described in Table 17-26.

Table 17-26. PLMRESPCSR Field Descriptions

Bits	Name	Description
0	RV	Response valid: status fields are valid (clear on read)
0–24	—	Reserved
25–27	AS	AckID_status field from LINK_RESPONSE
28–31	LS	Link_status field from LINK_RESPONSE

17.3.2.22 Port Local AckID Status Command and Status Register (PLASCSR)

The PLASCSR is shown in Figure 17-24.

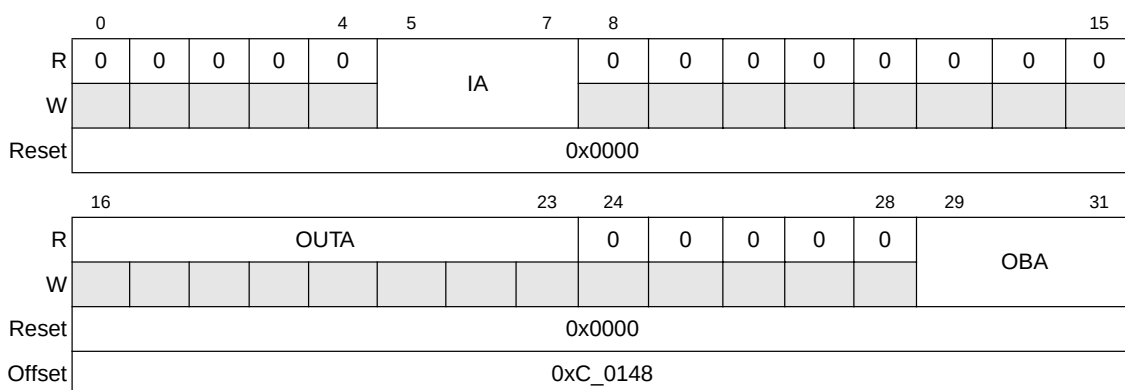


Figure 17-24. Port Local AckID Status Command and Status Register (PLASCSR)

PLASCSR fields are described in Table 17-27.

Table 17-27. PLASCSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IA	Input port next expected ackID value
8–15	—	Reserved
16–23	OUTA	Outstanding unacknowledged ackIDs. A set bit indicates that the corresponding ackID value is used a packet to an attached device but a corresponding acknowledge control symbol has not been received. (read-only field)
24–28	—	Reserved
29–31	OBA	Outbound AckID: output port next transmitted ackID value. Software writing this value can force re-transmission of outstanding unacknowledged packets in order to manually implement error recovery. This field can only be written by software when the RapidIO port is enabled.

17.3.2.23 Port Error and Status Command and Status Register (PESCSR)

The PESCSR is shown in Figure 17-25.

	0										10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	ORE	OR	ORS	OEE	OES	
W																
Reset	0x0000															

	16	20				21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	IRS	IEE	IES	0	0	0	0	PP	PE	PO	PU
W																
Reset	0x0000															
Offset	0xC_0158															

Figure 17-25. Port Error and Status Command and Status Register (PESCSR)

PESCSR fields are described in Table 17-28.

Table 17-28. PESCSR Field Descriptions

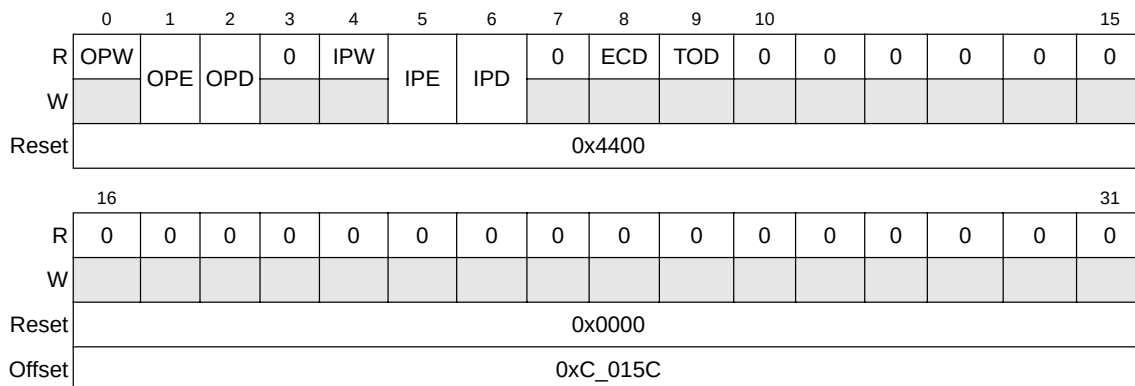
Bits	Name	Description
0–10	—	Reserved
11	ORE	Output port encountered a retry condition. This bit is set when bit 13 is set. (Bit reset, write 1 clear)
12	OR	Output port received a packet retry and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read-only field)
13	ORS	Output port is stopped due to a retry (read-only field)
14	OEE	Output port encountered an error. This bit is set when bit 15 is set. (Bit reset, write 1 clear)

Table 17-28. PESCSR Field Descriptions (continued)

Bits	Name	Description
15	OES	Output port is stopped due to a transmission error (read-only field)
16–20	—	Reserved
21	IRS	Input port is stopped due to a retry (read-only field)
22	IEE	Input port encountered a transmission error. This bit is set when bit 23 is set. (Bit reset, write 1 clear)
23	IES	Input port is stopped due to a transmission error (read-only field)
24–27	—	Reserved
28	PP	The input clock pin is toggling (there is a device attached) (read-only field). Note: this bit is set shortly after reset, due to asynchronous interface crossing and clock detection delay.
29	PE	Input port encountered an unrecoverable error or Output port failed error recovery (Bit reset, write 1 clear).
30	PO	Input and output ports are initialized and can communicate with the adjacent device (read-only).
31	PU	Input and output ports are not initialized and are in training mode (read-only). Note: this bit is set shortly after reset, due to asynchronous interface crossing delay.

17.3.2.24 Port Control Command and Status Register (PCCSR)

The PCCSR is shown in Figure 17-26.

**Figure 17-26. Port Control Command and Status Register (PCCSR)**

PCCSR fields are described in Table 17-29.

Table 17-29. PCCSR Field Descriptions

Bits	Name	Description
0	OPW	Operating width of the output port (read-only field) 0 Port operating in 8-bit mode 1 Reserved
1	OPE	Output port transmit enable 0 Port is stopped and not enabled to issue any packets except to respond to maintenance packets 1 Port is enabled to issue any packets This bit value must equal the value of PCCSR[IPE] in order for the RapidIO controller to function properly.
2	OPD	Output port drivers are disabled On the RapidIO controller, if this bit is set both the input receivers and output drivers are disabled.
3	—	Reserved
4	IPW	Operating width of the input port (read-only field) 0 Port operating in 8-bit mode 1 Reserved
5	IPE	Input port receive enable 0 Port is stopped and only enabled to receive maintenance requests packets. Other requests return packet-not-accepted control symbols with "General error" cause to force an error condition to be signalled by the sending device 1 Port is enabled to issue any packets This bit value must equal the value of PCCSR[OPE] in order for the RapidIO controller to function properly.
6	IPD	Input port receivers are disabled On the RapidIO controller, if this bit is set both the input receivers and output drivers are disabled.
7	—	Reserved
8	ECD	Error Checking Disable: This bit disables all RapidIO transmission error checking 0 Error checking and recovery is enabled 1 Error checking and recovery is disabled Device behavior while error checking is disabled and an error condition occurs is undefined
9	TOD	Hard-wired to a logic 0. The RapidIO controller treats a TOD control symbol as an IDLE control symbol.
10–31	—	Reserved

17.3.3 Implementation Registers

Following are the RapidIO implementation registers.

17.3.3.1 General Registers

17.3.3.1.1 Configuration Register (CR)

The CR is shown in Figure 17-27.

	0														15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															

	16													29		30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0		PTE	AA	
W																	
Reset	0x0001																
Offset	0xD_0000																

Figure 17-27. Configuration Register (CR)

CR fields are described in Table 17-30.

Table 17-30. CR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	PTE	Pass-through mode enable 0 Packets whose target ID doesn't match this device's ID are not sent out on the other RapidIO port; error occurs if not in "accept all mode". 1 Packets whose target ID doesn't match this device's ID are sent out on the other RapidIO port.
31	AA	Accept all 0 Detect an error upon receiving a packet with a target ID not equal to its own. 1 All packets are accepted without checking the target ID. NOTE: This bitfield must initially be cleared by the user as part of initialization.

17.3.3.1.2 Port Configuration Register (PCR)

The PCR is shown in Figure 17-28.

	0															15		
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																		
Reset	0x0000																	
	16						22				23	24	25	26	27	28	31	
R	0	0	0	0	0	0	0	RTD	LTD	QD	CB	CCE	0	0	0	0		
W																		
Reset	0x0010																	
Offset	0xD_0010																	

Figure 17-28. Port Configuration Register (PCR)

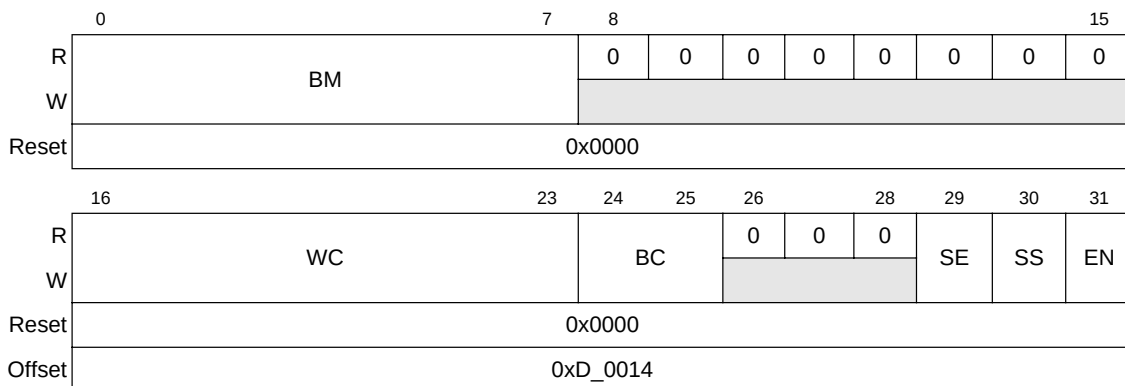
PCR fields are described in Table 17-31.

Table 17-31. PCR Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23	RTD	Response timeout disable 0 Response timeout function is enabled 1 Response timeout function is disabled
24	LTD	Link timeout disable 0 Link timeout function is enabled 1 Link timeout function is disabled
25	QD	Attached device doesn't support the QIN/QOUT quiesce protocol, or the port is not connected to another functioning device 0 Attached device supports RapidIO port quiescing 1 Attached device doesn't support RapidIO port quiescing, or no connected functioning device. If the RapidIO controller is put to sleep or soft stopped, then a QIN (which is an acknowledgement for QOUT) is implied.
26	CB	Data forwarding before CRC checked 0 Data is not forwarded to the processor before is checked 1 Data is forwarded to the process before CRC is checked.
27	CCE	CRC checking enable 0 No CRC is checked on received packets 1 CRC is checked on received packets.
28–31	—	Reserved

17.3.3.1.3 Port Error Injection Register (PEIR)

The RapidIO error injection registers offer a simple mechanism to inject errors into an outbound packet or a control symbol. The RapidIO error injection register for the target port is loaded with control information for injecting an error in a packet stream or control symbol. After the register is written, the hardware starts to count up to the programmed number of words and, upon reaching that number, an error is injected in the specified byte using the byte mask. The error is injected once and the enable bit is then cleared. After setting the enable bit, software can monitor the injection of the error by polling the register and observing the enable bit being cleared. The PEIR is shown in Figure 17-29.

**Figure 17-29. Port Error Injection Register (PEIR)**

PEIR fields are described in Table 17-32.

Table 17-32. PEIR Field Descriptions

Bits	Name	Description
0–7	BM	Byte mask to indicate which bit of the byte to inject error on.
8–15	—	Reserved
16–23	WC	Number of words to count before injecting error.
24–25	BC	Byte within the specified word count to injecting error on.
26–28	—	Reserved
29	SE	If the Enable bit is set, an error continues to be injected until the enable bit is cleared by software.
30	SS	Packet or control symbol selection 0 Inject error in a packet 1 Inject error in a control symbol
31	EN	Enabled to start counting and inject error at the specified word count, byte within the word, and bit within the byte. Hardware injects error and clears this bit.

17.3.3.2 ATMU Registers

Although, the RapidIO Specification allows for 48 and 64-bit addresses, the MPC8540 implementation only allows for 34-bit RapidIO addresses. ATMU window misses use the window 0 register set by default. Overlapping window hits 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 4G.

17.3.3.2.1 RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR_n)

The RapidIO outbound window translation address registers (ROWTARs) 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. The target ID is formed from the top eight bits of the translated address. The ROWTARs are shown in Figure 17-30.

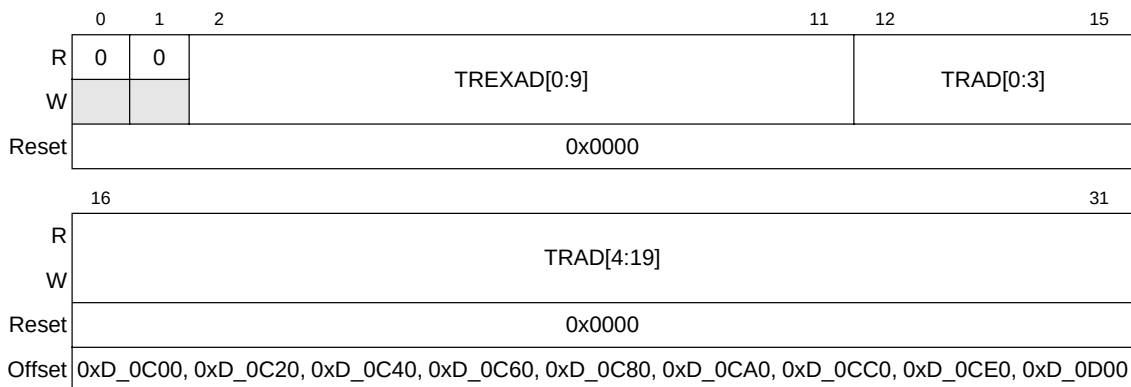


Figure 17-30. RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR_n)

ROWTARs fields are described in Table 17-33.

Table 17-33. ROWTAR_n Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–11	TREXAD	Translation extended address of outbound window The target ID for outbound requests is formed from bits 0–7 of the translation extended address. Bits 8–9 of the translation extended address correspond to bits 0–1 of the 34-bit RapidIO address and is reserved for maintenance transactions.
12–31	TRAD	Translation address of outbound window System address that represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. The translation address corresponds to bits 2–21 of the 34-bit RapidIO address.

The ROWTAR register description for maintenance transactions is described in Figure 17-31 below.

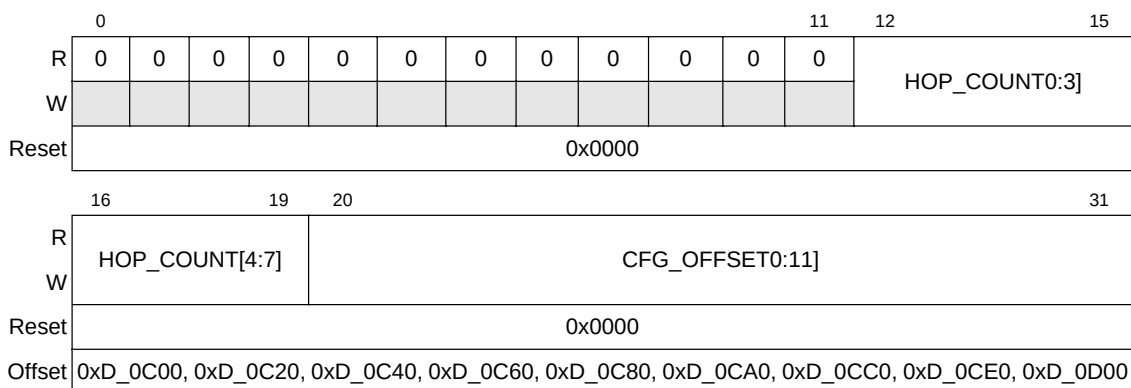


Figure 17-31. RapidIO Outbound Window Translation Address Registers 0–8 for Maintenance Transactions (ROWTAR_n)

Table 17-34 shows the field descriptions for ROWTARs for maintenance transactions. For maintenance transactions, the hop count is formed from bits 12–19 of the ROWTAR register. Similarly, the upper 12 bits of the maintenance offset is formed from bits 20–31 of the ROWTAR register. The rest of the maintenance offset is formed from the untranslated address.

Table 17-34. ROWTAR_n Maintenance Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–19	HOP_COUNT	Hop count of maintenance transaction
20–31	CFG_OFFSET	Upper 12 bits of maintenance offset. The lower 9 bits of the 21-bit RapidIO maintenance offset are formed from the untranslated address.

17.3.3.2.2 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR_n)

The RapidIO outbound window translation address registers (ROWBAR_ns) 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 window 0 register set. Window 0 ROWBAR is reserved. The ROWBARs are shown in Figure 17-32.

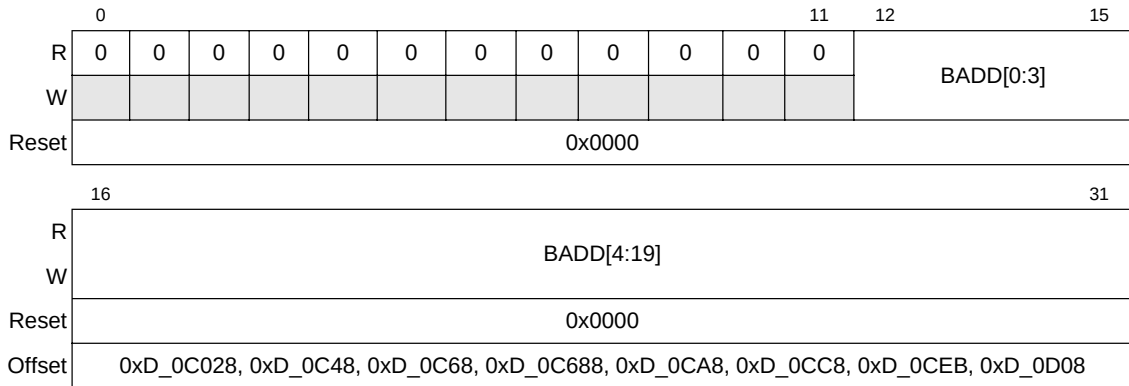


Figure 17-32. RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR_n)

ROWBAR_ns fields are described in Table 17-35.

Table 17-35. ROWBAR_n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	BADD	base address of outbound window Source address that 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 0–19 of a 32-bit address.

17.3.3.2.3 RapidIO Outbound Window Attributes Registers 0–8 (ROWAR_n)

The RapidIO outbound window attributes registers (ROWARs) define the window size to translate and other attributes for the translation. 4 Gbytes is the largest window size allowed. The ROWARs are shown in Figure 17-33.

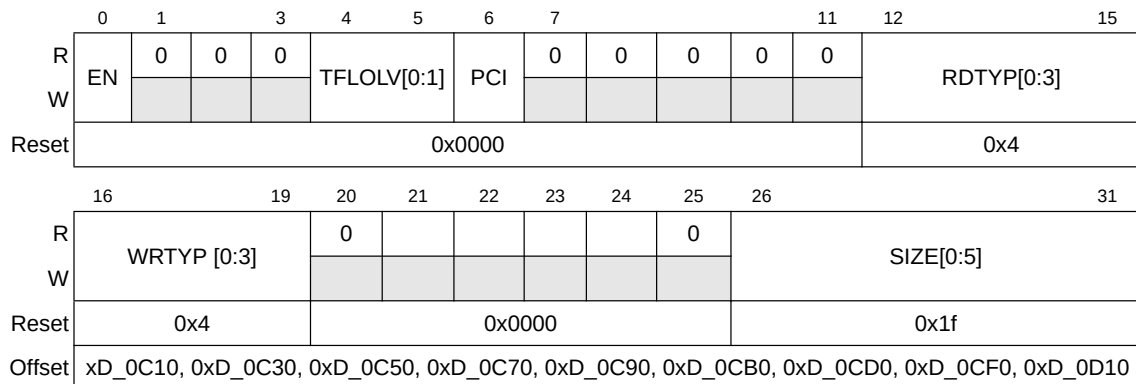


Figure 17-33. RapidIO Outbound Window Attributes Registers 0–8 (ROWAR_n)

ROWARs fields are described in Table 17-36.

Table 17-36. ROWAR_n Field Descriptions

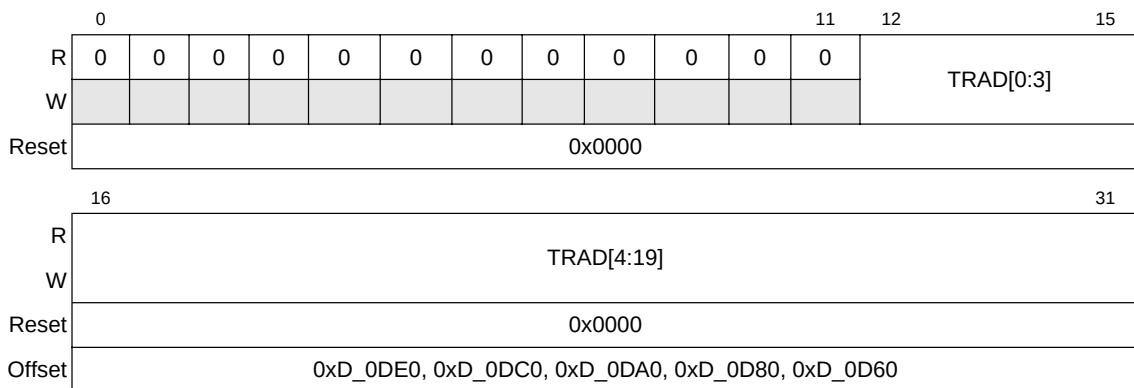
Bits	Name	Description
0	EN	Enable. Enables this window address translation. Note that for Window 0 ROWAR the enable bit is hardwired to 1. 0 Address translation disabled 1 Address translation enabled
1–3	—	Reserved
4–5	TFLOLV	Transaction flow level—priority of transaction 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6	PCI	PCI. Follow PCI transaction ordering rules. 0 Do not follow PCI transaction ordering rules. 1 Follow PCI transaction ordering rules (elevate write priority one level over reads).
7–11	—	Reserved
12–15	RDTYP	Read transaction type. Transaction type to run on RapidIO interface if access is a read. 0000–0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101–0110 Reserved 0111 MAINTENANCE read 1000–011 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear Note: See Section 17.7.1.2, “Outbound Special Transactions and Requirements,” on page 17-113 for more information on MAINTENANCE transactions.
16–19	WRTYP	Write transaction type. Transaction type to run on RapidIO interface if access is a write. 0000 Reserved 0001 FLUSH_WITH_DATA 0010 DOORBELL 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000–1111 Reserved Note: See Section 17.7.1.2, “Outbound Special Transactions and Requirements,” on page 17-113 for more information on MAINTENANCE and DOORBELL transactions.

Table 17-36. ROWAR_n Field Descriptions

Bits	Name	Description
20–25	—	Reserved
26–31	SIZE	Outbound window size. Outbound window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. The largest window size is 4 Gbytes. 00_0000–00_1010 Reserved 00_1011 4-Kbyte window size 00_1100 8-Kbyte window size 01_1111 4-Gbyte window size 10_0000–11_1111 = Reserved

17.3.3.2.4 RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR_n)

The RapidIO inbound window translation address registers (RIWTARs) select the starting addresses in the internal 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. The RIWTARs are shown in Figure 17-34.

**Figure 17-34. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR_n)**

RIWTARs fields are described in Table 17-37.

Table 17-37. RIWTAR_n Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TRAD	Translation address of inbound window. System address that represents the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 0–19 of the 32-bit address.

17.3.3.2.5 RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR_n)

The RapidIO inbound window translation address registers (RIWBARs) select the base address for the windows which are translated to an alternate system address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within the LCSBAR window or one of these spaces the transaction is forwarded through the window 0 register set. Window 0 RIWBAR is reserved. The RIWBARs are shown in Figure 17-35.

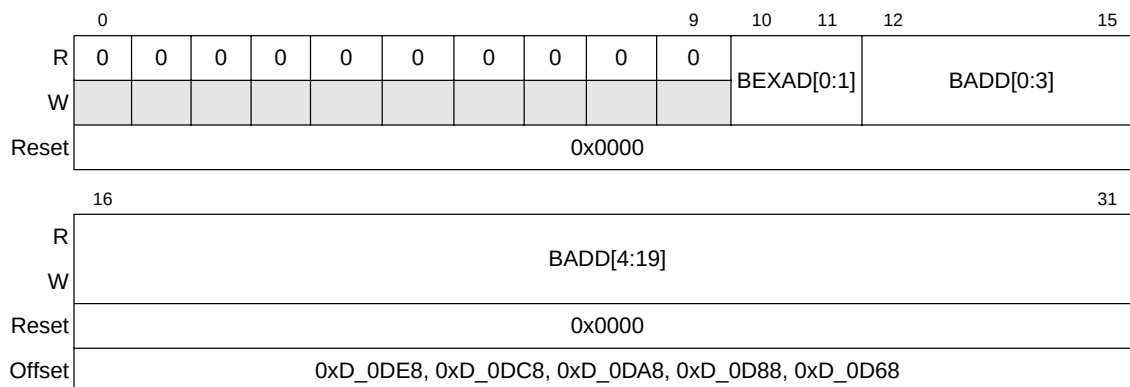


Figure 17-35. RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR_n)

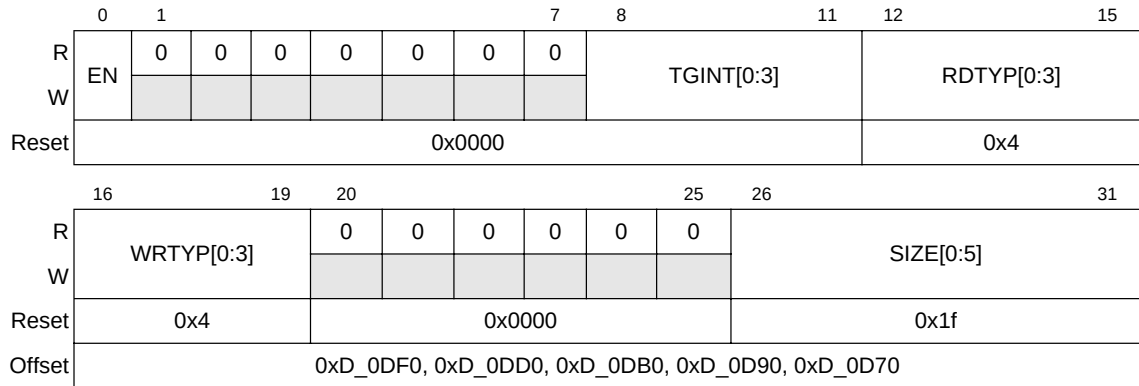
RIWBARs fields are described in Table 17-38.

Table 17-38. RIWBAR_n Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–11	BEXAD	Base extended address of inbound window. Field represents bits 0–1 of a 34-bit address.
12–31	BADD	Base address of inbound window. Source address that 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 a 34-bit address.

17.3.3.2.6 RapidIO Inbound Window Attributes Registers 0–4 (RIWAR_n)

The RapidIO inbound window attributes registers (RIWARs) define the window size to translate and other attributes for the translation. 4Gb is the largest window size allowed. The RIWARs are shown in Figure 17-36.

**Figure 17-36. RapidIO Inbound Window Attributes Registers 0–4 (RIWAR_n)**

RIWARs fields are described in Table 17-39.

Table 17-39. RIWAR_n Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables this window address translation. Note that for Window 0 RIWAR the enable bit is hardwired to 1. 0 Address translation disabled 1 Address translation enabled
1–7	—	Reserved
8–11	TGINT	Target Interface. 0000 PCI/PCI-X 0001–1110 Reserved 1111 Local Memory (DDR SDRAM, Local Bus, SRAM) Note: If this field is set to an I/O port rather than local memory space, attributes for the external I/O transaction are assigned in an outbound ATMU of that I/O controller.
12–15	RDTYP	Read transaction type. Transaction type to run if access is a read. The field description depends on the transaction being targeted (to I/O interface or to local memory). Following are the transaction type settings for reads to the PCI/PCI-X interface: 0000–0011 Reserved 0100 READ 0101–1111 Reserved Following are the transaction type settings for reads to local memory: 0000–0011 Reserved 0100 Received read; don't snoop local processor 0101 Received read; snoop local processor 0110 Reserved 0111 Received read; unlock L2 cache line 1000–1111 Reserved

Table 17-39. RIWAR_n Field Descriptions

Bits	Name	Description
16–19	WRTYP	Write transaction type. Transaction type to run if access is a write. The field description depends on the transaction being targeted (to I/O interface or to local memory). Following are the transaction type settings for writes to the PCI/PCI-X interface: 0000–0011 Reserved 0100 WRITE 0101–1111 Reserved Following are the transaction type settings for writes to local memory: 0000–0011 Reserved 0100 Received write; don't snoop local processor 0101 Received write; snoop local processor 0110 Received write; allocate L2 cache line 0111 Received write; allocate and lock L2 cache line 1000–1111 Reserved
20–25	—	Reserved
26–31	SIZE	Inbound window size. Inbound window size <i>N</i> which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. The largest window size is 4 Gbytes. 00_0000–00_1010 Reserved 00_1011 4-Kbyte window size 00_1100 8-Kbyte window size 01_1111 4-Gbyte window size 10_0000–11_1111 = Reserved

17.3.3.3 Error Management Registers

17.3.3.3.1 Port Notification/Fatal Error Detect Register (PNFEDR)

The PNFEDR keeps track of the notification and fatal errors that RapidIO has detected. Each bit in this register can be cleared when a 1 is written to it. The PNFEDR is shown in Figure 17-37.

	0	1	2		5	6	7	8	9	10	11	12	13	14	15	
R	MFE	AXE	0	0	0	0	ITE	MSE	NTIE	SE	DSE	BME	ETE	RTE	RERE	LTE
W																
Reset	0x0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RSTE	NAE	IRFE	URDE	URE	URSE	IWRE	IRSFE	IRTE	BRRE	BWRE	URSE	URSE	URSDE	MERE	MDFE
W																
Reset	0x0000															
Offset	0xD_0E00															

Figure 17-37. Port Notification/Fatal Error Detect Register (PNFEDR)

PNFEDR fields are described in Table 17-40.

Table 17-40. PNFEDR Field Descriptions

Bits	Name	Description
0	MFE	Multiple fatal errors (Bit reset, write 1 clear) 0 Multiple fatal RapidIO errors of the same type were not detected 1 Multiple fatal RapidIO errors of the same type were detected
1	AXE	ATMU window crossing error (Bit reset, write 1 clear). 0 No window crossing error 1 This error indicates that an inbound transaction with a data payload (write) crosses the boundary of its ATMU window.
2–5	—	Reserved
6	ITE	Illegal transaction target error (Bit reset, write 1 clear). 0 No illegal transaction detected 1 Received a packet whose target ID does not match the RapidIO controller deviceID while accept_all & pass through modes are disabled.
7	MSE	Message size error (Bit reset, write 1 clear). 0 No message size error detected 1 Received MESSAGE packet data payload is not of the size specified in the ssize field (exception for last packet which may be less)
8	NTIE	Missing idle after training (Bit reset, write 1 clear). 0 Idle received after a requested training sequence 1 Idle not received after a requested training sequence completes.
9	SE	Message segment error (Bit reset, write 1 clear). 0 No message segment error 1 Received a message packet with the segment field greater than the message length field (msgseg > msglen)
10	DSE	Duplicate message segment (Bit reset, write 1 clear). 0 No duplicate message segment received 1 Received a duplicate message segment from RapidIO
11	BME	Message length error (Bit reset, write 1 clear). 0 No message length error detected 1 Received a message packet from RapidIO with a bad message length
12	ETE	Error recovery threshold error (Bit reset, write 1 clear). 0 No error recovery threshold error detected 1 Error recovery threshold count exceeded
13	RTE	Retry threshold error (Bit reset, write 1 clear). 0 No retry threshold error detected 1 Consecutive ACK retry threshold count exceeded
14	RERE	Received request ERROR response (Bit reset, write 1 clear). 0 No error response received 1 Received a response of type error for anything other than a message.
15	LTE	Link response time-out (Bit reset, write 1 clear). 0 No link response time-out 1 A Link response is not received within the specified time-out interval
16	RSTE	Packet response time-out (Bit reset, write 1 clear). 0 No packet response time-out 1 A packet response is not received within the specified time-out interval

Table 17-40. PNFEDR Field Descriptions (continued)

Bits	Name	Description
17	NAE	Non-sensical ACKID (Bit reset, write 1 clear). 0 ACKID ok 1 ACKID received with Link_ response does not make sense
18	IRFE	Illegal request fields (Bit reset, write 1 clear). 0 Request packet fields ok 1 Illegal combinations of fields in the request packet
19	URDE	Unexpected request data (Bit reset, write 1 clear). 0 Request data ok 1 Read type with a data payload
20	URE	Unsupported request (Bit reset, write 1 clear). 0 Packe format type ok 1 Unsupported packet format type, or transport type (tt bits)
21	URSE	Unexpected request size (Bit reset, write 1 clear). 0 Packet data size ok 1 Packet data is not expected size
22	IWRE	Illegal write request (Bit reset, write 1 clear). 0 Write request ok 1 Write type with no data payload
23	IRSFE	Illegal response fields (Bit reset, write 1 clear). 0 Response packet ok 1 Illegal combinations of fields in the response packet
24	IRTE	Illegal response type (Bit reset, write 1 clear). 0 Response type ok 1 Illegal response for a given request type
25	BREE	Bad read response (Bit reset, write 1 clear). 0 Read response ok 1 Read "data" response with no data payload
26	BWRE	Bad write response (Bit reset, write 1 clear). 0 Write response ok 1 Write response with a data payload
27	URSSE	Unexpected response data size (Bit reset, write 1 clear). 0 Response packet size ok 1 Response packet data is not expected size
28	URESE	Unsolicited response without data (Bit reset, write 1 clear). 0 No response without data condition 1 A response without data is received if there exists no outstanding request matching that TID)
29	URSDE	Unsolicited response with data (Bit reset, write 1 clear). 0 No unsolicited response with data condition 1 A response with data is received if there exists no outstanding request matching that TID
30	MERE	Message error response (Bit reset, write 1 clear). 0 No message error response 1 Received a response of type error for an outbound message packet
31	MDFE	Message descriptor fetch error (Bit reset, write 1 clear). 0 No message descriptor fetch error 1 A message descriptor fetch from local memory gets an error

17.3.3.3.2 Port Notification/Fatal Error Disable Register (PNFEDiR)

PNFEDiR allows the reporting of each notification and fatal error in the PNFEDR to be disabled. The PNFEDiR is shown in Figure 17-38.

	0	1	2		5	6	7	8	9	10	11	12	13	14	15	
R	0		0	0	0	0	ITD	MSD	NTID	SD	DSD	BMD	ETD	RTD	RERD	LTD
W		AXD														
Reset	0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RSTD	NAD	IRFD	URDD	URD	URSD	IWRD	IRSF	IRTD	BRRD	BW RD	URS SD	URE SD	URSDD	MERD	MD FD
W																
Reset	0x0000															
Offset	0xD_0E04															

Figure 17-38. Port Notification/Fatal Error Disable Register (PNFEDiR)

PNFEDiR fields are described in Table 17-41.

Table 17-41. PNFEDiR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	AXD	ATMU window crossing error disable 0 ATMU window crossing error enabled 1 ATMU window crossing error disabled
2–5	—	Reserved
6	ITD	Illegal transaction target error disable 0 Illegal transaction target error enabled 1 Illegal transaction target error disabled
7	MSD	Message size error disable 0 Message size error enabled 1 Message size error disabled
8	NTID	Missing idle after training error disable 0 Missing idle after training error enabled 1 Missing idle after training error disabled
9	SD	Message segment error disable 0 Message segment error enabled 1 Message segment error disabled
10	DSD	Duplicate message segment error disable 0 Duplicate message segment error enabled 1 Duplicate message segment error disabled
11	BMD	Message length error disable 0 Message length error enabled 1 Message length error disabled

Table 17-41. PNFEDiR Field Descriptions (continued)

Bits	Name	Description
12	ETD	Error recovery threshold error disable 0 Error recovery threshold error enabled 1 Error recovery threshold error disabled
13	RTD	Retry threshold error disable 0 Retry threshold error enabled 1 Retry threshold error disabled
14	RERD	Received ERROR response error disable 0 Received ERROR response error enabled 1 Received ERROR response error disabled
15	LTD	Link response time-out error disable 0 Link response time-out error enabled 1 Link response time-out error disabled
16	RSTD	Packet response time-out error disable 0 Packet response time-out error enabled 1 Packet response time-out error disabled
17	NAD	Non-sensical ACKID error disable 0 Non-sensical ACKID error enabled 1 Non-sensical ACKID error disabled
18	IRFD	Illegal request fields error disable 0 Illegal request fields error enabled 1 Illegal request fields error disabled
19	URDD	Unexpected request data error disable 0 Unexpected request data error enabled 1 Unexpected request data error disabled
20	URD	Unsupported request error disable 0 Unsupported request error enabled 1 Unsupported request error disabled
21	URSD	Unexpected request size error disable 0 Unexpected request size error enabled 1 Unexpected request size error disabled
22	IWRD	Illegal write request error disable 0 Illegal write request error enabled 1 Illegal write request error disabled
23	IRSFD	Illegal response fields error disable 0 Illegal response fields error enabled 1 Illegal response fields error disabled
24	IRTD	Illegal response type error disable 0 Illegal response type error enabled 1 Illegal response type error disabled
25	BRED	Bad read response error disable 0 Bad read response error enabled 1 Bad read response error disabled
26	BWRD	Bad write response error disable 0 Bad write response error enabled 1 Bad write response error disabled

Table 17-41. PNFEDiR Field Descriptions (continued)

Bits	Name	Description
27	URSSD	Unexpected response data size error disable 0 Unexpected response data size error enabled 1 Unexpected response data size error disabled
28	URES	Unsolicited response without data error disable 0 Unsolicited response without data error enabled 1 Unsolicited response without data error disabled
29	URSDD	Unsolicited response with data error disable 0 Unsolicited response with data error enabled 1 Unsolicited response with data error disabled
30	MERD	Message error response error disable 0 Message error response error enabled 1 Message error response error disabled
31	MDFD	Message descriptor fetch error disable 0 Message descriptor fetch error enabled 1 Message descriptor fetch error disabled

17.3.3.3.3 Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

The PNFEIER, shown in Figure 17-39, allows each notification and fatal error to generate an interrupt.

	0	1	2		5	6	7	8	9	10	11	12	13	14	15	
R	0	AXI E	0	0	0	0	ITIE	MSIE	NTIIE	SIE	DSIE	BMIE	ETIE	RTIE	RERIE	LTIE
W																
Reset	0x0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RSTIE	NAI E	IRFI E	URDI E	URI E	URSI E	IWRI E	IRSFIE	IRTIE	BRRIE	BWRI E	URS SIE	URS IE	URSD IE	MERIE	MDF IE
W																
Reset	0x0000															
Offset	0xD_0E08															

Figure 17-39. Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

PNFEIER fields are described in Table 17-42.

Table 17-42. PNFEIER Field Descriptions

Bits	Name	Description
0	—	Reserved
1	AXIE	ATMU window crossing error interrupt enable
2–5	—	Reserved

Table 17-42. PNFEIER Field Descriptions (continued)

Bits	Name	Description
6	ITIE	Illegal Transaction Target Error interrupt enable
7	MSIE	Message size error interrupt enable
8	NTIE	Missing idle after training interrupt enable
9	SIE	Message segment error interrupt enable
10	DSIE	Duplicate message segment error interrupt enable
11	BMIE	Message length error interrupt enable
12	ETIE	Error recovery threshold error interrupt enable
13	RTIE	Retry threshold error interrupt enable
14	RERIE	Received ERROR response error interrupt enable
15	LTIE	Link response time-out error interrupt enable
16	RSTIE	Packet response time-out error interrupt enable
17	NAIE	Non-sensical ACKID error interrupt enable
18	IRFIE	Illegal request fields error interrupt enable
19	URDIE	Unexpected request data error interrupt enable
20	URIE	Unsupported request error interrupt enable
21	URSIE	Unexpected request size error interrupt enable
22	IWRIE	Illegal write request error interrupt enable
23	IRSFIE	Illegal response fields error interrupt enable
24	IRTIE	Illegal response type error interrupt enable
25	BREIE	Bad read response error interrupt enable
26	BWRIE	Bad write response error interrupt enable
27	URSSIE	Unexpected response data size error interrupt enable
28	URSIE	Unsolicited response without data error interrupt enable
29	URSDIE	Unsolicited response with data error interrupt enable
30	MERIE	Message error response error interrupt enable
31	MDFIE	Message descriptor fetch error interrupt enable

17.3.3.3.4 Port Error Control Register (PECR)

The PECR, shown in Figure 17-40, indicates that a valid packet has been captured when a notification or fatal error has occurred.

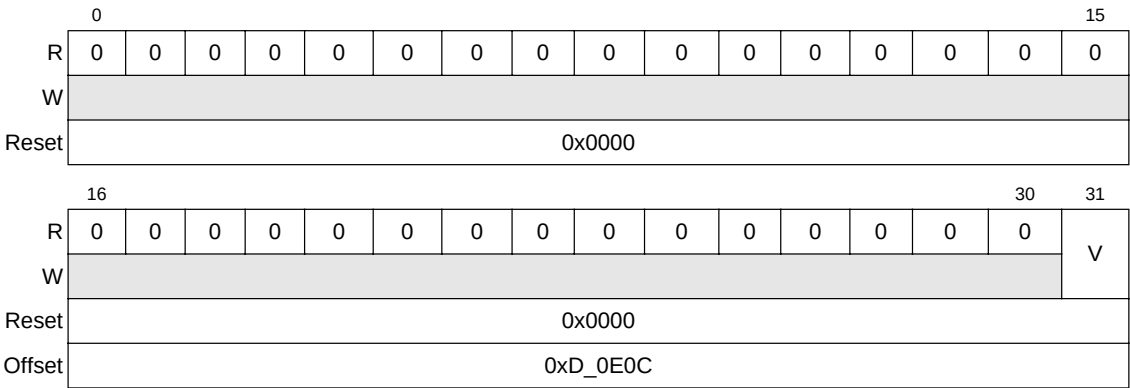


Figure 17-40. Port Error Control Register (PECR)

PECR fields are described in Table 17-43.

Table 17-43. PECR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	V	Packet registers contain valid information.

17.3.3.3.5 Port Error Packet/Control Symbol Register 0 (PEPCSR0)

The port error packet/control symbol register 0 (PEPCSR0), shown in Figure 17-41, captures the first 32 bits of an inbound packet which caused a fatal error. The first 32 bits of an inbound packet are the same for every packet format type.

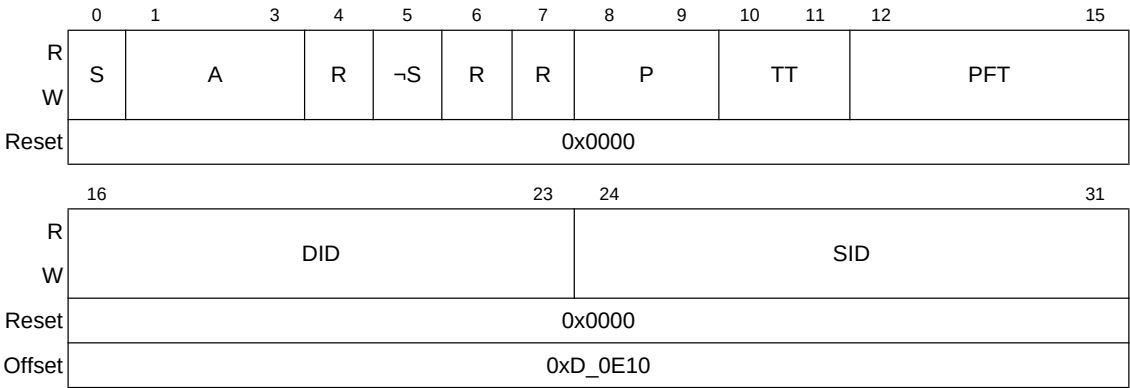


Figure 17-41. Port Error Packet/Control Symbol Register 0 (PEPCSR0)

PEPCSR0 fields are described in Table 17-44.

Table 17-44. PEPCSR0 Field Descriptions

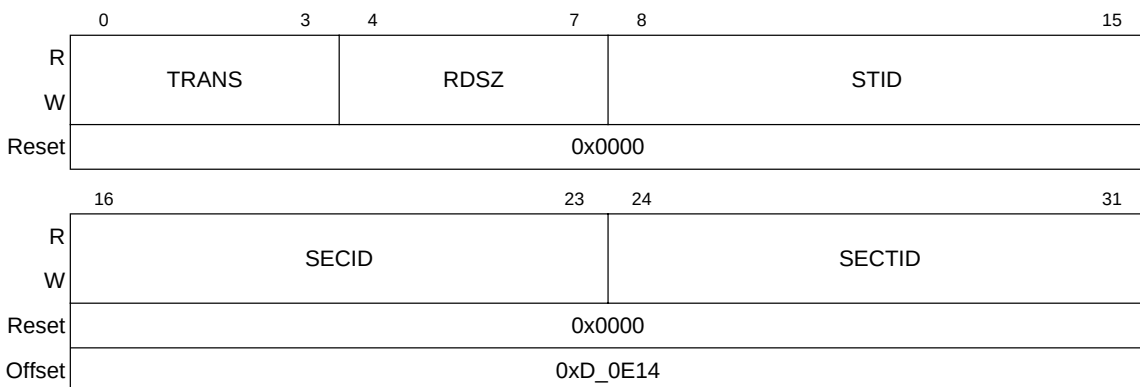
Bits	Name	Description
0	S	S-bit of the packet
1–3	A	ACK ID field of a packet
4	R	Reserved packet field - not a reserved register field.
5	¬S	These bits correspond to bits 4–7 of a packet header. In the packet definition bit 4 is reserved, bit 5 is S-complement, bits 6–7 are reserved.
6	R	Reserved packet field - not a reserved register field.
7	R	Reserved packet field - not a reserved register field.
8–9	P	Priority field of a packet
10–11	TT	Transport type field of a packet
12–15	PFT	Packet format type field of a packet
16–23	DID	Destination ID field of a packet
24–31	SID	Source ID field of a packet

17.3.3.3.6 Port Error Packet Register 1 (PEPR1)

This register captures a portion of an inbound packet that caused a fatal error. The contents of this register depends on the packet format type. The various formats of this register for the various packet format types are described as unique registers in Section 17.3.3.3.7, “RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1),” through Section 17.3.3.3.15, “RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13).”

17.3.3.3.7 RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1)

Figure 17-42 displays the RapidIO packet error capture register 1 for a type 1 packet format (PECR1T1).

**Figure 17-42. RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1)**

PECR1T1 fields are described in Table 17-45.

Table 17-45. PECR1T1 Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	RDSZ	Read size
8–15	STID	Source transaction ID
16–23	SECID	Secondary source ID
24–31	SECTID	Secondary transaction ID

17.3.3.3.8 RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2)

Figure 17-43 displays the RapidIO packet error capture register 1 for a type 2 packet format (PECR1T2).

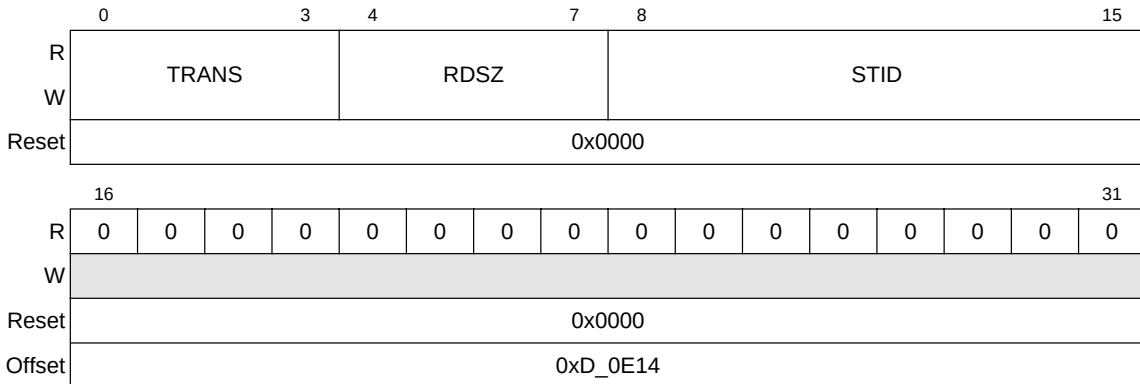


Figure 17-43. RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2)

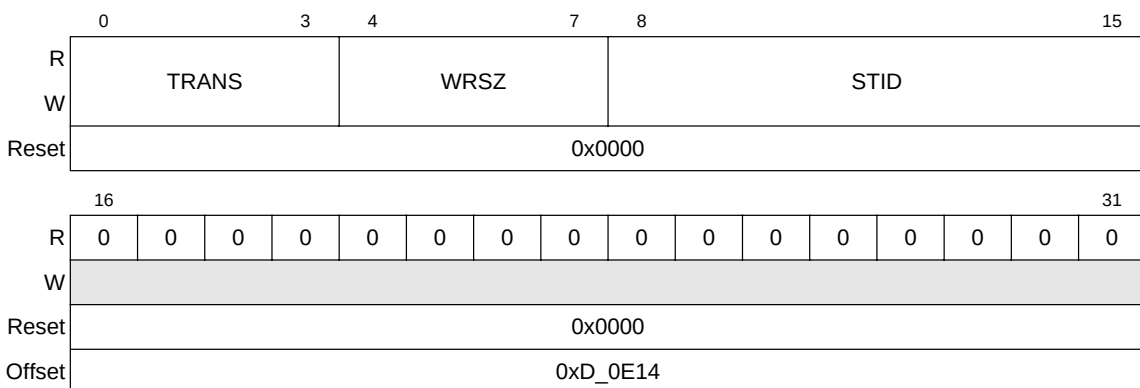
PECR1T2 fields are described in Table 17-46.

Table 17-46. PECR1T2 Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	RDSZ	Read size
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.9 RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5)

Figure 17-44 displays the RapidIO packet error capture register 1 for a type 5 packet format (PECR1T5).

**Figure 17-44. RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5)**

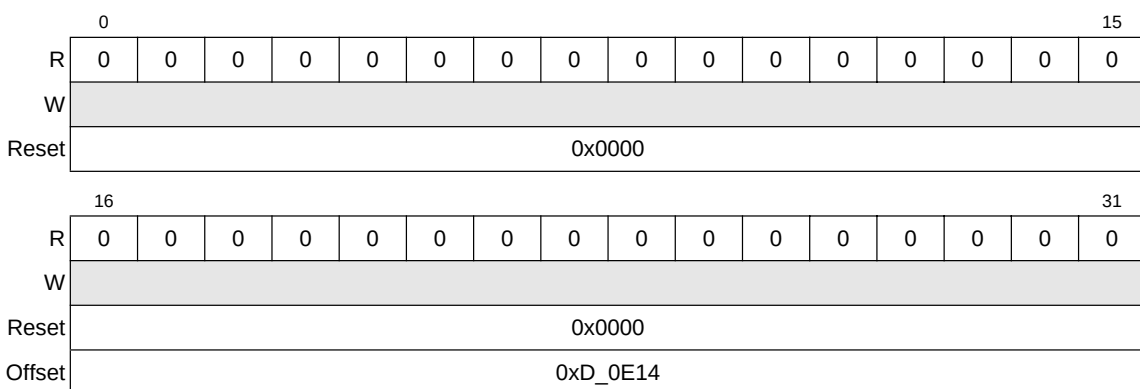
PECR1T5 fields are described in Table 17-47.

Table 17-47. PECR1T5 Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	WRSZ	Write size
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.10 RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6)

Figure 17-45 displays the RapidIO packet error capture register 1 for a type 6 packet format (PECR1T6).

**Figure 17-45. RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6)**

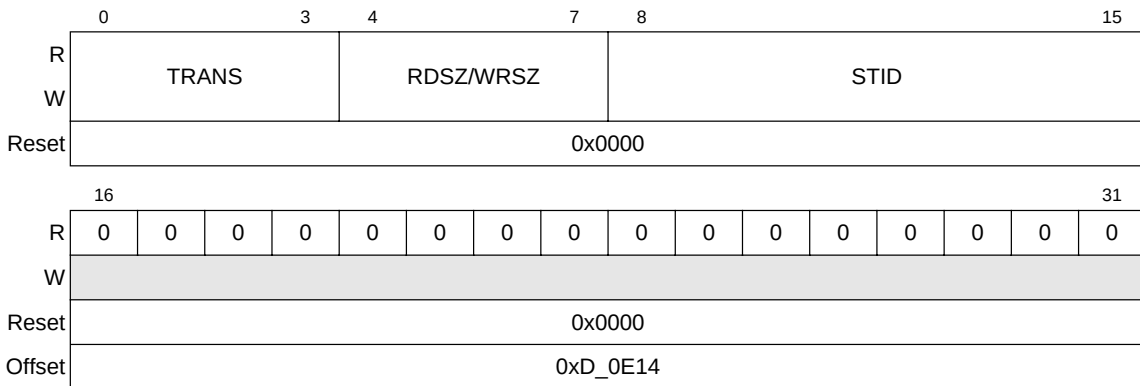
PECR1T6 fields are described in Table 17-48.

Table 17-48. PECR1T6 Field Descriptions

Bits	Name	Description
0–31	—	Reserved

17.3.3.3.11 RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ)

Figure 17-46 displays the RapidIO packet error capture register 1 for a type 8 request packet format (PECR1T8RQ).

**Figure 17-46. RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ)**

PECR1T8RQ fields are described in Table 17-49.

Table 17-49. PECR1T8RQ Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	RDSZ/WTSZ	Read or write size depending on whether it is a maintenance read or write request
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.12 RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)

Figure 17-47 displays the RapidIO packet error capture register 1 for a type 8 response packet format (PECR1T8R).

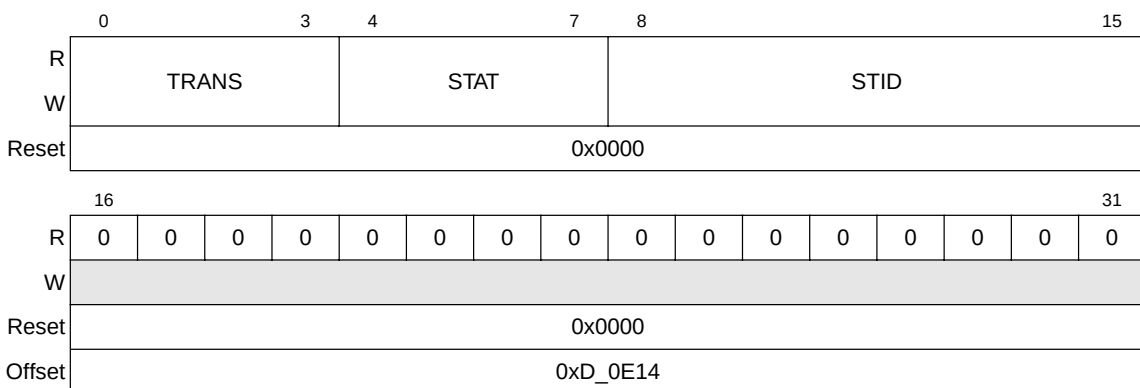


Figure 17-47. RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)

PECR1T8R fields are described in Table 17-50.

Table 17-50. PECR1T8R Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	STAT	Status field of a packet
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.13 RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10)

Figure 17-48 displays the RapidIO packet error capture register 1 for a type 10 packet format (PECR1T10).

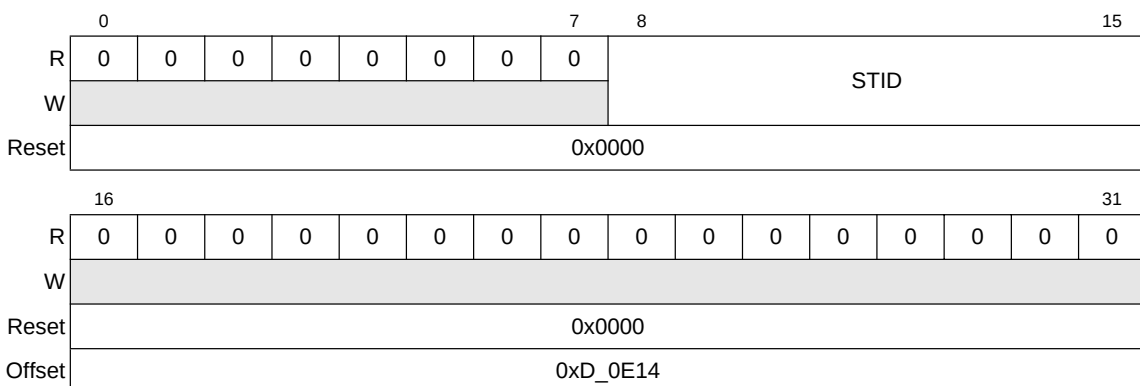


Figure 17-48. RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10)

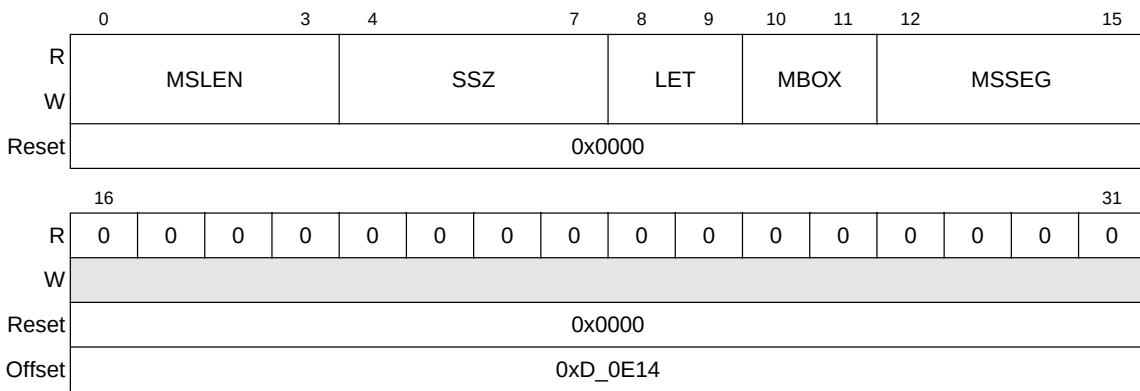
PECR1T10 fields are described in Table 17-51.

Table 17-51. PECR1T10 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.14 RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11)

Figure 17-49 displays the RapidIO packet error capture register 1 for a type 11 packet format (PECR1T11).

**Figure 17-49. RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11)**

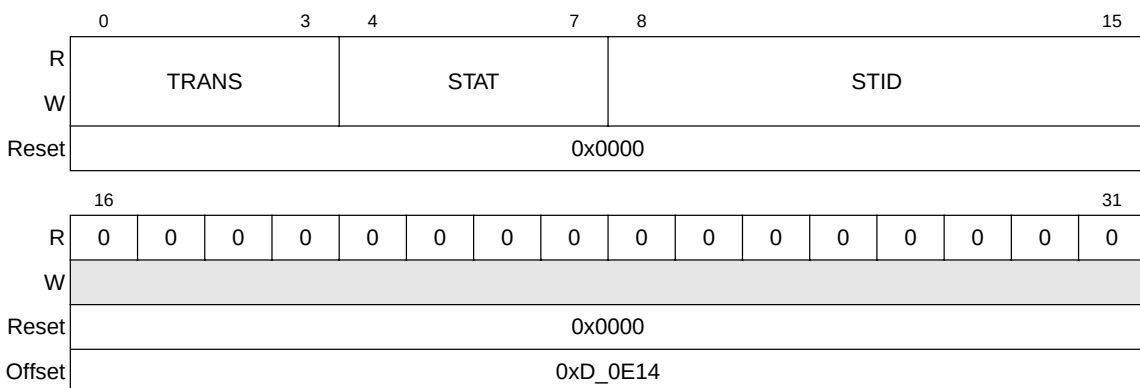
PECR1T11 fields are described in Table 17-52.

Table 17-52. PECR1T11 Field Descriptions

Bits	Name	Description
0–3	MSLEN	Message length field of a packet
4–7	SSZ	Segment size field of a packet
8–9	LET	Letter field of a packet
10–11	MBOX	Mailbox field of a packet
12–15	MSSEG	Message segment field of a packet
16–31	—	Reserved

17.3.3.3.15 RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13)

Figure 17-50 displays the RapidIO packet error capture register 1 for a type 13 packet format (PECR1T13).

**Figure 17-50. RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13)**

PECR1T13 fields are described in Table 17-53.

Table 17-53. PECR1T13 Field Descriptions

Bits	Name	Description
0–3	TRANS	Transaction field of a packet
4–7	STAT	Status field of a packet
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.3.3.16 RapidIO Port Error Packet Register 2 (PEPR2)

This register captures another portion of an inbound packet that caused a fatal error. The contents of this register depends on the packet format type. The various formats of this register for the various packet format types are described as unique registers in sections 17.3.3.3.17 through 17.3.3.3.22.

17.3.3.3.17 RapidIO Packet Error Capture Register 2, Types 1, 2, and 5 (PECR2T125)

Figure 17-51 displays the RapidIO packet error capture register 2 for a type 1, 2, and 5 packet format (PECR2T125).

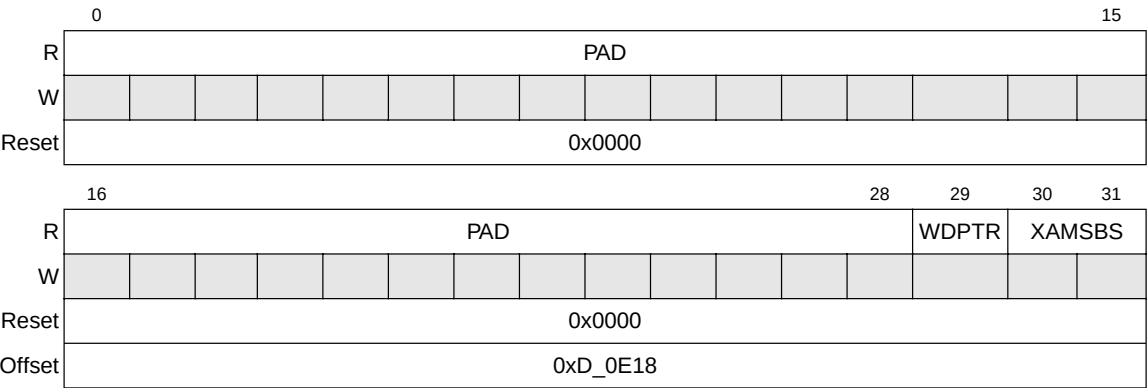


Figure 17-51. RapidIO Packet Error Capture Register 2, Type 1 (PECR2T125)

PECR2T125 fields are described in Table 17-54.

Table 17-54. PECR2T125 Field Descriptions

Bits	Name	Description
0–28	PAD	Address field of a packet
29	WDPTR	WdPtr field of a packet
30–31	XAMSBS	Address[0:1] field of a packet

17.3.3.3.18 RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6)

Figure 17-52 displays the RapidIO packet error capture register 2 for a type 6 packet format (PECR2T6).

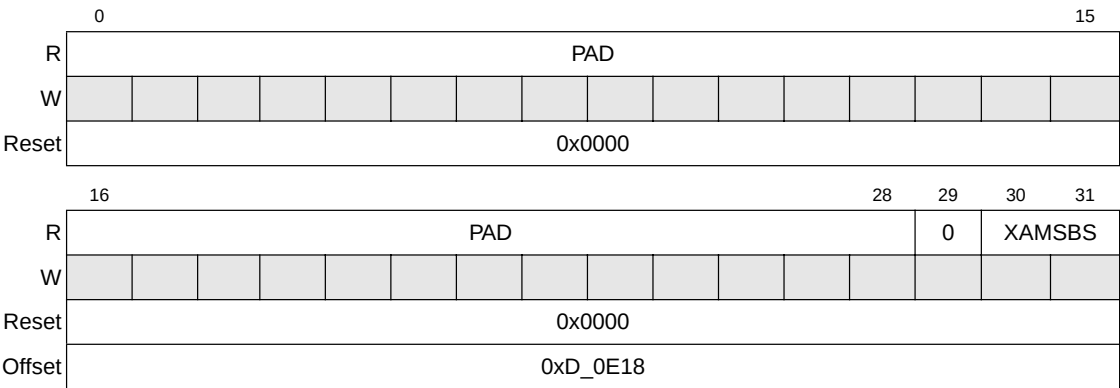


Figure 17-52. RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6)

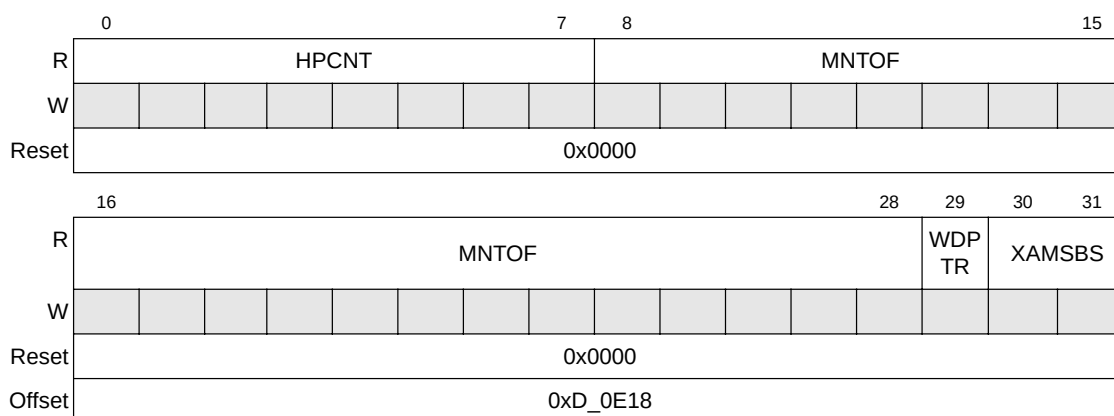
PECR2T6 fields are described in Table 17-55.

Table 17-55. PECR2T6 Field Descriptions

Bits	Name	Description
0–28	PAD	Address field of a packet
29	—	Reserved
30–31	XAMSBS	Address[0:1] field of a packet

17.3.3.3.19 RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8)

Figure 17-53 displays the RapidIO packet error capture register 2 for a type 8 packet format (PECR2T8).

**Figure 17-53. RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8)**

PECR2T8 fields are described in Table 17-56.

Table 17-56. PECR2T8 Field Descriptions

Bits	Name	Description
0–7	HPCNT	Hop count field of a packet
8–28	MNTOF	Maintenance offset field of a packet
29	WDPTR	WdPtr field of a packet
30–31	XAMSBS	Address[0:1] field of a packet

17.3.3.3.20 RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)

Figure 17-54 displays the RapidIO packet error capture register 2 for a type 8 response packet format (PECR2T8R).

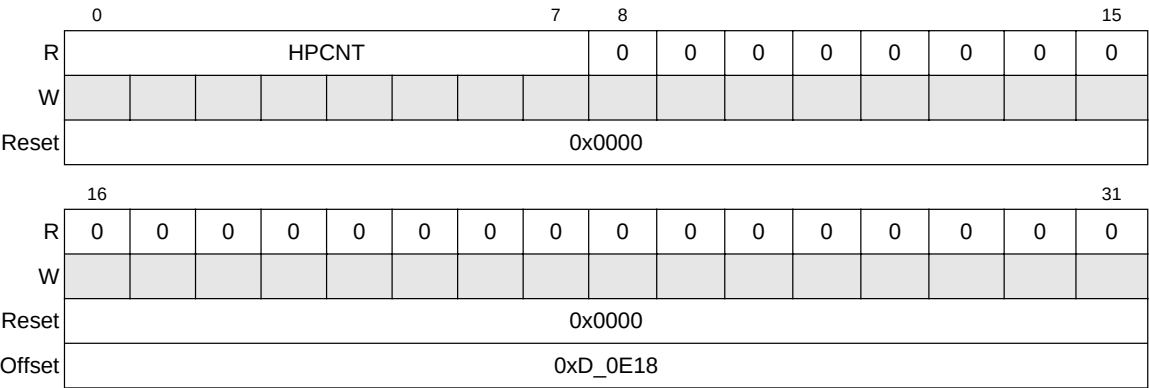


Figure 17-54. RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)

PECR2T8R fields are described in Table 17-57.

Table 17-57. PECR2T8R Field Descriptions

Bits	Name	Description
0–7	HPCNT	Hop count field of a packet
8–31	—	Reserved

17.3.3.3.21 RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10)

Figure 17-55 displays the RapidIO packet error capture register 2 for a type 10 packet format (PECR2T10).

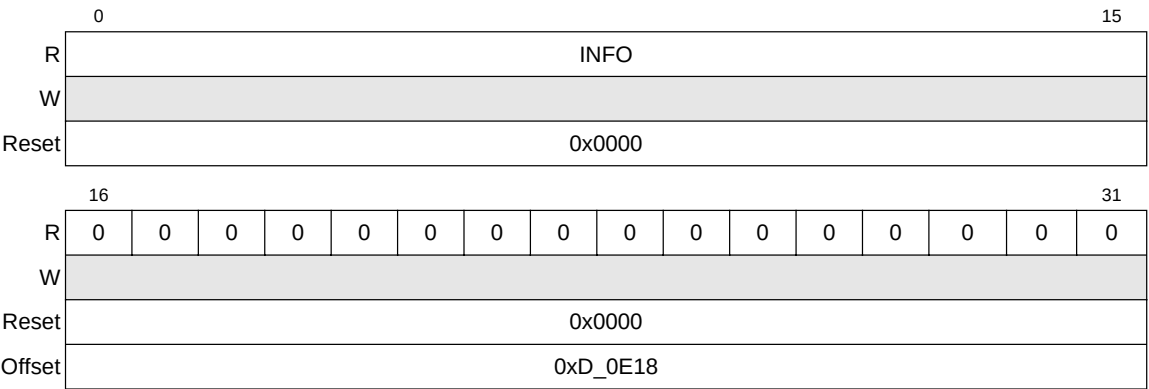


Figure 17-55. RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10)

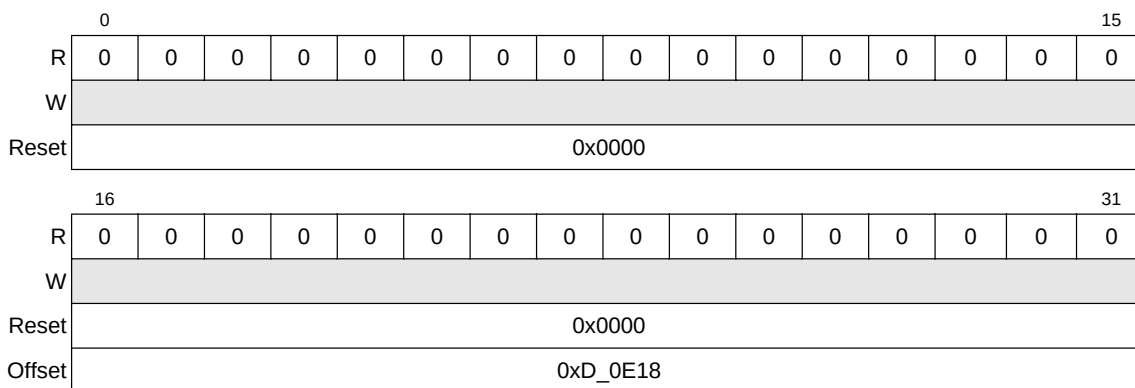
PECR2T10 fields are described in Table 17-58.

Table 17-58. PECR2T10 Field Descriptions

Bits	Name	Description
0–15	INFO	Information field of a packet
16–31	—	Reserved

17.3.3.3.22 RapidIO Packet Error Capture Register 2, Types 11 and 13 (PECR2T11)

Figure 17-56 displays the RapidIO packet error capture register 2 for types 11 and 13 packet formats (PECR2T11).

**Figure 17-56. RapidIO Packet Error Capture Register 2, Type 11 (PECR2T11)**

PECR2T11 fields are described in Table 17-59.

Table 17-59. PECR2T11 Field Description

Bits	Name	Description
0–31	—	Reserved

17.3.3.3.23 Port Recoverable Error Detect Register (PREDR)

The PREDR, shown in Figure 17-57, indicates the recoverable errors detected on RapidIO.

	0	1				6	7	8	9	10	11	12	13	14	15	
R	MRE	0	0	0	0	0	0	ETE	ISE	ICCE	IAOSE	IUAE	IBCE	IE276E	INAIE	INAUA
W																
Reset	0x0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INACC	INAPS	INABC	INASE	INAGE	URE	UAE	ABRE	UTE	USE	ULE	UEE	LRE	FAE	FTEE	ATE
W																
Reset	0x0000															
Offset	0xD_0E20															

Figure 17-57. Port Recoverable Error Detect Register (PREDR)

PREDR fields are described in Table 17-60.

Table 17-60. PREDR Field Descriptions

Bits	Name	Description
0	MRE	Multiple recoverable errors 0 Multiple recoverable RapidIO errors of the same type were not detected (Bit reset, write 1 clear) 1 Multiple recoverable RapidIO errors of the same type were detected
1–6	—	Reserved
7	ETE	Embedded training 0 No requested training pattern 1 Requested training pattern is embedded in a packet (Bit reset, write 1 clear)
8	ISE	Inbound S-bit error 0 No inbound S-bit error 1 Received a packet/control symbol with an S-bit parity error (Bit reset, write 1 clear)
9	ICCE	Inbound corrupt control symbol 0 No inbound corrupt control symbol 1 Received a control symbol in which true does not match complement counterpart (Bit reset, write 1 clear)
10	IAOSE	Inbound out-of-sequence ACK symbol 0 No inbound out-of-sequence ACK symbol 1 Received an ACK control symbol with an unexpected ACKID (Bit reset, write 1 clear)
11	IUAE	Inbound packet with unexpected ACKID 0 No inbound packet with unexpected ACKID 1 Received packet with unexpected ackID value—out-of-sequence ACKID (Bit reset, write 1 clear)
12	IBCE	Inbound packet with bad CRC 0 No inbound packet with bad CRC 1 Received packet with a bad CRC value (Bit reset, write 1 clear)
13	IE276E	Inbound packet exceeds 276 bytes 0 Inbound packet does not exceed 276 bytes 1 Received packet which exceeds the maximum allowed size by the RapidIO spec (Bit reset, write 1 clear)

Table 17-60. PREDR Field Descriptions (continued)

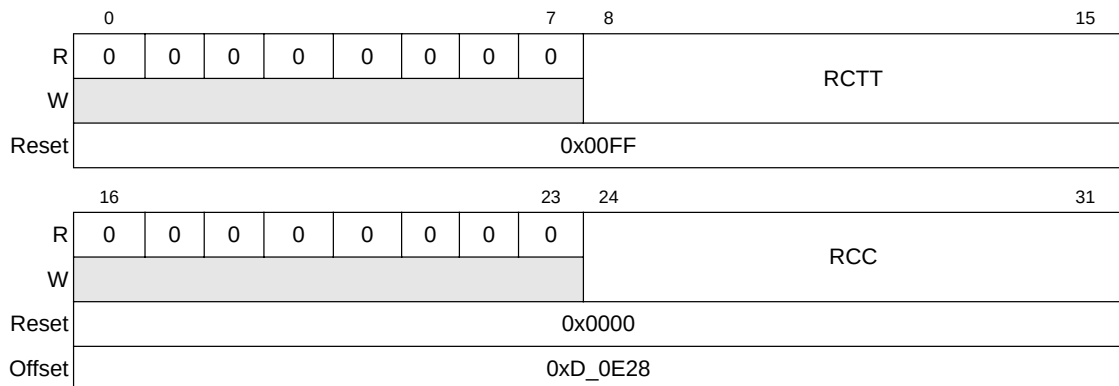
Bits	Name	Description
14	INAIE	Packet-not-accepted; encountered internal error 0 No packet-not-accepted; internal error encountered 1 Received ACK control symbol with packet-not-accepted of type 'encountered internal error' (Bit reset, write 1 clear)
15	INAUA	Packet-not-accepted; unexpected ACKID on packet 0 No packet-not-accepted; unexpected ACKID on packet condition 1 Received ACK control symbol with packet-not-accepted of type 'unexpected ACKID' (Bit reset, write 1 clear)
16	INACC	Packet-not-accepted; corrupt control symbol 0 No packet-not-accepted; corrupt control symbol condition 1 Received ACK control symbol with packet-not-accepted of type 'corrupt control symbol' (Bit reset, write 1 clear)
17	INAPS	Packet-not-accepted; input port stopped 0 No packet-not-accepted; input port stopped condition 1 Received ACK control symbol with packet-not-accepted of type 'input port stopped' (Bit reset, write 1 clear)
18	INABC	Packet-not-accepted; bad CRC 0 No packet-not-accepted; bad CRC condition 1 Received ACK control symbol with packet-not-accepted of type 'bad CRC' (Bit reset, write 1 clear)
19	INASE	Packet-not-accepted; S-bit parity error 0 NO packet-not-accepted; S-bit parity error condition 1 Received ACK control symbol with packet-not-accepted of type 'S-bit parity error' (Bit reset, write 1 clear)
20	INAGE	Packet-not-accepted; general error 0 No packet-not-accepted; general error condition 1 Received ACK control symbol with packet-not-accepted of type 'general error' (Bit reset, write 1 clear)
21	URE	Unexpected restart-from-retry symbol 0 No unexpected restart-from-retry symbol 1 Received a restart-from-retry control symbol while in the 'ok' state (Bit reset, write 1 clear)
22	UAE	Unsolicited ACK symbol 0 No unsolicited ACK symbol encountered 1 Received an ACK control symbol while no packets are posted on the outstanding ACK history queue (Bit reset, write 1 clear)
23	ABRE	ACK before restart-from-retry 0 No ACK before restart-from-retry 1 Received an ACK control symbol after receiving an ACK retry and before sending a restart-from-retry (Bit reset, write 1 clear)
24	UTE	Unexpected training symbol 0 No unexpected training symbol 1 Received a training symbol while in the 'ok' state (Bit reset, write 1 clear)
25	USE	Unexpected stomp symbol 0 No unexpected stomp symbol 1 Received a stomp control symbol while there is no packet being received (Bit reset, write 1 clear)

Table 17-60. PREDR Field Descriptions (continued)

Bits	Name	Description
26	ULE	Unexpected link response symbol 0 No unexpected link response symbol 1 Received a Link response control symbol while there is no outstanding request (Bit reset, write 1 clear)
27	UEE	Unexpected EOP symbol 0 No unexpected EOP symbol 1 Received an EOP control symbol while there is no packet being received (Bit reset, write 1 clear)
28	LRE	Link request error 0 No link request error 1 Received a link request control symbol before the previous link request has been serviced (Bit reset, write 1 clear)
29	FAE	Frame toggle alignment 0 No frame toggle alignment problem 1 Received FRAME signal toggles at a non 32-bit boundary (Bit reset, write 1 clear)
30	FTEE	Frame toggle edge 0 No frame toggle edge problem 1 Received FRAME signal toggles on the negative edge of the clock (Bit reset, write 1 clear)
31	ATE	ACK time-out 0 No ACK time-out condition 1 An ACK control symbol is not received within the specified time-out interval (Bit reset, write 1 clear)

17.3.3.3.24 Port Error Recovery Threshold Register (PERTR)

The PERTR is shown in Figure 17-58.

**Figure 17-58. Port Error Recovery Threshold Register (PERTR)**

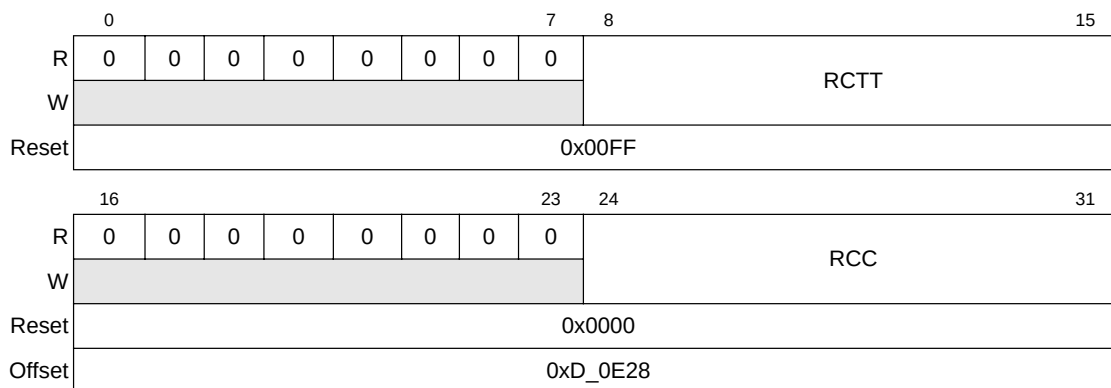
PERTR fields are described in Table 17-61.

Table 17-61. PERTR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	RCTT	Recovery threshold trigger. These bits provide the threshold value for the number of successful transmission error recoveries before reporting an error condition.
16–23	—	Reserved
24–31	RCC	Recovery counter. These bits maintain a count of the number of times the port has successfully recovered from a transmission error. An error is reported if this value equals the value contained in the recovery threshold trigger register and if reporting is enabled.

17.3.3.3.25 Port Retry Threshold Register (PRTR)

The PRTR is shown in Figure 17-59.

**Figure 17-59. Port Retry Threshold Register (PRTR)**

PRTR fields are described in Table 17-62.

Table 17-62. PRTR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	RCTT	Recovery threshold trigger. These bits provide the threshold value for the number of successful transmission error recoveries before reporting an error condition.
16–23	—	Reserved
24–31	RCC	Recovery counter. These bits maintain a count of the number of times the port has successfully recovered from a transmission error. An error is reported if this value equals the value contained in the recovery threshold trigger register and if reporting is enabled.

17.3.4 RapidIO Message Unit Registers

17.3.4.1 RapidIO Outbound Message Registers

The registers in this section control RapidIO outbound messages. The following provide partial descriptions of these registers.

17.3.4.1.1 Outbound Mode Register (OMR)

The outbound mode register (OMR) allows software to start a message operation and to control various message operation characteristics. The OMR is shown in Figure 17-60.

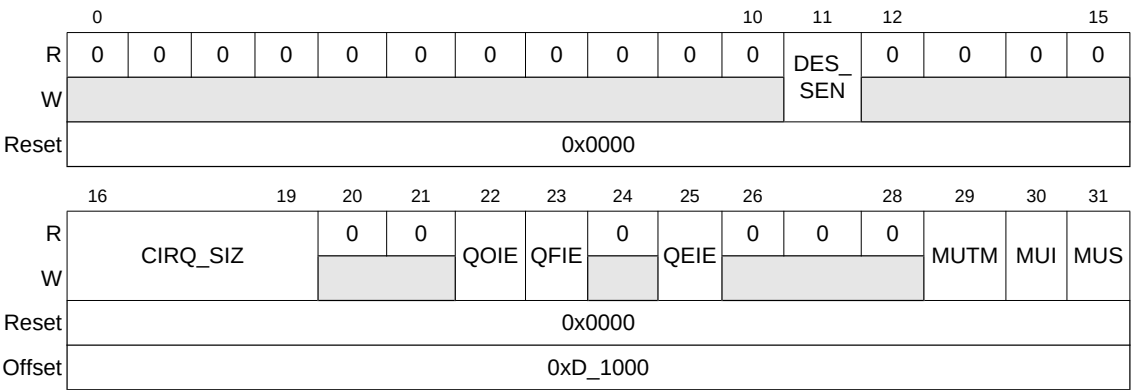


Figure 17-60. Outbound Mode Register (OMR)

OMR fields are described in Table 17-63.

Table 17-63. OMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	DES_SEN	Descriptor snoop enable. 0 Snooping of local processor disabled 1 Snooping of the local processor enabled when reading descriptors from memory.
12–15	—	Reserved

Table 17-63. OMR Field Descriptions (continued)

Bits	Name	Description
16–19	CIRQ_SIZ	Circular descriptor queue size. Determines the number of descriptors that can be placed on the circular queue without overflow. 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64 0110 128 (4-kbyte page boundary) 0111 256 1000 512 1001 1024 1010 2048 1011 Reserved ... 1111 Reserved
20–21	—	Reserved
22	QOIE	Queue overflow interrupt enable. Only applicable to chaining mode. 0 No QO interrupt is generated 1 Generates an interrupt upon 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).
23	QFIE	Queue full interrupt enable. Only applicable to chaining mode. 0 No QF interrupt is generated 1 Generates an interrupt when the queue transitions to full (that is, the enqueue and dequeue pointers are equal after being incremented by the processor).
24	—	Reserved
25	QEIE	Queue empty interrupt enable. Only applicable to chaining mode. 0 No QE interrupt is generated 1 Generates an interrupt at the completion of all outstanding message operations (that is, the enqueue and dequeue pointers are equal after an increment by the message unit controller).
26–28	—	Reserved
29	MUTM	Message unit transfer mode. 0 Message unit is configured to be in chaining mode. 1 Puts the message unit into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the message transmission.
30	MUI	Message unit increment. Software sets this bit after writing a descriptor to memory. Hardware increments the ODQEPAR and clears this bit. Always reads as 0 while MUS is set.
31	MUS	Message unit start Direct mode: a 0 to 1 transition while the message unit is not busy (MUB bit is cleared) starts the message unit. A 1 to 0 transition has no effect. Chaining mode: a 0 to 1 transition while the message unit is not busy (MUB bit is cleared) initializes the message unit in chaining mode and causes the outbound descriptor queue dequeue pointer address to be stored as the base address of the circular descriptor queue. If this bit is set, the message unit starts each time the enqueue and dequeue pointers are not equal and an overflow of the circular descriptor queue has not been detected. A 1 to 0 transition while the message unit is busy halts the message unit after the current descriptor is processed. The message unit must be re-initialized after a halt condition.

17.3.4.1.2 Outbound Status Register (OSR)

The outbound status register (OSR) reports various message unit conditions during and after a message operation. Setting a condition bit clears it. The OSR is shown in Figure 17-61.

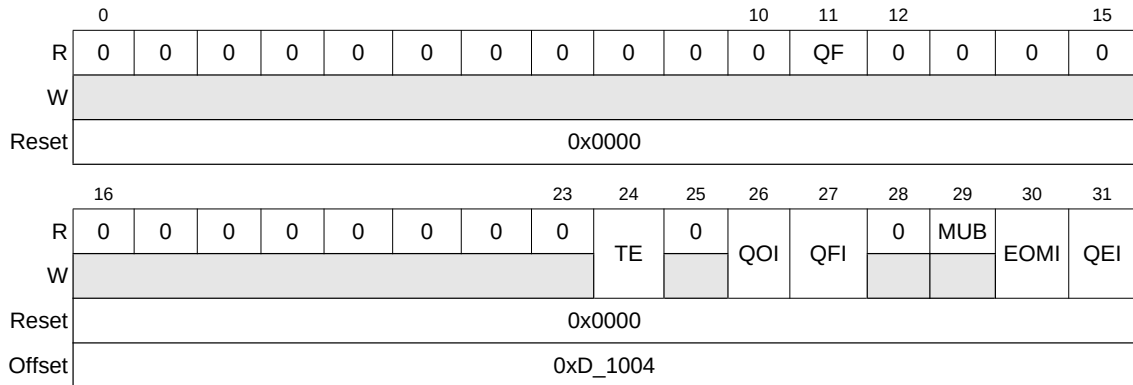


Figure 17-61. Outbound Status Register (OSR)

OSR fields are described in Table 17-64.

Table 17-64. OSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10	—	Reserved
11	QF	Queue full. (Read-only) 0 Queue not full 1 The queue has become full
12–13	—	Reserved
14	—	Reserved.
15	—	Reserved.
16–23	—	Reserved
24	TE	Transaction error 0 No transaction error 1 There was an error condition during the message operation. (Bit reset, write 1 clear)
25	—	Reserved
26	QOI	Queue overflow interrupt. Only applicable to chaining mode. (Bit reset, write 1 clear) 0 No overflow interrupt condition 1 A queue overflow condition is detected and OMR[QOIE] is set. An interrupt is generated in this case.
27	QFI	Queue full interrupt. Only applicable to chaining mode. (Bit reset, write 1 clear) 0 No queue full interrupt 1 The queue became full and OMR[QFIE] is set. An interrupt is generated in this case.

Table 17-64. OSR Field Descriptions (continued)

Bits	Name	Description
28	—	Reserved
29	MUB	Message unit busy. (Read-only) 0 Message unit not busy. This bit is cleared as a result of receiving an error or the completion of a message operation. 1 A message operation is currently in progress.
30	EOMI	End-of-message interrupt. 0 No end-of-message interrupt condition 1 After finishing this message operation, if ODATR[EOMIE] is set, this bit is set and an interrupt is generated. (Bit reset, write 1 clear)
31	QEI	Queue empty interrupt. Only applicable to chaining mode. (Bit reset, write 1 clear) 0 No queue empty interrupt condition 1 When the last message operation in the outbound descriptor queue is finished, if OMR[QEIE] is set, then this bit is set and an interrupt is generated. Otherwise, no interrupt is generated.

17.3.4.1.3 Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

The outbound descriptor queue dequeue pointer address register (ODQDPAR) contains the address of the first descriptor in memory to be processed. Software must initialize this register to point to the first descriptor in memory. After processing this descriptor, the message unit controller increments the outbound descriptor queue dequeue pointer address in ODQDPAR to point to the next descriptor. If the outbound descriptor queue enqueue pointer and the outbound 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 OMR[QEIE] is set, OSR[QEI] is set and an interrupt is generated. The ODQDPAR is shown in Figure 17-62.

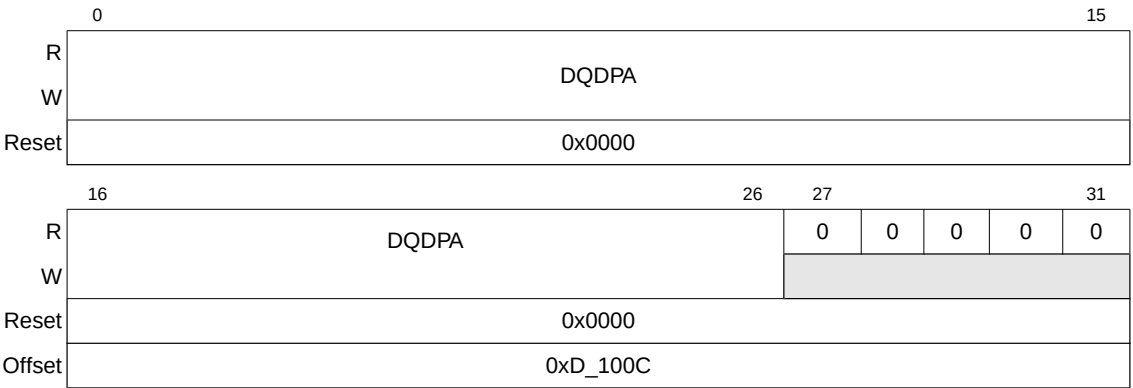


Figure 17-62. Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

ODQDPAR fields are described in Table 17-65.

Table 17-65. ODQDPAR 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.
27–31	—	Reserved

17.3.4.1.4 Outbound Unit Source Address Register (OSAR)

The outbound unit source address register (OSAR) indicates 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. The OSAR is shown in Figure 17-63.

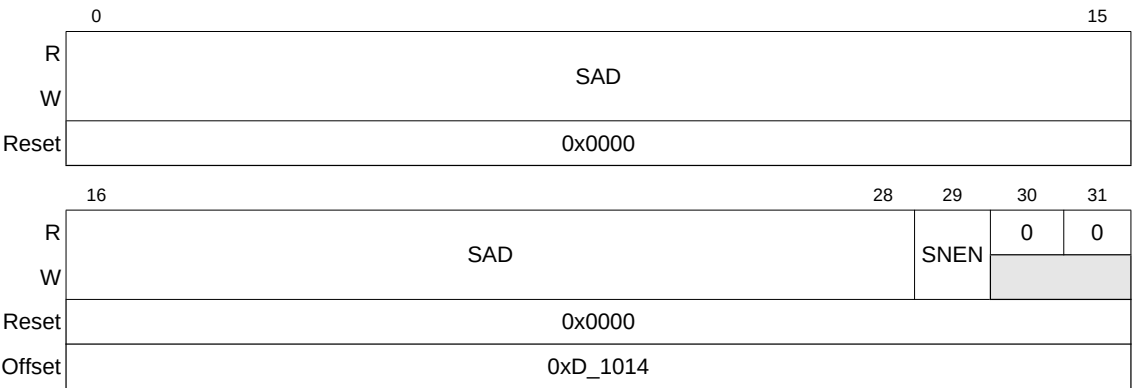


Figure 17-63. Outbound Unit Source Address Registers (OSAR)

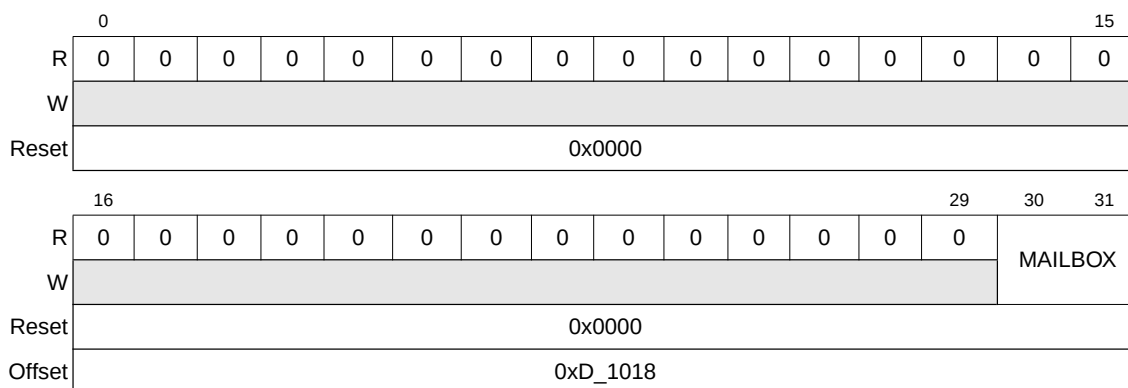
OSAR fields are described in Table 17-66.

Table 17-66. OSAR 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.
29	SNEN	Snoop enable 0 Snooping of the local processor caches disabled 1 Snooping of the local processor caches for data reads from local memory enabled
30–31	—	Reserved

17.3.4.1.5 Outbound Destination Port Register (ODPR)

The destination port register (ODPR) indicates the port to which the message unit controller is to send data. The software must ensure that this is a valid port in the receiving device. The ODPR is shown in Figure 17-64.

**Figure 17-64. Outbound Destination Port Registers (ODPR)**

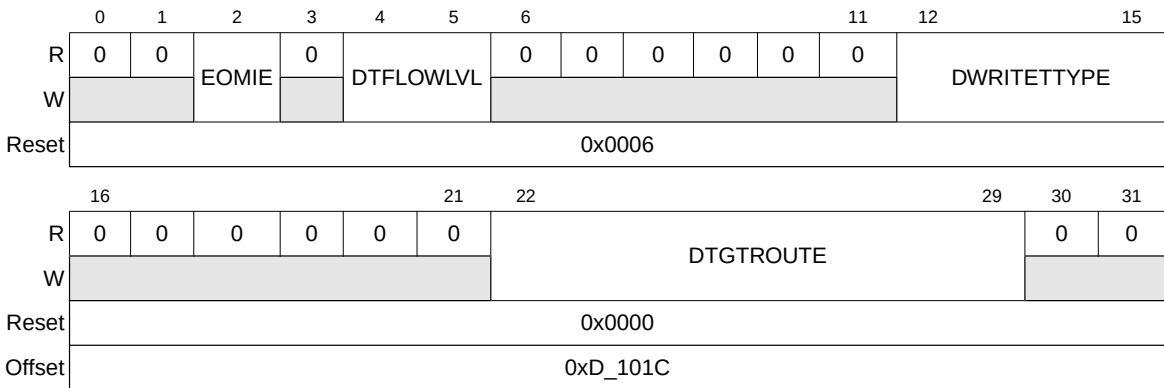
ODPR fields are described in Table 17-67.

Table 17-67. ODPR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30–31	MAILBOX	Value for mbox field in MESSAGE packet

17.3.4.1.6 Outbound Destination Attributes Register (ODATR)

The outbound destination attributes register (ODATR) contains the transaction attributes to be used for the message operation. The ODATR is shown in Figure 17-65.

**Figure 17-65. Outbound Destination Attributes Register (ODATR)**

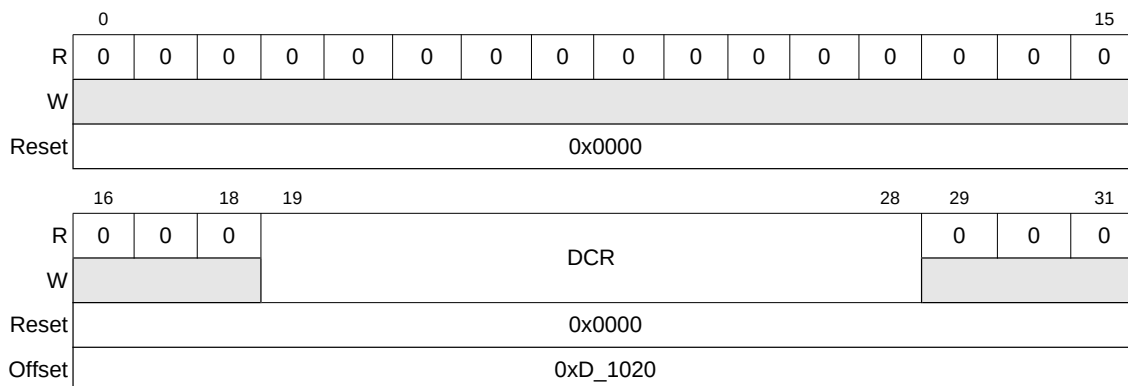
ODATR fields are described in Table 17-68.

Table 17-68. ODATR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	—	Reserved, hardwired to 0
2	EOMIE	End-of-message interrupt enable 0 End-of-message interrupt disabled 1 Generates an interrupt upon completion of the current message operation.
3	—	Reserved
4–5	DTFLOWLVL	Transaction flow level 00 Lowest priority transaction flow 01 Next highest priority transaction flow 10 Highest priority transaction flow 11 Reserved
6–11	—	Reserved
12–15	DWRITETYPE	Hardwired to 0110 (MESSAGE)
16–21	—	Reserved
22–29	DTGTRROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target)
30–31	—	Reserved

17.3.4.1.7 Outbound Double-Word Count Register (ODCR)

The outbound double-word count register (ODCR) contains the number of double-words for the message operation. The maximum message operation size is 4-kbytes and the minimum is eight bytes. The ODCR is shown in Figure 17-66.

**Figure 17-66. Outbound Double-word Count Register (ODCR)**

ODCR fields are described in Table 17-69.

Table 17-69. ODCR Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–28	DCR	Double-word Count Register. Contains the number of double-words for the message operation. The value in this register is decremented after each memory read 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 00 1000 0000 1024 bytes 01 0000 0000 2048 bytes 10 0000 0000 4096 bytes All other values yield undefined behavior
29–31	—	Reserved

17.3.4.1.8 Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)

The outbound descriptor queue enqueue pointer address register (ODQEPAR) contains the address for the next descriptor in memory to be added to the queue. Software must initialize this register to match the outbound 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 ODQEPAR), and then writes OMR[MUI]. The message unit then increments ODQEPAR to point to the next descriptor location in memory and clears OMR[MUI]. This can result in a number of actions:

- If the enqueue and dequeue pointers match, the queue is now full. If OMR[QFIE] is set, then OSR[QFI] 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 OMR[QOIE] is set, OSR[QOI] is set and an interrupt is generated.
- If the enqueue and dequeue pointer are the same before reading the register and no overflow condition is detected, the message unit controller starts (if enabled).

The ODQEPAR is shown in Figure 17-67.

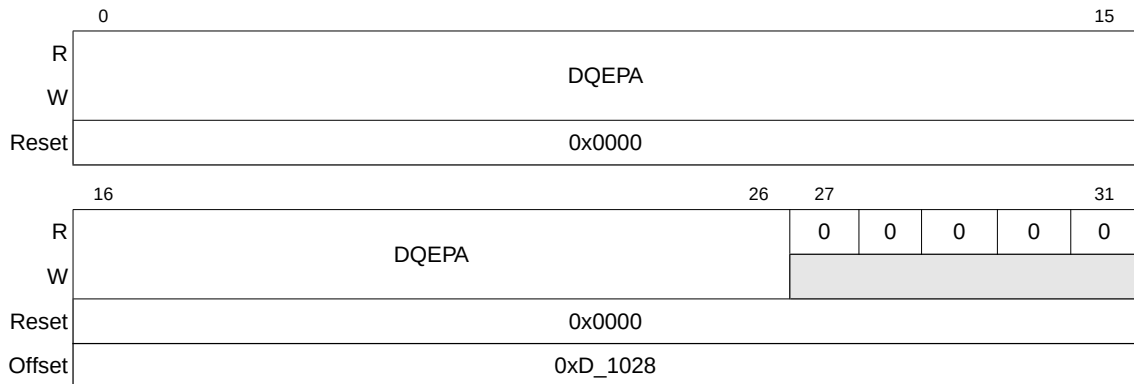


Figure 17-67. Outbound Descriptor Queue Enqueue Pointer Register (ODQEPAR)

ODQEPAR fields are described in Table 17-70.

Table 17-70. ODQEPAR 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.
27–31	—	Reserved

17.3.4.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

17.3.4.2.1 Inbound Mode Register (IMR)

The inbound mode register (IMR) allows software to enable the mailbox controller and to control various message operation characteristics. The IMR is shown in Figure 17-68.

Memory Map/Register Definition

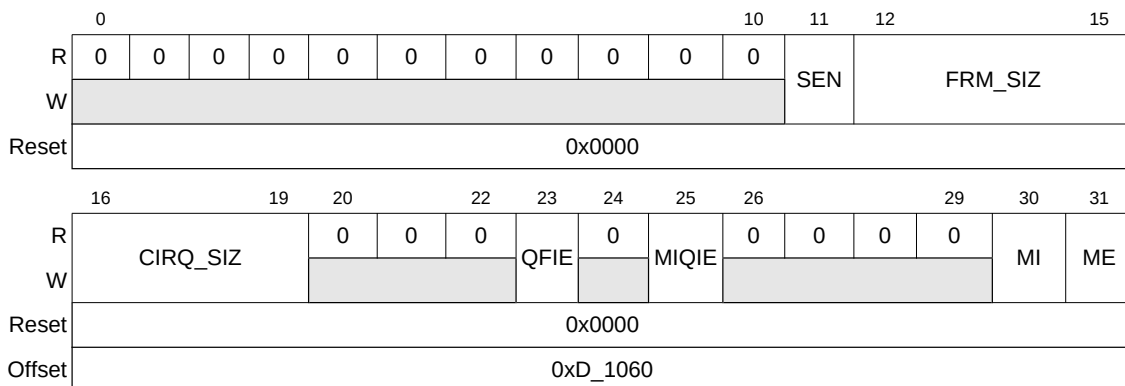


Figure 17-68. Inbound Mode Register (IMR)

IMR fields are described in Table 17-71.

Table 17-71. IMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping of the local processor while writing messages into memory disabled 1 Snooping of the local processor while writing messages into memory 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 ... 0010 8 bytes 0011 16 bytes 0100 32 bytes 0101 64 bytes 0110 128 bytes 0111 256 bytes 1000 512 bytes 1001 1024 bytes 1010 2048 bytes 1011 4096 bytes 1100 Reserved ... 1111 Reserved

Table 17-71. IMR Field Descriptions (continued)

Bits	Name	Description
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 determines the maximum contiguous memory space allocated to the mailbox. 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64 0110 128 (4-kbyte page boundary) 0111 256 1000 512 1001 1024 1010 2048 1011 Reserved ... 1111 Reserved
20–22	—	Reserved
23	QFIE	Queue full interrupt enable. 0 No QFI 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 mailbox controller).
24	—	Reserved
25	MIQIE	Message in queue interrupt enable. 0 No MIQ interrupt is generated 1 Generates an interrupt when the queue transitions to not empty (that is, the enqueue and dequeue pointers are no longer equal after an increment by the mailbox controller).
26–29	—	Reserved
30	MI	Mailbox increment. Software sets this bit after processing an inbound message. Hardware increments the IFQDPAR and clears this bit. Always reads as 0 while ME is set.
31	ME	Mailbox enable. 0 Mailbox disabled 1 The mailbox has been initialized and can service incoming message operations.

17.3.4.2.2 Inbound Status Register (ISR)

The inbound status register (ISR) reports various mailbox conditions during and after a message operation. Setting a condition bit clears it. The ISR is shown in Figure 17-69.

Memory Map/Register Definition

	0										10		11	12		14		15
R	0	0	0	0	0	0	0	0	0	0	0	QF	0	0	0	MIQ		
W																		
Reset	0x0000																	

	16							23							24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TE	0	0	QFI	0	MB	0	MIQI						
W																						
Reset	0x0000																					
Offset	0xD_1064																					

Figure 17-69. Inbound Status Register (ISR)

ISR fields are described in Table 17-72.

Table 17-72. ISR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full (Read-only) 0 Queue is not full 1 If the queue becomes full, then this bit is set.
12–14	—	Reserved
15	MIQ	Message-in-queue(Read-only) 0 No message in queue 1 If the queue goes from empty to not empty, then this bit is set.
16–23	—	Reserved
24	TE	Transaction error 0 No transaction error 1 This bit is set when there is an error condition during the message operation. (Bit reset, write 1 clear)
25–26	—	Reserved
27	QFI	Queue full interrupt. 0 No queue full interrupt condition 1 If the queue becomes full and OMR[QFIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 clear)
28	—	Reserved
29	MB	Mailbox busy (Read-only) 0 This bit is cleared as a result of receiving an error or the completion of a message operation. 1 When set indicates that a message operation is currently in progress.
30	—	Reserved
31	MIQI	Message-in-queue interrupt 0 No message-in-queue interrupt condition 1 If the queue goes from empty to not empty and OMR[MIQIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 clear)

17.3.4.2.3 Inbound Frame Queue Dequeue Pointer Address Register (IFQDPA)

The inbound frame queue dequeue pointer address register (IFQDPA) contains the address for the first message in memory to be processed. Software must initialize this register to the first frame location in memory. When a message has been processed, the processor sets IMR[MI]. The mailbox hardware then increments IFQDPA to point to the next frame in memory and clears the IMR[MI] bit. If the inbound frame queue enqueue pointer and the inbound 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. The IFQDPA is shown in Figure 17-70.

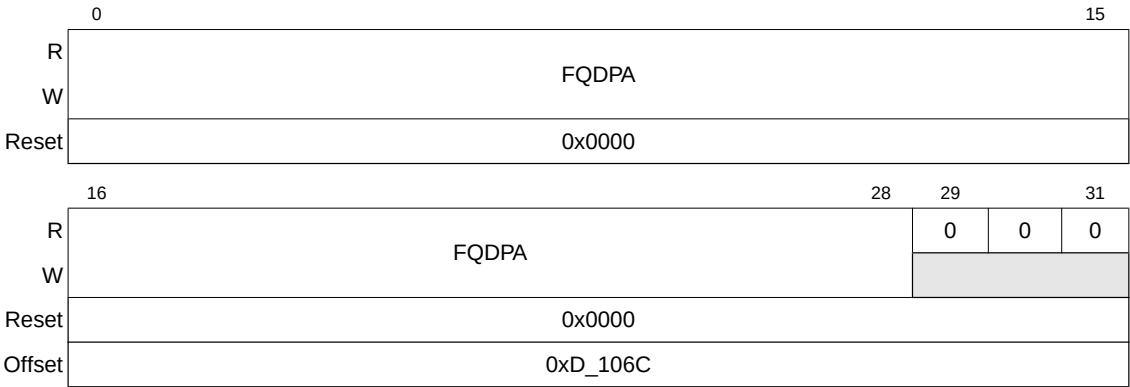


Figure 17-70. Inbound Frame Queue Dequeue Pointer Address Registers (IFQDPA)

IFQDPA fields are described in Table 17-73.

Table 17-73. IFQDPA 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

17.3.4.2.4 Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)

The inbound frame queue enqueue pointer address register (IFQEPAR) contains the address for the next message frame in memory to be added to the queue. Software must initialize this register 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 IFQEPAR) and then increments IFQEPAR 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 IMR[QFIE] is set, then ISR[QFI] is set and an interrupt is generated.
- If the enqueue and dequeue pointer were the same before the register was read, the queue has transitioned from empty to not empty. If IMR[MIQIE] is set, ISR[MIQI] is set and an interrupt is generated.

The IFQEPAR is shown in Figure 17-71.

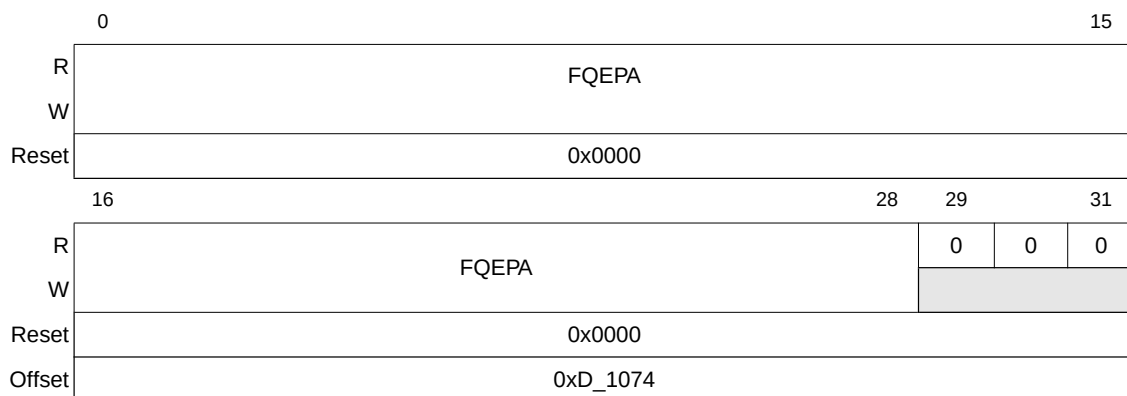


Figure 17-71. Inbound Frame Queue Enqueue Pointer Address Registers (IFQEPAR)

IFQEPAR fields are described in Table 17-74.

Table 17-74. IFQEPAR Field Descriptions

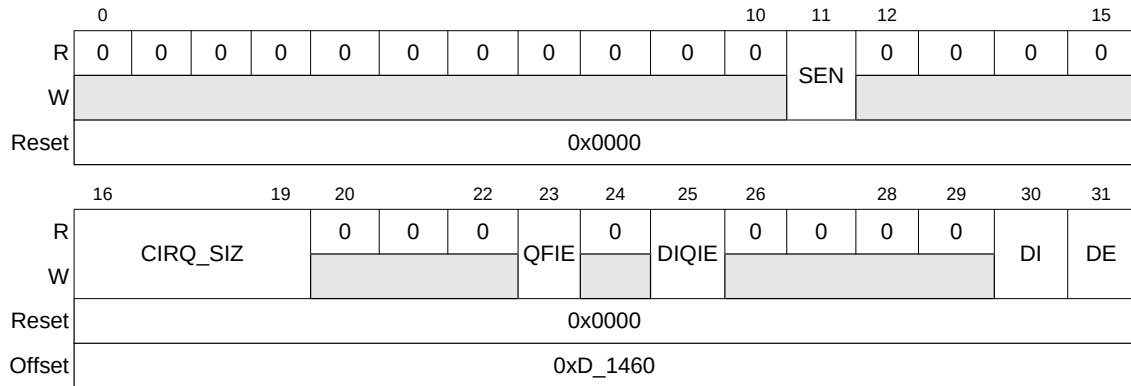
Bits	Name	Description
0–28	FQEPA	Frame queue enqueue pointer address. Contains the address of the next message frame to be added to the queue.
29–31	—	Reserved

17.3.4.3 RapidIO Doorbell Registers

These registers control the RapidIO doorbell unit. The following sections provide descriptions of these registers.

17.3.4.3.1 Doorbell Mode Register (DMR)

The doorbell mode register (DMR) allows software to enable the doorbell controller to control various doorbell operation characteristics. The DMR is shown in Figure 17-72.

**Figure 17-72. Doorbell Mode Register (DMR)**

DMR fields are described in Table 17-75.

Table 17-75. DMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping of the local processor while writing doorbell information into memory disabled 1 Snooping of the local processor while writing doorbell information into memory 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. 0000 2 0001 4 0010 8 0011 16 0100 32 0101 64 0110 128 0111 256 1000 512 (4-kbyte page boundary) 1001 1024 1010 2048 1011 Reserved ... 1111 Reserved
20–22	—	Reserved
23	QFIE	Queue full interrupt enable 0 No QOI 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).
24	—	Reserved

Table 17-75. DMR Field Descriptions (continued)

Bits	Name	Description
25	DIQIE	Doorbell in queue interrupt enable. 0 No MIQ interrupt is generated 1 Generates an interrupt when the queue transitions to not empty (that is, the enqueue and dequeue pointers are no longer equal after an increment by the doorbell controller).
26–29	—	Reserved
30	DI	Doorbell increment . Software sets this bit after processing an inbound doorbell. Hardware increments the DQDPAR and clears this bit. Always reads as 0 while DE is set.
31	DE	Doorbell enable 0 Doorbell disabled 1 The doorbell has been initialized and can service incoming doorbell operations.

17.3.4.3.2 Doorbell Status Register (DSR)

The doorbell status register (DSR) reports various doorbell conditions after a doorbell operation. Writing a 1 to the corresponding set bit clears it. The DSR is shown in Figure 17-73.

	0										10	11	12		14	15
R	0	0	0	0	0	0	0	0	0	0	0	QF	0	0	0	DIQ
W																
Reset	0x0000															
	16							23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TE	0	0	QFI	0	DB	0	DIQI
W																
Reset	0x0000															
Offset	0xD_1464															

Figure 17-73. Doorbell Status Register (DSR)

DSR fields are described in Table 17-76.

Table 17-76. DSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full (Read-only) 00 Queue is not full) 1 If the queue becomes full, then this bit is set.
12–14	—	Reserved
15	DIQ	Doorbell-in-queue (Read-only) 0 No doorbell-in-queue conditions) 1 If the queue goes from empty to not empty, then this bit is set.

Table 17-76. DSR Field Descriptions (continued)

Bits	Name	Description
16–23	—	Reserved
24	TE	Transaction error 0 There was no error condition during the doorbell operation. 1 There was an error condition during the doorbell operation. (Bit reset, write 1 clear)
25–26	—	Reserved
27	QFI	Queue full interrupt. 0 No queue full interrupt condition 1 The queue became full and the queue fill interrupt is enabled (IMR[QFIE] = 1). An interrupt is generated. (Bit reset, write 1 clear)
28	—	Reserved
29	DB	Doorbell busy (Read-only) 0 This bit is cleared as a result of receiving an error or the completion of a doorbell operation. 1 Indicates that a doorbell operation is currently in progress.
30	—	Reserved
31	DIQI	Doorbell-in-queue interrupt 0 No doorbell-in-queue interrupt condition 1 The queue went from empty to not empty and OMR[DIQIE] is set. An interrupt is generated. (Bit reset, write 1 clear)

17.3.4.3.3 Doorbell Queue/Dequeue Pointer Address Registers (DQDPAR)

The doorbell queue dequeue pointer address register (DQDPAR) contains the double-word address for the first doorbell in memory to be processed. Software must initialize this register to the first doorbell entry location in memory. When a current doorbell has been processed, the processor sets DMR[DI]. The doorbell hardware then increments the doorbell queue dequeue pointer address register to point to the next doorbell in memory and clear DMR[DI].

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 DQDPAR is shown in Figure 17-74.

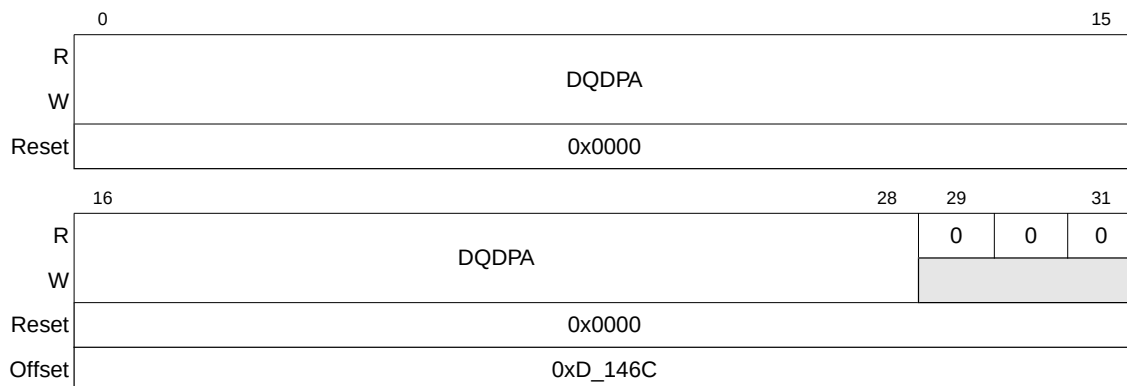


Figure 17-74. Doorbell Queue Dequeue Pointer Address Registers (DQDPA)

DQDPA fields are described in Table 17-77.

Table 17-77. DQDPA 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

17.3.4.3.4 Doorbell Queue/Enqueue Pointer Address Registers (DQEPAR)

The doorbell queue enqueue pointer address register (DQEPAR) contains the double-word address for the next doorbell entry in memory to be added to the queue. Software must initialize DQEPAR 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 DQEPAR) and then increments DQEPAR to point to the next doorbell location in memory. This can result in a number of actions:

- If the enqueue and dequeue pointers match, then the queue is now full and the doorbell controller does not accept any more incoming doorbell packets, returning RETRY responses to the sending devices until the queue is no longer full. If DMR[QFIE] is set, then DSR[QFI] is set and an interrupt is generated.
- If the enqueue and dequeue pointer were the same before reading the register, the queue has transitioned from empty to not empty. If DRM[MIQIE] is set, then DSR[MIQI] is set and an interrupt is generated.

The DQEPAR is shown in Figure 17-75.

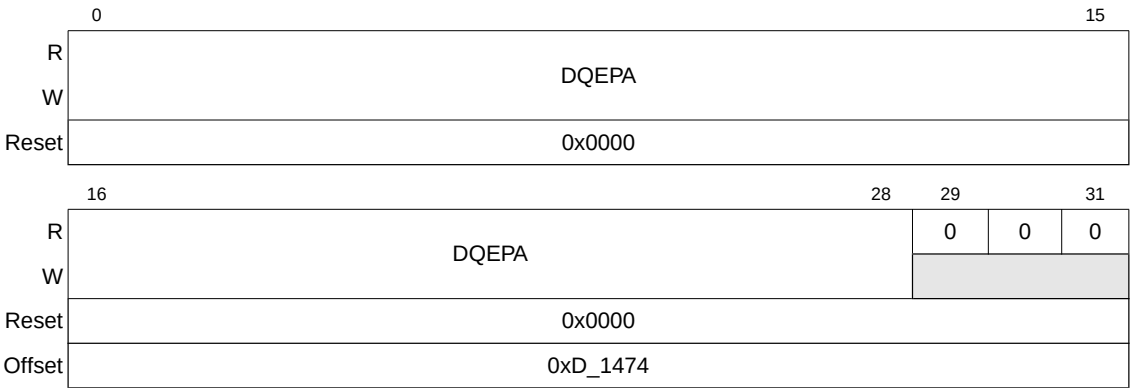


Figure 17-75. Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

DQEPAR fields are described in Table 17-78.

Table 17-78. DQEPAR 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.
29–31	—	Reserved

17.3.4.4 RapidIO Port-Write Registers

These registers control the RapidIO port-write unit. The following sections provide partial descriptions of these registers.

17.3.4.4.1 Port-Write Mode Register (PWMR)

The port-write mode register (PWMR) allows software to enable the port-write controller and to control various port-write operation characteristics. The PWMR is shown in Figure 17-76.

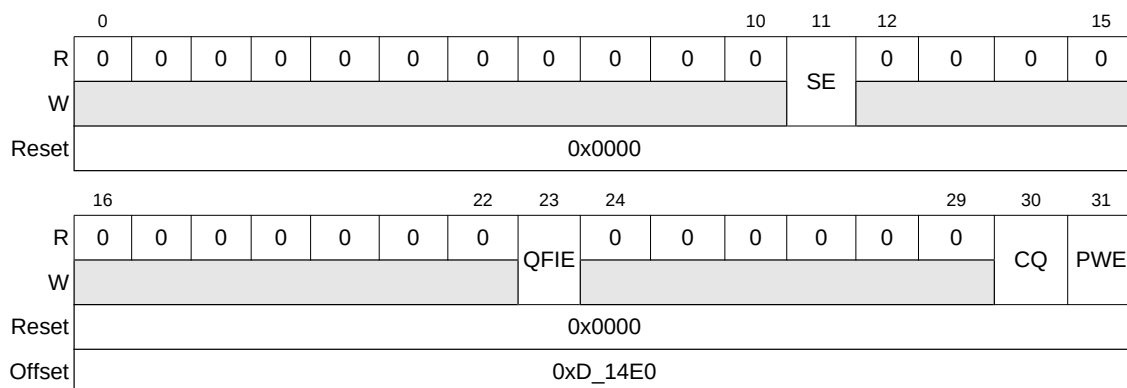


Figure 17-76. Port-Write Mode Register (PWMR)

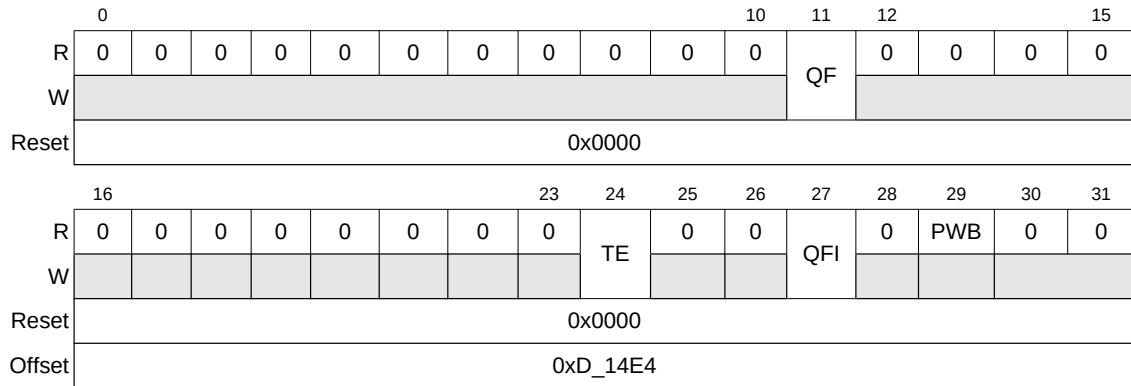
PWMR fields are described in Table 17-79.

Table 17-79. PWMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping the local processor while writing port-write data payload into memory disabled 1 Snooping the local processor while writing port-write data payload into memory enabled
12–22	—	Reserved
23	QFIE	Queue full interrupt enable 0 No QOI interrupt is generated 1 Generates an interrupt when the queue becomes full (that is, the controller has written an port-write data payload into memory).
24–29	—	Reserved
30	CQ	Clear queue. Software sets this bit after processing an inbound port-write operation. Hardware clears the previous port-write operation from the port-write queue and clears this bit. Always reads as 0 while We is set.
31	PWE	Port write enable 0 Port writes are disabled 1 The port-write controller has been initialized and can service an incoming port-write operation.

17.3.4.4.2 Port-Write Status Register (PWSR)

The port-write status register (PWSR) reports various port-write conditions after a port-write operation. Writing a 1 to the corresponding set bit clears the bit. The PWSR is shown in Figure 17-77.

**Figure 17-77. Port-Write Status Register (PWSR)**

PWSR fields are described in Table 17-80.

Table 17-80. PWSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full (Read-only) 0 The queue is not full 1 The queue became full
12–23	—	Reserved
24	TE	Transaction error (Bit reset, write 1 clear) 0 No transaction error occurred 1 There was an error condition during the port-write operation. The detected errors are: - If the port-write transaction is greater than 64 bytes. - A port write is received and the controller is not enabled.
25–26	—	Reserved
27	QFI	Queue full interrupt. 0 This bit must be manually cleared by software if a port write is serviced to re-enable the port-write controller. 1 The queue became full and OMR[QFIE] is set. An interrupt is generated. (Bit reset, write 1 clear)
28	—	Reserved
29	PWB	Port-write busy (Read-only) 0 This bit is cleared as a result of receiving an error or the completion of a port-write operation. 1 Indicates that a port-write operation is currently in progress.
30–31	—	Reserved

17.3.4.4.3 Port-Write Queue Base Address Register (PWQBAR)

The port-write queue base address register (PWQBAR) contains the 64-byte cache line address for the port-write data payload. Software must initialize this register to the desired location in memory. The PWQBAR is shown in Figure 17-78.

Functional Description

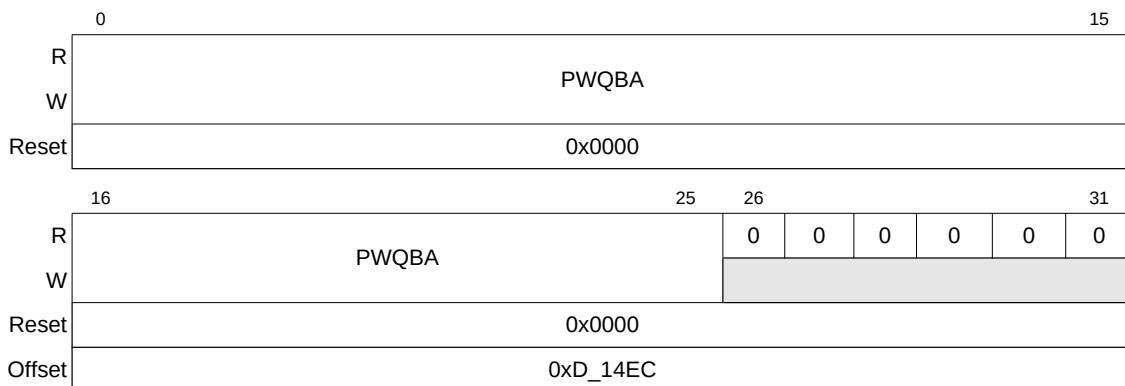


Figure 17-78. Port-Write Queue Base Address Register (PWQBAR)

PWQBAR fields are described in Table 17-81.

Table 17-81. PWQBAR Field Descriptions

Bits	Name	Description
0–25	PWQBA	Port-write queue base address. Contains the cache line address of the port-write data payload.
26–31	—	Reserved

17.4 Functional Description

17.4.1 RapidIO Transaction, Packet, and Control Symbol Summary

Table 17-82 summarizes the transaction types supported by the RapidIO controller. The table lists the packet format type, the RapidIO transaction, the programming model where the transaction is used, the processor operations that may cause the RapidIO transaction, possible responses for the transactions, and comments that note RapidIO-specific details associated with the transaction, or response. Transactions that are not supported by the RapidIO implementation are noted in the comments section of the table.

Table 17-82. RapidIO Transaction Summary

Packet Format	RapidIO Transaction	Programming Model	Possible Processor Operations	Possible RapidIO Response	Comments
2 non-intervention request class	IO_read_home	GSM	I = 1, M = 1 loads, and instruction fetches where the target of the request is a remote address space. M = 0 read type operations may also map to io_read_home.	done, (data_only + done_interv) retry, error	None
2 non-intervention request class	NREAD	IO	M = 0 read type operations where the target of the request is a remote address space	done, error	None
	atomic	IO	M = 0 read type mapped operation	done, error	Atomic transactions should always map in the ATMU to the DDR memory controller, or the behavior is boundedly undefined.
5 Write class	flush (with data)	GSM	M = 1, W = 1 or I = 1, store. M = 0 store type operation may also map to flush.	done, retry, error	None
5 (continued)	atomic swap	IO	N/A	N/A	RapidIO doesn't support atomic swap transactions.
	nwrite	IO	M = 0 store type operation	N/A	None
	nwrite_r	IO	M = 0 store type operation	done, error	None
6 Streaming write class	SWRITE	IO	M = 0 store type operation	N/A	None

Table 17-82. RapidIO Transaction Summary (continued)

Packet Format	RapidIO Transaction	Programming Model	Possible Processor Operations	Possible RapidIO Response	Comments
8 Maintenance class	Maintenance read	All	M = 0 read type operation	done, error	None
	maintenance write	All	M = 0 write type operation	done, error (port write doesn't get a response)	None
10 Doorbell class	DOORBELL	Message passing	M = 0 write type operation	done, retry, error	None
11 Message class	MESSAGE	Message passing	M = 0 write type operation	done, retry, error	None
13 Response class	RESPONSE	All	N/A	Depends on the request	

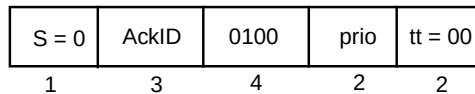
Table 17-83 summarizes the packet formats for all the defined packet formats. The following definitions and Figure 17-79 accompany the table. The bits in the physical maintenance field (PMF) are prefixed to each packet format.

- PMF—physical maintenance field
- TYPE —packet type (1, 2, 5, 6, 8, 10, 11, 13)
- TARID —target_id: the receiver's ID within its domain
- SRCTID— the packet transaction ID
- TTYPE—transaction type to be performed by the recipient
- RDSIZE—read data size requested
- WRSIZE—write data size requested
- SRCID—source_id: the packet sender's ID within its domain
- ADDR—address for the requested operation
- MSB—most significant byte
- LSB —least significant byte
- U—use hop_count to address switches that have no domain/device ID
- HOPCNT—hop_count selects the distance into the switch network to access
- CONFIG_OFFSET—double-word offset into the CAR/CSR register file
- INFO—doorbell information field - provided and used by software only
- MSGLEN—total number of packets comprising the message
- MSGSEG—portion of the message supplied by the packet

- SSIZE—standard message packet data size
- MBOX—recipient mailbox number at the target device
- LETTER—identifies message within the mailbox
- STATUS—packet response status field
- TARINFO—target_info used for messages only. Field length is eight bits: letter (2) + mbox (2) + msgseg (4)
- W—word pointer: used in the decoding of the transaction size

NOTE

Shaded areas in the table indicate reserved fields.

**Figure 17-79. Physical Maintenance Field (PMF) Definition****Table 17-83. RapidIO Packet Format Summary (tt = 0b00)**

Type	Packet Format																						
2 (0010)	Request To Home: <table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>RDSIZE</td><td>SRCTID</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr></table> <table><tr><td>ADDRESS</td><td>W</td><td>ADDRESS</td></tr><tr><td>29</td><td>1</td><td>2</td></tr></table> TTYPE:0010 (IO_READ_HOME); 0100 (NREAD); 1100 (ATOMIC INC); 1101 (ATOMIC DEC); 1110 (ATOMIC SET); 1111 (ATOMIC CLR)	PMF	TYPE	TARID	SRCID	TTYPE	RDSIZE	SRCTID	12	4	8	8	4	4	8	ADDRESS	W	ADDRESS	29	1	2		
PMF	TYPE	TARID	SRCID	TTYPE	RDSIZE	SRCTID																	
12	4	8	8	4	4	8																	
ADDRESS	W	ADDRESS																					
29	1	2																					
5 (0101)	Write Class: <table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>WRSIZE</td><td>SRCTID</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr></table> <table><tr><td>ADDRESS</td><td>W</td><td>ADDRESS</td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>29</td><td>1</td><td>2</td><td></td></tr></table> TTYPE:0001 (FLUSH); 0010–0011 (Reserved); 0100 (NWRITE); 0101 (NWRITE_R); 0110–0111 (Reserved);	PMF	TYPE	TARID	SRCID	TTYPE	WRSIZE	SRCTID	12	4	8	8	4	4	8	ADDRESS	W	ADDRESS	MSB for DW0 --> LSB for DWn	29	1	2	
PMF	TYPE	TARID	SRCID	TTYPE	WRSIZE	SRCTID																	
12	4	8	8	4	4	8																	
ADDRESS	W	ADDRESS	MSB for DW0 --> LSB for DWn																				
29	1	2																					
6 (0110)	Streaming-Write Class: <table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>ADDRESS</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>29</td></tr></table> <table><tr><td></td><td>ADDRESS</td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>1</td><td>2</td><td></td></tr></table>	PMF	TYPE	TARID	SRCID	ADDRESS	12	4	8	8	29		ADDRESS	MSB for DW0 --> LSB for DWn	1	2							
PMF	TYPE	TARID	SRCID	ADDRESS																			
12	4	8	8	29																			
	ADDRESS	MSB for DW0 --> LSB for DWn																					
1	2																						

Table 17-83. RapidIO Packet Format Summary (tt = 0b00) (continued)

Type	Packet Format														
8 (1000)	Maintenance Class Request:														
	<table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>RDSIZE</td><td>SRCTID</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr></table>	PMF	TYPE	TARID	SRCID	TTYPE	RDSIZE	SRCTID	12	4	8	8	4	4	8
	PMF	TYPE	TARID	SRCID	TTYPE	RDSIZE	SRCTID								
	12	4	8	8	4	4	8								
	<table><tr><td>HOPCNT</td><td>CONFIG_OFFSET</td><td>W</td><td></td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>8</td><td>21</td><td>1</td><td>2</td><td></td></tr></table>	HOPCNT	CONFIG_OFFSET	W		MSB for DW0 --> LSB for DWn	8	21	1	2					
HOPCNT	CONFIG_OFFSET	W		MSB for DW0 --> LSB for DWn											
8	21	1	2												
TTYPE: 0000 (READ); 0001 (WRITE); 0100 (PORT WRITE)															
Maintenance Class Response:															
10 (1010)	<table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>STATUS</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td></tr></table>	PMF	TYPE	TARID	SRCID	TTYPE	STATUS	12	4	8	8	4	4		
	PMF	TYPE	TARID	SRCID	TTYPE	STATUS									
	12	4	8	8	4	4									
	<table><tr><td>TARTID</td><td>HOPCNT</td><td></td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>8</td><td>8</td><td>24</td><td></td></tr></table>	TARTID	HOPCNT		MSB for DW0 --> LSB for DWn	8	8	24							
	TARTID	HOPCNT		MSB for DW0 --> LSB for DWn											
8	8	24													
TTYPE: 0010 (READ RESP); 0011 (WRITE RESP)															
STATUS: 0000 (DONE); 0111 (ERROR);															
11 (1011)	Doorbell Class:														
	<table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td></td><td>SRCTID</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>8</td><td>8</td></tr></table>	PMF	TYPE	TARID	SRCID		SRCTID	12	4	8	8	8	8		
	PMF	TYPE	TARID	SRCID		SRCTID									
12	4	8	8	8	8										
<table><tr><td>INFO</td></tr><tr><td>16</td></tr></table>	INFO	16													
INFO															
16															
13 (1101)	Message Class:														
	<table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>MSGLEN</td><td>SSIZE</td><td>LETTER</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>2</td></tr></table>	PMF	TYPE	TARID	SRCID	MSGLEN	SSIZE	LETTER	12	4	8	8	4	4	2
	PMF	TYPE	TARID	SRCID	MSGLEN	SSIZE	LETTER								
12	4	8	8	4	4	2									
<table><tr><td>MBOX</td><td>MSGSEG</td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>2</td><td>4</td><td></td></tr></table>	MBOX	MSGSEG	MSB for DW0 --> LSB for DWn	2	4										
MBOX	MSGSEG	MSB for DW0 --> LSB for DWn													
2	4														
13 (1101)	Response Class:														
	<table><tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>STATUS</td></tr><tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td></tr></table>	PMF	TYPE	TARID	SRCID	TTYPE	STATUS	12	4	8	8	4	4		
	PMF	TYPE	TARID	SRCID	TTYPE	STATUS									
	12	4	8	8	4	4									
<table><tr><td>TARTID</td><td>MSB for DW0 --> LSB for DWn</td></tr><tr><td>8</td><td></td></tr></table>	TARTID	MSB for DW0 --> LSB for DWn	8												
TARTID	MSB for DW0 --> LSB for DWn														
8															
STATUS: 0000 (DONE); 0001 (DATA_ONLY); 0011 (RETRY); 0101 (DONE_INTERVENTION); 0110 (Reserved); 0111 (ERROR); 1000–1011 (Reserved); 1100–1111 (User) TTYPE: 0000 (REQUEST RESP); 0001 (MESSAGE RESP); 1000 (REQUEST RESP WITH DATA)															

Table 17-84 summarizes the defined control symbols, and RapidIO specific details are noted in the comments section of the table.

Table 17-84. Control Symbol Formats

Symbol Type	Symbol Format	Comments												
0 (000)	Packet accepted: <table><tr><td>S = 1</td><td>TgtAckID</td><td>00000</td><td>BUF_STATUS</td><td>000</td></tr><tr><td>1</td><td>3</td><td>5</td><td>4</td><td>3</td></tr></table>	S = 1	TgtAckID	00000	BUF_STATUS	000	1	3	5	4	3	MPC8540 sets buf_status to 0xf when it generates a packet accepted control symbol.		
S = 1	TgtAckID	00000	BUF_STATUS	000										
1	3	5	4	3										
1 (001)	Packet retry: <table><tr><td>S = 1</td><td>TgtAckID</td><td>00000</td><td>0000</td><td>001</td></tr><tr><td>1</td><td>3</td><td>5</td><td>4</td><td>3</td></tr></table>	S = 1	TgtAckID	00000	0000	001	1	3	5	4	3	None		
S = 1	TgtAckID	00000	0000	001										
1	3	5	4	3										
2 (010)	Packet not accepted: <table><tr><td>S = 1</td><td>TgtAckID</td><td>00000</td><td>1</td><td>CAUSE</td><td>010</td></tr><tr><td>1</td><td>3</td><td>5</td><td>1</td><td>3</td><td>3</td></tr></table>	S = 1	TgtAckID	00000	1	CAUSE	010	1	3	5	1	3	3	Cause 000: Encountered internal error 001: Received unexpected ackID on packet 010: Uncorrectable error on control symbol 011: Input port has been stopped 100: Received bad CRC on packet 111: General error
S = 1	TgtAckID	00000	1	CAUSE	010									
1	3	5	1	3	3									
4 (100)	Packet control: <table><tr><td>S = 1</td><td>SUB_TYPE</td><td>00000</td><td>CONTENTS</td><td>100</td></tr><tr><td>1</td><td>3</td><td>5</td><td>4</td><td>3</td></tr></table>	S = 1	SUB_TYPE	00000	CONTENTS	100	1	3	5	4	3	Sub_type (contents) 000: Idle (MPC8540 sets to 1111) 001: Stomp (unused) 010: EOP (MPC8540 sets to 1111) 011: Restart From Retry (unused) 100: Throttle (number of aligned pacing idles) 101: TOD-sync (number of max sized packets)		
S = 1	SUB_TYPE	00000	CONTENTS	100										
1	3	5	4	3										
5 (101)	Link request: <table><tr><td>S = 1</td><td>CMD</td><td>00000</td><td>BUF_STATUS</td><td>100</td></tr><tr><td>1</td><td>3</td><td>5</td><td>4</td><td>3</td></tr></table>	S = 1	CMD	00000	BUF_STATUS	100	1	3	5	4	3	MPC8540 sets buf_status to 0xf when it generates a link request Cmd 000 Send training 010 Override width 011 Reset 100 Input status		
S = 1	CMD	00000	BUF_STATUS	100										
1	3	5	4	3										
6 (110)	Link response: <table><tr><td>S = 1</td><td>ACK_STAT</td><td>00000</td><td>LINK_STAT</td><td>100</td></tr><tr><td>1</td><td>3</td><td>5</td><td>4</td><td>3</td></tr></table>	S = 1	ACK_STAT	00000	LINK_STAT	100	1	3	5	4	3	Ack status 000: Expecting ackID 0 001: Expecting ackID 1 111: Expecting ackID 7 Link status 0000: Un-initialized 0001: Initializing 0010: Error 0011: Stopped 0100: Retry-stopped 0101: Error-stopped 1000: OK, ackID 0 111: OK, ackID 7		
S = 1	ACK_STAT	00000	LINK_STAT	100										
1	3	5	4	3										

17.5 RapidIO Functionality

17.5.1 General Device Functionality Checklists

The general device functionality checklists are applicable to devices of all classes, including end point free switch devices.

17.5.1.1 General Specification Functionality List

The list shown in Table 17-85 specifies general requirements for a compliant device.

Table 17-85. General Device Functionality List

Item No.	Item Functionality	Meets Reqs?
1	All device generated packets must adhere to the bit streams defined in the appropriate logical, transport, and physical layer specifications	Y
2	All device generated control symbols must adhere to the bit streams defined in the physical layer specification	Y
3	Device assigns reserved packet fields to logic 0s	Y
4	Device does not use reserved packet formats	Y
5	Reserved packet field contents are ignored	Y
6	Device never assigns reserved encodings for packet fields	Y
7	Implementation-defined packet fields are ignored unless the function is understood by the receiving device	Y
8	Received reserved packet field encodings are ignored if it is not necessary for the field to be defined for the requested transaction	Y
9	Reads to reserved CAR bits return logic 0s	Y
10	Writes to reserved CAR bits are ignored	Y
11	Reads to implementation-defined CAR bits return the implementation defined value	Y
12	Writes to implementation-defined CAR bits are ignored	Y
13	Reads to reserved CARs do not cause an error	Y
14	Reads to reserved CARs return logic 0s when read	Y
15	Writes to reserved CARs do not cause an error	Y
16	Writes to reserved CARs are ignored	Y
17	Reads to reserved CSRs and Extended Features register bits return logic 0s	Y
18	Writes to reserved CSRs and Extended Features register bits are ignored	Y
19	Reads to implementation-defined CSR and Extended Features register bits return the implementation defined value	Y
20	Writes to implementation-defined CSR and Extended Features register bits are determined by the implementation	Y

Table 17-85. General Device Functionality List (continued)

Item No.	Item Functionality	Meets Reqs?
21	Reads to reserved CSRs and Extended Features registers do not cause an error	Y
22	Reads to reserved CSRs and Extended Features registers return logic 0s when read	Y
23	Writes to reserved CSRs and Extended Features registers do not cause an error	Y
24	Writes to reserved CSRs and Extended Features registers are ignored	Y
25	Reads and writes to implementation-defined registers and register bits is implementation-defined	Y

17.5.1.2 8/16 LP-LVDS Layer Functionality Lists

The physical layer 8/16 LP-LVDS layer functionality is broken down into a number of sub-lists.

17.5.1.3 Basic Functionality List

Table 17-86 lists requirements for basic functionality for the device.

Table 17-86. General Device 8/16 LP-LVDS Physical Layer Basic Functionality List

Item No.	Item Functionality	Meets Reqs?
1	CLK signal(s) rising edge is on the 32-bit boundary	Y
2	CLK signal(s) toggle every clock (true free running clock)	Y
3	RIO_TFRAME/RIO_RFRAME signal only toggles on the 32-bit boundary	Y
4	RIO_TFRAME/RIO_RFRAME signal only toggles to mark the first byte of a control symbol or the first byte of a new packet	Y
5	RIO_RFRAME/RIO_TFRAME signal is associated with RIO_RCLK/RIO_TCLK	Y
6	Device meets target AC specifications	Y
7	Device must follow the link initialization and training procedure	Y
	7A ACTION: Device must respond to link-request/send-training control symbol with 256 training patterns followed by at least one idle control symbol	Y
	7B DETAIL: Device port is enabled to transmit or receive packets only after device is both sending and receiving idle control symbols	Y
	7C DETAIL: Training pattern is not embedded in a packet or used to terminate a packet.	Y
8	Training symbol is correct for port width	Y
9	16-bit ports must be able to function as 8-bit ports	Y
10	Control symbols are 16 bits followed by an inverted 16-bit copy	Y
11	Undefined control symbols are treated as idle control symbols	Y

**Table 17-86. General Device 8/16 LP-LVDS Physical Layer
Basic Functionality List (continued)**

Item No.	Item Functionality	Meets Reqs?
12	Device responds to a throttle control symbol with a pacing idle control symbol within the specified 60 data ticks time	Y
13	Device follows native RapidIO ordering rules at flow level 1	Y
	13A. DETAIL: Read packets are transmitted at priority level 1	Y
	13B. DETAIL: Write packets are transmitted at priority level 1	Y
	13C. DETAIL: Response packets are transmitted at priority level 1, 2, or 3	Y
	13D. DETAIL: Packets of the same priority level can not pass each other	Y
	13E. DETAIL: Packets of higher priority may pass packets of a lower priority level	Y
14	All registers in the 8/16 LP-LVDS Physical Layer Specification Extended Features data structure can be read and written	Y
	14A. DETAIL: Port Maintenance Block Header 0 CSR; reset value is implementation dependant (read-only)	Y
	14B. DETAIL Port Maintenance Block Header 0 CSR has the proper Extended Features block ID	Y
	14C. DETAIL: Port Link Time-out Control CSR; reset value is 0xFFFF_FFFF	Y
	14D. DETAIL: Port Response Time-out Control CSR; reset value is 0xFFFF_FFFF (end point devices only)	Y
	14E. DETAIL: Port General Control CSR; reset value is implementation dependant	Y
	14F. DETAIL: Port 0 Error and Status CSR; reset value is 0x0000_0001	Y
	14G. DETAIL: Port 0 Control CSR; reset value is implementation dependant	Y
15	Device follows system initialization and exploration inter-operability procedure	Y
	15A. ACTION: End point agent device that contain a boot ROM is initially assigned base device ID = 0xFE at power-up	Y
	15B. ACTION: End point agent device without a boot ROM is initially assigned base device ID = 0xFF at power-up	Y
	15C. ACTION: System host device is initially assigned a non-reserved base device ID at power-up (end point device only)	Y
	15D. ACTION: Host device sets PGCCSR[D] at power-up (end point device only)	Y
	15E. ACTION: Agent device resets PGCCSR[D] at power-up (end point device only)	Y
	15F. ACTION: Host device sets the Master Enable bit at power-up (end point device only)	Y
	15G. ACTION: Agent device resets the Master Enable bit at power-up (end point device only)	Y
	15H. ACTION: End point device must respond to all received maintenance request packets	Y
	15I. ACTION: Device with switch functionality defaults route information to allow boot ROM access by system host	Y
	15J. ACTION: Switch device must send maintenance responses that it generates to the port that the maintenance request was received on	Y

17.5.1.4 General Device Link Maintenance List

Table 17-87 lists requirements for link maintenance functionality.

Table 17-87. General Device 8/16 LP-LVDS Physical Layer Link Maintenance List

Item No.	Item Functionality	Meets Reqs?
1	Link-request/input-status control symbol causes a link-response control symbol	Y
2	Link-response control symbol link_status field indicates input port status	Y
	2A. DETAIL: Uninitialized means disabled	Y
	2B. DETAIL: Initializing means following the training sequence	Y
	2C. DETAIL: Error means unrecoverable problem encountered; no packets are accepted	Y
	2D. DETAIL: Retry-stopped means that input port is stopped due to a retry; no packets are accepted until receiving a restart-from-retry or a restart-from-error control symbol	Y
	2E. DETAIL: Error-stopped means that input port is stopped due to a transmission error; no packets are accepted until receiving a restart-from-error control symbol	Y
	2F. DETAIL: Stopped means that input is stopped for some other (not retry or transmission error) reason; no packets are accepted	Y
	2G. DETAIL: OK means that the input port is operating and can communicate with the connected device	Y
3	Link-request/send-training control symbol causes 256 training patterns to be sent	Y
4	Link-request/reset control symbol causes the device to reset after four requests	Y
5	Link-request/input-status, link-response control symbol pair forces completion of all preceding activity	Y
	5A. DETAIL: All preceding packets have generated acknowledge control symbol if applicable	Y
	5B. DETAIL: Returned link_status correctly indicates next expected ackID	Y
6	Device doesn't issue more than one outstanding link-request control symbol (exception in case of time-out)	Y
	6A. DETAIL: Must hold off another link-request control symbol until expected link-response control symbol is received	Y

17.5.1.5 General Device Packet Transmission List

Table 17-88 lists requirements for the device while transmitting a packet (a general device is required to generate response packets).

Table 17-88. General Device 8/16 LP-LVDS Physical Layer Packet Transmission List

Item No.	Item Functionality	Meets Reqs?
1	Device cannot issue more than eight unacknowledged packets	Y
2	ackIDs are always issued sequentially unless a retry or error causes the device to back up	Y
3	Start with ackID = 0 after reset	Y
4	Physical layer three reserved bits are cleared	Y
5	A response packet for a request packet can never be transmitted before the acknowledge control symbol for the request packet	Y
6	Embedded CRC properly inserted	Y
7	Packets that are not aligned to the 32-bit boundary are padded into alignment with 16 bits of logic 0	Y
8	CRC values are correctly calculated	Y
9	Switch devices preserve error coverage internally	Y
10	Packets are completed normally with a new packet or EOP control symbol	Y
11	Packets are canceled using stomp, restart-from-retry, or link-request control symbols	Y
12	Stomp and restart-from-retry control symbols have the cause field set to all logic 0s	Y
13	Device receiving packet-retry control symbol follows the defined retry behavior	Y
	13A. ACTION: Move from the Port OK state to the Output Retry-stopped state when a packet-retry control symbol is received	Y
	13B. ACTION: Complete transmission of current packet if applicable, don't transmit new packets	Y
	13C. ACTION: Send a restart-from-retry control symbol to the receiving device	Y
	13D. ACTION: Move from the Output Retry-stopped state to the Port OK state upon completion of the output port internal retry recovery procedure	Y
	13E. ACTION: Begin retransmitting packets from the returned expected ackID value	Y
	13F. DETAIL: Retried packet must eventually be re-transmitted (end point devices only; not true for cut-through switch devices)	Y

17.5.1.6 General Device Packet Reception List

Table 17-89 lists the requirements for the device to receive a packet.

Table 17-89. General Device 8/16 LP-LVDS Physical Layer Packet Reception List

Item No.	Item Functionality	Meets Reqs?
1	ackIDs are only accepted sequentially	Y
2	Start with ackID = 0 after reset	Y
3	Every packet causes an acknowledge control symbol with the corresponding ackID field	Y

**Table 17-89. General Device 8/16 LP-LVDS Physical Layer
Packet Reception List (continued)**

Item No.	Item Functionality	Meets Reqs?
4	Device must send acknowledge control symbols in same order as packets are received (ackIDs are issued sequentially)	Y
5	Device must not send a packet-accepted control symbol before the entire packet has been received and is error free (if error checking is enabled)	Y
6	Physical layer three reserved bits are ignored	Y
7	Packets can be completed normally with a new packet or EOP control symbol	Y
8	The cause field in stomp and restart-from-retry control symbols is ignored	Y
9	Device can accept embedded control symbols	Y
	9A. DETAIL: Packet-accepted control symbol	Y
	9B. DETAIL: Packet-retry control symbol	Y
	9C. DETAIL: Packet-not-accepted control symbol	Y
	9D. DETAIL: Idle control symbol	Y
	9E. DETAIL: Throttle control symbol	Y
	9F. DETAIL: Tod-sync control symbol	Y
	9G. DETAIL: Link-response control symbol	Y
10	Tod-sync control symbol is treated as idle control symbol if tod-sync control symbol is not supported	Y
11	Packets can be canceled using stomp, restart-from-retry, or link-request control symbols	Y
	11A ACTION: Device sends packet-retry control symbol for canceled packet (exception: don't send packet-retry control symbol while already in Input Retry-stopped or Input Error-stopped states)	Y
12	Device encountering a packet retry situation follows the defined retry behavior	Y
	12A. DETAIL: Packet retry situation is an internal hazard or a canceled packet (exception: packet canceled with a link-request control symbol does not follow retry behavior)	Y
	12B. ACTION: Move from the Port OK state to the Input Retry-stopped state upon detection of a packet retry situation.	Y
	12C. ACTION: Send a packet-retry control symbol with the expected ackID value	Y
	12D. DETAIL: Device silently discards packets while in the Input Retry-stopped state	Y
	12E. DETAIL: Device must detect control symbol errors while in Input Retry-stopped state	Y
	12F. ACTION: Move from the Input Retry-stopped state to the Port OK state upon reception of a restart-from-retry or a restart-from-error (link-request/input-status) control symbol	Y
13	Accepting an input packet of a priority cannot be contingent on successfully transmitting a packet of a less than or equal priority	Y
14	A generated packet-not-accepted control symbol uses a defined cause field encoding	Y

17.5.1.7 General Device Recoverable Errors List

Recoverable errors are typically transmission errors or physical layer protocol violation errors, and the recovery algorithm can be used to maintain communications on the link without losing any data. A compliant device must recover from a single error. Recovery from multiple errors is not required. The device is always the target for error detection.

Table 17-90 lists recoverable errors at the physical layer.

Table 17-90. General Device 8/16 LP-LVDS Physical Layer Recoverable Errors List

Item No.	Item Functionality	Meets Reqs?
1	Requested training pattern is embedded in a packet or used to terminate a packet	Y
	1A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	1B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	1C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	1D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y
2	Device encounters S-bit parity failure	Y
	2A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	2B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	2C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	2D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y
3	Device encounters incorrect CRC for a packet	Y
	3A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	3B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	3C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	3D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y
4	Device encounters unexpected ackID value in a packet	Y
	4A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	4B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	4C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	4D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y

Table 17-90. General Device 8/16 LP-LVDS Physical Layer Recoverable Errors List (continued)

Item No.	Item Functionality	Meets Reqs?
5	Device encounters a packet that exceeds the 276 byte maximum packet size limit	Y
	5A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	5B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	5C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	5D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y
6	Device encounters a corrupt control symbol	Y
	6A. ACTION: Move from the Port OK state to the Input Error-stopped state	Y
	6B. ACTION: Send a packet-not-accepted control symbol to the sending device	Y
	6C. ACTION: Device silently discards all packets until a restart-from-error (link-request/input-status) control symbol is encountered	Y
	6D. ACTION: Move from the Input Error-stopped state to the Port OK state	Y
7	Device detects protocol violations	Y
	7A. DETAIL: Any acknowledge control symbol with an unexpected ackID value	Y
	7B. DETAIL: Unsolicited acknowledge control symbol	Y
	7C. DETAIL: Received packet-accepted control symbol before packet transmission has completed	N
	7D. DETAIL: Received link-request control symbol before sending link-response control symbol for previous link-response/input-status control symbol	Y
	7E. DETAIL: Unexpected link-response control symbol	Y
	7F. DETAIL: Unexpected restart-from-retry control symbol	Y
	7G. DETAIL: Unexpected training pattern	Y
	7H. DETAIL: Unexpected stomp control symbol	Y
	7I. DETAIL: Unexpected eop control symbol	Y
8	Device encounters a protocol violation	Y
	8A. ACTION: Move from the Port OK state to the Output Error-stopped state	Y
	8B. ACTION: Device stops transmitting new packets	Y
	8C. ACTION: Device sends link-request/input-status (restart-from-error) control symbol if no link-request control symbol is already outstanding (must wait for previous one to complete) and waits for link-response control symbol	Y
	8D. ACTION: Move from the Output Error-stopped state to the Port OK state	Y
	8E. ACTION: Start re-transmitting at the ackID value returned with the received link-response control symbol	Y

Table 17-90. General Device 8/16 LP-LVDS Physical Layer Recoverable Errors List (continued)

Item No.	Item Functionality	Meets Reqs?
9	Device encounters a packet-not-accepted control symbol	Y
	9A. ACTION: Move from the Port OK state to the Output Error-stopped state	Y
	9B. ACTION: Device stops transmitting new packets	Y
	9C. ACTION: Device sends link-request/input-status (restart-from-error) control symbol if no link-request control symbol is already outstanding (must wait for previous one to complete) and waits for link-response control symbol	Y
	9D. ACTION: Move from the Output Error-stopped state to the Port OK state	Y
	9E. ACTION: Start re-transmitting at the ackID value returned with the received link-response control symbol	Y
10	Device encounters an acknowledge control symbol time-out	Y
	10A. ACTION: Move from the Port OK state to the Output Error-stopped state	Y
	10B. ACTION: Device stops transmitting new packets	Y
	10C. ACTION: Device sends link-request/input-status (restart-from-error) control symbol if no link-request control symbol is already outstanding (must wait for previous one to complete) and waits for link-response control symbol	Y
	10D. ACTION: Move from the Output Error-stopped state to the Port OK state	Y
	10E. ACTION: Start re-transmitting at the ackID value returned with the received link-response control symbol	Y
11	Any transmission error encountered during error recovery	Y

17.5.1.8 General Device Nonrecoverable Errors List

Relatively few nonrecoverable errors occur at the physical layer. Behavior of the interface is not defined for these events. lists nonrecoverable errors at the physical layer.

Table 17-91. General Device 8/16 LP-LVDS Physical Layer Nonrecoverable Errors List

Item No.	Item Functionality	Meets Reqs?
1	Device does not respond to link-request/send-training control symbol with 256 training patterns followed by at least one idle control symbol	Y
2	Late or no response to throttle request	N
3	Device is attempting error recovery and ackID received in link-response control symbol does not make sense (device cannot complete recovery)	Y
4	Link-request control symbol to link-response control symbol time-out	Y
5	Device port transmits packets before device is both sending and receiving idle control symbols	Y
6	Any transmission error encountered during error recovery	Y

17.5.2 Common Transport Layer Functionality Lists

The common transport layer functionality is broken down into a number of sub-lists.

17.5.2.1 Common Transport Layer Basic Functionality List

Table 17-92 lists the basic functionality for the device at the common transport layer.

Table 17-92. General Device Common Transport Layer Basic Functionality List

Item No.	Item Functionality	Meets Reqs?
1	All necessary CSRs (command and status registers) exist and can be read	Y
	1A. DETAIL: Base Device ID CSR; reset value is application and implementation dependant (end point devices only)	Y
	1B. DETAIL: Host Base Device ID Lock CSR; Host_base_deviceID field reset value is 0xFFFF	Y
	1C. DETAIL: Component Tag CSR; reset value is 0x0000_0000	Y
2	Writable CSR fields can be written	Y

17.5.2.2 Common Transport Layer Packet Transmission List

Table 17-93 lists the functionality for the device to transmit a packet (a slave device is required to generate response packets).

Table 17-93. General Device Common Transport Layer Packet Transmission List

Item No.	Item Functionality	Meets Reqs?
1	tt field is always logic 0s (small device ID fields)	Y
2	End point-free switch responds to maintenance requests with the c_count = 0 when received, otherwise decrement hop_count and route to proper output port	Y
3	MAINTENANCE response packets are issued with the hop_count field set to 0xFF	Y

17.5.2.3 Common Transport Layer Packet Reception List

Table 17-94 lists the functionality for the device to receive a packet.

Table 17-94. General Device Common Transport Layer Packet Reception List

Item No.	Item Functionality	Meets Reqs?
1	Response packets are sent with the target and destination fields reversed from the corresponding request packet	Y

17.5.2.4 Common Transport Layer Detectable Errors List

Logical layer errors are generally not recoverable in hardware. They may be recoverable in software. The device is always the target of a packet for error detection.

Table 17-95. General Device Common Transport Layer Detectable Errors List

Item No.	Item Functionality	Meets Reqs?
1	Received reserved tt field encoding	Y

17.5.3 Logical Layer Functionality Lists

The logical layer functionality is broken down into a number of sub-lists.

17.5.3.1 Logical Layer Basic Functionality List

Table 17-96 lists the requirements for basic device functionality. A class 1 device is always the target of a request transaction.

Table 17-96. General Device Logical Layer Basic Functionality List

Item No.	Item Functionality	Meets Reqs?
1	The address field of a packet is a double-word (8 byte) aligned address	Y
2	Multiple double-word data payloads are linear starting at the specified address	Y
3	Multiple double-word data payloads are aligned to a double-word boundary	Y
4	Multiple double-word data payloads are not required to be aligned to the transfer size boundary	Y
5	Sub-double-word data payloads have a defined data payload, properly aligned and padded to a double-word boundary	Y
6	Response packets have the transaction ID of the associated request packet	Y
7	Responses that are not expected to have a data payload must not have a data payload	Y
8	Responses may not contain a data payload if the response status is ERROR	Y

Table 17-96. General Device Logical Layer Basic Functionality List (continued)

Item No.	Item Functionality	Meets Reqs?
9	All necessary CARs (capabilities registers) exist and can be read	Y
	9A. DETAIL: Device Identity CAR; value is vendor and implementation dependant	Y
	9B. DETAIL: Device Information CAR; value is vendor and implementation dependant	Y
	9C. DETAIL: Assembly Identity CAR; value is vendor and implementation dependant	Y
	9D. DETAIL: Assembly Information CAR; value is vendor and implementation dependant	Y
	9E. DETAIL: Processing Element Features CAR; value is implementation dependant; must indicate support for 34-bit address format packets	Y
	9F. DETAIL: Switch Port Information CAR; value is implementation dependant (switch devices only)	Y
	9G. DETAIL: Source Operations CAR; value is implementation dependant (end point devices only)	Y
	9H. DETAIL: Destination Operations CAR; value is implementation dependant (end point devices only)	Y
10	All necessary CSRs (command and status registers) exist and can be read	Y
	10A. DETAIL: Processing Element Logical Layer Control CSR: reset value of the Extended addressing control field is 0b001 (end point devices only)	Y
11	Writable CSR fields can be written	Y
12	All registers in the Extended Features data structure are double-word (8 byte) aligned (the three Isbs of the Extended Features Pointers must be logic 0s)	Y
13	All Extended Features blocks lie completely within the Extended Features Space in the register address map	Y
14	Extended Features list is terminated with an Extended Features pointer value of logic 0s	Y

17.5.3.2 Logical Layer Target Transaction Support List

Table 17-97 lists the requirements for the device to function as a transaction target.

Table 17-97. General Device Logical Layer Target Transaction Support List

Item No.	Item Functionality	Meets Reqs?
1	MAINTENANCE read transaction	Y
	1A. DETAIL: MAINTENANCE read request may be for four bytes	Y
	1B. DETAIL: MAINTENANCE read request generates a MAINTENANCE read response	Y
	1C. DETAIL: MAINTENANCE read response data payload is four bytes	Y
2	MAINTENANCE write transaction	Y
	2A. DETAIL: MAINTENANCE write request may be for four bytes	Y
	2B. DETAIL: MAINTENANCE write request generates a MAINTENANCE write responses	Y
	2C. DETAIL: MAINTENANCE write response does not contain a data payload	Y

17.5.3.3 Logical Layer Source Transaction Support List

General functionality does not require a device to initiate any operations.

17.5.3.4 Logical Layer Detectable Errors List

Logical layer errors are generally not recoverable in hardware. They may be recoverable in software. The device is always the target of a packet for error detection.

Table 17-98. General Device Logical Layer Detectable Errors List

Item No.	Item Functionality	Meets Reqs?
1	MAINTENANCE write packet data payload exceeds size specified in wrsize field	Y
2	MAINTENANCE write packet data payload exceeds supported size	Y
3	Received MAINTENANCE read packet has a data payload	Y
4	Received MAINTENANCE write packet has no data payload	Y
5	Received MAINTENANCE packet uses a reserved field encoding for a required field	Y
6	Received MAINTENANCE packet uses illegal combinations of field encodings	Y

17.5.3.5 Logical Layer Extended Functionality Checklist

The RapidIO Controller supports IO_READ and Flush_with_data GSM transactions, all I/O and Message transactions as described by the RapidIO 1.2 Specifications except for ATOMIC test-and-swap and ATOMIC swap transactions.

17.6 RapidIO Errors

RapidIO errors are classified under three categories:

- **Recoverable Errors:** RapidIO supports hardware error detection and recovery mechanism as specified in the RapidIO specification. RapidIO detects and attempts to recover from corrupt packet and control symbol errors and general protocol errors. In these cases, the appropriate bit is set in the RapidIO recoverable error detect register. Logging of information in capture registers may or may not take place.
- **Notification Errors:** RapidIO detects a number of hardware nonrecoverable errors that are non-fatal such as exceeded threshold counts. Upon detection of a notification error the appropriate bit is set in the RapidIO fatal/notification error detect register. An interrupt is generated. Hardware continues to operate after notification errors.
- **Fatal Errors:** RapidIO also detects a number of fatal errors that are hardware nonrecoverable, such as time-outs. Upon detection of a fatal error relevant information may be logged in the RapidIO error packet register if the error is

enabled and the appropriate bit is set in the RapidIO fatal/notification error detect register. An interrupt is generated if the fatal error reporting is enabled in the RapidIO fatal error enable register. The RapidIO hardware either goes into training mode, or remains in the OK state based on the error type.

If bit 0 of the corresponding RapidIO error control register is set, then the information contained in the following corresponding capture registers is valid:

- RapidIO error control register
- RapidIO error packet/control symbol register
- RapidIO error packet registers 1 through 3

Fatal errors caused by a bad link invoke training in attempt to recover the link. If communication is not re-established, a RESET is required to clean up RapidIO's internal queues to resume normal operation. Table 17-99 lists all the errors detected by the RapidIO implementation and the actions taken.

Table 17-99. Errors Detected by RapidIO

Category	Error	Description	Action	Comments
Recoverable	Inbound corrupt control symbol	Received a control symbol in which true does not match complement counterpart.	Send "PACKET_NOT_ACCEPTED" ACK with "Error on control symbol" cause.	No information logged in capture register(s).
	Inbound packet/control symbol with S-bit parity error	Received a packet or a control symbol with an S-bit parity error.	Send "PACKET_NOT_ACCEPTED" ACK with "S-bit parity error" cause.	No information logged in capture register(s).
	Inbound out-of-sequence ACK symbol	Received an ACK control symbol with an unexpected ACKID.	Start Link recovery process.	No information logged in capture register(s).
	Inbound packet with unexpected ACKID	Received packet with unexpected ackID value (out-of-sequence ACKID)	Send "PACKET_NOT_ACCEPTED" ACK with "Received unexpected ACKID with packet" cause.	No information logged in capture register(s).
	Inbound packet with Bad CRC	Received packet with a bad CRC value.	Send "PACKET_NOT_ACCEPTED" ACK with "Bad CRC" cause.	No information logged in capture register(s).
	Embedded Training	Requested training pattern is embedded in a packet	Send "PACKET_NOT_ACCEPTED" ACK with "General error" cause.	No information logged in capture register(s).
	Inbound packet exceeds 276 Bytes	Received packet which exceeds the maximum allowed size by the RapidIO spec. MPC8540 performs this function by detecting a data payload greater than 256 bytes.	Send "PACKET_NOT_ACCEPTED" ACK with "General error" cause.	No information logged in capture register(s).

Table 17-99. Errors Detected by RapidIO (continued)

Category	Error	Description	Action	Comments
Recoverable	Packet-not-accepted: Encountered internal error	Received ACK control symbol with packet-not-accepted of type 'encountered internal error'.	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: Unexpected ACKID on packet	Received ACK control symbol with packet-not-accepted of type 'unexpected ACKID'	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: Corrupt control symbol	Received ACK control symbol with packet-not-accepted of type 'corrupt control symbol'	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: Input port stopped	Received ACK control symbol with packet-not-accepted of type 'input port stopped'	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: Bad CRC	Received ACK control symbol with packet-not-accepted of type 'Bad CRC'.	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: S-bit parity error	Received ACK control symbol with packet-not-accepted of type 'S-bit parity error'.	Start Link recovery process.	No information logged in capture register(s).
	Packet-not-accepted: General Error	Received ACK control symbol with packet-not-accepted of type 'General Error'. This error type could be created because of header/data problems at the receiver.	Start Link recovery process.	No information logged in capture register(s).
	Unexpected Restart-from-retry symbol	Received a Restart-from-retry control symbol while in the "ok" state.	Start Link recovery process.	No information logged in capture register(s).
	Unsolicited ACK symbol	Received an ACK control symbol while no packets are outstanding on the ACK History Queue.	Start Link recovery process.	No information logged in capture register(s).
	ACK before restart-from-retry	Received an ACK control symbol after receiving an ACK retry and before sending a restart-from-retry.	Start Link recovery process.	No information logged in capture register(s).
	Unexpected training symbol	Received a training symbol while in the "OKstate".	Start Link recovery process.	No information logged in capture register(s).

Table 17-99. Errors Detected by RapidIO (continued)

Category	Error	Description	Action	Comments
Recoverable	Unexpected Stomp symbol	Received a stomp control symbol while there is no packet being received.	Start Link recovery process.	No information logged in capture register(s).
	Unexpected Link response symbol	Received a Link response control symbol while there is no outstanding request.	Start Link recovery process.	No information logged in capture register(s).
	Unexpected EOP symbol	Received an EOP control symbol while there is no packet being received.	Start Link recovery process.	No information logged in capture register(s).
	Link request error	Received a Link request control symbol before the previous link request has been serviced.	Start Link recovery process.	No information logged in capture register(s).
	Frame toggle alignment	Received FRAME signal toggles at a non 32-bit boundary.	Send "PACKET_NOT_ACCEPTED" ACK with "General error" type	No information logged in capture register(s).
	Frame toggle edge	Received FRAME signal toggles on the negative edge of the clock.	Send "PACKET_NOT_ACCEPTED" ACK with "General error" type	No information logged in capture register(s).
	ACK time-out	An ACK control symbol is not received within the specified time-out interval.	Start Link recovery process.	No information logged in capture register(s).
Notification	Error Recovery threshold error	Error recovery threshold count exceeded.	Generate interrupt if enabled.	No information logged in capture register(s).
	Retry threshold error	Consecutive ACK retry threshold count exceeded.	Generate interrupt if enabled.	No information logged in capture register(s).
	Received request ERROR response	Received a response of type error for anything other than a message.	Generate interrupt if enabled.	Information logged in CCU capture registers.
Fatal	Link response time-out	A Link response is not received within the specified time-out interval.	Generate interrupt if enabled.	No information logged in capture register(s). Resume training on the interface.
	Packet response time-out	A packet response is not received within the specified time-out interval.	Generate interrupt if enabled.	Information logged in CCU capture registers.
	Missing Idle after training	Idle not received after a requested training sequence completes	Generate interrupt if enabled.	No Information logged in capture registers.

Table 17-99. Errors Detected by RapidIO (continued)

Category	Error	Description	Action	Comments
Fatal	Missing Idle After Training	Idle not received after a requested training sequence completes.	Generate interrupt if enabled.	No Information logged in capture registers.
	Message Segment Error	Received a message packet with the segment field greater than the message length field (msgseg > msglen)	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Duplicate Message Segment	Received a duplicate message segment from RapidIO	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Message Length Error	Received a message packet from RapidIO with a bad message length	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Non-sensical ACKID	ACKID received with Link_response does not make sense.	Generate interrupt if enabled.	No information logged in capture register(s). Resume training on the interface. HRESET required to resume normal operation.
	Illegal request fields	Illegal combinations of fields in the request packet. These include: packet format type in combination with size packet format type in combination with transaction type packet format type in combination with prio	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Unexpected request data	Read type with a data payload.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Unsupported request	Unsupported packet format type, or transport type (tt bits).	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Unexpected request size	Packet data is not expected size.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Illegal write request	Write types with no data payload.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Illegal response fields	Illegal combinations of fields in the response packet.	Generate interrupt if enabled.	Information logged in RapidIO capture registers

Table 17-99. Errors Detected by RapidIO (continued)

Category	Error	Description	Action	Comments
Fatal	Illegal response type	Illegal response for a given request type.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Bad read response	Read “done” response with no data payload.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Bad write response	Write response with a data payload.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Unexpected response data size	Response Packet data is not expected size.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Unsolicited response without data	A response without data is received while there is no corresponding request that matches that TID.	Generate interrupt if enabled.	No information logged in capture register(s).
	Unsolicited response with data	A response with data is received while there is no corresponding request that matches that TID.	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Message error response	Received a response of type error for an outbound message packet.	Generate interrupt if enabled.	No information logged in capture register(s).
	Message descriptor fetch error	A message descriptor fetch from local memory gets an error (ECC for example).	Generate interrupt if enabled.	No information logged in capture register(s).
	Message size error	Received MESSAGE packet data payload is not of the size specified in the ssize field (exception for last packet which may be less)	Generate interrupt if enabled.	Information logged in RapidIO capture registers.
	Illegal Transaction Target Error	Received a packet whose target ID does not match RapidIO deviceID while accept_all & pass through modes are disabled	Generate interrupt if enabled.	Information logged in RapidIO capture registers.

17.7 ATMU (Address Translation and Mapping Unit)

Outbound address translation refers to the translation of an address from the local address space to that of RapidIO. In the same context, inbound address translation and mapping refers to the translation of an address from the external address space of RapidIO to the local address space understood by the internal interfaces.

Although, the RapidIO Specification allows for 48 and 64-bit addresses, the MPC8540 implementation only supports 34-bit RapidIO addresses. Outbound and inbound window

misses use the window 0 register set by default. The size field in the attributes register of window 0 for both inbound and outbound translation is used to determine how much of the translation address will be used when the default window is selected. Overlapping window hits 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 4G

17.7.1 Outbound ATMU Translation

Outbound ATMU windows perform the mapping from the internal 32-bit address space to the 34-bit address space of RapidIO. Outbound ATMU windows also map attributes such as transaction type and priority level. There exist nine outbound ATMU windows (0–8). Window 0 is always enabled and is used as the default window if the address does not match one of the other eight windows. Overlapping outbound window hits result in the use of the lowest number window register set hit. Note that OCeaN read types can only map to RapidIO read types and OCeaN write types can only map to RapidIO write types including doorbells.

17.7.1.1 Outbound ATMU Bypass Mode

It is important to note that certain transactions may not require ATMU translation because the address associated with the original transaction is already mapped to the appropriate address space. Although, the DMA Controller can initiate outbound RapidIO transactions that may or may not bypass ATMU translation, it is the only source of transactions that can bypass outbound ATMU translation. The DMA Controller may bypass ATMU translation for any outbound transaction that it supports. For more information about the DMA Controller and the transactions it supports, reference Chapter 14, “DMA Controller”.

17.7.1.2 Outbound Special Transactions and Requirements

Due to requirements by the RapidIO Specification, IO_READ_HOME and FLUSH_WITH_DATA transactions require special handling. Outbound IO_READ transactions cannot cross a 32-byte cache-line boundary and cannot be greater than 32 bytes in size because the RapidIO interface requires wrapping around a 32-byte cache-line for IO_READ_HOME transactions. Therefore, regardless of the size of the original request, the address for outbound IO_READ transactions is double-word aligned, and the size is set to 32 bytes in order to fetch an entire cache-line. There is also a requirement for outbound requests originated by the PCI interface. These transactions must map to an ATMU window in which the ROWAR specifies a transaction flow level of zero and has the PCI bit set. This causes the outbound transaction to be sent through the RapidIO interface with a priority of one because the transaction flow level is incremented when the PCI bit is set.

In order to comply with the RapidIO Specification other transactions also have special requirements. Outbound maintenance transactions, for example, may not exceed 64 bytes.

Also, outbound doorbells are generated through the ATMU by using write transactions whose sizes are 2, 4 or 8 bytes and whose address is aligned to a doubleword boundary. A doorbell generated using a 2-byte write gets its INFO field from the data associated with the write transaction. A doorbell generated using a 4 or 8 byte write gets its INFO field from the most significant 2 bytes of the data associated with the write transaction.

Outbound SWRITE transactions must not be smaller than a double-word, and ATOMIC transactions may only request one, two or four bytes. Furthermore, ATOMIC transactions must meet the RapidIO alignment requirements as specified in the RapidIO Specification.

The RapidIO Specification should be referenced for any additional requirements that are not listed in this section.

17.7.2 Inbound ATMU Translation

Inbound ATMU windows perform the address translation from the 34-bit external RapidIO address space to the 32-bit internal address space. Inbound ATMU windows also attach attributes, transaction type and the target interface to the transaction. The RapidIO controller has four inbound ATMU windows plus the default window 0. The default window is always enabled. Overlapping inbound window hits result in the use of the lowest number window register set hit.

17.7.2.1 Inbound ATMU LCSBA1CSR Window

There is an additional inbound window that is used for inbound translation. It is a special condition for external accesses to LCSBA1CSR (Local Configuration Space Base Address Register) address space. The purpose of the LCSBA1CSR window is to allow external devices to access the LCSBA1CSR address space without knowledge of the internal memory map. When a remote device attempts to access the LCSBA1CSR register set (address hitting LCSBA1CSR window), the lower 20 bits are used as the offset to access the specific register in the LCSBA1CSR address space. The LCSBA1CSR window has the highest priority for inbound translation. If the inbound transaction's address does not hit the LCSBA1CSR window or any of the four inbound ATMU windows (1–4), the translation attributes defined by the default window (window 0) are used.

17.7.2.2 Inbound ATMU Bypass Mode

Just as some outbound transactions may bypass outbound ATMU translation, some inbound transactions may also bypass ATMU translation because the address associated with the original transaction is already mapped to the appropriate address space. Inbound messages, doorbells and port writes go through a different form of address mapping and are not mapped through the ATMU. These transactions have the stash and cache_lock attributes set to zero despite the values specified in the RIWAR of the selected window. Similarly, the no_snoop bit is set according to the corresponding programming model of the specific

transaction type regardless of the value specified in the RIWAR of the selected ATMU window. For more information about these transactions, refer to Section 17.8.4.1, “Data Message Controller,” Section 17.8.4.4, “Doorbell Message Controller,” and Section 17.8.4.5, “Port-Write Controller Structure.”

17.7.2.3 Inbound Special Transactions and Requirements

Just as with outbound translation, some inbound transactions require special handling in order to comply with the RapidIO Specification. For example, inbound IO_READ_HOME and FLUSH_WITH_DATA transactions require special handling. Snooping is enabled for these transactions regardless of the no-snoop value specified in the RIWAR of the selected window. Furthermore, these transactions should target local memory. Targeting any other interface is a programming error and results in undefined behavior. Also, IO_READ_HOME transactions can request less data than the maximum of 32 bytes.

There is a requirement for inbound requests that target the PCI interface. Inbound reads intended for the PCI interface should have a priority of zero. Similarly, inbound writes targeting the PCI interface must have a priority of one. Inbound NWRITE_R transactions that target the PCI interface are not allowed. These requirements insure proper processing through the PCI interface.

In order to prevent inbound write transactions from corrupting local memory space, some inbound write transactions may flag an error. Inbound write transactions with a data payload that cross the selected ATMU window boundary may generate an error condition and signal an interrupt. The settings of the port notification/fatal error register set described in Section 17.3.3.3, “Error Management Registers,” determine if the error condition is detected and if PNFEDR[AXE] is set. The settings in this register set also determine if an interrupt is signalled upon detection of the error condition. Note that inbound reads that cross the selected ATMU window boundary are not detected and thus, may return data from a different address space than what was intended.

Inbound SWRITE transactions must not be smaller than a double-word, and ATOMIC transactions may only request one, two or four bytes. Furthermore, ATOMIC transactions must meet the RapidIO alignment requirements as specified in the RapidIO Specification and must target the DDR memory controller. Inbound ATOMIC transactions targeting interfaces other than the DDR memory controller result in undefined behavior. Also, all inbound maintenance transactions must have a size of four bytes.

The RapidIO Specification should be referenced for any additional requirements that are not listed in this section.

17.8 RapidIO Message Unit

17.8.1 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 main memory. A message may consist of one or more packets depending on the size of the message. When the entire message has been received, an interrupt (if enabled) is generated for the processor to process the message. Messages can be queued for transmission in the producer's memory and the message hardware processes them sequentially. Messages can also be queued in the consumer's memory while software processes them sequentially. Software controls the depth of the circular queue in the producer, or consumer.

The message unit also supports another form of message called a doorbell message. Doorbell messages do not have a data payload and are basically used for interrupting the processor. Like data messages, doorbell messages can be queued in a consumer's local memory while being processed by the local processor. When a doorbell is received, an interrupt (if enabled) is communicated to the processor.

This chapter is an extension of the inbound and outbound sections. It describes the operation of the data message and doorbell message controllers in the message unit. The message unit is compliant with the message passing logical specification contained in the RapidIO specifications. 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.

There exist two types of RapidIO messages. The data message controller handles data messages and the doorbell message controller handles doorbell messages. A data message may consist of up to 16 packets and each packet may contain up to 256 bytes of data. A doorbell message contains a small amount of software-defined information embedded in the packet header and never has a data payload.

The message and doorbell controller are controlled through a set of run-time registers which are described in the programming model section of the document.

17.8.2 Features

- One inbound data message structure (inbox)
- One outbound data message structure (outbox)
- Support for chaining and direct modes in the outbox
- Support of up to 16 segments per message.

- Support of up to 256 bytes per packet, up to 4 Kbytes of data per message
- Support for one inbound doorbell message structure
- Support for transmitting and receiving messages to any mailbox, any letter

17.8.3 Modes of Operation

- Direct mode: software is expected to program all the necessary registers for sending an outbound message.
- Chaining mode: a transfer descriptor that describes the message information is fetched from local memory before a message is sent.

17.8.4 RapidIO Messaging Description

This section relates to controller operations, formats, interrupts, and response conditions.

17.8.4.1 Data Message Controller

RapidIO supports a data message passing programming model for inter-processor and inter-device communication. This model enables a producer to send a data 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 its memory. A data message may consist of one or more packets depending on the size of the message. When the entire message has been received, an interrupt is generated (if enabled) for the local processor to process the message. Data messages can be queued for transmission in the producer's memory and the message hardware processes them sequentially. Messages can also be queued in the consumer's memory while software processes them sequentially. Software controls the depth of the circular queue in the producer, or consumer.

The data message controller on RapidIO contains the following features:

- One inbound mailbox (inbox) structure
- One outbound mailbox (outbox) structure
- Support for one active letter in the inbox (letters are assigned and maintained by hardware)
- Support for up to three letters (0–2) in outbox for message unit initiated messages
- Support for up to 16 segments (packets) per message
- Support for up to 256 bytes per segment (packet)
- Chaining and direct mode support for outbound messages

The following sections describe the structure and operation of the outbox and inbox hardware in the data message controller.

17.8.4.2 Outbox Controller Operation

The outbox controller is responsible for sending messages stored in a circular queue in local memory. The outbox controller supports two modes of operation, direct and chaining 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 hardware then reads the descriptor to load all the necessary registers to start the message transfer.

17.8.4.2.1 Direct Mode Operation

In direct mode, OMR[MUTM] is set in the outbound mode register and the outbox controller does not read descriptors from memory, but instead uses the current parameters programmed in the outbox 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 outbox start bit, OMR[MUS], in the outbound mode register transitions from a 0 to 1 and the outbox is not already busy. Software is expected to program all the appropriate registers before setting OMR[MUS]. The sequence of events to start and complete a transfer in direct mode is as follows:

- Poll the status register message unit busy bit, OSR[MUB], to make sure the is not busy with a previously initiated message.
- Initialize the source address (OSAR), destination port (ODPR), destination attributes (ODATR) and double-word count (ODCR) registers.
- Initialize the outbound mode register message unit transfer mode bit, OMR[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, OMR[MUS], to start the message transfer.
- OSR[MUB] is set by the outbox controller to indicate the message transfer is in progress.
- OSR[MUB] is cleared by the outbox controller after the transfer is finished or if a transfer error occurs.
- End of message interrupts are generated if ODATR[EOMIE] is set.

17.8.4.2.2 Chaining Mode Operation

In chaining mode, OMR[MUTM] = 0 in the outbound mode register, message descriptors are built in local memory in a circular queue. Two options are available to the programmer in chaining mode: normal mode and list mode. In normal mode, software can initialize the outbox controller registers and then build the descriptors in memory. The enqueue pointer (ODQEPAR) is maintained and incremented by hardware, but the increment is controlled

by software. Software should build descriptors in memory using the enqueue pointer address and can have the hardware increment the enqueue pointer by writing OMR[MUI] to a value of 1. Software can then read the value of the enqueue pointer to write the next descriptor in memory and then increment the enqueue pointer by writing OMR[MUI] = 1 and repeat this process until the descriptor queue is full. Hardware performs the enqueue pointer address calculation, queue fullness and queue wrap checks and software writes new descriptors using the computed enqueue pointer address. In list mode, software can build one or more descriptors in memory before initializing the outbox controller registers. Software is responsible for maintaining and controlling the enqueue pointer (ODQEPAR). Software is also responsible for keeping track of queue fullness and wrap conditions. OMR[MUI] is not used in this mode.

The outbox message controller takes descriptors off the tail of the queue and processes them, incrementing the dequeue pointer (ODQDPAR) to point to the next descriptor in the queue and repeats this process until the enqueue and dequeue pointers are equal.

Figure 17-80 below depicts a sample structure of the outbox portion of the outbox message controller, a descriptor list queue, and the message data queue. In this example, the descriptor list queue has eight entries, four of which are currently valid. The local processor adds descriptors to the head of the queue and the outbox message controller takes them off from the tail.

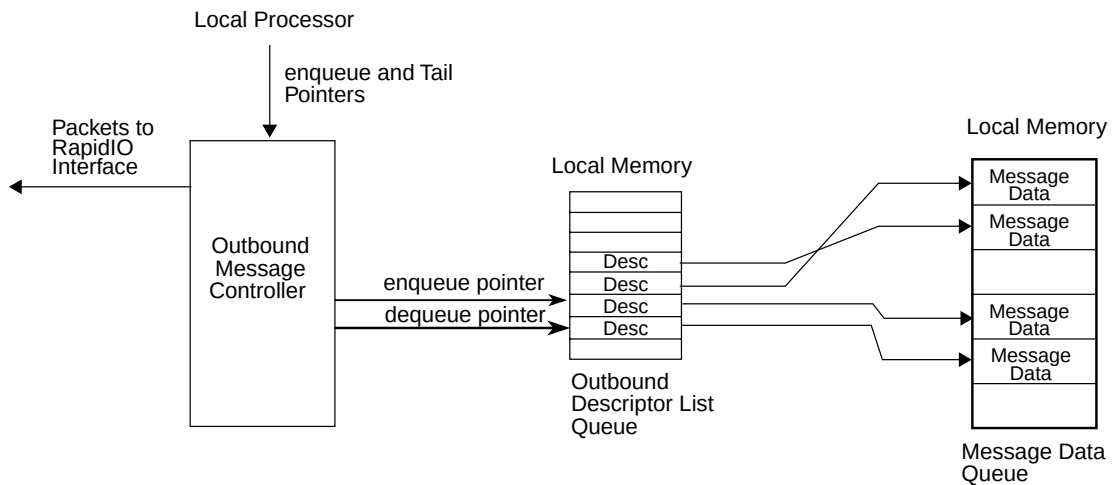


Figure 17-80. Outbound Message Queue Structure

The sequence of events to start and complete a transfer by adding descriptors after initializing the message unit in normal chaining mode is as follows:

- Poll the status register message unit busy bit, OSR[MUB], to make sure the outbox controller is not busy with a previously initiated message.

- Initialize the outbound descriptor queue dequeue pointer address registers (ODQDPAR) and the outbound descriptor queue enqueue pointer address registers (ODQEPAR). These need to be initialized to the same value for proper operation.
- Initialize the outbound mode register message unit transfer mode bit, OMR[MUTM] = 0, to indicate chaining mode and set OMR[MUS] = 0 to disable the data message controller. Other control parameters must also be initialized in the mode register.
- Set OMR[MUS], which enables the outbox controller and causes the dequeue pointer (ODQDPAR) to be saved off as the base address of the circular queue. The enqueue pointer is incremented when OMR[MUI] is set by software. If the enqueue and dequeue pointers are not equal, the descriptor fetch begins immediately from the address pointed to by the dequeue pointer (ODQDPAR). If they are equal, the outbox waits until the enqueue and dequeue pointers are not equal. OMR[MUI] is cleared by hardware after successfully incrementing the enqueue pointer.
- OSR[MUB] is set by the message unit to indicate that the message transfer is in progress while the tail and enqueue pointers are not equal.
- The above process continues until the dequeue pointer equals the enqueue pointer once again.
- OSR[MUB] is cleared by the message unit after finishing the transfer of the last descriptor segment, or if an error occurs during any of the transfers.
- Software can continue adding descriptors as needed for new transfers by setting OMR[MUI] as long as the circular queue in memory is not full.

The sequence of events to start and complete the transfer of a list of message descriptors built in memory before initializing the message unit in list chaining mode is as follows:

- Poll the status register message unit busy bit, OSR[MUB], to make sure the outbox controller is not busy with a previously initiated message.
- Initialize the outbound descriptor queue dequeue pointer address registers (ODQDPAR) to point to the first descriptor in memory and the outbound descriptor queue enqueue pointer address registers (ODQEPAR, EODQEPAR) to point to the circular queue entry in memory following the last descriptor in the list. The enqueue and dequeue pointers should not be the same.
- Initialize the outbound mode register message unit transfer mode bit, OMR[MUTM] = 0, to indicate chaining mode and set OMR[MUS] = 0 to disable the data message controller. Other control parameters must also be initialized in the mode register.
- Set OMR[MUS], which enables the outbox controller. The descriptor fetch begins immediately from the address pointed to by the dequeue pointer (ODQDPAR).
- OSR[MUB] is set by the message unit to indicate that the message transfer is in progress.

- OMR[MUI] is not used in this mode since software directly controls the enqueue pointer. So new descriptors cannot be added when the message unit is busy as indicated by OSR[MUB] = 1.
- OSR[MUB] is cleared by the message unit after finishing the transfer of the last descriptor segment, or if an error occurs during any of the transfers.
- New descriptors can be added by just updating the enqueue (ODQEPAR) pointer address registers as long as the enqueue pointer address does not cause a wrap condition.
- If adding new descriptors causes a wrap condition, software can build descriptors in memory but not update the enqueue pointer (ODQEPAR) address registers until the message unit is idle as indicated by OSR[MUB] = 0. Before updating the enqueue and dequeue pointers, software must disable the message unit by setting OMR[MUS] = 0. It can then update the enqueue (ODQEPAR) and dequeue (ODQDPAR) pointer address registers before enabling the message unit by setting OMR[MUS] = 1 to start transferring the new list of descriptors.

Software must guarantee that descriptors are not added to an already full queue. This can be accomplished in one of two ways:

- When OMR[QFIE] is set, the hardware keeps track of queue fullness in normal chaining mode and reports an interrupt if the queue becomes full. Once the interrupt is set, it is the responsibility of the software to clear the interrupt condition by setting OSR[QFI]. OSR[QFI] is not cleared with the write if the queue is still full. When OSR[QFI] is cleared, the queue is no longer full and a descriptor may be added to the queue.
- When OMR[QFIE] is cleared or if operating in list chaining mode, OSR[QFI] is not set by hardware on a queue full condition, and it is the responsibility of the software to calculate queue fullness before adding a new descriptor. Software can detect queue fullness by comparing the enqueue (ODQEPAR) and the dequeue pointer (ODQDPAR) and monitoring the queue busy bit (OSR[MUB]). The queue is full if the enqueue and dequeue pointers are equal and the queue busy bit is set.

The process for adding a descriptor to the circular memory queue in normal mode is as follows:

- Ensure that the circular queue is not full using either of the above methods.
- Write the descriptor to be enqueued to the 32 bytes pointed to by the enqueue pointer (ODQEPAR).
- Write OMR[MUI] = 1 keeping the same values of all the other bits in the OMR.
- A new descriptor may be added by incrementing the enqueue pointer if the queue is not full.

17.8.4.2.3 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 OSR[MUB] being cleared. When switching from direct mode to chaining mode, if OMR[MUS] is cleared and set, the message unit will be re-initialized in chaining mode and the outbound descriptor queue dequeue pointer address will be saved off as the new base address of the circular queue in memory. If this is not desired, OMR[MUS] should stay set when switching from direct mode to chaining mode. When the enqueue and dequeue pointers are not equal, the message unit will begin the new descriptor fetch while retaining the same circular queue parameters as the previous chaining mode transfer. When switching from chaining mode to direct mode, OMR[MUS] has to be cleared and set. This will have no effect on the circular queue parameters thereby saving the queue state for the next chaining mode transfer.

17.8.4.2.4 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. For each descriptor in the queue, the message unit controller starts a new message operation with the control parameters specified by the descriptor.

Table 17-100. Outbound Message Unit Descriptor Summary

Descriptor Field	Description
Reserved	
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.
Reserved	—
Reserved	—
Double-word Count	Contains the number of double-words 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 17-81 below depicts the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

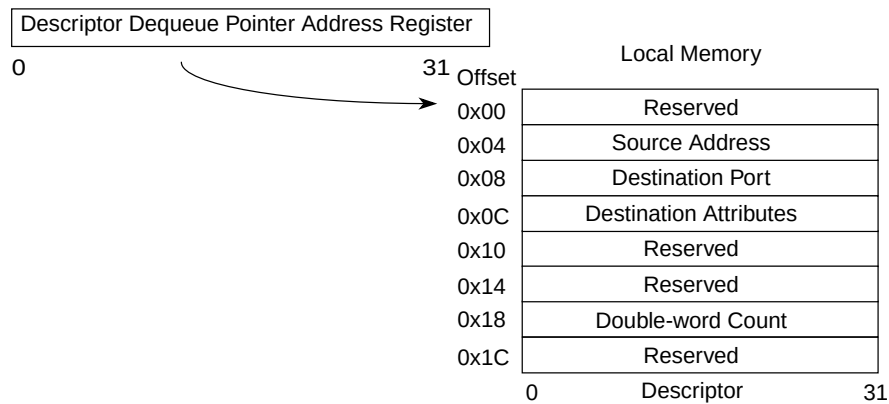


Figure 17-81. Descriptor Dequeue Pointer and Descriptor

17.8.4.2.5 Outbox Controller Interrupts

The outbox generates four different interrupts which can be individually enabled.

- Queue overflow interrupt. This interrupt is generated if the enqueue pointer passes the dequeue pointer and the queue is full and the interrupt is enabled. OSR[QOI] is set if OMR[QOIE] is set.
- Queue Full Interrupt. This interrupt is generated if the enqueue pointer catches up to the dequeue pointer and the queue is not empty and the interrupt is enabled. OSR[QFI] is set if OMR[QFIE] is set.
- Queue empty interrupt. This interrupt is generated if the dequeue pointer catches up to the enqueue pointer and the interrupt is enabled. OSR[QEI] is set if OMR[QEIE] is set.
- End-of-message interrupt. This interrupt is generated after the completion of each message if enabled. OSR[EOMI] is set if ODATR[EOMIE] is set.

17.8.4.2.6 Special Error Case Condition

If a data message is in progress and the outbox encounters an error with one of the segments, it causes an error at the recipient by placing an illegal field in the packet to cause the error. This is to prevent the recipient from using the data received in the message. One example of this is if the outbox controller is reading message data from its DRAM memory and it gets an uncorrectable ECC error. This scheme prevents the recipient from hanging or using bad data.

17.8.4.3 Inbox Controller Operation

The inbox controller is responsible for receiving messages and placing them in a circular queue in local memory. Although, it only supports one mailbox and one letter, it can receive messages with any mailbox or letter number. Furthermore, the inbox 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 outbox where the enqueue pointer is controlled by software and the dequeue pointer is controlled by hardware, the inbox controls the enqueue pointer and the software controls the dequeue pointer.

Figure 17-82 depicts a sample structure of the inbound mailbox message frames and the frame pointers. In this example, the frame queue has eight entries, three of which are currently valid. The mailbox controller adds frames to the head of the queue and the local processor takes them off from the tail. After processing a message, the local processor writes the inbox mode register mailbox increment bit (IMR[MI]) to point to the next message frame in the queue. This process is repeated for each received message.

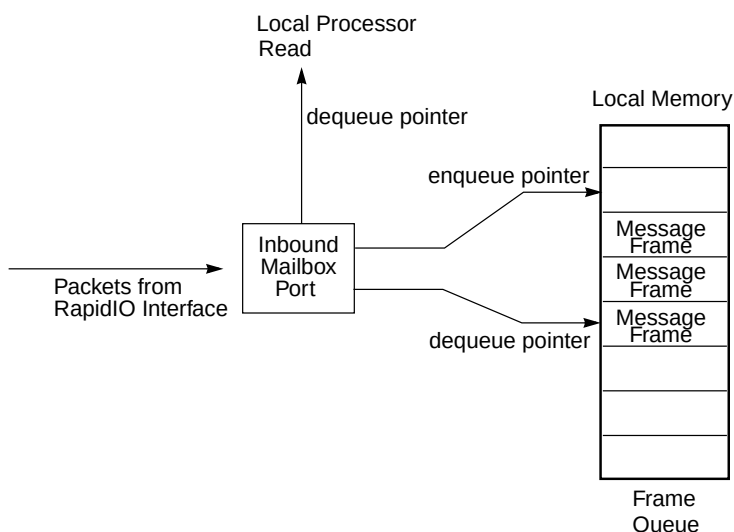


Figure 17-82. Inbound Mailbox Structure

The sequence of events that take place during the reception of a data message is as follows:

- The inbox receives a message request from the RapidIO port. If the inbox is enabled (IMR[ME] = 1) and it is not busy with another message (ISR[MB] = 1), then the message is accepted.
- The address is computed for each segment of the message (up to 16 segments/packets per message) using the value of the inbound frame queue enqueue pointer address registers and properly adjusted based on segment number.
- Data is stored in the circular queue in local memory at the computed address.
- Once the entire message is received, the enqueue pointer is incremented to point to the next message frame in local memory.
- An interrupt is generated if enabled (IMR[MIQIE] = 1). The interrupt is held until the dequeue pointer equals the enqueue pointer and the message queue is empty. That is, as long as messages exist in memory that have not been serviced and the interrupt is enabled, the interrupt remains asserted.

17.8.4.3.1 Retry Response Conditions

Two conditions generate a logical layer retry. (response “retry”).

- Inbox receives a message, but it is busy handling another message.
- Local memory circular queue is full and a message is received.

17.8.4.3.2 Error Response Conditions

Several conditions generate a logical layer error response (response “error”).

- Inbox receives a message and it is not enabled.
- Inbox is in an error state caused by a previous message.
- Inbox receives a segment which exceeds the message length ($\text{msgseg} > \text{msglen}$).
- Inbox receives a segment of a message that is being processed with a bad message length.
- Inbox receives a duplicate segment of a message being processed that was already received successfully.

17.8.4.3.3 Inbox Controller Interrupts

The inbox generates two different interrupts which can be individually enabled.

- Message-in-queue interrupt. This interrupt is generated each time the circular queue becomes not empty and the interrupt is enabled. $\text{ISR}[\text{MIQI}]$ is set if $\text{IMR}[\text{MIQIE}]$ is set.
- Queue full interrupt. This interrupt is generated each time the circular queue becomes full and the interrupt is enabled. $\text{ISR}[\text{QFI}]$ is set if $\text{IMR}[\text{QFIE}]$ is set.

17.8.4.3.4 Data Message Controller Limitations and Restrictions

This section addresses some of the limitations and restrictions of the data message controller. This is intended to help software maximize the message passing performance and avoid programming errors.

Software must guarantee that the enqueue pointer is not pointing to an invalid descriptor in chaining mode if the enqueue and dequeue pointers are not equal.

17.8.4.4 Doorbell Message Controller

RapidIO supports a doorbell message type that contains no data payload. The RapidIO architecture references outbound and inbound doorbell message controllers, however, RapidIO supports only an inbound doorbell message queue. Outbound doorbells are generated outside RapidIO. Inbound doorbell messages are handled by the doorbell message controller similar to how the data message controller handles inbound data

messages. The doorbell controller receives the doorbell message and places it in a circular queue located in the local memory.

Figure 17-83 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 adds doorbell information to the head of the queue and the local processor takes it off from the tail.

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.

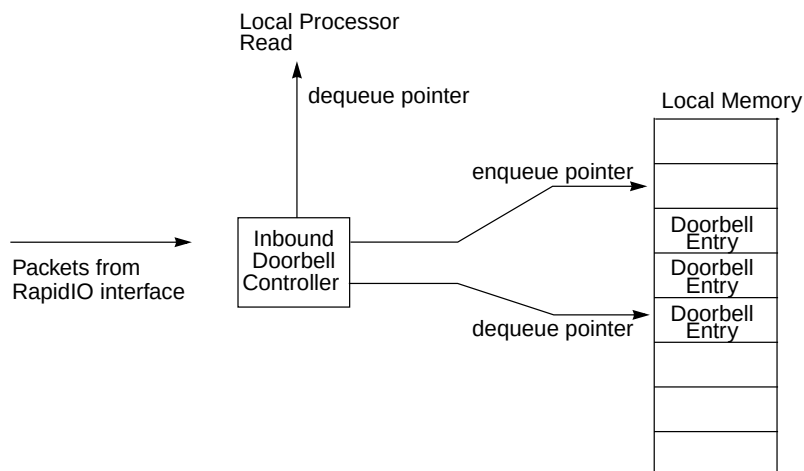


Figure 17-83. Inbound Doorbell Queue and Pointer Structure

17.8.4.4.1 Inbound Doorbell Reception

The sequence of events that take place during the reception of a doorbell message is as follows:

- The inbox receives a message request from the RapidIO port. If the inbox is enabled ($\text{DMR}[\text{DE}] = 1$) and it is not busy with another doorbell message ($\text{DSR}[\text{DB}] = 1$), then the doorbell message is accepted.
- The 16-bit information field is stored in local memory using the value of the doorbell queue enqueue pointer address registers (DQDPAR).
- The enqueue pointer is incremented to point to the next doorbell queue entry in local memory.
- An interrupt is generated if enabled ($\text{DMR}[\text{DIQIE}] = 1$). The interrupt is held until the dequeue pointer equals the enqueue pointer and the doorbell queue is empty. That is, as long as doorbell messages exist in memory that have not been serviced and the interrupt is enabled, the interrupt remains asserted.

17.8.4.4.2 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 two offsets of 32 bits, one for target information and one for source information.

Table 17-101. Target Info Definition

Bits	Name	Description
0–23	—	Reserved
24–31	TID	Target ID field from the received doorbell packet

Table 17-102. Source Info Definition

Bits	Name	Description
0–7	—	Reserved
8–15	SID	Source ID field from the received doorbell packet
16–31	INFO	Info field from the received doorbell packet

Figure 17-84 depicts the doorbell queue entry fields and their related offsets.

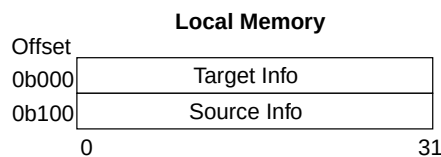


Figure 17-84. Doorbell Entry Format

17.8.4.4.3 Retry Response Conditions

There is one condition in which a doorbell message is logical layer retried (response “retry”).

- Doorbell request received and the doorbell circular queue is full.

17.8.4.4.4 Error Response Conditions

Several conditions cause a logical layer error response (response “error”).

- Doorbell controller is error state.
- Doorbell controller is not enabled.

17.8.4.4.5 Doorbell Controller Interrupts

The doorbell controller generates two different interrupts which can be individually enabled.

- Doorbell-in-queue interrupt. This interrupt is generated each time the circular queue becomes not empty and the interrupt is enabled. DSR[DIQI] is set if DMR[DIQIE] is set.
- Queue full interrupt. This interrupt is generated each time the circular queue becomes full and the interrupt is enabled. DSR[QFI] is set if DMR[QFIE] is set.

17.8.4.5 Port-Write Controller Structure

The implementation of the port-write controller is very similar to the inbound message hardware. 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 17-85 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 puts an incoming port-write data payload in the location indicated by the pointer and then sets its PWSR[QFI] if PWMR[QFIE] is set. The local processor must manually clear PWSR[QFI] after servicing the port-write. While the full bit is set, or if it is busy handling a previous port write, or the port-write controller is not enabled, the controller silently discards all incoming port-write packets.

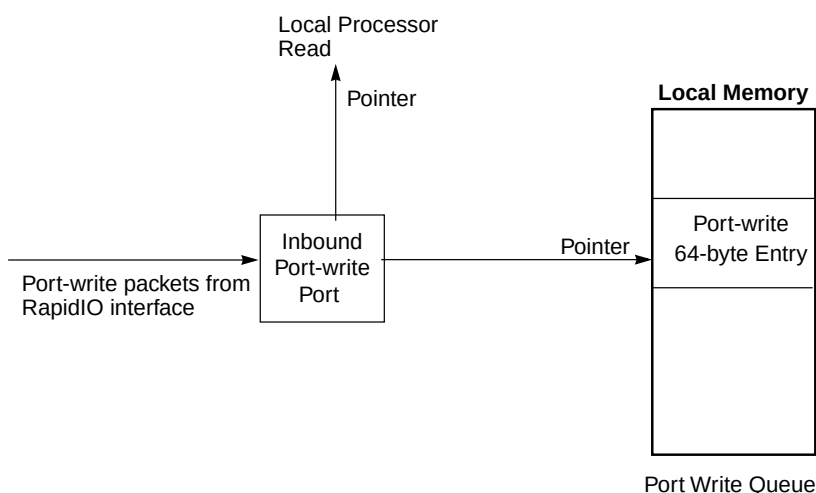


Figure 17-85. Inbound Port-Write Structure

Upon successfully receiving a port-write, if PWMR[QFIE] is set, then PWSR[QFI] is set and an interrupt is generated to the local processor.

17.9 Initialization and Application Information

The following steps outline the RapidIO port initialization procedure:

1. Configure the transmit clock select pins to select either the receive clock, externally provided transmit clock or the platform clock for the RapidIO transmit clock domain.
2. Initialize the host/agent field (using configuration pins) to put the end-point in host or agent mode over RapidIO.
3. Initialize the Device ID field (using configuration pins) to assign a device identifier at power-up.
4. The address translation and mapping unit (ATMU) for RapidIO must be initialized for RapidIO accesses.
5. Clear the CR[AA] bit to allow the logic to detect an error when it receives packets with a target ID not equal to its own.
6. Follow the system initialization and discovery mechanism described in the RapidIO inter-operability specifications for configuring the RapidIO system.

Part IV

Global Functions and Debug

This part defines other global blocks of the MPC8540. It contains the following chapters:

- Chapter 18, “Global Utilities,” defines the global utilities of the MPC8540. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general purpose I/O signal use and multiplexing for the interrupt and local bus chip select signals
- Chapter 19, “Performance Monitor,” describes the performance monitor of the MPC8540.
- Chapter 20, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8540.
- Chapter 21, “10/100 Ethernet Controller,” includes information about using this dual-speed Ethernet controller of the MPC8540 for debug purposes.

Chapter 18

Global Utilities

18.1 Introduction

This chapter describes the global utilities of the MPC8540. It provides signal descriptions, register descriptions and a functional description of these utilities.

18.1.1 Overview

The global utilities block controls several device-level functions. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal configuration, and alternate function selection for multiplexed signals.

18.1.2 Global Utilities Features

This section provides a summary overview of the features that comprise the global utilities.

18.1.2.1 Power Management and Device Disables

The following features can be enabled/disabled to affect the overall power consumption of the device:

- Dynamic power management mode
- Software-controlled power management (doze, nap, sleep)
- Externally controlled power management (doze, sleep)
- Static power management (I/O device disables)

18.1.2.2 Software Access to 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.

18.1.2.3 General-Purpose I/O

Two groups of signals can optionally be used as general-purpose I/O signals when not used for their primary function. These signals include specific PCI and TSEC2 data buses. Memory-mapped registers in the global utilities block provide control and status for these signals.

18.1.2.4 Interrupt and Local Bus Signal Multiplexing

Other signals can have multiple functions that can be selected by configuration registers in the global utilities block. These signals include the following:

- IRQ[9:11]
- $\overline{\text{LCS}}$ [5:7]

18.1.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

18.2 External Signal Description

18.2.1 Signals Overview

Table 18-1 summarizes the external signals used by the global utilities block.

Table 18-1. External Signal Summary

Signal Name	I/O	Idle State	State Meaning	Timing/Comments	Section No.
ASLEEP	O	LOW	Signals that the device has reached a sleep state.	—	Table 18-2
$\overline{\text{CKSTP\_IN}}$	I	HIGH	Checkstop input	Digitally filtered	Table 18-2
$\overline{\text{CKSTP\_OUT}}$	O	HIGH	Checkstop output	Negated with the negation of $\overline{\text{HRESET}}$	Table 18-2
CLK_OUT	O		Clock out. Selected by CLKOCR values.	—	Table 18-2
SYSCLK	I	—	System clock input (also used as PCI clock)	—	Chapter 4, "Reset, Clocking, and Initialization"
RTC	I	—	Real time clock input to e500 time base and/or system timers.	Synchronized to CCB clock (see Section 4.4.3.1, "System PLL Ratio").	Chapter 4, "Reset, Clocking, and Initialization"
$\overline{\text{SRESET}}$	I	HIGH	Soft reset	Digitally filtered	Chapter 4, "Reset, Clocking, and Initialization"

Table 18-1. External Signal Summary (continued)

Signal Name	I/O	Idle State	State Meaning	Timing/Comments	Section No.
$\overline{\text{HRESET}}$	I	HIGH	Hard reset Assertion takes affect asynchronously. Negation should be synchronous	Sampled by SYSCLK after analog filtering	Chapter 4, "Reset, Clocking, and Initialization"
$\overline{\text{HRESET\_REQ}}$	O	HIGH	Hard reset request	—	Chapter 4, "Reset, Clocking, and Initialization"
$\overline{\text{UDE}}$	I	HIGH	Unconditional debug event	Digitally filtered	Chapter 10, "Programmable Interrupt Controller"
$\overline{\text{MCP}}$	I	HIGH	Machine check processor	Digitally filtered	Chapter 10, "Programmable Interrupt Controller"

18.2.2 Detailed Signal Descriptions

Table 18-2 shows the detailed descriptions of the signals in the global utilities block.

Table 18-2. Detailed Signal Descriptions

Signal	I/O	Description
$\overline{\text{CKSTP\_IN}}$	I	Checkstop in
		State Meaning Asserted—Indicates that the MPC8540 processor core must enter a hard stop condition. Clocks in the e500 core will be turned off. $\overline{\text{CKSTP\_OUT}}$ will be asserted. System logic including memory controllers, internal memories and registers, and I/O interfaces will remain active . 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 system has been reset with a hard reset.

Table 18-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
CKSTP_OUT	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e500 core of the MPC8540 is in a checkstop state. The system logic in the MPC8540 platform will remain functional . Negated—Indicates normal operation. After CKSTP_OUT has been asserted, it is negated after the next negation of HRESET.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the system has been reset with a hard reset.
ASLEEP	O	Asleep. See Section 18.4.1.6, “Sleep Mode.” After the negation of HRESET the ASLEEP signal is asserted until the system 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 core and the system logic have exited the power-on sequence, and the device is not in sleep mode.
		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.
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR register (see Section 18.3.1.16, “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].

18.3 Memory Map/Register Definition

Table 18-3 summarizes the global utilities registers and their addresses.

Table 18-3. Global Utilities Block Register Summary

Offset	Register	Access	Reset	Section/Page
Power-On Reset Configuration Values				
0xE_0000	POR PLL ratio status register (PORPLLSR)	R	0x00nn_00nn	18.3.1.1/18-5
0xE_0004	POR boot mode status register(PORBMSR)	R	0xnnnn_0000	18.3.1.2/18-6
0xE_0008	POR I/O impedance status and control register (PORIMPSCR)	R/W	0x000n_003F	18.3.1.3/18-8
0xE_000C	POR I/O device status register (PORDEVSR)	R		18.3.1.4/18-9
0xE_0010	POR debug mode status register (PORDBGMSR)	R		18.3.1.5/18-10
0xE_0020	General-purpose POR configuration register (GPPORCR)	R		18.3.1.6/18-11
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIO control register (GPIOCR)	R/W	0x0000_0000	18.3.1.7/18-11
0xE_0040	General-purpose output data register (GPOUTDR)	R/W	0x0000_0000	18.3.1.8/18-12
0xE_0050	General-purpose input data register (GPINDR)	R	0xnnnn_nnnn	18.3.1.9/18-13

Table 18-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0060	Alternate function signal multiplex control (PMUXCR)	R/W	0x0000_0000	18.3.1.9/18-13
Device Disables				
0xE_0070	Device disable control (DEVDISR)	R/W	0x0000_0000	18.3.1.11/18-15
Power Management Registers				
0xE_0080	Power management status and control register (POWMGTCSR)	R/W	0x0000_0000	18.3.1.12/18-17
Interrupt Reporting				
0xE_0090	Machine check summary register (MCPSUMR)	Read/Clear	0x0000_0000	18.3.1.13/18-19
Version Registers				
0xE_00A0	Processor version register (PVR)	R	e500 processor version	18.3.1.14/18-20
0xE_00A4	System version register (SVR)	R	MPC8540 system version	18.3.1.15/18-20
Debug Control				
0xE_0E00	Clock out select register (CLKOCR)	R/W	0x0000_0000	18.3.1.16/18-21
0xE_0E10	DDR DLL control register (DDRDLPCR)	R/W	0x0000_0000	18.3.1.17/18-21
0xE_0E20	LBC DLL control register (LBDLLCR)	R/W	0x0000_0000	18.3.1.18/18-22

18.3.1 Register Descriptions

This section describes the global utilities registers in detail.

18.3.1.1 POR PLL Status Register (PORPLLSR)

Shown in Figure 18-1, the PORPLLSR contains the current settings for the PLL ratios as set by power-on reset (POR) configuration pins.

	0						6	7	8	9	10				15
R	0	0	0	0	0	0	0	0	0	0	0	e500_Ratio			
W															
Reset	0000_0000_00nn_nnnn														
	16						22	23	24	25	26				31
R	0	0	0	0	0	0	0	0	0	0	0	Plat_Ratio			BYP
W															
Reset	0000_0000_00nn_nnnn														
Offset	0xE_0000														

Figure 18-1. POR PLL Status Register (PORPLLSR)

Table 18-4 describes the bit settings of PORPLLSR.

Table 18-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	e500_Ratio	Clock ratio between the e500 core and the CCB clock. 000100—2:1 000101—5:2 000110—3:1 000111—7:2 All other patterns are reserved.
16–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the CCB (platform) clock and SYSCLK. 00010 - 2:1 00011 - 3:1 00100 - 4:1 00101 - 5:1 00110 - 6:1 01000 - 8:1 01001 - 9:1 01010 - 10:1 01100 - 12:1 10000 - 16:1 All other patterns are reserved.
31	BYP	PLL bypass mode

18.3.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 18-2, reports the boot mode settings sampled from power-on-reset configuration pins. The configuration pins affect the default settings of RapidIO and PCI host/agent modes. For further information about setting these configurations, see Section 4.4.3.4, “Host/Agent Configuration.” Also, for more information about the RapidIO interface and how it is configured, see Section 17.3.2.19,

“Port General Control Command and Status Register (PGCCSR).” For more information about the PCI configurations, see Section 16.3.2.15, “PCI Bus Function Register (PBFR).”

	0	1	2	3	4	5	7	8	9	10	11	12	13	14	15
R	BCFG	0	0	0	0	ROM_LOC		0	0	BSCFG		0	0	HA	
W															
Reset	n000_0nnn_00nn_00nn														
	16														31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
Offset	0xE_0004														

Figure 18-2. POR Boot Mode Status Register (PORBMSR)

Figure 18-5 describes the bit settings of the PORBMSR.

Table 18-5. PORBMSR Field Descriptions

Bits	Name	Description
0	BCFG	CPU boot configuration 0 The CPU is prevented from booting until configuration by an external master is complete. 1 The CPU is allowed to start fetching boot code.
1–4	—	Reserved
5–7	ROM_LOC	Location of boot ROM: 000 PCI/PCI-X 001 DDR SDRAM 011 RapidIO 101 Local bus GPCM: 8-bit 110 Local bus GPCM:16-bit 111 Local bus GPCM: 32-bit
8–9	—	Reserved
10–11	BSCFG	Boot sequencer configuration 00 Reserved 01 Boot sequencer enabled with normal I ² C addressing 10 Boot sequencer enabled with extended I ² C addressing 11 Boot sequencer disabled
12–13	—	Reserved
14–15	HA	Host/agent mode configuration. When the MPC8540 is an agent on an interface, it is prevented from mastering transactions on that interface until the external host configures the interface appropriately. 00 PCI and RapidIO agent mode 01 RapidIO agent mode 10 PCI/PCI-X agent mode 11 Host mode
16–31	—	Reserved

18.3.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

Shown in Figure 18-3, the PORIMPSCR contains the current settings for the I/O driver impedances for the PCI and local bus interfaces. The local bus clock signal is controlled by writing to this register. Other local bus and PCI signals are controlled by POR configuration pins and are reported by this register. Refer to the *MPC8540 Hardware Specification* for exact I/O impedances.

	0										14				15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PCI_Z	
W																
Reset	0000_0000_0000_000n															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	LADP_Z	LA_CKE_Z	LCS12_Z	LWE_Z	LGPL_Z	LCLK_Z
W																
Reset	0000_0000_0011_1111															
Offset	0xE_0008															

Figure 18-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

Table 18-6 describes the bit settings of PORIMPSCR.

Table 18-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	PCI_Z	PCI/PCI-X I/O impedance 0 Low impedance 1 High impedance
16–25	—	Reserved
26	LADP_Z	I/O impedance for local bus address/data and data parity (LAD[0:31] and LDP[0:7]) 0 Low impedance 1 High impedance
27	LA_CKE_Z	I/O impedance for local bus address and clock enable (LA[27:31] and LCKE) 0 Low impedance 1 High impedance
28	LCS12_Z	I/O impedance for two local bus chip selects (LCS[1] and LCS[2] only). NOTE: Other chip selects use a fixed high I/O impedance 0 Low impedance 1 High impedance
29	LWE_Z	I/O impedance for local bus write enables (LWE[0:3]) 0 Low impedance 1 High impedance

Table 18-6. PORIMPSR Field Descriptions (continued)

Bits	Name	Description
30	LGPL_Z	I/O impedance for local bus general-purpose lines (LGPL0 - LGPL5) 0 Low impedance 1 High impedance
31	LCLK_Z	I/O impedance for local bus clocks (LCLK[0:2]) 0 Low impedance 1 High impedance

18.3.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 18-4, the PORDEVSR reports the power-on-reset settings for I/O devices.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ECW	0	0	0	0	0	ECP		PCI	0	0	0	0	0	PCI_ARB	PCI32
W																
Reset	n000_00nn_n000_00nn															

	16	17	18	19	20	21	22	23	24	31						
R	0	0	0	0	0	0	RIOCLK		DEV_ID							
W																
Reset	0000_00nn_nnnn_nnnn															
Offset	0xE_000C															

Figure 18-4. POR Device Status Register (PORDEVSR)

Table 18-7 describes the bit settings of PORDEVSR.

Table 18-7. PORDEVSR Field Descriptions

Bits	Name	Description
0	ECW	Gigabit Ethernet controller width 0 Reduced (RGMII, RTBI) 1 Full (MII, GMII, TBI)
1–5	—	Reserved
6–7	ECP	Gigabit Ethernet controller protocol 00 Both TSEC blocks use a Media Independent Interface protocols (MII/GMII/RGMII) 01 TSEC1 uses MII, TSEC2 uses TBI 10 TSEC1 uses TBI, TSEC2 uses MII 11 Both TSEC blocks use a ten bit interface protocol (TBI or RTBI)
8	PCI	PCI/PCI-X mode 0 PCI-X mode 1 PCI mode
9–13	—	Reserved

Table 18-7. PORDEVSR Field Descriptions (continued)

Bits	Name	Description
14	PCI_ARB	PCI/PCI-X arbiter enable 0 PCI/PCI-X arbiter is disabled 1 PCI/PCI-X arbiter is enabled
15	PCI32	PCI/PCI-X interface width 0 64-bit 1 32-bit
16–21	—	Reserved
22–23	RIOCLK	RapidIO transmit clock source 00 Reserved 01 RapidIO receive clock 10 RapidIO transmit clock input 11 CCB clock (default)
24–31	DEV_ID	RapidIO device ID

18.3.1.5 POR Debug Mode Status Register (PORDBGMSR)

Shown in Figure 18-5, the PORDBGMSR contains the debug mode settings from the power-on-reset configuration pins.

	0	1	2	3	4	5	6	7	8						15
R	0	0	0	0	0	MEM_SEL	DDR_DBG	PCI_DBG	0	0	0	0	0	0	0
W															
Reset	0000_0nnn_0000_0000														
	16														31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
Offset	0xE_0010														

Figure 18-5. POR Debug Mode Status Register (PORDBGMSR)

Table 18-8 describes the bit settings of PORDBGMSR.

Table 18-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	MEM_SEL	Memory select. Indicates which memory controller is driving the MSRCID[0:4] and MDVAL memory debug signals 0 Local bus controller is driving debug information 1 DDR SDRAM controller is driving debug information

Table 18-8. PORDBGMSR Field Descriptions (continued)

Bits	Name	Description
6	DDR_DBG	DDR debug configuration 0 SourceID and data valid information is being driven on ECC pins of DDR SDRAM interface 1 Normal mode. ECC information is being driven on ECC pins of DDR SDRAM interface
7	PCI_DBG	PCI/PCI-X debug configuration 0 PCI/PCI-X drives source ID onto address bits 63:59. 1 PCI/PCI-X drives address 63:59 onto address bits 63:59.
8–31	—	Reserved

18.3.1.6 General-Purpose POR Configuration Register (GPPORCR)

Shown in Figure 18-6, the GPPORCR contains the general-purpose power-on-reset configuration register value sampled from the local bus address/data signals LAD[0:31].

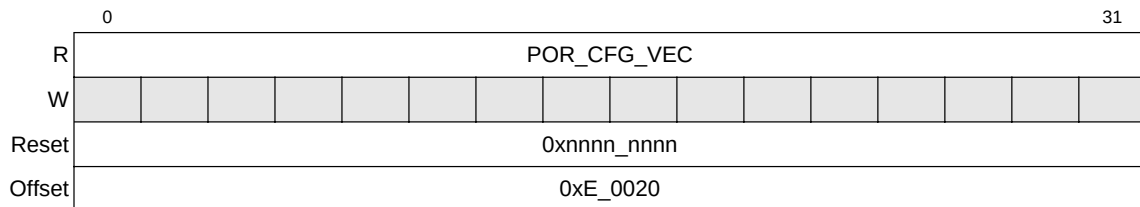
**Figure 18-6. POR Configuration Register (GPPORCR)**

Table 18-9 describes the bit settings of GPPORCR.

Table 18-9. 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 HRESET.

18.3.1.7 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 18-7, the GPIOCR contains the enable bits for each group of pins that may be used for general-purpose I/O. These bits only have meaning if the pins are not being used for their primary function.

**Figure 18-7. General-Purpose I/O Control Register (GPIOCR)**

Table 18-10 describes the bit settings of GPIOCR.

Table 18-10. GPIOCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	Tx2out	Enables the TSEC2_Tx[0:7] pins to be used as general-purpose output if the TSEC2 interface is disabled.
7	Rx2in	Enables the TSEC2_Rx[0:7] pins to be used as general-purpose input if the TSEC2 interface is disabled.
8–13	—	Reserved
14	PClout	Enables the PCI_AD[47:40] pins to be used as general-purpose output if the PCI interface is in 32-bit mode or is disabled.
15	PClin	Enables the PCI_AD[39:32] pins to be used as general-purpose input if the PCI interface is in 32-bit mode or is disabled.
16–31	—	Reserved

18.3.1.8 General-Purpose Output Data Register (GPOUTDR)

Shown in Figure 18-8, the GPOUTDR contains the data currently being driven as general-purpose output on the TSEC2_TxD[0:7] pins and/or the PCI_AD[47:40] pins. Writes to this register only affect the pins that are enabled as general-purpose outputs. Reads only return valid data for bits corresponding to pins that are enabled as general-purpose outputs. This register may be accessed using single byte writes (using big endian addressing) so that writes to one byte do not affect outputs controlled by other bytes.

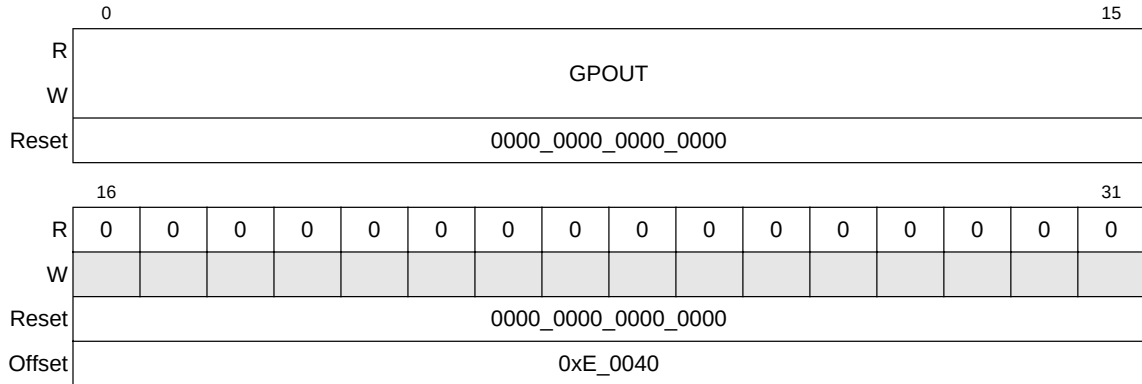


Figure 18-8. General-Purpose Output Data Register (GPOUTDR)

Table 18-11 describes the fields of GPOUTDR.

Table 18-11. GPOUTDR Field Descriptions

Bits	Name	Description
0–15	GPOUT	General-purpose output data. When the corresponding signals are configured to be general-purpose output signals, the values of the bits of GPOUT are driven onto those pins. GPOUTDR[0:7] corresponds to TSEC2_TxD[0:7]. GPOUTDR[8:15] corresponds to PCI_AD[47:40] as follows: GPOUTDR[8] -> PCI_AD[47] GPOUTDR[9] -> PCI_AD[46] GPOUTDR[10] -> PCI_AD[45] GPOUTDR[11] -> PCI_AD[44] GPOUTDR[12] -> PCI_AD[43] GPOUTDR[13] -> PCI_AD[42] GPOUTDR[14] -> PCI_AD[41] GPOUTDR[15] -> PCI_AD[40]
16–31	—	Reserved, should be cleared

18.3.1.9 General-Purpose Input Data Register (GPINDR)

Shown in Figure 18-9, the GPINDR contains the data currently sampled as general-purpose input on the TSEC2_RxD[0:7] pins and/or the PCI_AD[39:32] pins. Bits in this register are only updated if the associated bits are configured as general-purpose input pins rather than their primary functions.

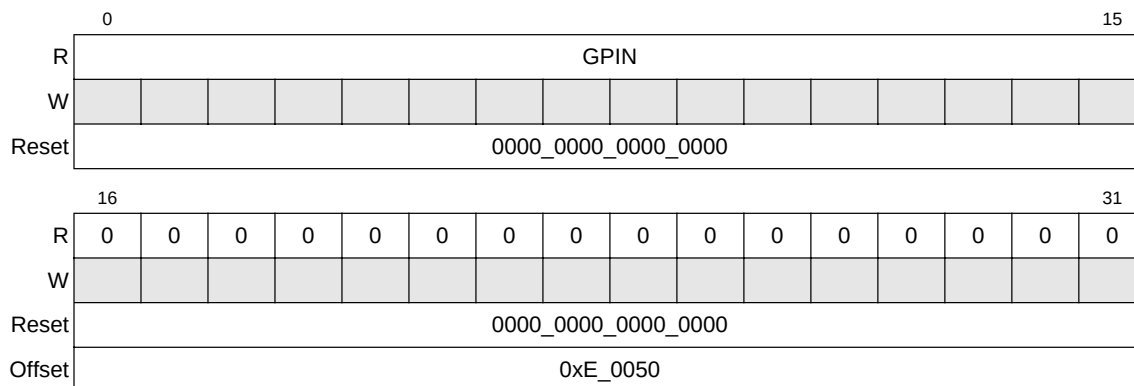
**Figure 18-9. General-Purpose Output Data Register (GPINDR)**

Table 18-11 describes the fields of GPINDR.

Table 18-12. GPINDR Field Descriptions

Bits	Name	Description
0–15	GPIN	General-purpose input data. When the corresponding signals are configured to be general-purpose input signals, the values sampled on these signals are reflected in GPIN. GPINDR[0:7] corresponds to TSEC2_RxD[0:7]. GPINDR[8:15] corresponds to PCI_AD[39:32] as follows: GPINDR[8] -> PCI_AD[39] GPINDR[9] -> PCI_AD[38] GPINDR[10] -> PCI_AD[37] GPINDR[11] -> PCI_AD[36] GPINDR[12] -> PCI_AD[35] GPINDR[13] -> PCI_AD[34] GPINDR[14] -> PCI_AD[33] GPINDR[15] -> PCI_AD[32]
16–31	—	Reserved

18.3.1.10 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 18-10, the PMUXCR contains bits that enable DMA channels 2 and 3 which exist as alternate functions on local bus chip select pins LCS[5:7], and interrupt input pins IRQ[9:11], respectively. Specifically, DMA request, acknowledge, and done signals comprise the secondary functions for the associated IRQ and local bus chip select signals.

	0													14	15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DMA2
W																
Reset	0000_0000_0000_0000															
	16													30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DMA3
W																
Reset	0000_0000_0000_0000															
Offset	0xE_0060															

Figure 18-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 18-13 describes the bit settings of PMUXCR.

Table 18-13. PMUXCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	DMA2	Enables DMA channel 2 signals. 0 DMA channel 2 is disabled; the pins retain their primary function as local bus chip selects 1 DMA channel 2 is enabled and: LCS5 acts as <u>DMA_DREQ2</u> LCS6 acts as <u>DMA_DACK2</u> LCS7 acts as <u>DMA_DDONE2</u>
16–30	—	Reserved
31	DMA3	Enables DMA channel 3 signals. 0 DMA channel 3 is disabled; the pins retain their primary function as interrupt requests . 1 DMA channel 3 is enabled and: IRQ9 acts as <u>DMA_DREQ3</u> IRQ10 acts as <u>DMA_DACK3</u> IRQ11 acts as <u>DMA_DDONE3</u>

18.3.1.11 Device Disable Register (DEVDISR)

Shown in Figure 18-11, the DEVDISR contains disable bits for various functional blocks (devices). All functional blocks are enabled after reset, but un-needed blocks can be disabled in order to reduce power consumption or to use their signals as general-purpose I/O signals.

Memory Map/Register Definition

	0	1	2	3	4	5					10	11	12	13	14	15
R	PCI	0	0	0	LBC	0	0	0	0	0	0	L2	RIO	0	0	DDR
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	e500	TB	0	0	0	DMA	0	0	TSEC1	TSEC2	DSEC	0	0	I2C	DUART	0
W																
Reset	0000_0000_0000_0000															
Offset	0xE_0070															

Figure 18-11. Device Disable Register (DEVDISR)

Table 18-14 describes the bit settings of DEVDISR

Table 18-14. DEVDISR Field Descriptions

Bits	Name	Description
0	PCI	PCI/PCI-X controller disable 0 PCI/PCI-X controller enabled 1 PCI_AD[63:32] may be used as general-purpose I/O
1–3	—	Reserved
4	LBC	Local bus controller disable 0 Local bus controller enabled 1 Local bus controller disabled
5–10	—	Reserved
11	L2	Level-2 cache disable 0 L2 cache enabled 1 L2 cache disabled
12	RIO	RapidIO controller disable 0 RapidIO controller enabled 1 RapidIO controller disabled, including RapidIO messaging, doorbell, and port write controllers.
13–14	—	Reserved
15	DDR	DDR SDRAM controller disable 0 DDR SDRAM controller enabled 1 DDR SDRAM controller disabled
16	e500	e500 core disable 0 e500 core is enabled 1 e500 core is disabled. This is equivalent to NAP mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core with the exception of the time base.
17	TB	Time base unit of the e500 core disable 0 Time base unit enabled 1 Time base unit disabled.
18–20	—	Reserved

Table 18-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
21	DMA	DMA controller disabled 0 DMA controller enabled 1 DMA controller disabled
22–23	—	Reserved
24	TSEC1	Three-speed Ethernet controller 1 disable 0 TSEC1 enabled 1 TSEC1 disabled
25	TSEC2	Three-speed Ethernet controller 2 disable 0 TSEC2 enabled 1 TSEC2 disabled. Rx/D and Tx/D pins may be used for general-purpose I/O
26	DSEC	Dual-speed Ethernet controller (10/100 maintenance port) disable 0 DSEC enabled 1 DSEC disabled
27–28	—	Reserved
29	I2C	I ² C controller disabled 0 I ² C controller enabled 1 I ² C controller disabled
30	DUART	Dual UART controller disabled 0 DUART enabled 1 DUART disabled
31	—	Reserved

18.3.1.12 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 18-12, the POWMGTCR contains control bits for placing the MPC8540 into low power states and for controlling when it wakes up. POWMGTCR also contains status bits for reporting the current power management state.

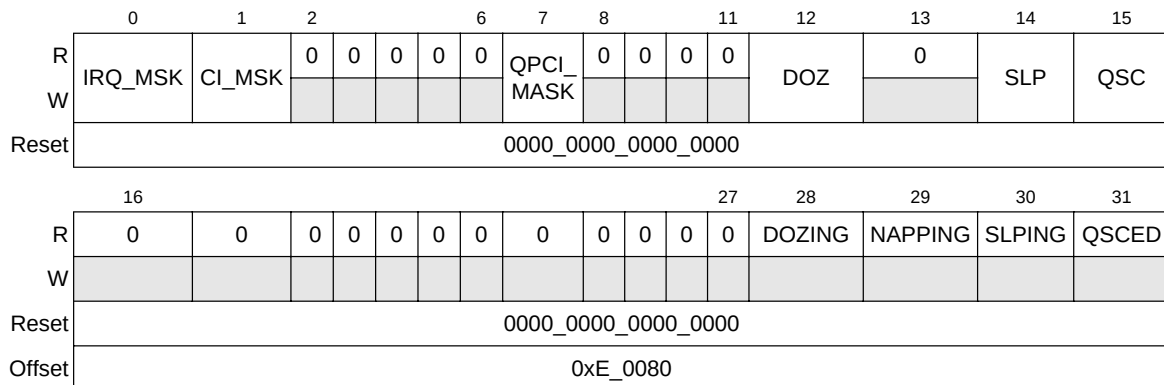
**Figure 18-12. Power Management Control & Status Register (POWMGTCSR)**

Table 18-15 describes the bit settings of POWMGTCR.

Table 18-15. POWMGTCR Field Descriptions

Bits	Name	Description
0	IRQ_MSK	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	CI_MSK	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.
2–6	—	Reserved
7	QPCI_MSK	PCI quiesce mask 0 In quiesce mode the PCI interface does not respond to inbound requests. PCI masters are not able to access the device during quiesce mode 1 The PCI interface is masked from quiesce mode. This enables debug via debug masters on the PCI interface in addition to the system access port (SAP).
8–11	—	Reserved
12	DOZ	Doze mode 0 No request to put device in doze mode. 1 Device is to be placed in doze mode. Instruction fetching is halted in the e500 core. Note that this bit is logically ORed with the HID0[DOZE] bit.
13	—	Reserved
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 in the e500 core, snooping of L1 caches is disabled, and most functional blocks are shut down in both the e500 core and the system logic.
15	QSC	Quiesce mode (useful for debug) 0 No request to put device in quiesce mode 1 Device is to be placed in quiesce mode. Instruction fetching is halted in the e500 core, and all interfaces that master transactions into the system are quiesced with the exception of the system access port (SAP). The PCI interface also continues to accept transactions if the QPCI_MSK bit is set. When this bit is cleared by the SAP or PCI master, the device exits quiesce mode. External interrupts do not cause the device to exit quiesce mode.
16–27	—	Reserved
28	DOZING	Doze status 0 Device is not in doze mode. 1 The e500 core is in doze mode because the POWMGTCR[DOZ] bit is set in the system or because the HID0[DOZE] bit and MSR[WE] bits in the e500 core are set. The core has halted instruction fetching, but all other functional blocks in the core and system are running.
29	NAPPING	Nap status 0 Device is not in nap mode. 1 The e500 core is in nap mode because the HID0[NAP] bit and MSR[WE] bits in the e500 core 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 time base unit are shut down. All functional blocks in the system are running.

Table 18-15. POWMGTCR Field Descriptions (continued)

Bits	Name	Description
30	SLPING	Sleep status 0 Device is not attempting to reach sleep mode. 1 The device is attempting to SLEEP because the POWMGTCR[SLP] bit is set in the system or because the HIDE[SLEEP] bit and MSR[WE] bits in the e500 core are set. Most functional blocks in the core and system are shut down or are attempting to shut down.
31	QSCED	Quiesce status 0 Device is not in quiesce mode. 1 The device is quiesced because the POWMGTCR[QSC] bit is set in the system or because a debug event in the system has quiesced the device. All memories, configuration registers, and most I/O blocks are accessible, but only the system access port (SAP) and the PCI port (if the POWMGTCR[QPCI_MSK] bit is set) may master reads and writes into the system. All other mastering interfaces are shut down.

18.3.1.13 Machine Check Summary Register (MCPSUMR)

Shown in Figure 18-13, the MCPSUMR contains bits summarizing the source of a pending machine check interrupt. Bits may be cleared by writing a 1 to the desired bit.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															

	16											27				28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	WRS	SRESET	MCP_IN				
W																			
Reset	0000_0000_0000_0000																		
Offset	0xE_0090																		

Figure 18-13. Machine Check Summary Register (MCPSUMR)

Table 18-16 describes the bit settings of MCPSUMR.

Table 18-16. MCPSUMR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	WRS	Watchdog timer MCP 0 MCP was not caused by watchdog timer. 1 MCP was caused by a soft reset condition from the e500 watchdog timer as configured in the TSR register in the core. Specifically TSR[WRS] = 0b01, and a watchdog reset condition has occurred.

Table 18-16. MCPSUMR Field Descriptions (continued)

Bits	Name	Description
30	SRESET	Soft reset MCP 0 MCP was not caused by $\overline{\text{SRESET}}$ assertion. 1 MCP was caused by the assertion of the $\overline{\text{SRESET}}$ input signal.
31	MCP_IN	$\overline{\text{MCP}}$ signal asserted 0 MCP was not caused by $\overline{\text{MCP}}$ assertion. 1 MCP was caused by the assertion of the $\overline{\text{MCP}}$ input signal.

18.3.1.14 Processor Version Register (PVR)

Shown in Figure 18-14, the PVR contains the processor version number for the e500 core. This register is a memory-mapped copy of the PVR in the e500 core (and is therefore, accessible to external devices).

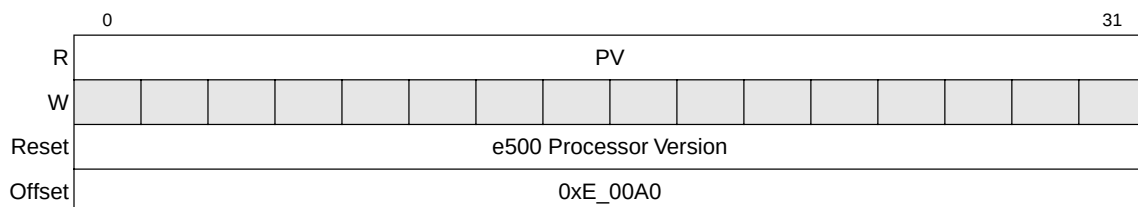
**Figure 18-14. Processor Version Register (PVR)**

Table 18-17 describes the fields of PVR.

Table 18-17. PVR Field Descriptions

Bits	Name	Description
0–31	PV	Processor version for the e500 CPU

18.3.1.15 System Version Register (SVR)

Shown in Figure 18-15, the SVR contains the system version number for the MPC8540 system logic. Note that value in this memory-mapped register is copied in the SVR register of the e500 core.

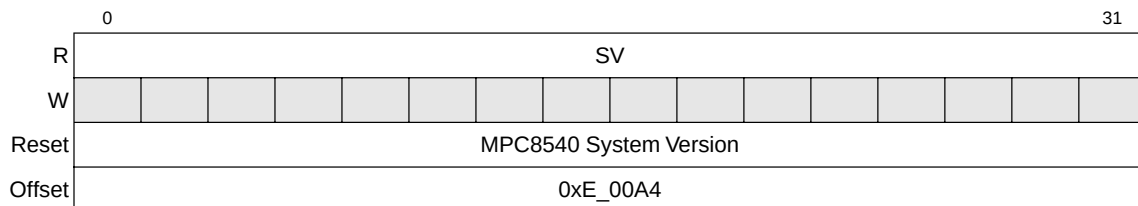
**Figure 18-15. System Version Register (SVR)**

Table 18-18 describes the fields of SVR.

Table 18-18. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version for the MPC8540 system logic

18.3.1.16 Clock Out Control Register (CLKOCR)

Shown in Figure 18-16, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

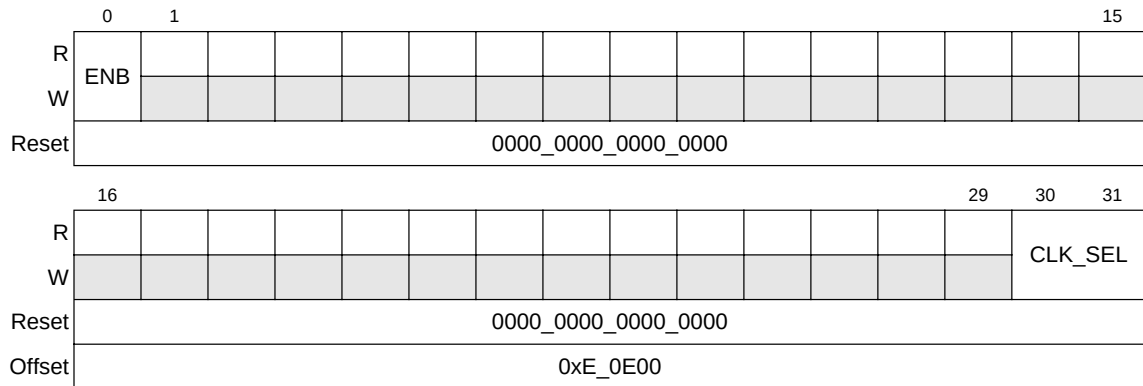
**Figure 18-16. Clock Out Control Register (CLKOCR)**

Table 18-19 describes the bit settings of CLKOCR.

Table 18-19. CLKOCR Field Descriptions

Bits	Name	Description
0	ENB	Clock out enable 0 CLK_OUT signal is tri-stated 1 CLK_OUT signal is driven according to CLKOCR[CLK_SEL]
1–29	—	Reserved
30–31	CLK_SEL	Clock out select 00 CCB (platform)clock 01 CCB (platform)clock divided by 2 10 SYSCLK (echos SYSCLK input) 11 SYSCLK divided by 2 (demonstrates platform PLL lock)

18.3.1.17 DDR DLL Control Register (DDRDLLCR)

Shown in Figure 18-17, the DDRDLLCR contains control bits that allow control and debug of the DDR SDRAM controller's DLL. The delay chain of the DLL is made up of 128 tap points.

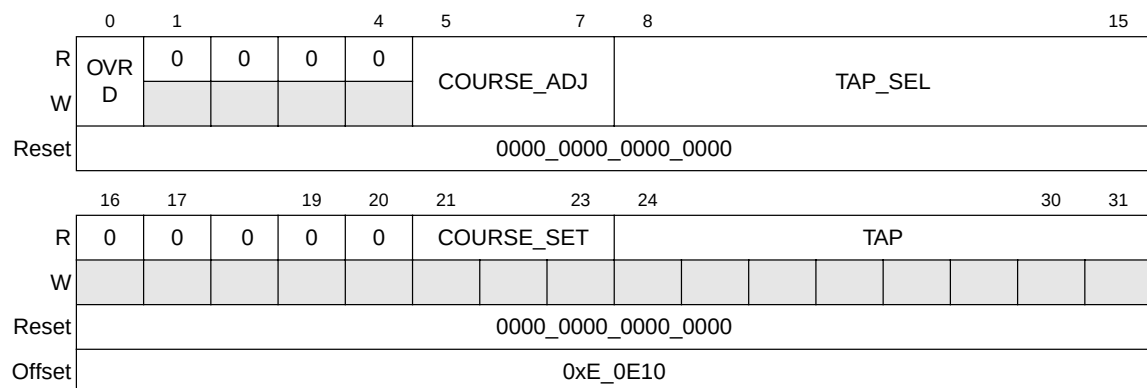


Figure 18-17. DDR DLL Control Register (DDRDLLCR)

Table 18-20 describes the bit settings of DDRDLLCR.

Table 18-20. DDRDLLCR Field Descriptions

Bits	Name	Description
0	OVRD	Override mode 0 Override mode disabled 1 Override of current delay chain tap point with the TAPSEL tap point enabled.
1–4	—	Reserved
5–7	COURSE_ADJ	Course adjustment value to be used by the DLL when in override mode (OVRD = 1). The course adjustment is the number of CCB (platform) clock cycles of delay to inject before the delay chain. When leaving override mode (OVRD cleared) this course adjustment point serves as the starting point for a dynamic search for a lock point.
8–15	TAP_SEL	Tap select value to be used by the DLL when in override mode (OVRD = 1). Selects the tap point within the delay chain. When leaving override mode (OVRD cleared) this tap point serves as the starting point for a dynamic search for a lock point.
16–20	—	Reserved
21–23	COURSE_SET	Reports the current course delay setting found by the dynamic search algorithm that produced a lock. Measured in CCB clock cycles.
24–31	TAP	Reports the tap value found by the dynamic search algorithm that produced a lock. Measured in tap points.

18.3.1.18 Local Bus DLL Control Register (LBDLLCR)

Shown in Figure 18-18, the LBDLLCR contains control bits that allow debug of the local bus controller's DLL. The delay chain of the DLL is made up of 128 tap points.

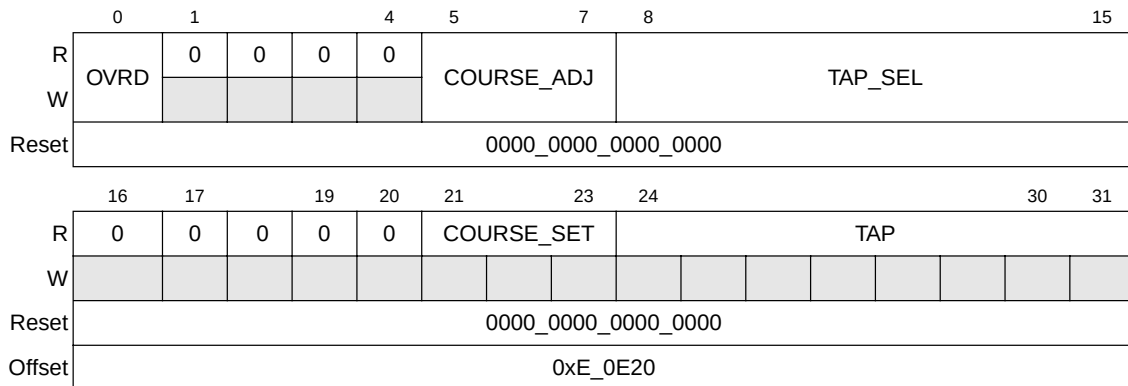
**Figure 18-18. Local Bus DLL Control Register (LBDLLCR)**

Table 18-21 describes the bit settings of LBDLLCR.

Table 18-21. LBDLLCR Field Descriptions

Bits	Name	Description
0	OVRD	Override mode 0 Override mode disabled 1 Override of current delay chain tap point with the TAPSEL tap point enabled.
1–4	—	Reserved
5–7	COURSE_ADJ	Course adjustment value to be used by the DLL when in override mode (OVRD = 1). The course adjustment is the number of CCB clock cycles of delay to inject before the delay chain. When leaving override mode (OVRD cleared) this course adjust point serves as the starting point for a dynamic search for a lock point.
8–15	TAP_SEL	Tap select value to be used by the DLL when in override mode (OVRD = 1). Selects the tap point within the delay chain. When leaving override mode (OVRD cleared) this tap point serves as the starting point for a dynamic search for a lock point.
16–20	—	Reserved
21–23	COURSE_SET	Reports the current course delay setting found by the dynamic search algorithm that produced a lock. Measured in CCB clock cycles.
24–31	TAP	Reports the tap value found by the dynamic search algorithm that produced a lock. Measured in tap points.

18.4 Functional Description

This section describes the global utilities from a functional perspective.

18.4.1 Power Management

The MPC8540 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. Additionally, software running on the e500 core can access the core's special purpose registers to put the

chip in three different power down states: doze, nap, or sleep. Finally, software running on an external host can access memory-mapped registers to put the device in the doze or sleep states. These features are described in further detail in this section.

18.4.1.1 Dynamic Power Management

Many blocks in the MPC8540 are capable of dynamically turning off clocks within the block when sections of the block are idle. This power savings feature is always enabled.

18.4.1.2 Shutting Down Unused Blocks

Several blocks within the MPC8540 can be shut down if they are not being used in a particular system by writing to the device disable register (DEVDISR) in the global utilities block. See Section 18.3.1.11, “Device Disable Register (DEVDISR). This register can be written by the e500 core or by an external master. Powering down a block in this way turns off all clocks to that block.

NOTE

If a block is powered down it cannot respond to configuration accesses. Access to configuration, control, and status registers of a block that is disabled is a programming error. If software writes to the DEVDISR register to re-enable clocks to a block, it should follow that write with a read of the DEVDISR register to force completion of the write before it attempts to access configuration, control, or status registers in that block.

18.4.1.3 Software-Controlled Power-Down States

Software running on the e500 core may place the device in doze, nap, or sleep power-down states by writing to the HID0 special purpose register in the core. In addition, external masters can write to the memory-mapped POWMGTCR control register in the MPC8540 to cause the device to enter doze or sleep modes.

18.4.1.4 Doze Mode

In doze mode, the e500 processor core of the MPC8540 suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1 cache is still supported in this mode. The data in the L1 caches is kept valid and coherent. Interrupts to the core cause the MPC8540 to exit doze mode to process the interrupt; however, unless the interrupt subroutine turns off or masks the control bits that enabled doze mode, the device re-enters doze mode after the interrupt has been serviced. See Section 18.4.1.8, “Interrupts and Power Management,” for more information.

The e500 core's time base is still enabled during doze mode, and time base interrupts can be generated. System logic external to the e500 core remains fully operational in doze mode.

18.4.1.5 Nap Mode

In nap mode all clocks internal to the e500 core are turned off with the exception of the 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.

In nap mode, interrupts cause the e500 core to wake up to service the interrupt, but unless the interrupt service routine changes the control bits that caused the device to enter nap mode, the MPC8540 returns to nap mode after the interrupt is serviced. See Section 18.4.1.8, “Interrupts and Power Management,” for more information.

System logic external to the e500 core remains fully operational in doze mode.

18.4.1.6 Sleep Mode

In sleep mode, all clocks internal to the e500 core are turned off, including the time base. In addition, all I/O interfaces in the system logic are shut down. Only the clocks to the MPC8540 interrupt controller are still running so that an external interrupt can wake the system up. After the core and the I/O interfaces are asleep, the ASLEEP signal is asserted.

Only external interrupts can wake the MPC8540 when it is asleep.

18.4.1.7 Power Management Control Bits

The e500 core provides the following control bits that are used to signal the power management requests to the MPC8540 device logic.

- MSR[WE]—The WE bit is used to qualify the values of the HID0[DOZE], HID0[NAP], and HID0[SLEEP] bits in the generation of the internal doze, nap and sleep signals to the system logic.
- HID0[DOZE]—Qualified with MSR[WE], signals the MPC8540 to initiate doze mode.
- HID0[NAP]—Qualified with MSR[WE], signals the MPC8540 to initiate nap mode.
- HID0[SLEEP]—Qualified with MSR[WE], signals the MPC8540 to initiate sleep mode.

Refer to the *e500 User's Manual* for more details on accessing these power management control bits.

18.4.1.8 Interrupts and Power Management

When an interrupt occurs, the core complex saves certain portions of the MSR to MCSRR1, CSRR1, or SRR1 (depending on the type of interrupt) on entry to the interrupt handler routine, and restores those values to the MSR on return from the interrupt handler routine. The MSR[WE] bit is always among the bits that are saved and restored. The MSR[WE] bit gates the doze, nap, and sleep power management outputs (internal device signals) from the core complex; hence these outputs negate to the MPC8540 power management logic when the interrupt begins processing in the core, and they return to their previous state when the core executes an **rfi**, **rfci**, or **rfmci** instruction.

This gated operation of the doze, nap, and sleep outputs by the MSR[WE] bit has the following implication for the design of power management software:

The return of the doze, nap, and sleep signals to their original state as the MSR[WE] bit is restored is a difference from the way in which power management was implemented on earlier devices that implemented the PowerPC architecture. In the previous devices, the MSR[POW] bit, which enabled power down requests, was cleared when the processor exited from a low-power state, and that bit was not automatically restored as it is in Book E implementations.

18.4.1.9 Snooping in the Power-Down State

During doze mode, the e500 core continues to snoop its L1 caches, and full coherency is maintained. In deeper power-down modes, however, the e500 core does not respond to snoops.

The MPC8540 does not perform dynamic bus snooping. That is, when the e500 core is in nap or sleep mode, the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, the L1 caches should be flushed if coherency is required during these power-down modes.

18.4.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8540 logic (external to the core complex) to enter a power saving state. The power saving state (doze, nap, or sleep) must be previously declared with the appropriate HID0 bit, typically at system start-up time. Setting MSR[WE] has no direct effect on instruction execution, but it is reflected on the internal doze, nap, sleep signals, depending on the setting of HID0[DOZE], HID0[NAP], and HID0[SLEEP], respectively. 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

```

18.4.2 General-Purpose I/O Signals

Several groups of signals can optionally be used as general-purpose I/O signals when not being used for their primary function. The general-purpose I/O functionality of these signals can be enabled through configuration registers in the global utilities block. These signals include:

- PCI_AD[47:40]
- PCI_AD[39:32]
- TSEC2_RxD[0:7]
- TSEC2_TxD[0:7]

When configured as general-purpose I/O signals, inputs can be read by software reading the associated GPIO data register. Output values can be set by software writing the desired values to the associated GPIO data register.

The PCI_AD pins have fixed directions when configured as general-purpose I/O. Specifically, PCI_AD[47:40] function as outputs, and PCI_AD[39:32] function as inputs.

The TSEC2 pins are fixed as either inputs or outputs based on the direction of the signal's primary function. The TSEC2_TxD pins are always outputs, so these signals may only be used as outputs when configured as general-purpose I/O. Similarly, the TSEC2_RxD pins are used as inputs when configured as general-purpose I/O.

The PCI_AD[47:32] pins can be used as general-purpose I/O if the PCI/PCI-X interface is disabled or if the PCI/PCI-X controller is not in 64-bit mode.

The TSEC2 TxD and RxD pins are available when the TSEC2 block is disabled. The TxD signals can then be enabled as general-purpose outputs and the RxD pins can be enabled as general-purpose inputs.

For details regarding the control and status of the general-purpose I/O signals, see Section 18.3.1.7, "General-Purpose I/O Control Register (GPIOCR)."

18.4.3 Interrupt and Local Bus Signal Multiplexing

The MPC8540 has very little signal multiplexing. Two sets of DMA channel triggering signals can alternately be placed on other signals as follows:

- IRQ[9:11]
- $\overline{\text{LCS}}$ [5:7]

The $\overline{\text{LCS}}$ [5:7] signals are multiplexed with DMA channel 2 $\overline{\text{DMA\_DREQ2}}$, $\overline{\text{DMA\_DACK2}}$, and $\overline{\text{DMA\_DONE2}}$ signals respectively, while the IRQ[9:11] signals are multiplexed with DMA channel 3 $\overline{\text{DMA\_DREQ3}}$, $\overline{\text{DMA\_DACK3}}$, and $\overline{\text{DMA\_DDONE3}}$ signals. For details regarding the selection of the alternate function DMA trigger, see Section 18.3.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

Chapter 19

Performance Monitor

19.1 Introduction

The MPC8540 includes a performance monitor facility that allows it to record/monitor selected behaviors of the integrated device. The performance monitor described in this chapter is distinctly different and separate from the performance monitor in the e500 core.

There are nine performance monitor counters (PMC0–PMC8) that can be used to count events selected by the performance monitor local control registers. Performance monitor counter 0 (PMC0) is a 64-bit counter that is specifically designated to count cycles. Performance monitor counters 1–8 (PMC1–PMC8) 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 MPC8540's 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.

19.1.1 Overview

Figure 19-1 shows a high level block diagram of the performance monitor. The design consists of a global control register (PMGC0), one 64-bit counter (PMC0), eight 32-bit counters, and two control registers per counter (18 total control registers). The global control register PMGC0 affects all the counters and takes priority over the local control registers. The local control registers are divided into two groups, local control A registers and local control B registers:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. The local control register PMLCA0 that controls counter PMC0 does not contain event selection because PMC0 only counts cycles.

Introduction

- Local control B registers control the start and stop triggering, and contain the counters' threshold values and the value of the threshold multiplier. Local control register PMLCB0 that also controls PMC0 does not contain threshold information because PMC0 only counts cycles.

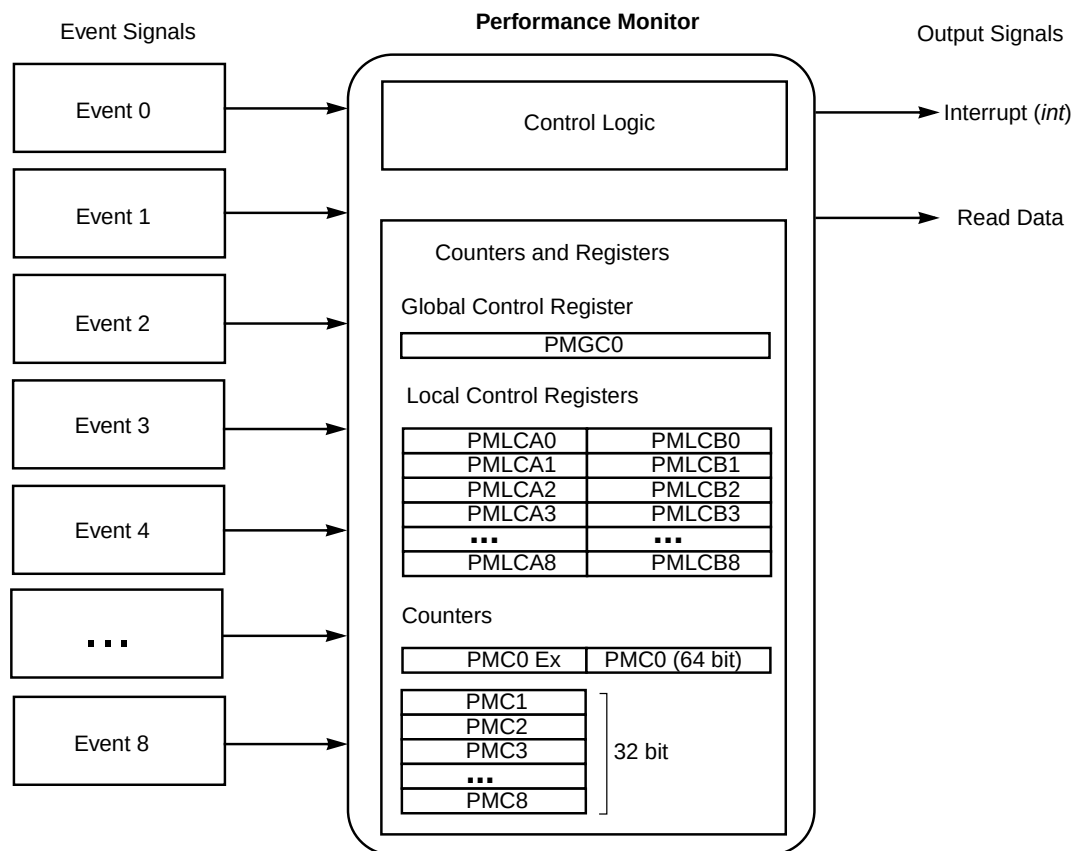


Figure 19-1. Performance Monitor Block Diagram

Performance monitor events are signalled by the functional blocks in the integrated device and are selectively recorded in the performance monitor counters. Sixty-four of these events are referred to as reference events. Reference events can be counted on any of the eight counters as opposed to the counter-specific events that can only be counted 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 performance monitor counters can also be programmed to freeze when an interrupt is signalled.

19.1.2 Features

The MPC8540 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
- Eight 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 eight 32-bit counters
- Ability to count up to 512 counter-specific events
- Triggering and chaining capability
- Duration threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

19.2 External Signals

Table 19-1 outlines the signals of the MPC8540 performance monitor facility. The inputs to the MPC8540 are internal event signals from other blocks on the device. 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.

Table 19-1. Performance Monitor Interface Signals

Group	Signal	I/O	Comments
Performance Monitor Interrupt	<i>int</i>	O	Performance monitor interrupt
Performance Monitor Events	Reference events and counter-specific event signals.	I	More information about these signals and their associated events can be found in Section 19.4.7, "Performance Monitor Events"

19.3 Memory Map and Register Definition

All of the performance monitor registers reside in the run-time register block starting at offset 0xE_1000. This section describes the registers implemented to support the performance monitor facilities. Table 19-2 lists the performance monitor control registers and the performance monitor counters. The performance monitor registers and counters are not byte-accessible and may only be read or written with 32-bit accesses.

19.3.1 Register Summary

The performance monitor uses nine counter registers and a group of local control registers that are used to specify the method of counting. There are two local control registers

associated with each counter in addition to a global control register that applies to all counters.

Table 19-2. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	Global control register (PMGC0)	R/W	0x0000_0000	19.3.2.1/19-5
0xE_1010	Local control register A0 (PMLCA0)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1014	Local control register B0 (PMLCB0)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1018	Performance monitor counter 0 (PMC0)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1010	Performance monitor counter 0 Extended (PMC0 Ex)	R/W	0x0000_0000	
0xE_1020	Local control register A1 (PMLCA1)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1024	Local control register B1 (PMLCB1)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1028	Performance monitor counter 1 (PMC1)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1030	Local control register A2 (PMLCA2)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1034	Local control register B 2 (PMLCB2)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1038	Performance monitor counter 2 (PMC2)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1040	Local control register A3 (PMLCA3)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1044	Local control register B3 (PMLCB3)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1048	Performance monitor counter 3 (PMC3)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1050	Local control register A4 (PMLCA4)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1054	Local control register B4 (PMLCB4)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1058	Performance monitor counter 4 (PMC4)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1060	Local control register A5 (PMLCA5)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1064	Local control register B 5 (PMLCB5)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1068	Performance monitor counter 5 (PMC5)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1070	Local control register A6 (PMLCA6)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1074	Local control register B6 (PMLCB6)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1078	Performance monitor counter 6 (PMC6)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1080	Local control register A7 (PMLCA7)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1084	Local control register B7 (PMLCB7)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1088	Performance monitor counter 7 (PMC7)	R/W	0x0000_0000	19.3.3.1/19-10
0xE_1090	Local control register A8 (PMLCA8)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1094	Local control register B8 (PMLCB5)	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1098	Performance monitor counter 8 (PMC8)	R/W	0x0000_0000	19.3.3.1/19-10

19.3.2 Control Registers

This section describes the performance monitor control registers in detail.

19.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 19-2, is a 32-bit register used to control all performance monitor counters.

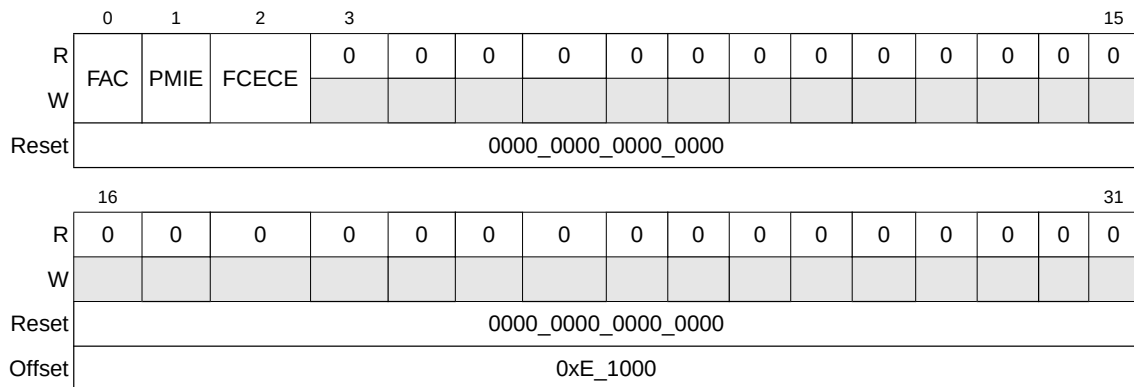


Figure 19-2. Performance Monitor Global Control Register (PMGC0)

Table 19-3 shows the field descriptions for the performance monitor global control register (PMGC0).

Table 19-3. PMGC0 Field Descriptions

Bits	Name	Description
0	FAC	Freeze all counters. 0 The PMCs are incremented (if permitted by other PMGC0/PMLC bits) 1 The PMCs are not incremented. Set by hardware when an interrupt is signalled and PMGC0[FCECE] is set.
1	PMIE	Performance monitor interrupt enable. Performance monitor interrupts are caused by performance monitor counter overflows. 0 Performance monitor interrupts are disabled. Software can clear PMIE to prevent performance monitor interrupts. 1 Performance monitor 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 most-significant-bit = 1 in PMC _n and PMLC _n [CE] = 1. The use of the trigger and freeze counter conditions depends on the enabled condition. 0 The PMCs can be incremented (if permitted by other control bits). 1 The 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 to 1. It is up to software to clear FAC to 0.
3–31	—	Reserved

19.3.2.2 Performance Monitor Local Control Registers (PMLC_n and PMLCB_n)

The performance monitor local control registers (PMLC_n and PMLCB_n) are used to control the operation of the performance monitor counters. The performance monitor local

Memory Map and Register Definition

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 19-3 shows the performance monitor local control A0 register (PMLCA0).

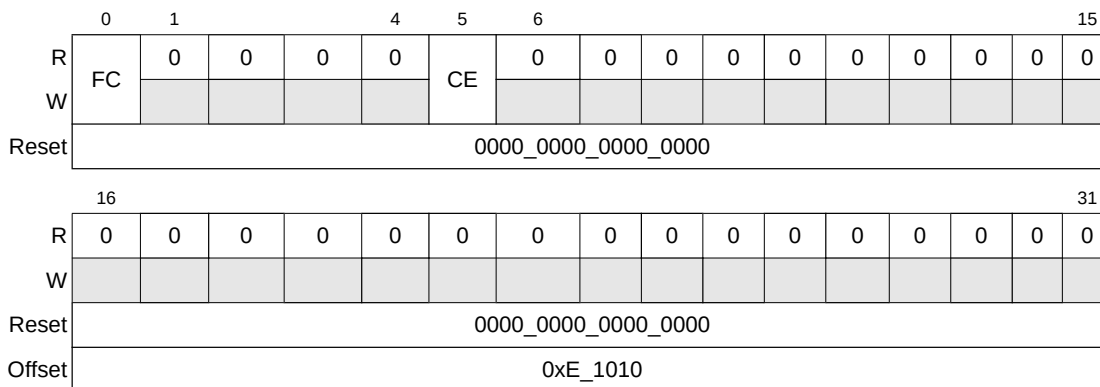


Figure 19-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 19-4 shows the field descriptions for the performance monitor local control A0 register (PMLCA0).

Table 19-4. 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, cannot freeze counters) 1 Overflow conditions occur when the most-significant-bit = 1 of PMC0
6–31	—	Reserved

Figure 19-4 shows the performance monitor local control registers A1–A8.

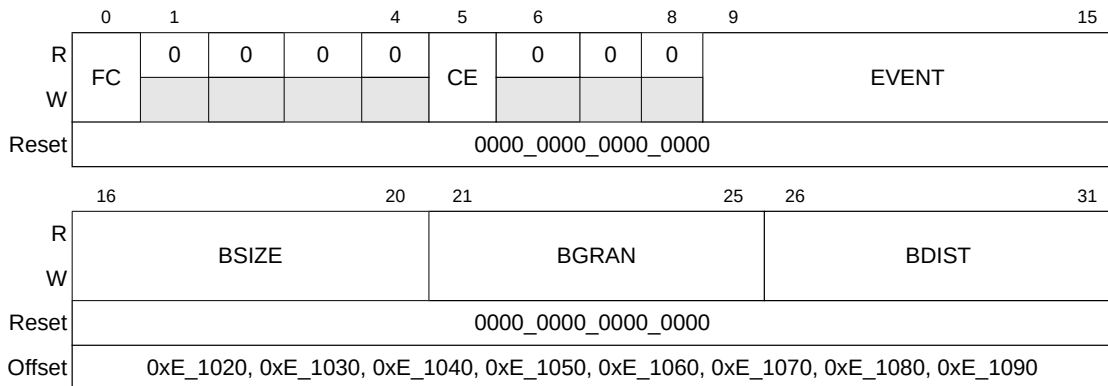
**Figure 19-4. Performance Monitor Local Control Register A[1–8] (PMLCA[1–8])**

Table 19-5 shows the field descriptions for the performance monitor local control A[1–8] registers (PMLCAn).

Table 19-5. PMLCA[1–8] Registers

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, cannot freeze counters) 1 Overflow conditions occur when the most-significant-bit = 1 of PMC _n CE should be cleared when counter PMC _n is used as a trigger or when PMC _n is selected for chaining.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable
16–20	BSIZE	Burst size. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion.
21–25	BGRAN	Burst granularity. See Section 19.4.6, “Burstiness Counting,” for more information.
26–31	BDIST	Burst distance (used with TBMULT). BDIST must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 19-5 shows the performance monitor local control B0 register (PMLCB0).

Memory Map and Register Definition

	0	1	2	5	6	7	8	11	12	13	14	15			
R	0	0	TRIGONSEL			0	0	TRIGOFFSEL			TRIGONCNTL		TRIGOFFCNTL		
W															
Reset	0000_0000_0000_0000														
	16	31													
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
Offset	0xE_1014														

Figure 19-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 19-6 shows the field descriptions for the performance monitor local control B0 register (PMLCB0).

Table 19-6. 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 overflows, the current counter begins counting events. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger off select. The number of the counter that stops event counting. When the specified counter overflows, the current counter stops counting events. 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 19-6 shows the performance monitor local control registers B1–B8.

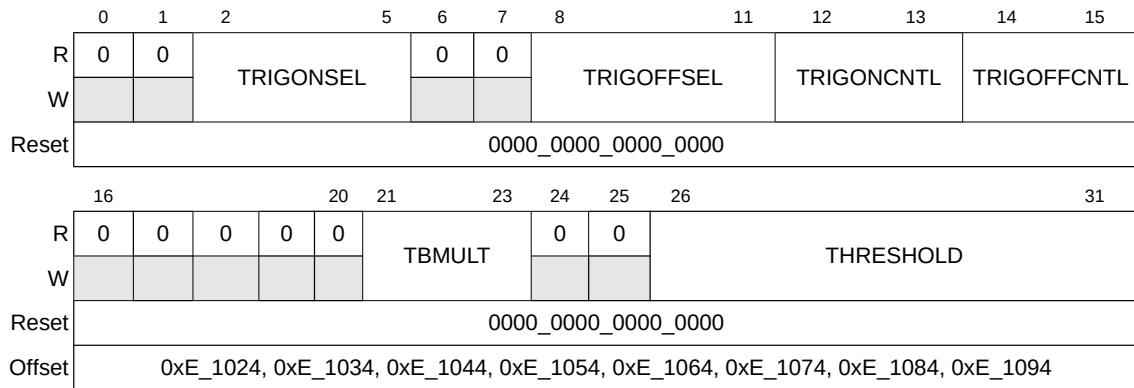


Figure 19-6. Performance Monitor Local Control Register B[1–8] (PMLCB[1–8])

Table 19-6 shows the field descriptions for the performance monitor local control B[1–8] registers (PMLCBn).

Table 19-7. PMLCB[1–8] 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 overflows, the current counter will begin counting. When TRIGONSEL = current counter, no triggering will occur due to another counter change
6–7	—	Reserved
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 overflows, the current counter will stop counting. When TRIGOFFSEL = current counter, no triggering occurs due to another counter change
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

Table 19-7. PMLCB[1–8] Field Descriptions (continued)

Bits	Name	Description
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 TBMULTIPLIER settings. The burst distance for burstiness counting is also scaled using the TBMULTIPLIER settings. For those events that scale the threshold: 000 Threshold field is multiplied by 1 001 Threshold field is multiplied by 2 010 Threshold field is multiplied by 4 011 Threshold field is multiplied by 8 100 Threshold field is multiplied by 16 101 Threshold field is multiplied by 32 110 Threshold field is multiplied by 64 111 Threshold field is multiplied by 128
24–25	—	Reserved
26–31	THRESHOLD	Threshold. Contains a threshold value, which is a value such that only events that exceed the value are counted. By varying the threshold value, software can obtain a profile of the characteristics of the events subject to the threshold. For example, if PMC2 counts cache misses for which the duration exceeds 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 value each time.

19.3.3 Counter Registers

This section describes the performance monitor counter registers in detail.

19.3.3.1 Performance Monitor Counters (PMC0, PMC1, PMC2, PMC3, PMC4, PMC5, PMC6, PMC7, PMC8)

There are nine performance monitor counters (PMC0, PMC1, PMC2, PMC3, PMC4, PMC5, PMC6, PMC7, and PMC8) that can be used to count events selected by the performance monitor local control registers. Performance monitor counter 0 (PMC0) is a 64-bit counter specifically designated to count only clock cycles. There are two 32-bit registers associated with PMC0. PMC0 extended represents the upper 32 bits of counter 0, and PMC0 represents the lower 32 bits.

Figure 19-7 and Figure 19-8 show the two registers which comprise the 64-bit performance monitor counter. Figure 19-7 shows the upper 32-bits of the counter and Figure 19-8 shows the lower 32-bits.

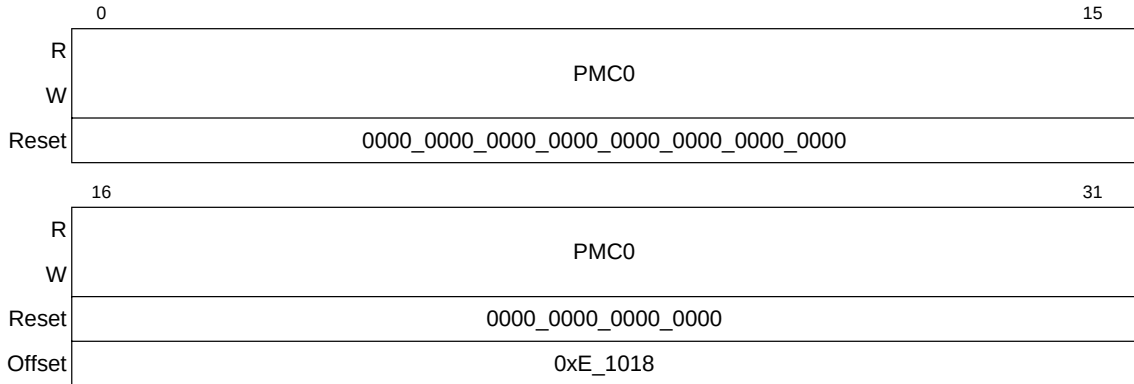


Figure 19-7. Performance Monitor Counter Register 0 Extended (PMC0) (Upper 32-bits)

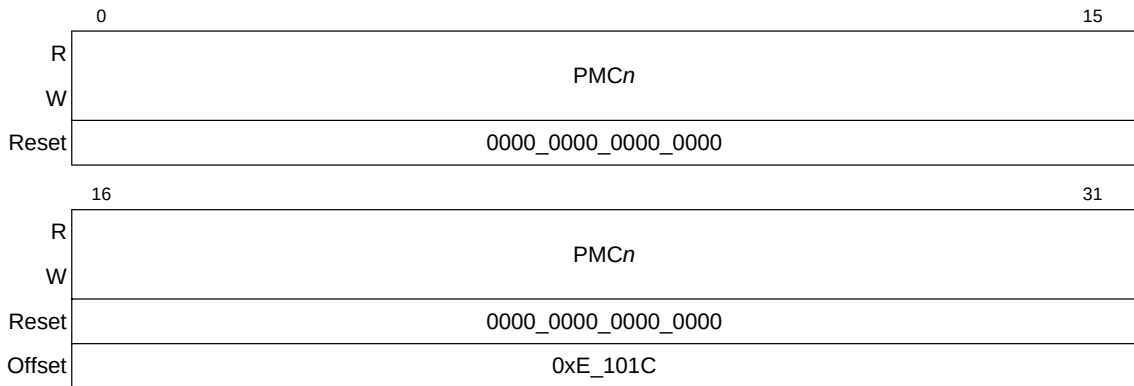


Figure 19-8. Performance Monitor Counter Register 0 (PMC0) (Lower 32-bits)

Table 19-8. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Counts only clock cycles

Performance monitor counters 1–8 (PMC1–PMC8) 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 19-9 shows the performance monitor counters (PMC[1-8]).

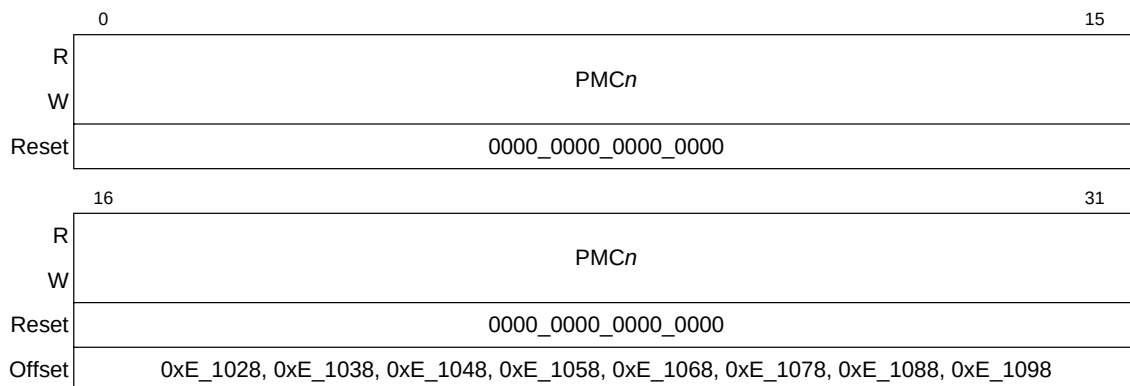


Figure 19-9. Performance Monitor Counter Register [1–8] (PMC[1–8])

Table 19-9 shows the field descriptions for the performance counters [1–8].

Table 19-9. PMC[1–8] Field Descriptions

Bits	Name	Description
0–31	PMCn	MSB indicates overflow. Setting the MSB can cause an immediate interrupt.

19.4 Functional Description

This section describes the use of some of the features of the performance monitor.

19.4.1 Performance Monitor Interrupt

The performance monitor counters can generate an interrupt on an overflow when the most significant bit of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit in the local control register (PMLCAN[CE]) and the performance monitor interrupt enable bit in the global control register (PMGC0[PMIE]) must be set. When an interrupt is signalled and the freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set in the global control register, the (PMGC0[FAC]) is set by hardware and all the registers are frozen. The interrupt condition can be cleared by software resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

19.4.2 Event Counting

Using the control registers described in Section 19.3.2, “Control Registers,” the nine performance monitor counters can count the occurrences of specific events. The 64-bit performance monitor counter 0 (PMC0) is designated to only count clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit

performance monitor counters (PMC1–PMC8). Additionally, 64 unique events can be counted on each of the 32-bit counters for a total of 576 possible events.

19.4.3 Threshold Events

The intent of threshold support is to allow 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. The threshold value is specified in the local control register (PMLCB n [THRESHOLD]).

For duration threshold event sequences, the performance monitor counter is only incremented when the duration of the event is equal to or greater than the threshold value. For this type of event the threshold value is scaled by a multiple also specified in the local control register (PMLCB n [TBMULT]).

A threshold event requires two signals. The first signal indicates when a threshold event sequence begins, and the second signal indicates when a threshold event sequence ends. An internal counter is used to determine when the threshold count has been exceeded and when the performance monitor counter can increment. This counter decrements during a threshold event sequence until it reaches the value of one. A new threshold event sequence cannot begin until the current threshold event sequence has completed. Additional threshold start signals are ignored during a threshold event sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle during a current event sequence, the stop terminates the current sequence, and the start signals the beginning of a new sequence. However, if both signals are asserted during the same cycle while not in a current event sequence, both signals are ignored. A timing diagram for duration threshold event counting is shown in Figure 19-10.

There are several illegal conditions for threshold counting which result in undefined behavior. An illegal condition exists if the threshold value obtained from the PMLCB n [THRESHOLD] and PMLCB n [TBMULT] fields is less than two. For these conditions the intent of threshold counting is ambiguous and thus results in an illegal condition.

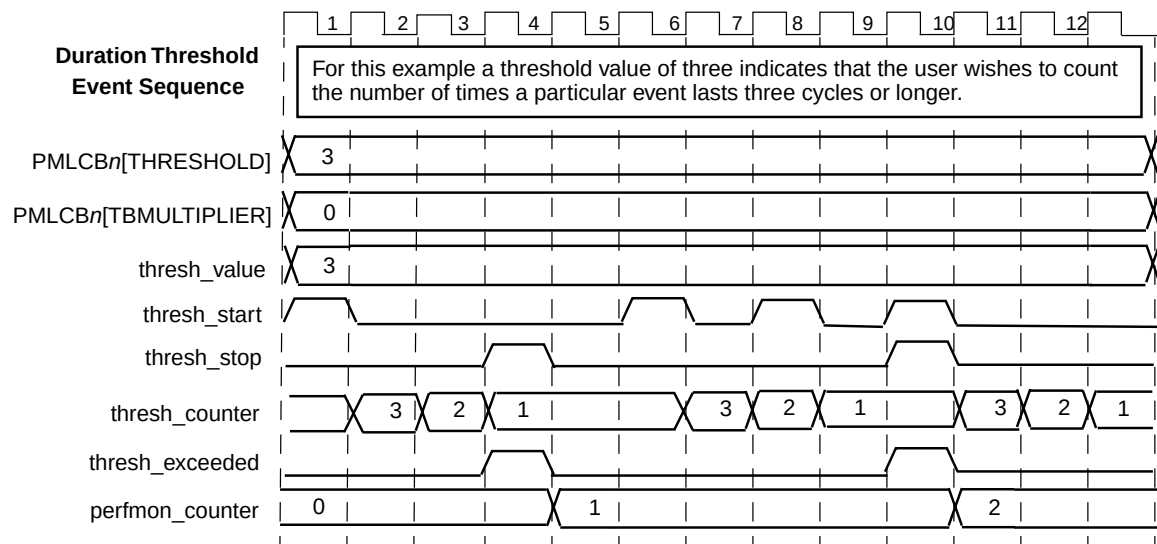


Figure 19-10. Duration Threshold Event Sequence Timing Diagram

19.4.4 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 the available chaining events, as shown in Table 19-11, 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, PMLCBn[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.

19.4.5 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 accomplished by appropriately setting the trigger on select (PMLCBn[TRIGONSEL]) or trigger off select (PMLCBn[TRIGOFFSEL]) bits in counter PMC1's local control register. Additionally, the condition that will trigger the counter must be selected by configuring the trigger on control (PMLCBn[TRIGONCNTL]) or trigger-off-control (PMLCBn[TRIGOFFCNTL]) bits in

that counter's local control register. 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 trigger 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 is not restarted by subsequent trigger on conditions. The state of the counter is preserved after the trigger off condition occurs.

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.

19.4.6 Burstiness Counting

The purpose of the burstiness counting feature is to facilitate characterizing events that occur in rapid succession followed by a relatively long pause. As shown in Table 19-10 event bursts are defined by three parameters; size, granularity, and distance. Figure 19-11 shows the same information graphically.

Table 19-10. Burst Definition

Parameter	Description	PMLCAn Field Name
size	the number of events constituting a burst	BSIZE
granularity	the time between individual events	BDIST
distance	the time between bursts	BGRAN

Burstiness counting can be performed for all events except threshold events.

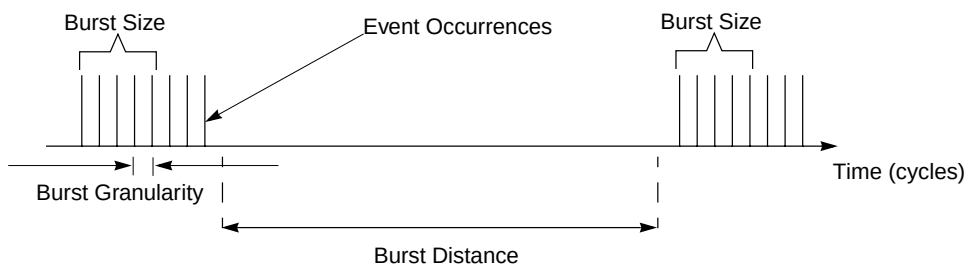


Figure 19-11. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCAn[BSIZE]) is used to set the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences is equal to or greater than the number specified in the burstiness size field. Furthermore, these event occurrences must be separated by no more cycles than what is specified in the burstiness granularity field (PMLCAn[BGRAN]). Also 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.

The burstiness granularity field (PMLCAn[BGRAN]) is used to specify the allowable number of cycles between individual event occurrences for it to be considered part of a burst sequence. If two event occurrences are separated by more cycles than what is specified by the burstiness granularity, then it cannot be considered part of the same burst sequence.

The burstiness distance field (PMLCAn[BDIST]) along with the threshold/burstiness multiplier field (PMLCBn[TBMULT]) are used to 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 performance monitor counter 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 19-12 illustrates 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 has expired.

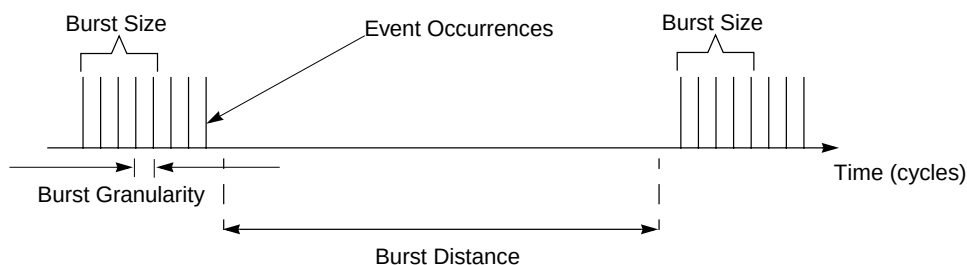


Figure 19-12. Description of Burstiness Counting

Three internal counters are used to track the different values required for burstiness counting.

- Burstiness size is monitored by a counter called `burst_size_ctr`. 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 an event occurs, the burst distance count equals zero, and it has not already reached zero.

- Burstiness granularity is monitored by a counter called `burst_gran_ctr` and is loaded with the specified value in the local control register on the rising edge of an event occurrence when the burst distance count is equal to zero. The `burst_gran_ctr` 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 called `burst_dist_ctr`. It is loaded with the product of `PMLCBn[BDIST]` and `PMLCBn[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 a value of zero.

A burst is counted at the end of a burst sequence when the `burst_size_ctr`, the `burst_gran_ctr` and the `burst_dist_ctr` are all equal to zero. A timing diagram for a burstiness counting example is shown in Figure 19-13.

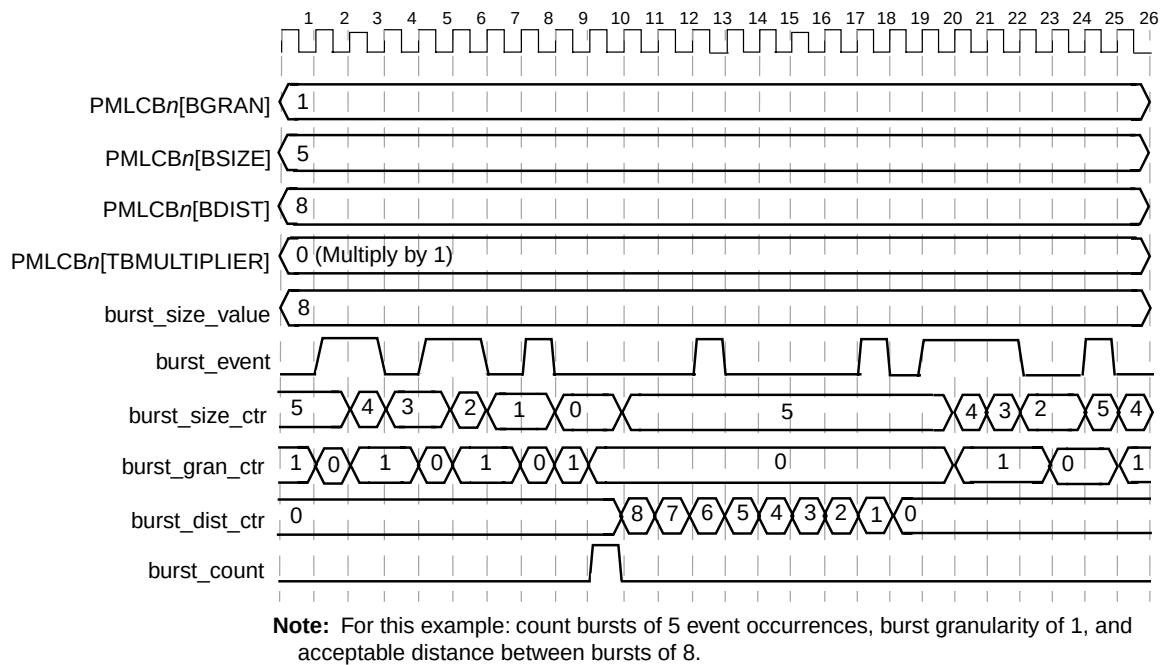


Figure 19-13. Burstiness Counting Timing Diagram

19.4.7 Performance Monitor Events

The events for counters PMC1 through PMC8 fall into two categories:

- Reference events, numbered 0–63
- Counter-specific events, numbered 0–63, but 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. To specify counter-specific event 0, for example, the event field must be

Functional Description

programmed to 64. Counter events that are not specified in Table 19-11 below are reserved.

In Table 19-11 the event assignment column indicates the type of event and the event number assigned to a given event. If the column uses the format Ref:#", then the event is considered a reference event and is shared across counters PMC1 through PMC8, with the specific event being indicated by the number following the word Ref. Ref:6, for example, means that reference event 6 is shared by counters PMC1–PMC8.

If the event is labeled in the format C[0–8]:#, then the event is considered a counter-specific event. The letter C followed by a number indicates the PMC counter to which the event is assigned: C2 for example specifies that the event is assigned to PMC2. The number after the colon specifies the event. Thus C8:62 means that PMC8 is assigned event 62, which in this case is PIC interrupt wait cycles. Remember that for counter-specific events, an offset of 64 must be used when programming the event in the event field of the local control registers.

Table 19-11. Performance Monitor Events

Event	Event Assignment	Description
General Events		
Nothing	Ref:0	Register counter holds current value
System Cycles	C0	Counts every platform cycle
DDR Memory Controller Events		
Total number of cycles a read is returning data from DRAM.	Ref:10	Asserts once for each beat of data that is returned to the memory controller on the DRAM interface
Total number of cycles a read or write is returning/sending data from/to DRAM.	Ref:11	Asserts once for each beat of data that is transferred from or to the DRAM.
Number of pipelined reads that missed in the Row Open Table.	C1:57	Asserts when a read that missed in the Row Open Table is issued while there is still a previous outstanding read.
Number of pipelined reads or writes that missed in the Row Open Table.	C2:0	Asserts when a read or write that missed in the Row Open Table is issued while there is still a previous outstanding read or write.
Number of non-pipelined reads that missed in the Row Open Table.	C3:60	Asserts when a read that missed in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that missed in the Row Open Table.	C4:0	Asserts when a read or write that missed in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of pipelined reads that hit in the Row Open Table.	C5:56	Asserts when a read that hit in the Row Open Table is issued while there is still a previous outstanding read.
Number of pipelined reads or writes that hit in the Row Open Table.	C6:0	Asserts when a read or write that hit in the Row Open Table is issued while there is still a previous outstanding read or write.

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of non-pipelined reads that hit in the Row Open Table.	C7:57	Asserts when a read that hit in the Row Open Table is issued while there are not any outstanding reads.
Number of non-pipelined reads or writes that hit in the Row Open Table.	C8:0	Asserts when a read or write that hit in the Row Open Table is issued while there are not any outstanding reads or writes.
Number of forced page closings not caused by a refresh.	C1:0	Asserts if a precharge has been issued to the DRAM for any reason other than a refresh. The three possible situations are: • A new transaction must be issued to an already active bank and sub-bank, which has a different row open; • A new transaction must be issued, but the Row Open Table is full and there is not a bank/sub-bank match between the current transaction and the Row Open Table; • The BSTOPRE interval has expired for an already open row.
Total number of Row Open Table misses.	C2:1	Asserts for each transaction that misses in the Row Open Table.
Total number of Row Open Table hits.	C3:0	Asserts for each transaction that hits in the Row Open Table.
Total number of force page closings.	C4:1	Asserts for all forced page closings including page closings due to refreshes.
Number of read-modify-write transactions due to ECC.	C5:0	If ECC is enabled and a transaction requires byte enables, a read-modify-write sequence is issued on the DRAM interface.
Number of forced page closings due to collision with bank and sub-bank.	Ref:12	Asserts if a new transaction must be issued to an already active bank and sub-bank which has a different row open.
Total number of reads or writes from core.	Ref:13	Total number of reads or writes from core.
Total number of reads or writes from TSEC 1	C3:1	Total number of reads or writes from TSEC 1
Total number of reads or writes from TSEC 2	C4:2	Total number of reads or writes from TSEC 2
Total number of reads or writes from Rapid IO	C3:2	Total number of reads or writes from Rapid IO
Total number of reads or writes from PCI	C4:3	Total number of reads or writes from PCI
Total number of reads or writes from DMA	C5:2	Total number of reads or writes from DMA
Total number of Row Open Table hits for reads or writes from core	Ref:14	Total number of Row Open Table hits for reads or writes from core
Total number of Row Open Table hits for reads or writes from TSEC 1	C6:1	Total number of Row Open Table hits for reads or writes from TSEC 1

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Total number of Row Open Table hits for reads or writes from TSEC 2	C7:0	Total number of Row Open Table hits for reads or writes from TSEC 2
Total number of Row Open Table hits for reads or writes from Rapid IO	C6:2	Total number of Row Open Table hits for reads or writes from Rapid IO
Total number of Row Open Table hits for reads or writes from PCI	C7:1	Total number of Row Open Table hits for reads or writes from PCI
Total number of Row Open Table hits for reads or writes from DMA	C8:2	Total number of Row Open Table hits for reads or writes from DMA
Number of times process detect measurement changed.	C5:3	Asserts if the measured tap point for the process detect circuit has drifted.
MEM TQ Events		
MEM TQ speculative reads	C1:36	Number of total speculative reads (committed or killed)
MEM TQ speculative reads committed	C3:43	Number of speculative reads committed
MEM TQ non-speculative reads	C2:45	Number of total non-speculative reads
MEM TQ killed writes	C4:44	Number of killed writes
MEM TQ read/write address collision	C5:5	Number of read/write address collisions
MEM TQ write flush timeout	C6:4	Number of write flushes caused by timeout counter
MEM TQ write flush FIFO full	C7:3	Number of write flushes caused by write FIFO full
MEM TQ read queue full	C3:3	Number of cycles that read queue is full
MEM TQ write queue full	C4:5	Number of cycles that write queue is full
MEM TQ read FIFO full	C5:4	Number of cycles that read FIFO is full
MEM TQ read FIFO full stall cycles	C6:3	Number of cycles stalled due to read FIFO full
MEM TQ write FIFO full	C7:2	Number of cycles that write FIFO is full
MEM TQ read data beats to MEM	C1:1	Number of data read beats to MEM
MEM TQ write data beats to MEM	C2:4	Number of data write beats to MEM
RapidIO Events		
Number of reads from OCeaN to RapidIO	C1:55	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read transaction is accepted by RapidIO.
Number of priority 0 reads from OCeaN to RapidIO	C2:9	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read transaction of priority 0 is accepted by RapidIO.
Number of priority 1 reads from OCeaN to RapidIO	C3:8	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read transaction of priority 1 is accepted by RapidIO.

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of priority 2 reads from OCeaN to RapidIO	C4:10	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read transaction of priority 2 is accepted by RapidIO.
Number of reads or writes from OCeaN to RapidIO	C5:54	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read or write transaction is accepted by RapidIO.
Number of priority 0 reads or writes from OCeaN to RapidIO	C6:59	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read or write transaction of priority 0 is accepted by RapidIO.
Number of priority 1 reads or writes from OCeaN to RapidIO	C7:55	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read or write transaction of priority 1 is accepted by RapidIO.
Number of priority 2 reads or writes from OCeaN to RapidIO	C8:53	Asserted when a RapidIO to OCeaN Gasket initiated RapidIO read or write transaction of priority 2 is accepted by RapidIO.
Number of reads from RapidIO to OCeaN	C8:54	Asserted when a RapidIO initiated read transaction is accepted by the RapidIO to OCeaN Gasket.
Number of priority 0 reads from RapidIO to OCeaN	C7:4	Asserted when a RapidIO initiated read transaction of priority 0 is accepted by the RapidIO to OCeaN Gasket.
Number of priority 1 reads from RapidIO to OCeaN	C6:5	Asserted when a RapidIO initiated read transaction of priority 1 is accepted by the RapidIO to OCeaN Gasket.
Number of priority 2 reads from RapidIO to OCeaN	C5:6	Asserted when a RapidIO initiated read transaction of priority 2 is accepted by the RapidIO to OCeaN Gasket.
Number of reads or writes from RapidIO to OCeaN	C4:59	Asserted when a RapidIO initiated read or write transaction is accepted by the RapidIO to OCeaN Gasket.
Number of priority 0 reads or writes from RapidIO to OCeaN	C3:9	Asserted when a RapidIO initiated read or write transaction of priority 0 is accepted by the RapidIO to OCeaN Gasket.
Number of priority 1 reads or writes from RapidIO to OCeaN	C2:10	Asserted when a RapidIO initiated read or write transaction of priority 1 is accepted by the RapidIO to OCeaN Gasket.
Number of priority 2 reads or writes from RapidIO to OCeaN	C1:6	Asserted when a RapidIO initiated read or write transaction of priority 2 is accepted by the RapidIO to OCeaN Gasket.
Number of 64-bit doubleword transfers from RapidIO to OCeaN Gasket to RapidIO	C7:56	Asserted for each "dval" across the gasket interface. Each "dval" validates 64-bits of data.
Number of misaligned transactions on RapidIO	C4:11	Asserted every time a new transaction gets loaded into the Misaligned engine.
Number of retried packets on RapidIO transmit due to resource limitations.	C2:62	Only includes packet that caused the retry, not the ignored packets that will need to be resent.
Number of flushed packets due to prior retries or error recovery on RapidIO transmit.	C3:10	This does not take into account the packet being ack retried.
Number of RapidIO Rx to RapidIO to OCeaN Gasket 64-bit or less doubleword transfers	C8:4	Asserted for each "dval" across the RapidIO Rx to gasket interface. Each "dval" validates 64-bits (double-word) or less of data.

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of retried or not accepted packets on RapidIO Rx	C5:7	Only includes packets received from a remote device that were either retried or not accepted, not the ignored packets that will need to be re-sent.
Number of ignored packets on RapidIO Rx	C6:6	Only includes packets received from a remote device that were ignored due to a prior retried or not accepted packet.
Number of RapidIO Rx maintenance training invocations	C7:5	RapidIO Rx invoked maintenance training.
RapidIO Rx to RapidIO to OCeaN Gasket generic interface request (start) RapidIO to OCeaN Gasket to RapidIO Rx generic interface ack or terminate (stop) (Duration Threshold)	Ref:32	Assert when a new request is active on the RapidIO Rx to gasket interface. Stop asserts when the new request has been acked or terminated on the RapidIO to OCeaN Gasket to RapidIO Rx generic interface.
Number of RapidIO Rx retries because the RapidIO Tx core response queue (CRQ) is unavailable	C1:7	RapidIO Rx retried a packet received from a remote device because the RapidIO Tx CRQ was unavailable for the packet's priority.
Number of RapidIO Rx retries because the inbound queue (IQ) is unavailable	C2:11	RapidIO Rx retried a packet received from a remote device because the IQ was unavailable for the packet's priority.
Number of RapidIO Rx retries because the doorbell unit is busy	C3:11	RapidIO Rx retried a packet received from a remote device because the doorbell unit was busy.
Number of RapidIO Rx retries because the message unit won arbitration	C4:12	RapidIO Rx retried a packet received from a remote device because the message unit won arbitration.
Number of RapidIO Rx retries because the packet was terminated by a restart-from-retry (RFR) (pro2)	C5:8	RapidIO Rx retried a packet received from a remote device because it was terminated by a RFR (determined in the pro2 pipeline stage).
Number of RapidIO Rx retries because the packet was terminated by a restart-from-retry (RFR) (log1)	C6:7	RapidIO Rx retried a packet received from a remote device because it was terminated by a RFR (determined in the log1 pipeline stage).
Number of RapidIO Rx retries because the packet was terminated by a link-request (LR) (pro2)	C7:6	RapidIO Rx retried a packet received from a remote device because it was terminated by a LR (determined in the pro2 pipeline stage).
Number of RapidIO Rx retries because the packet was terminated by a link-request (LR) (log1)	C8:5	RapidIO Rx retried a packet received from a remote device because it was terminated by a LR (determined in the log1 pipeline stage).
Number of RapidIO Rx retries because the packet was terminated by a stomp (S) (pro2)	C1:59	RapidIO Rx retried a packet received from a remote device because it was terminated by a S (determined in the pro2 pipeline stage).

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of RapidIO Rx retries because the packet was terminated by a stomp (S) (log1)	C2:63	RapidIO Rx retried a packet received from a remote device because it was terminated by a S (determined in the log1 pipeline stage).
Number of RapidIO Rx retries because the data queue (DQ) is full	C3:62	RapidIO Rx retried a packet received from a remote device because the DQ was full.
Number of RapidIO Rx retries because response retry from Tx (no tail)	C4:60	RapidIO Rx retried a response packet received from a remote device because Tx said so (response packet not yet fully received).
Number of RapidIO Rx retries because response retry from Tx (tail)	C5:57	RapidIO Rx retried a response packet received from a remote device because Tx said so (response packet fully received).
Number of RapidIO Rx retries because ack-out queue (AOQ) timeout	C6:61	RapidIO Rx retried a packet received from a remote device because the AOQ timeout counter expired. The timeout counter prevents a write packet from remaining as the arbitration winner forever in the inbound queue.
Number of RapidIO Rx retries because ack-out queue (AOQ) near full (pro2 and pro2o)	C7:58	RapidIO Rx retried a packet received from a remote device because the AOQ is nearly full (pro2 and pro2o pipeline stages valid). This guarantees room for future packets.
Number of RapidIO Rx retries because ack-out queue (AOQ) near full (pro2)	C8:55	RapidIO Rx retried a packet received from a remote device because the AOQ is nearly full (pro2 pipeline stage valid). This guarantees room for future packets.
Number of RapidIO Rx not accepted because s-bit error (pro2)	C1:8	RapidIO Rx did not accept a packet or control symbol from a remote device because it had a s-bit error (determined in the pro2 pipeline stage).
Number of RapidIO Rx not accepted because s-bit error (pro2o)	C2:12	RapidIO Rx did not accept a packet or control symbol from a remote device because it had a s-bit error (determined in the pro2o pipeline stage).
Number of RapidIO Rx not accepted because control symbol error (pro2)	C3:12	RapidIO Rx did not accept a control symbol from a remote device because it had a control symbol error (determined in the pro2 pipeline stage). The first 16 bits didn't match the inverse of the last 16 bits.
Number of RapidIO Rx not accepted because control symbol error (pro2o)	C4:13	RapidIO Rx did not accept a control symbol from a remote device because it had a control symbol error (determined in the pro2o pipeline stage). The first 16 bits didn't match the inverse of the last 16 bits.
Number of RapidIO Rx not accepted because frame error (pro2)	C5:9	RapidIO Rx did not accept a packet or control symbol from a remote device because it had a frame error (determined in the pro2 pipeline stage). Either the frame toggled on a negative edge or on a non-32-bit boundary.
Number of RapidIO Rx not accepted because frame error (pro2o)	C6:8	RapidIO Rx did not accept a packet or control symbol from a remote device because it had a frame error (determined in the pro2o pipeline stage). Either the frame toggled on a negative edge or on a non-32-bit boundary.

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of RapidIO Rx not accepted because no idle after maintenance training	C7:7	RapidIO Rx synthesized an ack not accepted control symbol in response to the remote device terminating maintenance training not with an idle control symbol.
Number of RapidIO Rx not accepted because embedded training pattern	C8:6	RapidIO Rx synthesized an ack not accepted control symbol in response to receiving a training pattern from the remote device which was embedded within a packet.
Number of RapidIO Rx not accepted because 256 byte data payload exceeded	C5:58	RapidIO Rx did not accept a packet from a remote device because its data payload exceeded 256 bytes.
Number of RapidIO Rx not accepted because port disabled	C8:56	RapidIO Rx did not accept a packet from a remote device because the Rx port was not enabled and the packet was not a format type 8 (maintenance) request.
Number of RapidIO Rx not accepted because CRC error (pro2)	C6:62	RapidIO Rx did not accept a packet from a remote device because it had a CRC error (determined in the pro2 pipeline stage). This will only assert when not in CRC bypass mode for non-pass-through responses.
Number of RapidIO Rx not accepted because CRC error (log1)	C7:59	RapidIO Rx did not accept a packet from a remote device because it had a CRC error (determined in the log1 pipeline stage). Only assert when not in CRC bypass mode for non-pass-through responses.
Number of RapidIO Rx not accepted because bad ackid	C8:57	RapidIO Rx did not accept a packet from a remote device because it had an unexpected ackid.
Number of RapidIO Rx to RapidIO to OCeAN Gasket request deassertions	C6:9	RapidIO Rx to RapidIO to OCeAN Gasket request deasserted due to either a higher priority request winning arbitration or the current request being terminated.
RapidIO Outbound Queue Full	C8:10	RIOOQ full for one of the priorities
Priority 3 transaction won RIOOQ local arbitration	C1:9	A priority 3 transaction has won local arbitration of RIOOQ.
Priority 2 transaction won RIOOQ local arbitration	C2:13	A priority 2 transaction won RIOOQ local arbitration
Priority 1 transaction won RIOOQ local arbitration	C3:13	A priority 1 transaction won RIOOQ local arbitration
Priority 0 transaction won RIOOQ local arbitration	C4:14	A priority 0 transaction won RIOOQ local arbitration
RIOOQ Priority 3 transaction won global arbitration	C5:10	A priority 3 transaction from RIOOQ has won global arbitration.
RIOOQ Priority 2 transaction won global arbitration	C6:10	A priority 2 transaction from RIOOQ has won global arbitration.
RIOOQ Priority 1 transaction won global arbitration	C7:8	A priority 1 transaction from RIOOQ has won global arbitration.
RIOOQ Priority 0 transaction won global arbitration	C8:7	A priority 0 transaction from RIOOQ has won global arbitration.

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
RapidIO Response Queue Full	C4:17	CRQ full for all priorities
Priority 3 transaction won CRQ local arbitration	C5:11	A priority 3 transaction has won local arbitration of CRQ.
Priority 2 transaction won CRQ local arbitration	C6:11	A priority 2 transaction has won local arbitration of CRQ.
Priority 1 transaction won CRQ local arbitration	C7:9	A priority 1 transaction has won local arbitration of CRQ.
Priority 0 transaction won CRQ local arbitration	C8:8	A priority 0 transaction has won local arbitration of CRQ.
CRQ Priority 3 transaction won global arbitration	C1:10	A priority 3 transaction has won global arbitration of CRQ.
CRQ Priority 2 transaction won global arbitration	C2:14	A priority 2 transaction has won global arbitration of CRQ.
CRQ Priority 1 transaction won global arbitration	C3:14	A priority 1 transaction has won global arbitration of CRQ.
CRQ Priority 0 transaction won global arbitration	C4:15	A priority 0 transaction has won global arbitration of CRQ.
Misaligned Engine Priority 2 occupied	C5:12	Misaligned engine for priority 2 transactions busy
MisAligned Engine Priority 2 local arbitration won event	C1:11	A priority 2 transaction from Misaligned engine has won local arbitration.
MisAligned Engine Priority 2 global arbitration won event	C4:16	A priority 2 transaction from Misaligned engine has won global arbitration.
MisAligned Engine priority 1 occupied	C6:12	Misaligned engine for priority 1 transactions busy
MisAligned Engine Priority 1 local arbitration won event	C2:15	A priority 1 transaction from Misaligned engine has won local arbitration.
MisAligned Engine Priority 1 global arbitration won event	C5:13	A priority 1 transaction from Misaligned engine has won global arbitration.
MisAligned Engine priority 0 occupied	C7:10	Misaligned engine for priority 0 transactions busy
MisAligned Engine Priority 0 local arbitration won event	C3:15	A priority 0 transaction from Misaligned engine has won local arbitration.
MisAligned Engine Priority 0 global arbitration won event	C8:9	A priority 0 transaction from Misaligned engine has won global arbitration.
Ack History Queue Full	C4:61	Outstanding ACK history queue is full with transactions.
PTS Full	C1:12	Packet header formatting queue full
RapidIO Outbound Data Queue Full	C2:16	Packet data formatting queue full

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
RapidIO Outbound Transmit FIFO Full	C3:16	Outbound asynchronous crossing FIFO full
ACK Transmit Queue Full	C5:14	Outbound ACK transmit FIFO full
Outbound RapidIO stopped for training event	C6:13	RapidIO Outbound port stopped for training. Event asserted for the cycles when packet transmission is stopped for training.
RapidIO Outbound retry -stopped event	C7:11	Outbound port is stopped due to an inbound ACK retry
Outbound RapidIO error-stopped event	C8:11	Outbound port is stopped due to an error recovery
RapidIO Outbound Software Link Validation stopped	C1:13	Outbound port is stopped for a software link validation
Message descriptor request event	C2:17	A message descriptor request from the outbound message unit is active in the system.
Message data request	C7:12	A message packet data request from the outbound message unit is active in the system.
Inbound Message packet protocol level retry	C4:18	An inbound message packet has encountered a protocol level retry
Inbound Doorbell packet protocol level retry	C6:14	An inbound doorbell packet has encountered a protocol level retry
RapidIO Outbound ACK Timeout / Out-of-order event	C8:12	An ACK timeout or an out of sequence ACK has been detected by the ACK history queue.
RapidIO to OCeaN Gasket Events		
Inbound Static Queue (ISQ) Full	C1:14	Inbound Static Queue (ISQ) contains 7 entries
Outbound Static Queue (OSQ) Full	C2:18	Outbound Static Queue (OSQ) contains 7 entries
Inbound Buffer Full	C3:17	Inbound SRAM Buffer contains 8 valid rows
Inbound Response Buffer Full	C4:19	Inbound Response SRAM Buffer contains 8 valid rows
Outbound Buffer Full	C5:15	Outbound SRAM Buffer contains 16 valid rows
Outbound Read waiting for Inbound Response Buffer Space	C6:15	An outbound read cannot be issued to RapidIO Tx because inbound buffer space for its response is not available.
DMA Events		
Channel 0 read request	C1:2	DMA channel 0 read request active in the system
Channel 1 read request	C2:5	DMA channel 1 read request active in the system
Channel 2 read request	C3:4	DMA channel 2 read request active in the system
Channel 3 read request	C4:6	DMA channel 3 read request active in the system
Channel 0 write request	C1:3	DMA channel 0 write request active in the system
Channel 1 write request	C2:6	DMA channel 1 write request active in the system

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Channel 2 write request	C3:5	DMA channel 2 write request active in the system
Channel 3 write request	C4:7	DMA channel 3 write request active in the system
Channel 0 descriptor request	C5:41	DMA channel 0 descriptor request active in the system
Channel 1 descriptor request	C6:44	DMA channel 1 descriptor request active in the system
Channel 2 descriptor request	C7:41	DMA channel 2 descriptor request active in the system
Channel 3 descriptor request	C8:41	DMA channel 3 descriptor request active in the system
Channel 0 read DW or less	C1:4 and C5:53	DMA channel 0 read doubleword valid
Channel 1 read DW or less	C2:7 and C6:58	DMA channel 1 read doubleword valid
Channel 2 read DW or less	C3:6 and C7:54	DMA channel 2 read doubleword valid
Channel 3 read DW or less	C4:8 and C8:52	DMA channel 3 read doubleword valid
Channel 0 write DW or less	C1:5	DMA channel 0 write doubleword valid
Channel 1 write DW or less	C2:8	DMA channel 1 write doubleword valid
Channel 2 write DW or less	C3:7	DMA channel 2 write doubleword valid
Channel 3 write DW or less	C4:9	DMA channel 3 write doubleword valid
ECM Events		
ECM request wait core	C8:13	Asserted for every cycle Core request occurs
ECM request wait TSEC1	C5:16	Asserted for every cycle TSEC1 request occurs
ECM request wait TSEC2	C6:16	Asserted for every cycle TSEC2 request occurs
ECM request wait RapidIO	C3:18	Asserted for every cycle RapidIO request occurs
ECM request wait PCI	C4:20	Asserted for every cycle PCI request occurs
ECM request wait DMA	C2:19	Asserted for every cycle DMA request occurs
ECM request wait other	C1:15	Asserted for every cycle "other" 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	C1:16	ECM dispatch from core (includes address only's)
ECM dispatch from TSEC1	C3:19	ECM dispatch from TSEC1
ECM dispatch from TSEC2	C4:21	ECM dispatch from TSEC2
ECM dispatch from RapidIO	C5:17	ECM dispatch from RapidIO
ECM dispatch from PCI	C6:17	ECM dispatch from PCI
ECM dispatch from DMA	C7:14	ECM dispatch from DMA
ECM dispatch from other	C8:14	ECM dispatch from other

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
ECM dispatch to DDR	C4:22	ECM dispatch to DDR
ECM dispatch to L2/SRAM	C5:18	ECM dispatch to L2/SRAM
ECM dispatch to LBIU	C6:18	ECM dispatch to LBIU
ECM dispatch to RapidIO	C7:15	ECM dispatch to RapidIO
ECM dispatch to PCI	C8:15	ECM dispatch to PCI
ECM dispatch snoopable	C3:20	ECM dispatch snoopable
ECM dispatch write	C1:17	ECM dispatch write
ECM dispatch write allocate	C2:21	ECM dispatch write allocate
ECM dispatch write allocate lock	C3:21	ECM dispatch write allocate lock
ECM dispatch read	C4:23	ECM dispatch read
ECM dispatch read unlock	C5:19	ECM dispatch read unlock
ECM dispatch read clear atomic	C6:19	ECM dispatch read clear atomic
ECM dispatch read set atomic	C7:16	ECM dispatch read set atomic
ECM dispatch read decrement atomic	C8:16	ECM dispatch read decrement atomic
ECM dispatch read increment atomic	C7:17	ECM dispatch read increment atomic
ECM Data Bus grant DDR	C1:18	ECM Data Bus grant DDR
ECM Data Bus grant LBIU	C2:22	ECM Data Bus grant LBIU
ECM Data Bus grant ECM to OCeaN Gasket	C3:22	ECM Data Bus grant ECM to OCeaN Gasket
ECM Data Bus grant SkyBlue	C4:24	ECM Data Bus grant SkyBlue
ECM Data Bus grant PIC	C1:19	ECM Data Bus grant PIC
ECM Data Bus grant TSEC1	C3:23	ECM Data Bus grant TSEC1
ECM Data Bus grant TSEC2	C4:25	ECM Data Bus grant TSEC2
ECM Data Bus wait DDR	C5:20	ECM Data Bus DDR requesting
ECM Data Bus wait LBIU	C6:20	ECM Data Bus LBIU requesting
ECM Data Bus wait ECM to OCeaN Gasket	C7:18	ECM Data Bus ECM to OCeaN Gasket requesting
ECM Data Bus wait SkyBlue	C8:17	ECM Data Bus SkyBlue requesting
ECM Data Bus wait PIC	C5:21	ECM Data Bus PIC requesting
ECM Data Bus wait TSEC1	C7:19	ECM Data Bus TSEC1 requesting
ECM Data Bus wait TSEC2	C8:18	ECM Data Bus TSEC2 requesting
ECM Global Data Bus beat	Ref:16	ECM Global Data Bus beat

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
ECM DRB beat	Ref:17	ECM Direct Read Bus beat
ECM DRB beat forwarded	C2:24	ECM DRB beat forwarded directly to Elf R1 Data Bus
ECM cancel	Ref:18	ECM cancel
Interrupt Controller Events		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt wait cycles	C8:62	Number of cycles there was any interrupt waiting to be acknowledge
PIC interrupt select 0 (Duration Threshold)	C1:56	THRESHOLD: select 0: interrupt count over threshold (Please note that a request is acknowledged only if it is not masked, and it's priority is greater than 0)
PIC interrupt select 1 (Duration Threshold)	C3:59	THRESHOLD: select 1: interrupt count over threshold
PIC interrupt select 2 (Duration Threshold)	C5:55	THRESHOLD: select 2: interrupt count over threshold
PIC interrupt select 3 (Duration Threshold)	C6:60	THRESHOLD: select 3: interrupt count over threshold
OCeaN to ECM to OCeaN Gasket Events		
Number of OCeaN inbound initiated transactions	C1:61	Number of inbound initiated transactions
Number of OCeaN inbound data beats	C3:33	Number of inbound data beats
Number of OCeaN outbound initiated transactions	C4:63	Number of outbound initiated transactions
Number of OCeaN outbound data beats	C6:30	Number of outbound data beats
Number of successful dispatches to ECM to OCeaN Gasket from ECM	C5:62	Number of successful dispatches to ECM to OCeaN Gasket from ECM
Number of ECM data beats to ECM to OCeaN Gasket	C6:31	Number of ECM data beats to ECM to OCeaN Gasket
Number of ECM to OCeaN Gasket initiated successful ECM request bus transactions	C7:63	Number of ECM to OCeaN Gasket initiated successful ECM request bus transactions
Num of Inbound nwrite trans	C2:34	Num of Inbound nwrite trans
Num of inbound nread_r trans	C3:34	Num of inbound nread_r trans
Num of inbound ECM read transactions	C8:61	Num of inbound ECM read transactions.
Number of outbound ECM reads	C6:63	Number of outbound ECM reads
PCI/PCI-X Common Events		

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
PCI/X clock cycles	Ref:28	PCI/X clock cycles
PCI/X inbound memory reads	C1:62	Number of inbound memory reads (all read types)
PCI/X inbound memory writes	C2:37	Number of inbound memory writes
PCI/X inbound config reads	C3:63	Number of inbound config reads
PCI/X inbound config writes	C4:37	Number of inbound config writes
PCI/X outbound memory reads	C5:30	Number of outbound memory reads (all read types)
PCI/X outbound memory writes	C6:32	Number of outbound memory writes
PCI/X outbound i/o reads	C3:37	Number of outbound i/o reads
PCI/X outbound i/o writes	C4:38	Number of outbound i/o writes
PCI/X outbound config reads	C7:26	Number of outbound config reads
PCI/X outbound config writes	C8:26	Number of outbound config writes
PCI/X inbound total read data beats	C5:32	Number of total (32-bit and 64-bit) inbound read data beats
PCI/X inbound total write data beats	C6:34	Number of total (32-bit and 64-bit) inbound write data beats
PCI/X outbound total read data beats	C7:28	Number of total (32-bit and 64-bit) outbound read data beats
PCI/X outbound total write data beats	C8:28	Number of total (32-bit and 64-bit) outbound write data beats
PCI/X inbound 32-bit read data beats	C1:30	Number of 32-bit inbound read data beats
PCI/X inbound 32-bit write data beats	C2:38	Number of 32-bit inbound write data beats
PCI/X outbound 32-bit read data beats	C3:38	Number of 32-bit outbound read data beats
PCI/X outbound 32-bit write data beats	C4:39	Number of 32-bit outbound write data beats
PCI/X inbound 64-bit read data beats	C5:31	Number of 64-bit inbound read data beats
PCI/X inbound 64-bit write data beats	C6:33	Number of 64-bit inbound write data beats
PCI/X outbound 64-bit read data beats	C7:27	Number of 64-bit outbound read data beats
PCI/X outbound 64-bit write data beats	C8:27	Number of 64-bit outbound write data beats
PCI/X total transactions	C7:29	Number of total (32-bit and 64-bit) transactions
PCI/X 64-bit transactions	C8:29	Number of 64-bit transactions

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
PCI/X inbound purgeable reads	C2:2	Number of inbound purgeable reads (speculative reads)
PCI/X inbound purgeable reads discarded	C8:63	Number of inbound purgeable reads (speculative reads) discarded
PCI/X idle cycles	C1:31	Number of PCI/X idle cycles
PCI/X dual address cycles	C2:40	Number of PCI/X dual address cycles
PCI/X internal cycles	C3:39	Number of PCI/X cycles that the internal port is using the bus
PCI/X device 0 cycles	C4:40	Number of PCI/X cycles that device 0 is using the bus
PCI/X device 1 cycles	C5:33	Number of PCI/X cycles that device 1 is using the bus
PCI/X device 2 cycles	C6:35	Number of PCI/X cycles that device 2 is using the bus
PCI/X device 3 cycles	C7:30	Number of PCI/X cycles that device 3 is using the bus
PCI/X device 4 cycles	C8:30	Number of PCI/X cycles that device 4 is using the bus
PCI inbound memory read PCI-X inbound memory dword read	C1:34	Number of PCI inbound memory read number of PCI-X inbound memory dword read
PCI inbound memory readline PCI-X inbound memory alias read	C2:44	Number of PCI inbound memory readline number of PCI-X inbound memory alias read
PCI inbound memory readmultiple PCI-X inbound memory block read	C3:42	Number of PCI inbound memory readmultiple number of PCI-X inbound memory block read
PCI outbound memory read PCI-X outbound memory dword read	C4:43	Number of PCI outbound memory read number of PCI-X outbound memory dword read
PCI outbound memory readline PCI-X outbound memory block read	C5:36	Number of PCI outbound memory readline number of PCI-X outbound memory block read
PCI wait PCI-X initial wait	C1:35	Number of PCI wait cycles number of PCI-X initial wait cycles (IRDY_ TRDY_ not both asserted)
PCI Specific Events		
PCI IRDY_	C6:36	Number of PCI cycles IRDY_ is asserted
PCI TRDY_	C7:31	Number of PCI cycles TRDY_ is asserted
PCI FRAME_	C8:31	Number of PCI cycles FRAME_ is asserted
PCI-X Specific Events		
PCI-X 1 split transaction	C2:41	Number of PCI-X cycles there are 1 or more outstanding split transactions
PCI-X 2 split transactions	C3:40	Number of PCI-X cycles there are 2 or more outstanding split transactions
PCI-X 3 split transactions	C4:41	Number of PCI-X cycles there are 3 or more outstanding split transactions

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
PCI-X 4 split transactions	C5:34	Number of PCI-X cycles there are 4 or more outstanding split transactions
PCI-X split responses	C6:37	Number of split responses received for an outbound transactions
PCI-X ADB_ disconnects	C7:32	Number of ADB_ disconnects
PCI-X Gasket Specific Events		
PCI/X snoopable	C1:32	Number of inbound read or write snoopable requests
PCI/X write stash	C2:42	Number of inbound write stash requests
PCI/X write stash with lock	C3:41	Number of inbound write stash-with-lock requests
PCI/X read unlock	C4:42	Number of inbound read unlock requests
PCI/X byte enable transactions	C1:33	Number of PCI/X transactions that have byte enables
PCI/X non-byte enable transactions	C2:43	Number of PCI/X transactions that do not have byte enables
Ethernet Events		
TSEC1 Address Data Filtering (ADF) Events		
Accepted frames	Ref:36	Number of accepted frames
Individual Hash table accepted frame	C7:35	Number of accepted frames by a Individual Hash table hit
Group Hash table accepted frame	C8:35	Number of accepted frames by a Group Hash table hit
Rejected frames	Ref:39	Number of rejected frames
Dropped frames	Ref:38	Number of dropped frames that could have been accepted (data overflow, status overflow and lack of buffer descriptors)
Dropped frames due to data overflow	C1:41	Number of frames dropped because of overflow in the FIFO and 256 byte buffer
Dropped frames due to status overflow	C2:50	Number of frames dropped because of inability to write status for one frame and a new frame starts
TSEC1 FIFO Events		
Receive FIFO data valid	C1:45	Number of cycles receive FIFO has data
Transmit frames without threshold	C7:44	Number of transmit frames that don't hit TX FIFO threshold
Receive FIFO above 1/4	Ref:47	Number of cycles RX FIFO is above 1/4
Receive FIFO above 1/2	Ref:48	Number of cycles RX FIFO is above 1/2
Receive FIFO above 3/4	Ref:49	Number of cycles RX FIFO is above 3/4
TSEC1 DMA Events		
DMA reads	C1:47	Number of reads (descriptor and data)
Buffer descriptor reads	C2:54	Number of transmit and receive buffer descriptor reads

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
RX Buffer descriptor reads	C3:51	Number of receive buffer descriptor reads (transmit number can be calculated by subtracting this number from total)
DMA Writes	C4:52	Number of writes (descriptor and data)
Buffer descriptor writes	C5:46	Number of transmit and receive buffer descriptors closed
RX Buffer descriptor writes	C6:49	Number of receive buffer descriptors closed (transmit number can be calculated by subtracting this number from total)
RX Buffer descriptor read latency (Duration Threshold)	Ref:41	Number of times latency of receive buffer descriptor read exceeds programmed threshold (threshold event) start: request asserted stop: data ack received
TX Buffer descriptor read latency (Duration Threshold)	Ref:42	Number of times latency of transmit buffer descriptor read exceeds programmed threshold (threshold event). start: request asserted stop: data ack received
RX Buffer descriptor write latency (Duration Threshold)	Ref:43	Number of times latency of getting a response on a receive Buffer descriptor write exceeds programmed threshold (threshold event). Only for writes which require a response. start: request asserted stop: End of Transaction received
TX Buffer descriptor write latency (Duration Threshold)	Ref:44	Number of times latency of getting a response on a transmit Buffer descriptor write exceeds programmed threshold (threshold event). Only for writes which require a response. start: request asserted stop: End of Transaction received
TX data read latency (Duration Threshold)	Ref:45	Number of times latency of getting a response on a data read exceeds programmed threshold (threshold event). start: request asserted stop: Data ackreceived
Data beats	C7:48	Number of data beats
Read data beats	C8:46	Number of read data beats
64-byte buffer empty	C6:51	Number of cycles 64 byte receive buffer is empty while receiving a frame.
No Ack from System	C8:48	Number of cycles DMA is requesting and is not receiving an ack.
RX frame interrupts	C1:49	Number of times a receive frame interrupt is set. Event occurs when the Buffer descriptor receive interrupt is set on a last buffer descriptor.
TX frame interrupts	C3:53	Number of times a transmit frame interrupt is set. Event occurs when the Buffer descriptor transmit interrupt is set on a buffer descriptor with L = 1.
RX frame processing (Duration Threshold)	Ref:46	Number of times the processing of a receive frame exceeds programmed threshold (threshold event). start: Open first buffer descriptor stop: Close last buffer descriptor

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
TSEC2 Address Data Filtering (ADF) Events		
Accepted frames	Ref:50	Number of accepted frames
Individual Hash table accepted frame	C7:36	Number of accepted frames by a Individual Hash table hit
Group Hash table accepted frame	C8:36	Number of accepted frames by a Group Hash table hit
Rejected frames	Ref:53	Number of rejected frames
Dropped frames	Ref:52	Number of dropped frames that could have been accepted (data overflow, status overflow and lack of buffer descriptors)
Dropped frames due to data overflow	C1:42	Number of frames dropped because of overflow in the FIFO and 256 byte buffer
Dropped frames due to status overflow	C2:51	Number of frames dropped because of inability to write status for one frame and a new frame starts
TSEC2 FIFO Events		
Receive FIFO data valid	C1:46	Number of cycles receive FIFO has data
Transmit frames without threshold	C7:45	Number of transmit frames that don't hit TX FIFO threshold
Receive FIFO above 1/4	Ref:61	Number of cycles RX FIFO is above 1/4
Receive FIFO above 1/2	Ref:62	Number of cycles RX FIFO is above 1/2
Receive FIFO above 3/4	Ref:63	Number of cycles RX FIFO is above 3/4
TSEC2 DMA Events		
DMA reads	C1:48	Number of reads (descriptor and data)
Buffer descriptor reads	C2:55	Number of transmit and receive buffer descriptor reads
RX Buffer descriptor reads	C3:52	Number of receive buffer descriptor reads (transmit number can be calculated by subtracting this number from total)
DMA Writes	C4:53	Number of writes (descriptor and data)
Buffer descriptor writes	C5:47	Number of transmit and receive buffer descriptors closed
RX Buffer descriptor writes	C6:50	Number of receive buffer descriptors closed (transmit number can be calculated by subtracting this number from total)
RX Buffer descriptor read latency (Duration Threshold)	Ref:55	Number of times latency of receive buffer descriptor read exceeds programmed threshold (threshold event) start: request asserted stop: data ack received
TX Buffer descriptor read latency (Duration Threshold)	Ref:56	Number of times latency of transmit buffer descriptor read exceeds programmed threshold (threshold event). start: request asserted stop: data ack received

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
RX Buffer descriptor write latency (Duration Threshold)	Ref:57	Number of times latency of getting a response on a receive Buffer descriptor write exceeds programmed threshold (threshold event). Only for writes which require a response. start: request asserted stop: End of Transaction received
TX Buffer descriptor write latency (Duration Threshold)	Ref:58	Number of times latency of getting a response on a transmit Buffer descriptor write exceeds programmed threshold (threshold event). Only for writes which require a response. start: request asserted stop: End of Transaction received
TX data read latency (Duration Threshold)	Ref:59	Number of times latency of getting a response on a data read exceeds programmed threshold (threshold event). start: request asserted stop: Data ack received
Data beats	C7:49	Number of data beats
Read data beats	C8:47	Number of read data beats
64-byte buffer empty	C6:52	Number of cycles 64 byte receive buffer is empty while receiving a frame.
No ack from System	C8:49	Number of cycles DMA is requesting and is not receiving an ack.
RX frame interrupts	C1:50	Number of times a receive frame interrupt is set. Event occurs when the Buffer descriptor receive interrupt is set on a last buffer descriptor.
TX frame interrupts	C3:54	Number of times a transmit frame interrupt is set. Event occurs when the Buffer descriptor transmit interrupt is set on a buffer descriptor with L = 1.
RX frame processing	Ref:60	Number of times the processing of a receive frame exceeds programmed threshold (threshold event). start: Open first buffer descriptor stop: Close last buffer descriptor
Local Bus Events		
Number of accesses hitting bank (chip-select) 1	C1:51	Number of accesses hitting bank (chip-select) 1
Number of accesses hitting bank (chip-select) 2	C2:56	Number of accesses hitting bank (chip-select) 2
Number of accesses hitting bank (chip-select) 3	C3:55	Number of accesses hitting bank (chip-select) 3
Number of accesses hitting bank (chip-select) 4	C4:54	Number of accesses hitting bank (chip-select) 4
Number of accesses hitting bank (chip-select) 5	C5:48	Number of accesses hitting bank (chip-select) 5
Number of accesses hitting bank (chip-select) 6	C6:53	Number of accesses hitting bank (chip-select) 6

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Number of accesses hitting bank (chip-select) 7	C7:50	Number of accesses hitting bank (chip-select) 7
Number of accesses hitting bank (chip-select) 8	C8:50	Number of accesses hitting bank (chip-select) 8
Number of requests granted to ECM port	C2:57	Number of requests granted to ECM port
Number of cycles atomic lock for ECM port is enabled	C4:55	Number of cycles atomic lock for ECM port is enabled
Number of atomic lock time-outs for ECM port	C6:54	Number of atomic lock time-outs for ECM port
Number of cycles a read is taking in GPCM	C1:53	Number of cycles a read is taking in GPCM
Number of cycles a read is taking in UPM	C2:58	Number of cycles a read is taking in UPM
Number of cycles a read is taking in SDRAM	C3:57	Number of cycles a read is taking in SDRAM
Number of cycles a write is taking in GPCM	C4:56	Number of cycles a write is taking in GPCM
Number of cycles a write is taking in UPM	C5:50	Number of cycles a write is taking in UPM
Number of cycles a write is taking in SDRAM	C6:55	Number of cycles a write is taking in SDRAM
Number of SDRAM bank misses	C7:51	Number of SDRAM bank misses
Number of SDRAM page misses	C8:51	Number of SDRAM page misses
L2 Events		
Core instruction accesses to L2 that hit	Ref:22	Core instruction accesses to L2 that hit
Core instruction accesses to L2 that miss	C2:59	Core instruction accesses to L2 that miss
Core data accesses to L2 that hit	Ref:23	Core data accesses to L2 that hit
Core data accesses to L2 that miss	C4:57	Core data accesses to L2 that miss
Non-core burst write to L2 (cache external write or SRAM)	C5:51	Non-core burst write to L2 (cache external write or SRAM)
Non-core non-burst write to L2	C6:56	Non-core non-burst write to L2
Non-core write misses cache external write window and SRAM memory range	C7:52	Non-core write misses cache external write window and SRAM memory range
Non-core read hit in L2	Ref:24	Non-core read hit in L2

Table 19-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
Non-core read miss in L2	C1:54	Non-core read miss in L2
L2 allocates, from any source	Ref:25	L2 allocates, from any source
L2 retries due to full write queue	C2:60	L2 retries due to full write queue
L2 retries due to address collision	C3:58	L2 retries due to address collision
L2 failed lock attempts due to full set	C4:58	L2 failed lock attempts due to full set
L2 victimizations of valid lines	C5:52	L2 victimizations of valid lines
L2 invalidations of lines	C6:57	L2 invalidations of lines
L2 clearing of locks	C7:53	L2 clearing of locks
Debug Events		
Ext. Event	C3:61	Number of cycles trig_in pin is asserted
Number of watchpoint monitor hits	C2:61	Number of watchpoint monitor hits
Number of trace buffer hits	C1:58	Number of trace buffer hits
DUART Events		
UART0 Baud Rate	C1:63	uart0 real baud rate
UART1 Baud Rate	C5:63	uart1 real baud rate
Chaining Events		
PMC0 carry-out	Ref:1	Counts the number of times PMC0[0] transitions from 1 to 0.
PMC1 carry-out	Ref:2	Counts the number of times PMC1[0] transitions from 1 to 0. This event is reserved for PMC1.
PMC2 carry-out	Ref:3	Counts the number of times PMC2[0] transitions from 1 to 0. This event is reserved for PMC2.
PMC3 carry-out	Ref:4	Counts the number of times PMC3[0] transitions from 1 to 0. This event is reserved for PMC3.
PMC4 carry-out	Ref:5	Counts the number of times PMC4[0] transitions from 1 to 0. This event is reserved for PMC4.
PMC5 carry-out	Ref:6	Counts the number of times PMC5[0] transitions from 1 to 0. This event is reserved for PMC5.
PMC6 carry-out	Ref:7	Counts the number of times PMC6[0] transitions from 1 to 0. This event is reserved for PMC6.
PMC7 carry-out	Ref:8	Counts the number of times PMC7[0] transitions from 1 to 0. This event is reserved for PMC7.
PMC8 carry-out	Ref:9	Counts the number of times PMC8[0] transitions from 1 to 0. This event is reserved for PMC8.

19.4.8 Performance Monitor Examples

This section 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

Table 19-12 lists the register settings for several counting examples. Settings are listed for regular counting, triggering, threshold counting, and burstiness counting. For all four examples the global control 0 register settings are identical. PMGC0[FAC] is cleared because all four counting examples require that the counters not be frozen. PMGC0[PMIE] and PMGC0[FCECE] are set to one indicating that performance monitor interrupts are enabled and that the counters should be frozen when an interrupt is signalled. Similarly, for all four examples the FC and CE fields in the local control A register (PMLCA_n) are cleared and set, respectively, since the counters cannot be frozen for counting, and overflow condition enable is required to allow interrupt signalling. For regular event counting a non-threshold event is selected in the PMLCA_n[EVENT] field, and all other features are disabled by clearing all the register fields except for CE.

For the triggering example any event can be selected in the PMLCA_n[EVENT] field. All other features are disabled by setting these register fields to zero except for CE to allow interrupt signalling. Values of three and five in the PMLCB_n[TRIGONSEL] and PMLCB_n[TRIGOFFSEL] fields indicate that the counter begins and ends counting based on the conditions in counters three and five. Furthermore, a value of one in the PMLCB_n[TRIGONCNTL] field indicates that the counter begins counting when PMC3 changes value. According to the setting in the PMLCB_n[TRIGOFFCNTL] field the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, the PMLCA_n[CE] field 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 the PMLCA_n[EVENT] field. For this example, the threshold value for the duration threshold event is scaled by one since the PMLCB_n[TBMULT] field is cleared. For the threshold counting example all other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values must be specified in the PMLCA_n[BSIZE], PMLCA_n[BGRAN], PMLCA_n[BDIST], and PMLCB_n[TBMULT] fields of the local control registers.

Table 19-12. Register Settings for Counting Examples MPC8540

Register	Register Field	Regular Counting	Triggering	Threshold	Burstiness
Global Control 0 Register	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1
Local Control A Registers	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
Local Control B Registers	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

Chapter 20

Debug Features and Watchpoint Facility

20.1 Introduction

This document describes all customer visible debug modes of the MPC8540 integrated device. The debug features on the MPC8540 cover the major interfaces such as the local bus interface, the DDR SDRAM interface, and the PCI interface. In addition to the external interfaces, the MPC8540 provides triggering capabilities based on user programmable events and internal visibility to the processor core interface, and the RapidIO interface via the watchpoint and trace buffer features.

This chapter describes the operation of the following interfaces:

- PCI interface debug (Section 20.4.1, “PCI/PCI-X Interface Debug”)
- DDR SDRAM interface debug (Section 20.4.2, “DDR SDRAM Interface Debug”)
- Local bus interface debug (Section 20.4.3, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 20.4.4, “Watchpoint Monitor” and Section 20.4.5, “Trace Buffer”)

Figure 20-1 is a block diagram that shows the details pertinent to the customer-visible debug features.

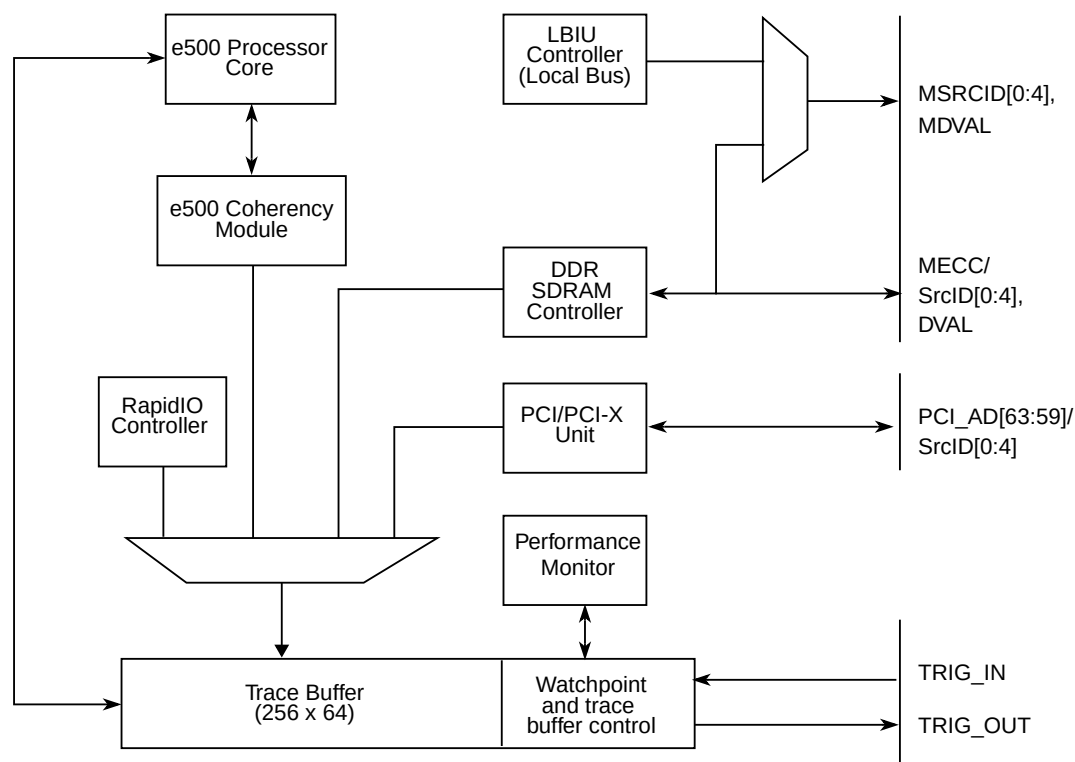


Figure 20-1. Debug and Watchpoint Monitor Block Diagram

20.1.1 Overview

As shown in Figure 20-1, debug information is provided through three major interfaces; PCI/PCI-X, local bus, and DDR SDRAM. Limited visibility, through a 256 x 64 trace buffer, is also provided for the processor core interface and on an internal RapidIO outbound interface. This visibility into internal device operation can help the user debug the system/software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID) and a data valid signal (MDVAL) indicates meaningful activity is visible on the local bus and DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID and MDVAL.

In addition to the visibility aids, features such as programmable triggering in the watchpoint monitor and trace buffer can be very useful in system troubleshooting. The watchpoint monitor and the trace buffer can be triggered by one of three sources:

- each other
- a performance monitor event
- an external source via TRIG_IN.

The watchpoint monitor can be configured to assert TRIG_OUT when a programmed event occurs. The trace buffer block can be used as a second watchpoint monitor.

20.1.2 Features

The debug and watchpoint monitor main features are as follows:

- PCI/PCI-X interface source port multiplexed onto AD[63:59]
- Local bus interface valid data and address indicators
 - Pins multiplexed between the local bus and DDR SDRAM interfaces.
- DDR interface data valid and address indicators
 - One mode multiplexes the local bus and DDR SDRAM signals onto the same pins
 - Other mode multiplexes data valid and address indicators onto the error checking and correcting (ECC) pins of the DDR interface
- Watchpoint monitor
 - Begin triggering for the programmed event when some other event occurs
 - Option to activate an external trigger (TRIG_OUT) when the programmed event occurs
 - Interlocked with the performance monitor to take advantage of the large number of counters
- Trace buffer feature
 - Begin triggering for the programmed event when some other event occurs
 - Option to activate an external trigger (TRIG_OUT) when the programmed event occurs
 - Interlocked with the performance monitor to take advantage of the large number of counters
 - 256-entry trace buffer, 64 bits each
 - Option to trace every cycle or to start tracing on specific events
 - Option to stop tracing on specific events

20.1.3 Modes of Operation

Two of the three interfaces where debug information is available, PCI/PCI-X and DDR SDRAM, have normal and debug operating modes where they carry normal traffic or debug information respectively. The watchpoint monitor and trace buffer features have a variety of operating modes.

20.1.3.1 Local Bus Interface Debug Modes of Operation

- Select local bus debug information to source MSRCID and MDVAL signals.
The user has the option to select between the local bus controller and the DDR SDRAM controller for driving source ID and DVAL indicators onto the MSRCID and MDVAL signals.

20.1.3.2 DDR SDRAM Interface Debug Modes of Operation

- Select DDR SDRAM debug information to put on MSRCID and MDVAL signals
This mode is controlled through the debug mode sampled during POR on the pin MSRCID0. If MSRCID0 equals 0, then the local bus SDRAM information is placed on MSRCID and MDVAL. If MSRCID0 equals 1, then the DDR SDRAM information is placed on MSRCID and MDVAL.
- Select ECC or SrcID/DVAL to drive onto the ECC pins on the DDR SDRAM interface

This mode is controlled through the debug mode sampled during POR on the pin MSRCID1. If MSRCID1 equals 0, then the SrcID/DVAL information is driven onto the ECC pins. When the debug information is selected, ECC checking is automatically turned off in the memory controller.

20.1.3.3 PCI/PCI-X Interface Debug Modes of Operation

- PCI/PCI-X debug mode

In this mode, the source ID (SID) information is driven on the high-order address bits (ADDR[63:59]) during the bus command phase. The bus command phase may be during the first cycle that FRAME is asserted or it may be after a dual address cycle phase in the case of addresses greater than 32 bits. In either case, the debug information is driven on the highest order address bits when FRAME is asserted and provided both IRDY and TRDY are negated.

This mode is controlled through the debug mode sampled during POR on the pin PCI_GNT3. If PCI_GNT3 equals 0, then the source ID information is driven onto ADDR[63:59].

20.1.3.4 Watchpoint Monitor Modes of Operation

- Wait for event to start triggering
The watchpoint monitor can be programmed to wait for a specific event before it starts triggering on the programmed event. That is, the watchpoint monitor does not indicate a hit until the start condition is satisfied.
- Assert TRIG_OUT pin on a hit
This mode enables the watchpoint monitor to assert TRIG_OUT when a programmed event is hit.

20.1.3.5 Trace Buffer Modes of Operation

- Variety of modes to trigger the start of tracing
The trace buffer may be programmed to start immediately or when one of the programmed “start tracing” events is detected.
- Variety of modes to trace specific events
The trace may be programmed to trace every cycle in which a valid transaction is present on the selected interface, when a watchpoint monitor event occurs, or when the programmed trace event is detected.
- Variety of modes to trigger the stop of tracing
The trace buffer may be programmed to stop on full, or stop when one of the programmed “stop tracing” events is detected.

20.2 External Signal Description

20.2.1 Overview

All the signals associated with the features described in this document are listed in Table 20-1. Some signals may be duplicated in other blocks, MECC for example, but are covered in this area for completeness and with an eye to explaining their use from a debug perspective.

Table 20-1. Signal Properties

Name	Port	Function	Reset State	Pull up
MDVAL	Debug	Selectable data valid sign from either DDR SDRAM controller or local bus controller. This is an output.	1	—
MECC[0:7]	DDR	In debug mode, the high order 6 bits carry debug information (source ID and data valid). This is an output in debug mode.	0x08	—
MSRCID[0:4]	Debug	Selectable source ID from either DDR SDRAM controller or local bus controller. This is an output.	0x05	—
TRIG_IN	Debug	Trigger input used as a trigger for various function in the watchpoint monitor and trace buffer blocks. This signal is required to remain active for a minimum of 3 system clocks to be recognized internally. This is an input.	1	—
TRIG_OUT	Debug	Trigger output which can be used externally for triggering a logic analyzer or for any other purpose. This signal remains asserted for a minimum of 3 clock cycles. Different functions are MUXed onto this signal depending on the value of TOSR[SEL]. See Table 20-24 for a description of these functions. This is an output.	1	—

Table 20-1. Signal Properties

Name	Port	Function	Reset State	Pull up
PCI_AD[63:59]	Debug	In debug mode these pins carry the source ID of the transaction.	1'b0	—
TCK	Debug	Test Clock		
TCI	Debug	Test clock input. Inputs instructions and data to the JTAG scan registers.		
TCO	Debug	Test clock output. Outputs instructions and data to the JTAG scan registers.		
TMS	Debug	Test mode select. Carries commands to the TAP controller for boundary scan operations.		
TRST	Debug	Test reset. Resets the TAP controller asynchronously.		
TEST_SEL	Test	Factory Test		
LSSD_MODE	Test	Factory Test		
L1_TSTCLK	Test	Factory Test		
L2_TSTCLK	Test	Factory Test		

20.2.2 Detailed Signal Descriptions

20.2.2.1 Memory Data Valid (MDVAL)—Output

The memory data valid (MDVAL) signal indicates that data is being received or is being driven during this cycle. MDVAL may be used by a logic analyzer to capture the data on the data bus.

State Meaning

Assertion/negation— When asserted, it indicates that data is being received or driven this cycle. When the DDR SDRAM interface is chosen to drive information on MDVAL, this signal is valid for every cycle that data is being driven or received on the DDR SDRAM interface. When the local bus interface is chosen, this signal is valid for every cycle that data is being driven or received on the local bus interface. The assertion of this signal may be used by a logic analyzer to capture data.

Timing Comments

Assertion is in reference to the selected interface, the DDR or local bus interface, and is held asserted for the duration of the transfer.

20.2.2.2 Memory ECC (MECC[0:7])—Output

The memory ECC (MECC) bus is bidirectional during normal operation. The alternate function of this bus is what is being covered in this portion of the documentation. The high

order 6 bits of this bus may be used in a debug mode to drive the source ID and data valid indicator. In this state, the bus is constantly being driven with this information.

State Meaning Assertion/negation— In debug mode, MECC is always driven with the value of the source ID and data valid values. The source ID has a value of 0x1F for cycles other than RAS and CAS.

Timing Comments Driven every cycle in debug mode.

20.2.2.3 Memory Source ID (MSRCID[0:4])—Output

The memory attribute signals are associated with the memory interface and provide information about the source of the memory operation being performed by the MPC8540. They are also reset configuration input signals. The memory source ID (MSRCID) bus indicates the source ID for a transaction on the SDRAM interface. The source of the value, DDR or local bus, is a selectable mode.

State Meaning Assertion/negation— In debug mode, always driven with the value of the source ID and data valid values. The source ID has a value of 0x1F for cycles other than RAS and CAS. These signals are encoded to provide more detailed information about a memory transaction.

Timing Comments Driven every cycle in debug mode.

20.2.2.4 Watchpoint Trigger Signals

There is one watchpoint trigger input and one watchpoint trigger output signal that together provide a programmable output signal and control of the watchpoint facility.

20.2.2.4.1 Trigger In (TRIG_IN)—Input

The trigger in (TRIG_IN) signal is used for triggering the watchpoint and trace buffer when programmed to respond to this event. Note that TRIG_IN is an active-high (rising-edge triggered) signal and it can alternately be used as the $\overline{\text{RCS2}}$ output signal.

State Meaning Assertion/negation— When asserted, it indicates that a programmed/armed external event has been detected. The assertion of this signal may be used internally to trigger internal trace buffers and watchpoint mechanisms. May cause the MPC8540 to exit the HOLD state or may cause the value of the WP_RUN bit in the WP_CONTROL register to toggle (turning the watchpoint facility on or off).

Negated: No action taken.

Timing Comments Assertion/Negation: The MPC8540 interprets TRIG_IN as asserted on detection of the rising edge of TRIG_IN. It may occur at any time.

Must remain asserted for a minimum of 3 system clocks to be recognized internally.

20.2.2.4.2 Trigger Out (TRIG_OUT)—Output

The trigger out (TRIG_OUT) signal is used for triggering external devices such as a logic analyzer. Note that the active sense of TRIG_OUT is controlled by the setting of WP_CONTROL[WP_TRIG] and it can alternately be used as the $\overline{\text{RCS2}}$ output signal.

State Meaning Asserted: Indicates that a final watchpoint match has occurred, as defined in the WP_MODE field of the WP_CONTROL register.

Negated: No final watchpoint match condition.

Timing Comments Assertion may occur at any time. It is asserted for a minimum of three system clocks.

20.2.2.5 PCI Address (PCI_AD[63:59])—Output

The PCI address signal is associated with the PCI bus and provides information about the source of the bus transaction being performed by the MPC8540.

State Meaning Assertion/negation— In debug mode, always driven with the value of the transaction source ID.

Timing Comments Driven every cycle in debug mode.

20.2.3 Test and Configuration Signals

The MPC8540 has several signals that are sampled during reset to determine the configuration of the phase-locked loop clock mode and the ROM, Flash, and dynamic memory.

To facilitate system testing, the MPC8540 provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes the JTAG test access port signals.

20.2.3.1 JTAG Test Clock (TCK)—Input

The JTAG test clock (TCK) signal is an input on the MPC8540. Following is the state meaning for the TCK input signal.

State Meaning Asserted/Negated: This input should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the test access port are clocked in on the rising edge of TCK. Changes to the test access port output signals occur on the falling edge of TCK. The test logic allows TCK to be stopped.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

20.2.3.2 JTAG Test Data Input (TDI)—Input

Following is the state meaning for the TDI input signal.

State Meaning Asserted/Negated: The value presented on this signal on the rising edge of TCK is clocked into the selected JTAG test instruction or data register.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

20.2.3.3 JTAG Test Data Output (TDO)—Output

Following is the state meaning for the TDO output signal.

State Meaning Asserted/Negated: The contents of the selected internal instruction or data register are shifted out onto this signal on the falling edge of TCK. The TDO signal remains in a high-impedance state except when scanning of data is in progress.

20.2.3.4 JTAG Test Mode Select (TMS)—Input

The test mode select (TMS) signal is an input on the MPC8540. Following is the state meaning for the TMS input signal.

State Meaning Asserted/Negated: This signal is decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

20.2.3.5 JTAG Test Reset ($\overline{\text{TRST}}$)—Input

The test reset ($\overline{\text{TRST}}$) signal is an input on the MPC8540. Following is the state meaning for the $\overline{\text{TRST}}$ input signal.

State Meaning Asserted: This input causes asynchronous initialization of the internal JTAG test access port controller. Note that the signal must be asserted during power-up reset in order to properly initialize the JTAG test access port and for normal operation of the MPC8540.

Negated: Indicates normal operation.

Note that this input contains an internal pull-up resistor to ensure that an unterminated input appears as a high signal level to the test logic.

20.2.3.6 $\overline{\text{LSSD\_MODE}}$ —Input

$\overline{\text{LSSD\_MODE}}$ is used for factory test.

20.2.3.7 L1_TSTCLK—Input

L1_TSTCLK is used for factory test.

20.2.3.8 L2_TSTCLK—Input

L2_TSTCLK is used for factory test.

20.2.3.9 TEST_SEL—Input

TEST_SEL is used for factory test and should be connected to binary 0.

20.3 Memory Map/Register Definition

Table 20-2 shows the memory mapped registers accessible through the internal register interface.

Table 20-2. Debug and Watchpoint Monitor Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
Watchpoint Monitor Registers				
0xE_2000	Watchpoint Monitor Control Register 0 (WMCR0)	R/W	0x0000_0000	20.3.1.1/20-11
0xE_2004	Watchpoint Monitor Control Register 1 (WMCR1)	R/W	0x0000_0000	20.3.1.2/20-12
0xE_200C	Watchpoint Monitor Address Register (WMAR)	R/W	0x0000_0000	20.3.1.3/20-13
0xE_2014	Watchpoint Monitor Address Mask Register (WMAMR)	R/W	0x0000_0000	20.3.1.4/20-14
0xE_2018	Watchpoint Monitor Transaction Mask Register (WMTMR)	R/W	0x0000_0000	20.3.1.5/20-15
0xE_201C	Watchpoint Monitor Status Register (WMSR)	R/W	0x0000_0000	20.3.1.6/20-17
Trace Buffer Registers				
0xE_2040	Trace Buffer Control Register (TBCR0)	R/W	0x0000_0000	20.3.1.1/20-11
0xE_2044	Trace Buffer Control Register (TBCR1)	R/W	0x0000_0000	20.3.1.2/20-12
0xE_204C	Trace Buffer Address Register (TBAR)	R/W	0x0000_0000	20.3.2.3/20-20
0xE_2054	Trace Buffer Address Mask Register (TBAMR)	R/W	0x0000_0000	20.3.2.4/20-21
0xE_2058	Trace Buffer Transaction Mask Register (TBTMR)	R/W	0x0000_0000	20.3.2.5/20-22
0xE_205C	Trace Buffer Status Register (TBSR)	R/W	0x0000_0000	20.3.2.6/20-22
0xE_2060	Trace Buffer Access Control Register (TBACR)	R/W	0x0000_0000	20.3.2.7/20-23

Table 20-2. Debug and Watchpoint Monitor Memory Map (continued)

Local Memory Offset	Register	Access	Reset	Section/Page
0xE_2064	Trace Buffer Access Data High Register (TBADHR)	R/W	0x0000_0000	20.3.2.8/20-24
0xE_2068	Trace Buffer Access Data Register (TBADR)	R/W	0x0000_0000	20.3.2.9/20-25
Context ID Registers				
0xE_20A0	Programmed Context ID Register (PCIDR)	R/W	0x0000_0000	20.3.3.1/20-25
0xE_20A4	Current Context ID Register (CCIDR)	R/W	0x0000_0000	20.3.3.2/20-26
Other Registers				
0xE_20B0	Trigger Output Source Register (TOSR)	R/W	0x0000_0000	20.3.4.1/20-27

20.3.1 Watchpoint Monitor Register Descriptions

20.3.1.1 Watchpoint Monitor Control Register 0 (WMCR0)

The watchpoint monitor control register 0 (WMCR0) contains part of the controls for setting up a watchpoint monitor event.

	0	1	2	3	4	5	6	7									15
R	EN	AMD	TMD	ECEN	NECEN	SIDEN	TIDEN	0	0	0	0	0	0	0	0	0	0
W																	
Reset	0000_0000_0000_0000																

	16				20	21		23	24								31
R	0	0	0	0	0	STRT			0	0	0	0	0	0	0	0	0
W																	
Reset	0000_0000_0000_0000																
Offset	0x000																

Figure 20-2. Watchpoint Monitor Control Register 0 (WMCR0)**Table 20-3. WMCR0 Field Descriptions**

Bits	Name	Description
0	EN	Enable 0 no watchpoint monitor event is flagged. 1
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 is ignored when detecting a watchpoint event.

Table 20-3. WMCR0 Field Descriptions

Bits	Name	Description
2	TMD	Transaction match disable. Qualifies transaction type match as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match is ignored when detecting watchpoint events.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion. The function of ECEN and NECEN are mutually exclusive so their values should be complementary. If both are set, watchpoint events never occur (are inhibited). 0 Current context match is ignored when detecting watchpoint events. 1 Watchpoint events are qualified by comparing current context with the programmed context event value.
4	NECEN	Not equal context enable 0 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. The value of ECEN and NECEN should be mutually exclusive. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved
21–23	STRT	Start condition. This field specifies the event that arms the Watchpoint monitor to start looking for the programmed event. 000 Armed immediately 001 Armed when Trace Buffer event is detected 010 Armed when Performance Monitor signals overflow 011 Armed when TRIG_IN goes from 0 to 1 100 Armed when TRIG_IN goes from 1 to 0 101 Armed when current context ID equals programmed context ID 110 Armed when current context ID not equal programmed context ID 111 Reserved
24–31	—	Reserved

20.3.1.2 Watchpoint Monitor Control Register 1 (WMCR1)

This register contains the other part of the controls for setting up a watchpoint monitor event.

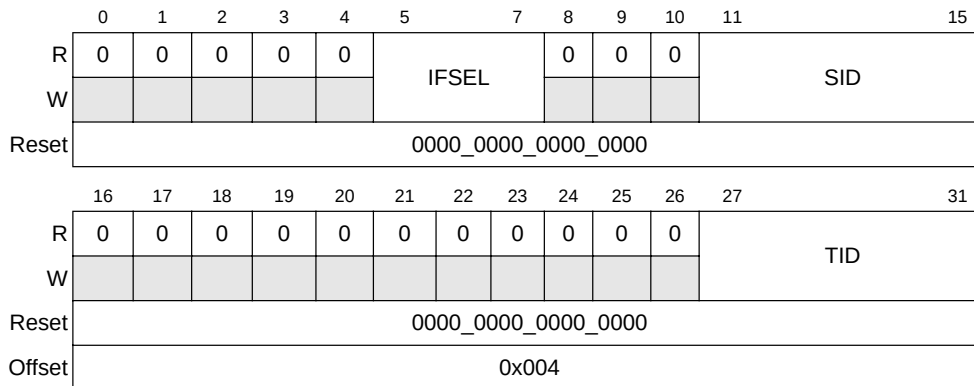


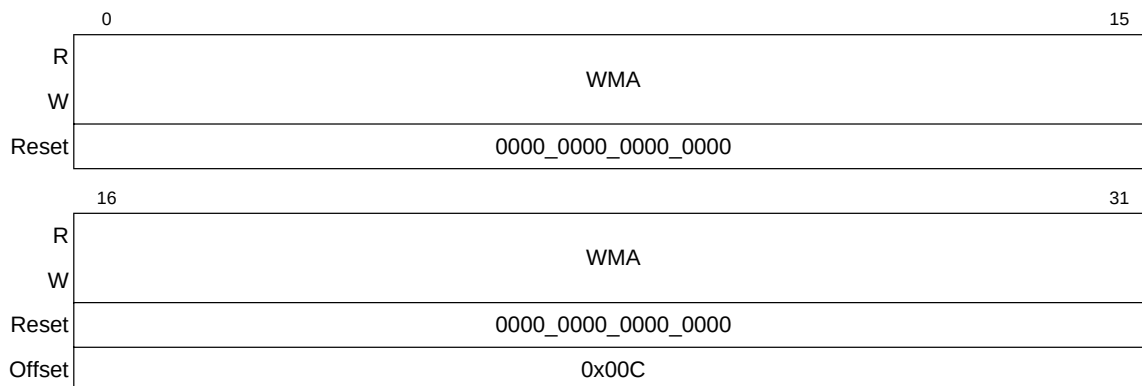
Figure 20-3. Watchpoint Monitor Control Register (WMCR1)

Table 20-4. WMCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. This field selects the address, transaction type and other attributes to be used for comparison. 000Select Coherency Module dispatch interface 001Select DRAM interface 010Select PCI outbound interface 011Select RapidIO interface 1xxReserved
8–10	—	Reserved
11–15	SID	Source ID. This field specifies the source ID associated with WMCR0[SIDEN]. For a definition of the Source ID decode see Table 20-25.
16–26	—	Reserved
27–31	TID	Target ID. This field specifies the target ID associated with WMCR0[TIDEN]. For a definition of the Target ID decode see Table 20-25.

20.3.1.3 Watchpoint Monitor Address Register (WMAR)

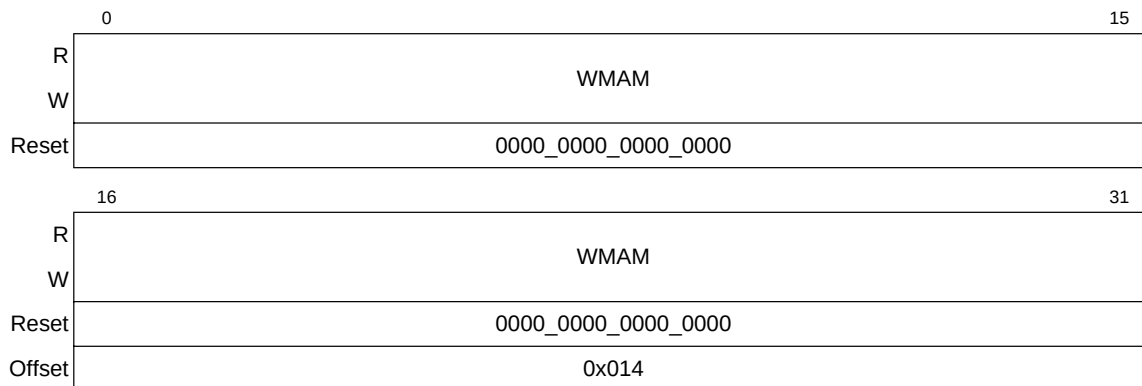
This register contains the address to match against if the WMCR[AMD] = 0.

**Figure 20-4. Watchpoint Monitor Address Register (WMAR)****Table 20-5. WMAR Field Descriptions**

Bits	Name	Description
0–31	WMA	Watchpoint monitor address. Low order bits of the match address. A value of 0 in a bit masks the address comparison for the corresponding address bit.

20.3.1.4 Watchpoint Monitor Address Mask Register (WMAMR)

This register contains the mask for the address contained in the register WMAR..Permits user to mask desired address bits from the comparison.

**Figure 20-5. Watchpoint Monitor Address Mask Register (WMAMR)****Table 20-6. WMAMR Field Descriptions**

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. Low order bits of the match address mask.

20.3.1.5 Watchpoint Monitor Transaction Mask Register (WMTMR)

This register contains the value for the transaction mask. The mask enables a user to match one or more transaction types. A value of 1 in the specific transaction field enables a user to match when this transaction is matched (along with other match criteria) if the WMCR[TMD] = 0. Each bit represents a transaction type on the selected interface.

Different interfaces support different transaction types and each bit may represent a different transaction type depending on the interface.

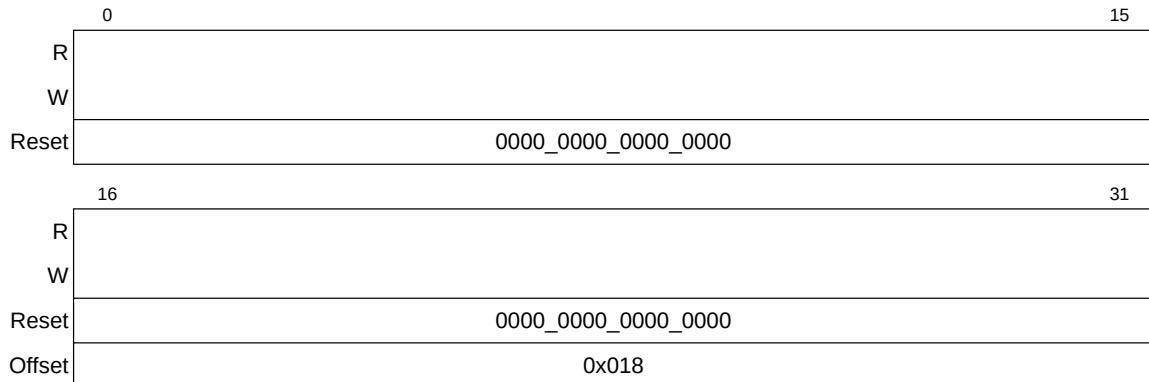


Figure 20-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 20-7. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit of this mask is associated with a transaction type. The transaction associated with any particular bit may be different depending on the interface being monitored.

The following tables define the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

Table 20-8. Coherency Module Dispatch Interface

Bits	Description
0	Match on Write with local processor snoop
1	Match on Write with no local processor snoop
2	Match on Write with allocate(L2 stashing)
3	Match on Write with allocate and lock (L2 stashing with locking)
4–7	Reserved
8	Match on Read with local processor snoop
9	Match on Read with no local processor snoop
10	Match on Read with unlock

Table 20-8. Coherency Module Dispatch Interface

Bits	Description
11–15	Reserved
16	Match on Atomic Clear
17	Match on Atomic Set
18	Match on Atomic Decrement
19	Match on Atomic Increment
20–31	Reserved

Table 20-9. DDR Controller Request Interface

Bits	Description
0	Match on Write
1	Reserved
2	Match on Write with allocate
3	Match on Write with allocate and lock
4–7	Reserved
8	Match on Read
9	Reserved
10	Match on Read with unlock
11–15	Reserved
16	Match on Atomic Clear
17	Match on Atomic Set
18	Match on Atomic Decrement
19	Match on Atomic Increment
20–31	Reserved

Table 20-10. PCI Outbound Request Interface

Bits	Description
0	Match on Memory Write transaction
1	Match on IO write
2–7	Reserved
8	Match on Memory Read
9	Match on I/O Read
10–31	Reserved

Table 20-11. RapidIO Outbound Request Interface

Bits	Description
0	Match on NWRITE
1	Match on NWRITE_R
2	Match on SWRITE
3	Match on FLUSH with data
4–7	Reserved
8	Match on NREAD
9	Match on IO_READ_HOME
10–15	Reserved
16	Match on ATOMIC Clear
17	Match on ATOMIC Set
18	Match on ATOMIC Decrement
19	Match on ATOMIC Increment
20–27	Reserved
28	Match on MAINTENANCE Write
29	Match on MAINTENANCE Read
30	Match on DOORBELL
31	Match on MESSAGE

20.3.1.6 Watchpoint Monitor Status Register (WMSR)

This register contains the status of the watchpoint monitor.

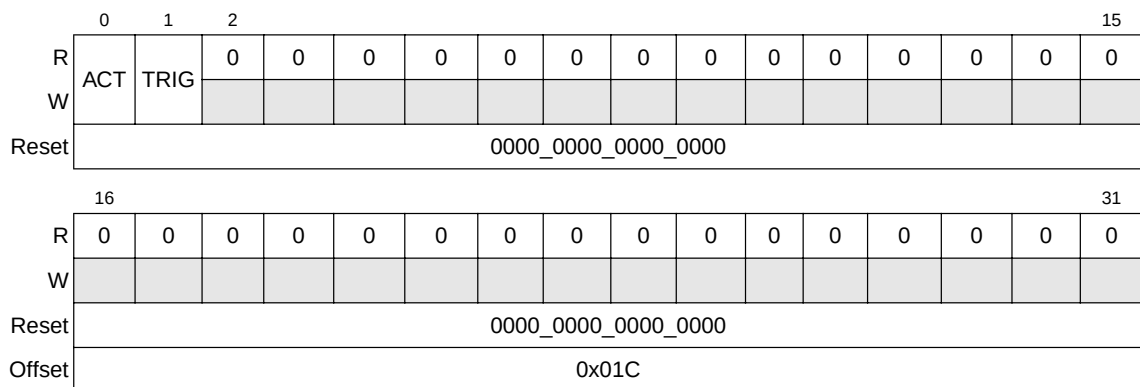
**Figure 20-7. Watchpoint Monitor Status Register (WMSR)**

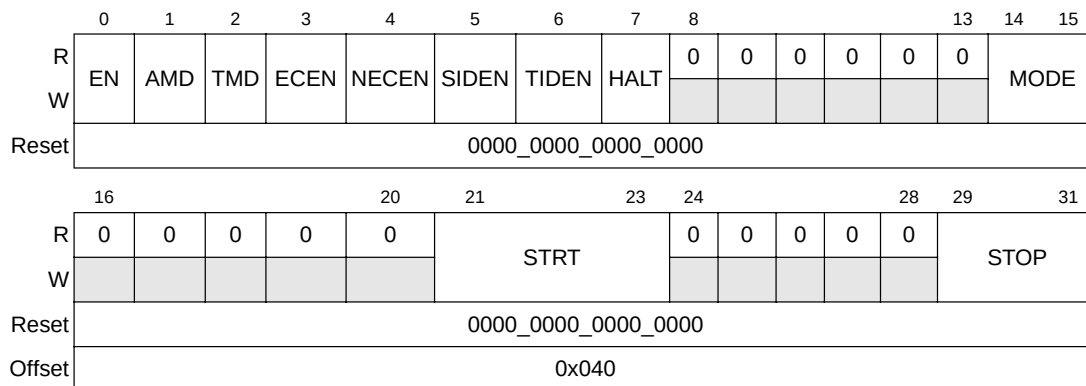
Table 20-12. WMSR Field Descriptions

Bits	Name	Description
0	ACT	Active. 1 The start triggering event has taken placed - watchpoint monitor is armed.
1	TRIG	TRIGGERED. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

20.3.2 Trace Buffer Register Descriptions

20.3.2.1 Trace Buffer Control Register (TBCR0)

This register contains part of the controls for setting up a trace buffer event. See Section 20.3.2.2, “Trace Buffer Control Register 1 (TBCR1),” for other trace buffer event controls.

**Figure 20-8. Trace Buffer Control Register 0 (TBCR0)****Table 20-13. TBCR0 Field Descriptions**

Bits	Name	Description
0	EN	Enable 0 No trace buffer event is flagged.
1	AMD	Address match disable 0 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 1 The transaction type match is ignored when detecting a trace buffer event.

Table 20-13. TBCR0 Field Descriptions

Bits	Name	Description
3	ECEN	Equal context enable 0 1 Trace buffer events are qualified by comparing current context with the programmed context event value. The value of ECEN and NECEN should be mutually exclusive. If both are set, trace events are inhibited (never occur).
4	NECEN	Not equal context enable 0 1 Trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. The value of ECEN and NECEN should be mutually exclusive. If both are set, trace events are inhibited (never occur).
5	SIDEN	Source ID enable 0 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the coherency module is selected for tracing (TBCR1[IFSEL] = 0b000).
7	HALT	Halt causes the trace buffer to stop tracing immediately.
8–13	—	Reserved
14–15	MODE	Trace mode 00 Trace every valid cycle 01 Reserved 10 Trace only cycles in which TRACE event is detected ¹ 11 Reserved
16–20	—	Reserved
21–23	STRT	Start condition. This field specifies the event that arms the Trace Buffer to start looking for the programmed event. 000 Armed immediately 001 Armed when Watchpoint monitor event is detected 010 Armed when Trace Buffer event is detected 011 Armed when Performance Monitor signals overflow 100 Armed when TRIG_IN goes from 0 to 1 101 Armed when TRIG_IN goes from 1 to 0 110 Armed when current context ID equals programmed context ID 111 Armed when current context ID not equal programmed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. This field specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stop after it has been triggered at least once. 000 Stop tracing when buffer is full 001 Stop tracing when Watchpoint monitor event is detected 010 Stop tracing when Trace Buffer event is detected 011 Stop tracing when Performance Monitor signals overflow 100 Stop tracing when the TRIG_IN goes from 0 to 1 101 Stop tracing when TRIG_IN goes from 1 to 0 110 Stop tracing when current context ID equals programmed context ID 111 Stop tracing when current context ID not equal programmed context ID

¹ When selecting a trace event as the trace mode, if no event is programmed in the trace buffer control register (TBCR0), then tracing takes place for every valid address. This is really considered a programming error, but this is the chosen behavior.

20.3.2.2 Trace Buffer Control Register 1 (TBCR1)

This register contains a portion of the controls for setting up a trace buffer event. See Section 20.3.2.1, “Trace Buffer Control Register (TBCR0)” for other trace buffer event controls.

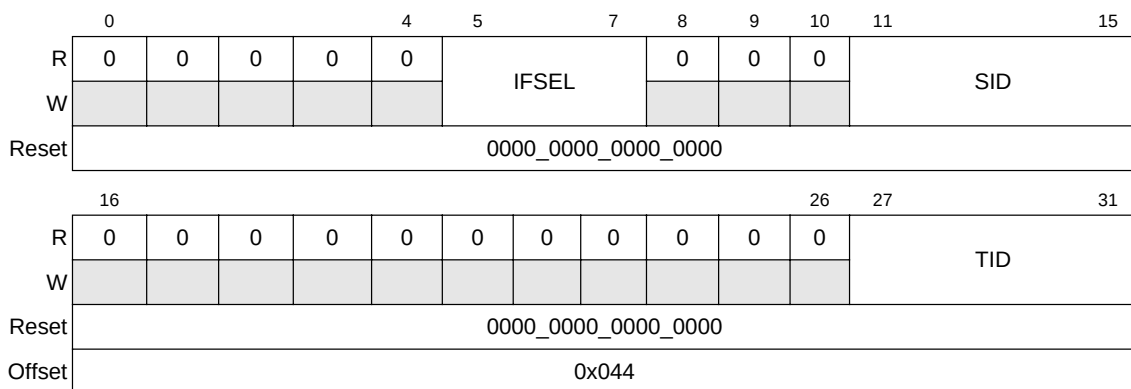


Figure 20-9. Trace Buffer Control Register (TBCR1)

Table 20-14. TBCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. This field selects the address, transaction type and other attributes to be used for comparison. for the address and transaction type comparison. 000 Select Coherency Module dispatch interface 001 Select DRAM interface 010 Select PCI outbound interface 011 Select RapidIO interface 1xx Reserved
8–10	—	Reserved
11–15	SID	Source ID. This field specifies the source ID associated with TBCR0[SIDEN]. For a definition of the Source ID decode see Table 20-25.
16–26	—	Reserved
27–31	TID	Target ID. This field specifies the target ID associated with TBCR0[TIDEN]. For a definition of the Target ID decode see Table 20-25.

20.3.2.3 Trace Buffer Address Register (TBAR)

This register contains the address to match against if TBCR[AMD] = 0.



Figure 20-10. Trace Buffer Address Register (TBAR)

Table 20-15. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address. Low order bits of the match address

20.3.2.4 Trace Buffer Address Mask Register (TBAMR)

This register contains the mask for the address contained in the register TBAR. Permits user to mask desired address bits from the comparison.

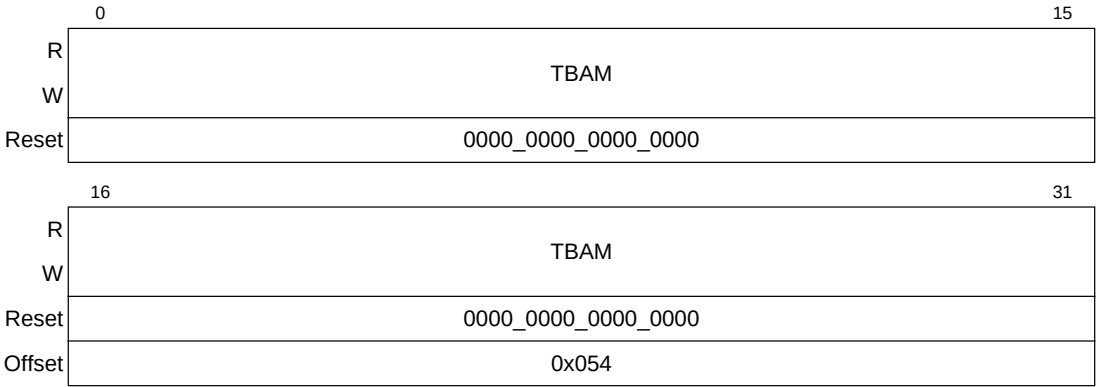


Figure 20-11. Trace Buffer Address Mask Register (TBAMR)

Table 20-16. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask. Low order bits of the match address mask.

20.3.2.5 Trace Buffer Transaction Mask Register (TBTMR)

This register contains the value for the transaction mask. This enables a user to match one or more transaction types. A value of 1 in the specific transaction field enables a user to match when this transaction is matched (along with other match criteria) if TBCR[TMD] = 0. Each bit represents a transaction type on the selected interface.

Different interfaces support different transaction types, and each bit may represent a different transaction type depending on the interface.

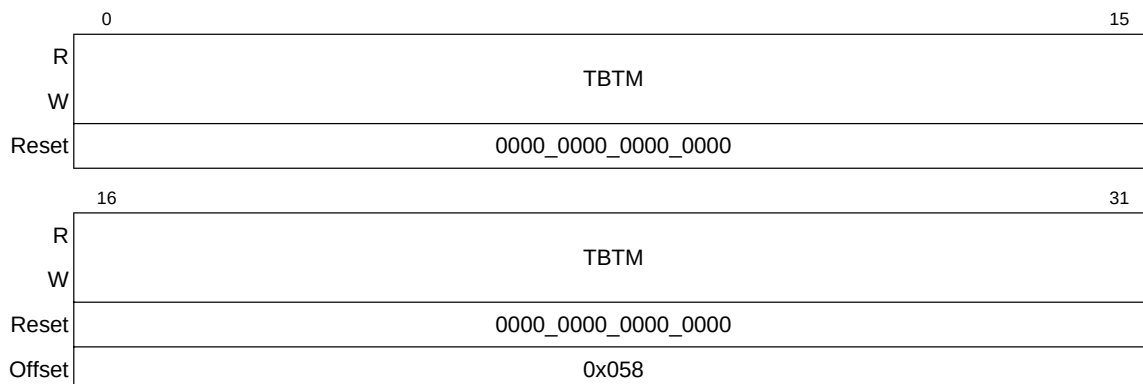


Figure 20-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 20-17. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit of this mask is associated with a transaction type. The transaction associated with any particular bit may be different depending on the interface being monitored. See Table 20-8 through Table 20-11 for the definition of the transaction types.

20.3.2.6 Trace Buffer Status Register (TBSR)

This register contains the status of the trace buffer.

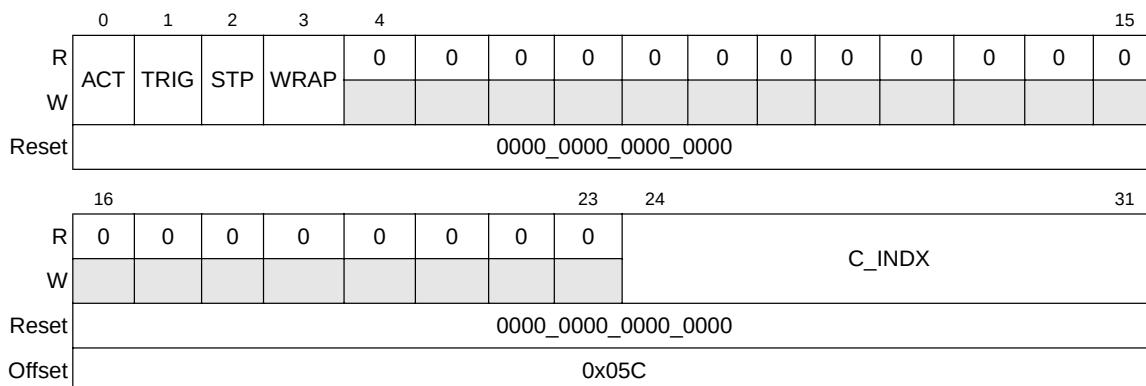


Figure 20-13. Trace Buffer Status Register (TBSR)

Table 20-18. TBSR Field Descriptions

Bits	Name	Description
0	ACT	Active 1 The start triggering event has taken place - trace buffer is armed.
1	TRIG	Triggered 1 The programmed event in TBCR0 has been triggered at least once.
2	STP	Stopped 1 The trace buffer hit a stopped condition.
3	WRAP	Wrapped 1 The trace buffer write pointer wrapped to the beginning at least once. Set whenever the last entry of the trace buffer is written.
4–23	—	Reserved
24–31	C_INDXX	Current index. This represents the current value of the write pointer at the time the TBSR register was read. This value may be written by software to initialize the write pointer, however, software is not allowed to write the write pointer when the trace buffer is active - writes are ignored when the trace buffer is active. It is recommended to write the status register before enabling the trace buffer to zero out any bits which might have been set during a prior run and to initialize the write pointer to 0.

20.3.2.7 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 the TBACR register and two reads of the access data register (TBADR and TBADHR) to read one entry of the 256-entry array. Similarly, it takes one write of the TBACR register and two writes of the access data register (TBADR and TBADHR) to write one entry of the 256-entry array. Software can access any entry by writing the appropriate index into the TBACR[INDX] field. If a user wishes to read or write the buffer sequentially, starting with entry 0, the user must start with an index of 0 and increment it by one every time a new entry is read or written.

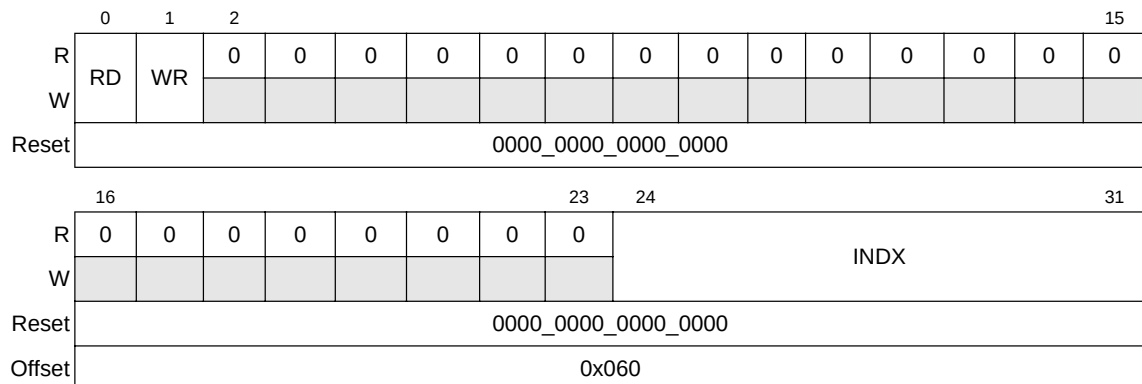
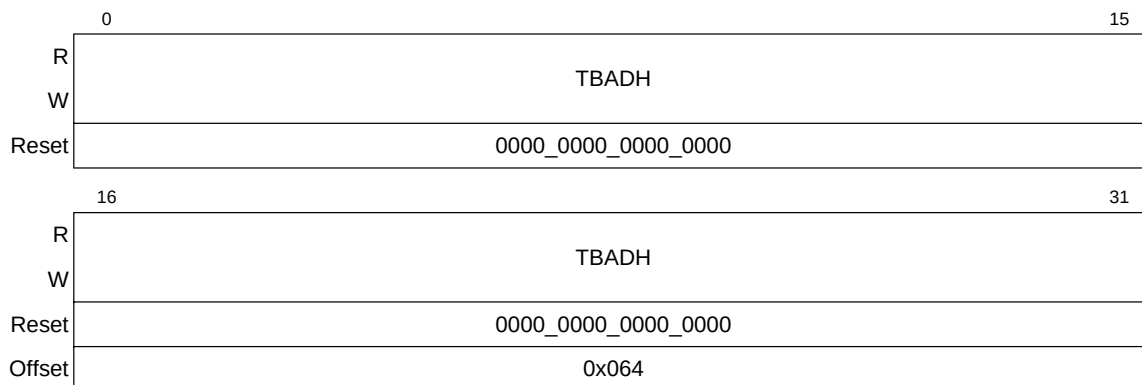
**Figure 20-14. Trace Buffer Access Control Register (TBACR)**

Table 20-19. TBACR Field Descriptions

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. Cleared when the read is performed.
1	WR	Write command. When written with a value of 1, a trace buffer write is performed using the value of TBACR[INDX]. Cleared when the write is performed. A write occurs only when the trace buffer is not active: write requests are ignored when the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer Index to read from or write into(0–255). This field is used in conjunction with TBACR[RD] and TBACR[WR].

20.3.2.8 Trace Buffer Access Data High Register (TBADHR)

This register 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]). The TBACR register must be set up first to perform a read and then this register contains valid data. After writing TBACR with the read command, no polling is required, and subsequent accesses to any register in the trace buffer are delayed until the data is placed in TBADHR and TBADR. This register must be set up by software before setting up the TBACR to perform a write command.

**Figure 20-15. Trace Buffer Read High Register (TBADHR)****Table 20-20. TBADHR Field Descriptions**

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. This corresponds to the higher 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.

20.3.2.9 Trace Buffer Access Data Register (TBADR)

This register 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]). The TBACR register must be set up first to perform a read and then this register contains valid data. After writing TBACR with the read command, no polling is required, and subsequent accesses to any register in the trace buffer are delayed until the data is placed in TBADHR and TBADR. This register must be set up by software before setting up the TBACR to perform a write command.

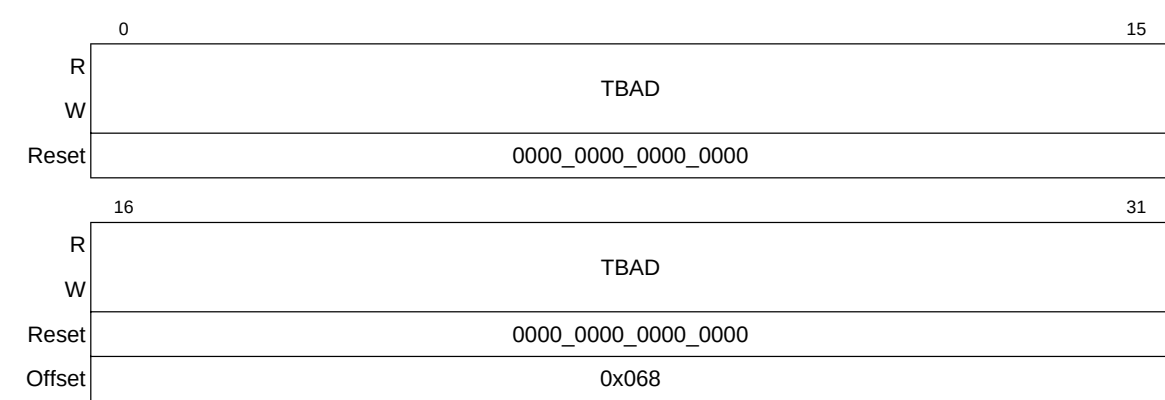


Figure 20-16. Trace Buffer Read High Register (TBADR)

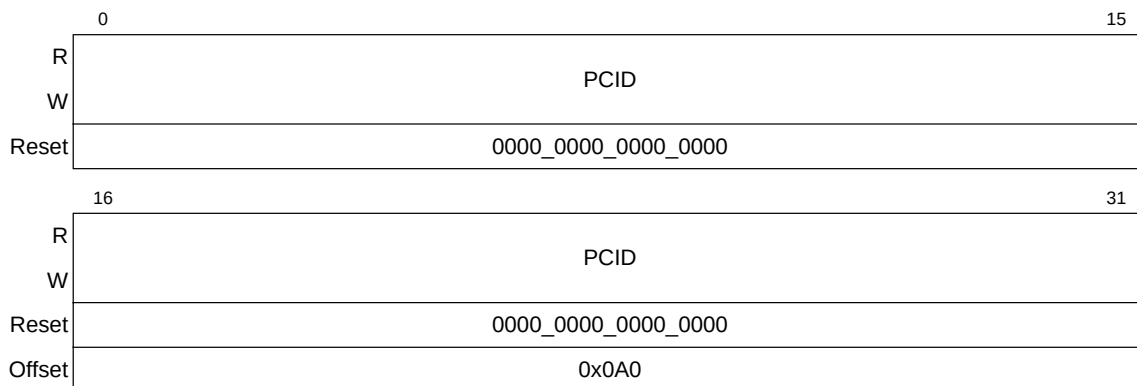
Table 20-21. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. This 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.

20.3.3 Context ID Registers

20.3.3.1 Programmed Context ID Register (PCIDR)

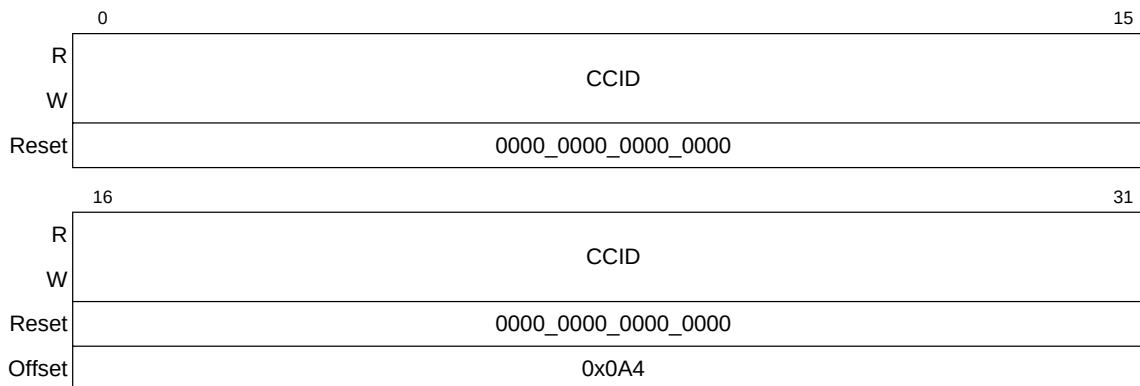
This register contains the context ID to be compared with the current context ID. This register is initialized by software and can be used to trigger events when it matches the current context ID register (CCIDR).

**Figure 20-17. Programmed Context ID Register (PCIDR)****Table 20-22. PCIDR Field Descriptions**

Bits	Name	Description
0–31	PCID	Programmed Context ID

20.3.3.2 Current Context ID Register (CCIDR)

This register contains the context ID to be compared with the programmed context ID. This register is written by software on a context switch and can be used to trigger events when it matches the programmed context ID register (PCIDR).

**Figure 20-18. Current Context ID Register (CCIDR)****Table 20-23. CCIDR Field Descriptions**

Bits	Name	Description
0–31	CCID	Current Context ID

20.3.4 Trigger Out Function

20.3.4.1 Trigger Out Source Register (TOSR)

This register contains the controls for selecting a source for the TRIG_OUT output pin. The sources for TRIG_OUT are the watchpoint monitor, the trace buffer, and the performance monitor event.



Figure 20-19. Trigger Out Source Register (TOSR)

Table 20-24. TOSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	SEL	Selects the source for TRIG_OUT. 000 Ready indication. Device is not in a quiesced state 001 Selects the watchpoint monitor. 010 Selects the trace buffer ¹ . 011 Selects the performance monitor.
8–31	—	Reserved

¹ When the trace buffer hit is selected, meaningful information is put out on the pins only when the TBCR0 is configured to hit on something. Otherwise, the trace buffer control logic indicates a hit all the time and TRIG_OUT is always asserted if the trace buffer is enabled.

20.4 Functional Description

The debug features on the MPC8540 cover the major interfaces such as the local bus interfaces, DDR SDRAM interface, and the PCI interface. Debug information that is common to all the interfaces is the source ID (SID). The SID field provides enough information to determine which block or port originated a transaction including the differentiation between instruction and data fetch from the local processor. Table 20-25 shows the decodes for the 5-bit SID field. Note that the table also includes ports which are only slaves, such as local memory. These ports would always be a target. As such, the decode represents a target ID (TID) and not a SID. For a port that can be a master or a slave,

Functional Description

the decode means SID when it is a producer and signifies a TID when it is a consumer of a transaction. The TID field is only meaningful when the coherency module dispatch bus is selected, WMCR1[IFSEL] = 0b000 for the watchpoint monitor or TBCR1[IFSEL] = 0b000 for the trace buffer.

Table 20-25. Source ID and Target ID Decodes

SID Value (5 bits)	Source Port
00000	PCI
00001	Reserved
00010	Reserved
00011	Reserved
00100	Local Bus
00101	Reserved
00110	Reserved
00111	Reserved
01000	Configuration space
01001	Reserved
01010	Boot Sequencer
01011	Reserved
01100	RapidIO
01101	Reserved
01110	Reserved
01111	Local space (DDR)
10000	Local Processor (instruction fetch)
10001	Local Processor (Data fetch)
10010	Reserved
10011	Reserved
10100	Reserved
10101	DMA
10110	Reserved
10111	System Access Port (SAP)
11000	Ethernet 0
11001	Ethernet 1
11010	Ethernet 2
11011	Reserved
11100	RapidIO message unit

Table 20-25. Source ID and Target ID Decodes (continued)

SID Value (5 bits)	Source Port
11101	RapidIO doorbell unit
11110	RapidIO port-write unit
11111	Non valid port indicator (reserved for debug info)

20.4.1 PCI/PCI-X Interface Debug

If $\overline{\text{PCI\_GNT3}}$ is cleared when sampled during POR, then the PCI interface is in debug mode and the scheme described in this section is valid.

When the PCI/PCI-X interface is in debug mode, the source ID (SID) information is driven on the high-order address bits ($\text{PCI\_AD}[63:59]$) during the bus command phase. The bus command phase may be during the first cycle that $\overline{\text{PCI\_FRAME}}$ is asserted or it may be after a dual address cycle phase in the case of addresses greater than 32 bits. In either case, the debug info is driven on the highest order address bits when $\overline{\text{PCI\_FRAME}}$ is asserted and both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are negated.

For accesses to the low 4 Gbytes of address space, in which only 32 address bits are needed and no dual address cycle is needed, the debug information is driven during the first (and only) address phase on $\text{PCI\_AD}[63:59]$. For accesses to address above 4 Gbytes, a dual address cycle must be run, and the debug information is placed on $\text{PCI\_AD}[63:59]$ during the second address cycle. In either case the logic analyzer should be set up to sample information on the first cycle of the assertion of $\overline{\text{PCI\_FRAME}}$ and the cycle after a dual address cycle command.

NOTE

It is not possible to capture an entire 64-bit address and the debug information in a single cycle because they share the same pins.

20.4.2 DDR SDRAM Interface Debug

The DDR interface has two debug modes distinguished by which pins are used to output the debug information. In one mode debug information appears on the ECC pins, the other mode uses dedicated pins. Both modes are discussed below.

20.4.2.1 Debug Information on Debug Pins

If MSRCID0 is set when sampled during POR, the debug information from the DDR SDRAM interface is output on the pins MSRCID (SID) and MDVAL . The the source ID appears on the MSRCID pins during a RAS or CAS cycle. During any other cycle the value

of MSRCID is 0x1F which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

20.4.2.2 Debug Information on ECC Pins

If MSRCID1 is cleared when sampled during POR, the debug information from the DDR SDRAM interface is selected to be placed onto the pins MECC[0:5] (SrcID, DVAL). The value of the source port is placed on the ECC[0:4] pins during a RAS or CAS cycle. During any other cycle the value of MSRCID is 0x1F. Similarly, MDVAL is asserted during valid data cycles on the DDR interface. In this mode, the user must disconnect the ECC pins at the DRAM device to prevent contention on those lines.

20.4.3 Local Bus Interface Debug

Like the DDR controller, the local bus controller can be selected as the source for the debug pins MSRCID and MDVAL. If MSRCID0 is cleared when sampled during POR, the debug information from the local bus interface is selected to be placed onto the pins MSRCID (SrcID) and MDVAL. Unlike the DDR controller, this is the only way to put the debug information on this interface.

20.4.4 Watchpoint Monitor

The watchpoint monitor can be programmed to arm and trigger on many events. It can be armed by an external event via TRIG_IN, a trace buffer event, a performance monitor overflow event, or an equal (or not equal) compare of the current and programmed context ID registers. Once armed, meaning it is ready to start looking for the programmed trigger event, the watchpoint monitor starts looking for the programmed event. A watchpoint event can be used to trigger logic analyzers via TRIG_OUT, arm/trigger the trace buffer, or trigger a performance monitor event. The interlock between the performance monitor and the watchpoint monitor enables a user to create very useful debug scenarios by using the large counters available in the performance monitor block.

The watchpoint monitor trigger event may be composed of several other events which are programmed in the control registers (WMCR0–1). The default coming out of POR is watchpoint monitor feature disabled. Hence, in order to take advantage of this debug feature, the corresponding memory mapped registers must be initialize by the user. Note that the address mask register (WMAMR) and the type mask register (WMTMR) are cleared. This means that the watchpoint monitor triggers on any address and any transaction type.

20.4.4.1 Watchpoint Monitor Performance Monitor Events

The watchpoint monitor produces a performance monitor event with every trigger. This can be done by setting up the performance monitor to count the debug events “Number of

watchpoint monitor hits” and “Number of trace buffer hits.” See Chapter 19, “Performance Monitor” to set up the performance monitor block to count these events. One can create a chain of events using the watchpoint monitor, the performance monitor, and the trace buffer. For example, the watchpoint monitor can be programmed to trigger on certain events that may cause the incrementing of a performance monitor counter (the performance monitor must also be programmed to respond to this event) and then have the output of the performance monitor (`perfmon_overflow`) trigger the start of tracing in the trace buffer block. `Perfmon_overflow` is an internal signal which indicates that a counter overflow condition has occurred.

20.4.5 Trace Buffer

The trace buffer is a 256×64 array that can be used to capture information which provides internal visibility of the processing of transactions to selected interfaces. The controls for the trace buffers are a superset of the controls for the watchpoint monitor. Close inspection of the trace buffer control registers (`TBCRn`) and the watchpoint monitor control registers (`WMCRn`) shows that the trace buffer controls that are not needed for the watchpoint monitor are marked reserved in `WMCRn`. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer controls provide great flexibility with respect to when to start tracing, when to trace, 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 when to stop tracing: on an event or when the buffer is full. Finally, with `TBCR0[IFSEL]` the user can select which interface to capture information on.

The trace buffer can be programmed to trace the dispatch bus from the coherency module, outbound host interface to the RapidIO controller, the outbound host interface to the PCI controller, or the host interface to the DDR controller. Transactions come into the e500 coherency module (ECM), arbitrate for common resources, and get dispatched to the target port. Information such as cache hits, local processor retries, and target port busy conditions can be captured in the trace buffer. Similarly, transaction types, source ID, and other attributes can be captured in any of the other selected interfaces. On the RapidIO interface, where one cannot tell whether the request is due to an instruction fetch or a data fetch, the trace buffer can be used to capture this type of information.

20.4.5.1 Traced Data (`TBCR[TRSEL]`)

The following tables describe the format of the information saved in the trace buffer entries based on the selected interface.

**Table 20-26. Buffer Entry Definition for TBCR[TRSEL] = 0b000
(Coherency Module Dispatch)**

Bits	Name	Function
0–4		Transaction type. The decoded values of this field map to the transactions specified in Table 20-8 on page 20-15. For example, a value of 0b0000 maps to “Write with local processor snoop” and so on.
5–9		Source ID. This identifies the source of the transaction. The decoded values of this field map to the source IDs specified in Table 20-25 on page 20-28. For example, a value of 0b01100 maps to a Source ID of RapidIO and so on.
10–13		Target ID. This identifies the target of the transaction. The decoded values of this field map to the target IDs specified in Table 20-25 on page 20-28 for the first 16 decodes (All targets are contained within the first 16 decodes). For example, a value of 0b1100 maps to a target ID of RapidIO and so on.
14–18		Byte count. A value of 0 indicates 32 bytes. 00000byte count = 32 00001byte count = 31
19–31		Reserved
32–63		Address bits 0–31.

Table 20-27. Buffer Entry Definition for TBCR[TRSEL] = 0b001 (DDR SDRAM)

Bits	Name	Function
0–4		Transaction type. The decoded values of this field map to the transactions specified in Table 20-9 on page 20-16. For example, a value of 0b0000 maps to “Write” and so on.
5–9		Source ID. This identifies the source of the transaction. The decoded values of this field map to the source IDs specified in Table 20-25 on page 20-28. For example, a value of 0b01100 maps to a Source ID of RapidIO and so on.
10–13		Reserved
14–18		Byte count
19–31		Reserved
32–63		Address bits 0–31

Table 20-28. Buffer Entry Definition for TBCR[TRSEL] = 0b010 (PCI)

Bits	Name	Function
0–4		Transaction type. The decoded values of this field map to the transactions specified in Table 20-10 on page 20-16. For example, a value of 0b0000 maps to “Write” and so on.
5–9		Source ID. This identifies the source of the transaction. The decoded values of this field map to the source IDs specified in Table 20-25 on page 20-28. For example, a value of 0b01100 maps to a Source ID of RapidIO and so on.
10–11		Size. 00 32 bytes 01 8 bytes 10 16 bytes 11 24 bytes

Table 20-28. Buffer Entry Definition for TBCR[TRSEL] = 0b010 (PCI)

Bits	Name	Function
12–31		Reserved
32–63		Address bits 0–31

Table 20-29. Buffer Entry Definition for TBCR[TRSEL] = 0b011 (RapidIO)

Bits	Name	Function
0–4		Transaction type. The decoded values of this field map to the transactions specified in Table 20-11 on page 20-17. For example, a value of 0b0000 maps to "NWRITE" and so on.
5–9		Source ID. This identifies the source of the transaction. The decoded values of this field map to the source IDs specified in Table 20-25 on page 20-28. For example, a value of 0b01100 maps to a Source ID of RapidIO and so on.
10–13		Reserved
14–18		Double word count. This number times 8 equals the byte count.
19–32		Reserved
33–63		Address bits 0–31 which represents a double word address on RapidIO

20.5 Initialization

The watchpoint and trace buffer have similar initialization requirements. The only difference is that the trace buffer adds to the watchpoint monitor functionality the capability to trace information in addition to triggering on an event. For either the watchpoint or trace buffer, first all required registers should be initialized and then the control register should be written with the corresponding enable bit set. Writing the enable bit to a value of 1 is what sets off all the events, otherwise, all other information is ignored.

Chapter 21

10/100 Ethernet Controller

21.1 Introduction

The MPC8540 provides a 10/100 Ethernet controller operating at 10 to 100 megabits per second (Mbps) as a device debug and maintenance port. The gigabit triple-speed Ethernet controller (TSEC) is separate from this 10/100 Ethernet controller, and it is defined and described in Chapter 13, “Three-Speed Ethernet Controller.”

The 10/100 Ethernet standard supports the IEEE 802.3 Ethernet frame format, backward compatibility for installed media, and the use of full or half duplex CSMA/CD.

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 3-1 depicts the typical Ethernet protocol stack and the relationship to the OSI model.

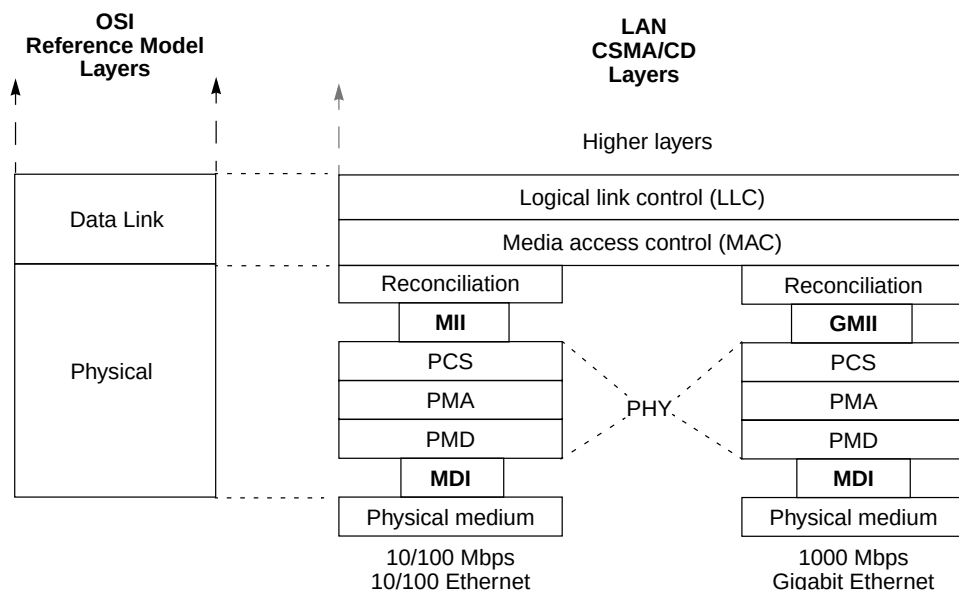


Figure 21-1. 10/100 Controller Location in the OSI Protocol Stack

Ethernet provides the following sublayers:

- Media access control (MAC)—The media access control sublayer provides a logical connection between the MAC and its peer station. Its main 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)—MII 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 coding and encoding data stream to and from the MAC layer. For fiber medium (1000BASEX), 8B/10B coding is used. For unshielded twisted pair (UTP) medium (1000BASET), 8B1Q coding is used.
- PMA (physical medium attachment)—The PMA sublayer is responsible for serializing code groups into bit stream suitable for serial bit-oriented physical devices and vice versa (SERDES). Synchronization is also done for proper data decoding in this sublayer. For fiber medium (1000BASEX), the interface on the PHY side is a 1 bit 1250 MHz signal while on the other side it is a ten-bit interface (TBI) with 10 bits at 125 MHz.
- For UTP medium the PHY side interface consist of four pair of 62.5 MHz PAM5 encoded signals while the other side provides the 1250 Mbps input to a 8B1Q4 PCS.
- PMD (physical medium dependent) 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) MDI is a connector. It defines different connector types for different physical media and PMD devices.

Figure 21-2 describes the 10/100 controller physical interface standard.

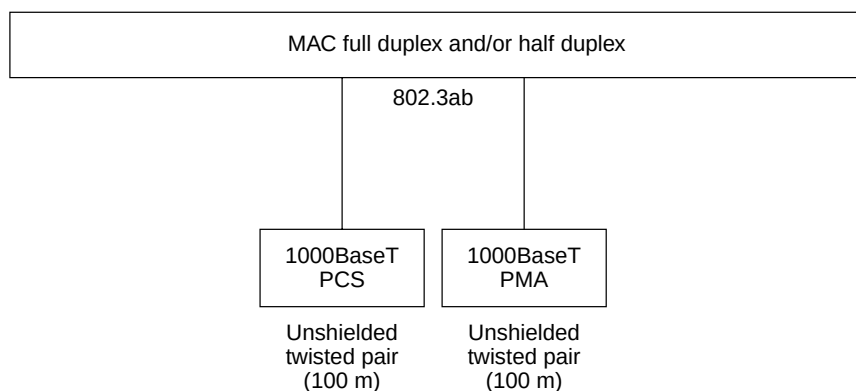


Figure 21-2. 802.3ab Physical Standard

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 21-3. The term packet is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

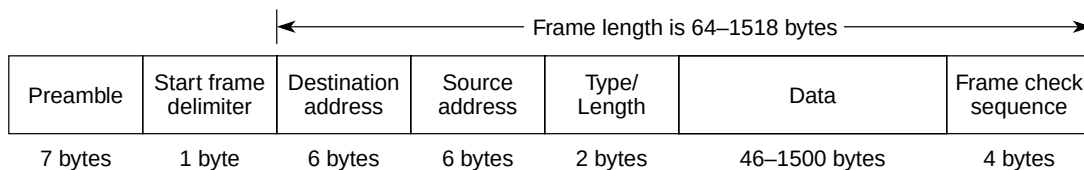
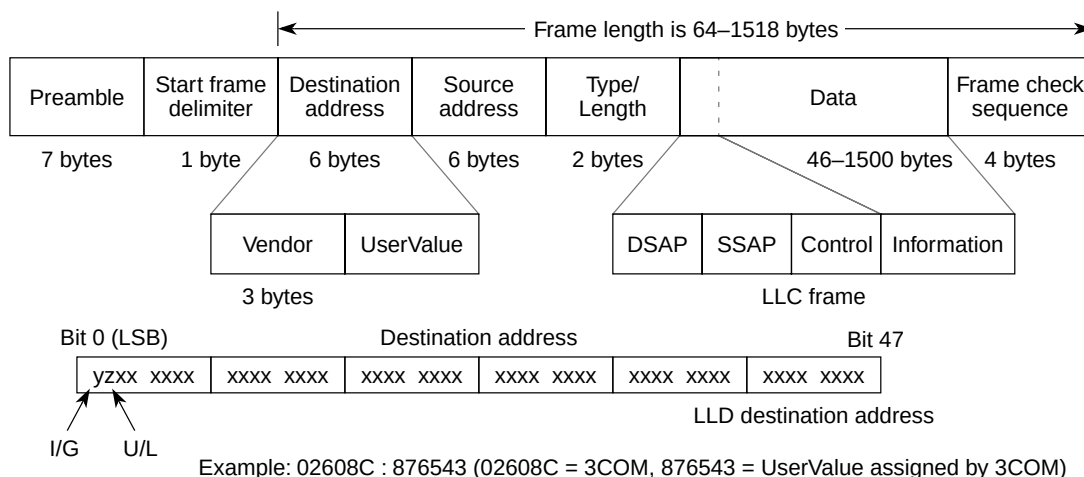


Figure 21-3. Ethernet/IEEE 802.3 Frame Structure

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 since 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 (1) or globally (0) defined.
- **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 used in the rest of the frame, such as TCP/IP. The length field specifies the length of the data portion of the frame. For 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 to 1536 (0x0600 through 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 1,536 was intentionally left undefined.
- **Data and padding.** Padding is optional. It is only needed when the data is smaller than 46 octets (1 octet = 1 bytes) to ensure the minimum frame size of 64 octets as specified in the standard. In 802.3x the first 2 octets of the data field are used as Opcode (OP) (pause = x0001) and the second 2 octets are used to transmit a pause time (PT) parameter (pausetime = x0000 for on and xFFFF for off). Since the use of these fields varies with the protocol used, the ability to examine them and report their content can significantly accelerate the processing of ethernet frames.
- **Frame-check sequence (FCS).** Contains the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD and CRC.



Note: The LSB of each octet is transmitted first.

Figure 21-4. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 21-4, 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 (first the preamble and last the FCS), and the bits of each octet are transmitted least significant bit (lsb) first. The destination address example shown in Figure 21-4 (02608C:876543) which would normally be written as:

0000 0010 0110 0000 1000 1100 1000 0111 0110 0101 0100 0011

will look like:

0100 0000 0000-0110 0011-0001 1110 0001 1010 0110 1100 0010

which is an individual address (because the lsb = 0) but locally defined (because the second least significant bit is 1).

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 left to 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 so on. 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. The LLC layer uses three variables to determine access into the upper layers via the LLC-PDU.

The variables include the destination service access point (DSAP), 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. Unfortunately, many protocols do not obey the rules for those layers. Therefore, 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, or SNAP frame. A SNAP encapsulation is indicated by the DSAP and SSAP addresses being set to “0xAA” and the LLC Control field equal to 0x03. When that address is seen, a SNAP header follows. The SNAP header is 5 bytes long. The first 3 bytes consist of the organization code (SNAP OUI), which is assigned by the IEEE. The second 2 bytes are the *type* value set from the original Ethernet specifications, if SNAP OUI = 00-00-00, or represent a SNAP Protocol Identifier when SNAP OUI is non zero.

The 10/100 controller allows the flexibility to accelerate the identification and retrieval of all of the standard and non-standard protocols mentioned above. Figure 21-5 shows the block diagram of the 10/100 controller.

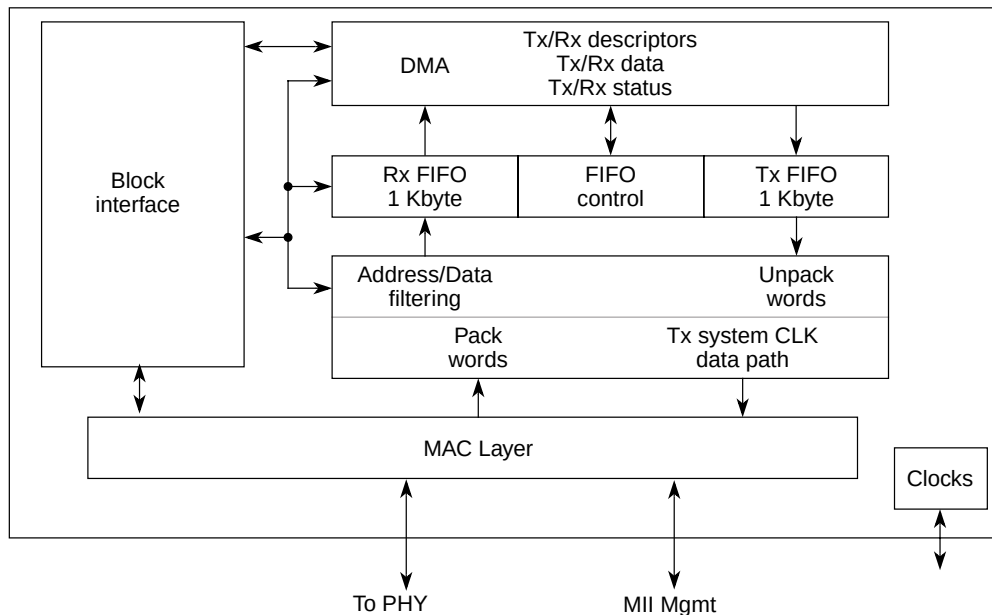


Figure 21-5. 10/100 ENET Controller Block Diagram

21.1.1 10/100 Ethernet Controller Overview

The 10/100 controller is designed to support 10/100 Mbps Ethernet/802.3 networks. It contains the Ethernet media access controller (MAC) functions, a first in first out (FIFO) controller, and a direct memory access (DMA) controller. Big-endian byte ordering format is used in accessing buffer data from memory. As data is sent onto the serial line (Ethernet

data bus) from the data buffer, the MSB of the buffer word contains data to be sent earlier than the LSB of the same buffer word.

The complete 10/100 controller is designed for single MAC applications. The 10/100 controller supports the 10/100 Mbps MII interface to connect to an external Ethernet transceiver.

The 10/100 controller software programming model is similar to the MPC8260 (PowerQUICC II) device, enabling Motorola customers to leverage already implemented Ethernet drivers, reducing software development cycle.

21.2 10/100 Ethernet Controller Features

The MPC8450 10/100 controller includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3ac compliant
- Support for different Ethernet physical interfaces:
 - 10/100 Mbps IEEE 802.3 MII
 - 10 Mbps IEEE 802.3 MII
- Full and half-duplex support
- IEEE 802.3 full duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
- Support of out-of-sequence transmit queue (for initiating flow-control)
- Programmable maximum frame length supports jumbo frames and IEEE 802.1 virtual local area network (VLAN) tags and priority
- Retransmission from transmit FIFO following a collision
- Support CRC generation and verification of inbound/outbound packets
- Address recognition
 - Each exact match can be programmed to be accepted or rejected
 - 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 processor's L2 cache to reduce access latency

21.3 Modes of Operation

The 10/100 controller's primary operational modes are the following:

- Full and half duplex operations

This is determined by the 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.

When configured in half-duplex mode (only 10 and 100 Mbps operations, MACCFG2[Full_Duplex] = 0), the MAC complies with the IEEE CSMA/CD access method.

When configured in full duplex mode (MACCFG2[Full_Duplex] = 1), the MAC supports flow control. When flow control is enabled, it allows the MAC to receive or send PAUSE frames.

- 10 Mbps and 100 Mbps MII interface operations

The MAC-PHY interface operates in MII mode by setting 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 TX_CLK and RX_CLK pins which are driven by the transceiver. The transceiver either auto negotiates the speed or it may be controlled by software via the serial management interface (EC_MDC/EC_MDIO pins) to the transceiver.
- Address recognition options
The options supported are promiscuous, broadcast, exact individual address hash or exact match and multicast hash match. Refer to Section 21.5.3.5.1, "Individual Address Registers 0–7 (IADDRn)," Section 21.5.3.5.2, "Group Address Registers 0–7 (GADDRn)," Section 21.5.3.2.1, "Receive Control Register (RCTRL)," and Section 21.6.2.4, "Frame Recognition," for a detailed description.
- Internal loop back
Internal loop back mode is selected via the loop back bit in the MACCFG1 register. See Section 21.6.2.8, "Internal and External Loop Back" for more details on loop back.

21.4 External Signals Description

This section describes the 10/100 Ethernet controller interface signals.

21.4.1 Signals Overview

The buses in this chapter are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention (which is TxD[3:0] with 0 as the least significant bit). Note that except for external physical interfaces, the buses and registers follow a big-endian format.

External Signals Description

The 10/100 controller network interface supports the MII option that requires 18 I/O pins (including the EC_MDIO and EC_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.

Table 21-1 summarizes the network interface signals.

Table 21-1. Signal Properties

10/100 Controller I/O Signal Names	Function	Input/Output	Reset State	Pull-up
FEC_COL	MII—collision, input	I		
FEC_CRS	MII—carrier sense, input	I		
EC_GTX_CLK125	Gigabit Reference Clock— input	I		
EC_MDC	MII—management clock, output	O	0	
EC_MDIO	MII—management data, bidirectional	I/O	Hi-Z (input)	
FEC_RX_CLK	MII— receive clock, input	I		
FEC_RX_DV	MII—receive data valid, input	I		
FEC_RXD[0:3]	MII—receive data bit 3:0 input	I		
FEC_RX_ER	MII—receive error, input	I	0	
FEC_TX_CLK	MII—transmit clock, input	I		
FEC_TXD[0:3]	MII—transmit data bit 3:0, output	O		
FEC_TX_EN	MII—transmit data valid, output	O	0	
FEC_TX_ER	MII—transmit error, output	O	0	

21.4.2 Detailed Signal Descriptions

This section provides a detailed description of the 10/100 controller interface signals. The modes follow the IEEE Std. 802.3, 2000 Edition. The signals are described in detail in Table 21-5.

Table 21-2. FEC Signals—Detailed Signal Descriptions

Signal	I/O	Description
FEC_COL	I	Collision. The behavior of this signal is not specified when in full duplex mode.
		State Meaning Asserted/Negated—This signal is asserted upon detection of a collision, and must remain asserted while the collision persists.
		Timing Assertion/Negation—This signal is not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.

Table 21-2. FEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
FEC_CRD	I	Carrier sense.	
		State Meaning	Asserted/Negated—This signal is asserted when the transmit or receive medium is not idle. In the event of a collision, FEC_CRD remains asserted through the duration of the collision.
		Timing	Assertion/Negation—This signal is not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.
EC_GTX_CLK125	I	Gigabit reference clock	
		State Meaning	Asserted/Negated—The gigabit reference clock
		Timing	Assertion/Negation—See the <i>MPC8540 Hardware Specifications</i>
EC_MDC	O	Management data clock	
		State Meaning	Asserted/Negated—Represents the management clock.
		Timing	Assertion/Negation—This is a MAC supplied clock, typically 2.5 MHz(IEEE sets a minimum period of 400 ns or 2.5 MHz), but the device can be configured up to 12.5 MHz and if the PHY would support it, MII could operate at that speed.
EC_MDIO	I/O	Management Data Input/Output, BIDI	
		State Meaning	Asserted/Negated—This is a bi-directional signal to latch PHY-supplied status during management read cycles and drive control during MII management write cycles.
		Timing	Assertion/Negation—
FEC_RX_CLK	I	Receive clock	
		State Meaning	Asserted/Negated—Receive clock
		Timing	Assertion/Negation—This 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	
		State Meaning	Asserted—When this signal is asserted, the PHY is indicating that valid data is present on the MII interface. Negated—The PHY is not indicating that valid data is present.
		Timing	Assertion/Negation—
FEC_RXD[3:0]	I	Receive data	
		State Meaning	Asserted/Negated—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. When FEC_RX_DV is not asserted, FEC_RXD has no meaning.

Table 21-2. FEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
FEC_RX_ER	I	Receive error	
		State Meaning	Asserted—When FEC_RX_ER and FEC_RX_DV are asserted, the PHY has detected an error in the current frame. Negated—The PHY has not detected an error in the current frame.
		Timing	Assertion/Negation—
FEC_TX_CLK	I	Transmit clock. Continuous clock that provides a timing reference for the FEC_TX_EN, FEC_TXD, and FEC_TX_ER. FEC_TX_CLK is generated in the PHY and provided to the MAC.	
		State Meaning	Asserted/Negated—Transmit clock
		Timing	Assertion/Negation—This is a continuous clock with a frequency of 2.5 or 25 MHz
FEC_TXD[3:0]	O	Transmit data	
		State Meaning	Asserted/Negated—Represents a nibble of data to be sent from the MAC to the PHY when FEC_TX_EN is asserted; has no meaning when FEC_TX_EN is negated.
FEC_TX_EN	O	Transmit data valid	
		State Meaning	Asserted—The MAC is indicating that valid data is present on the MII. Negated—Valid data is not guaranteed to be present on the MII
		Timing	Assertion/Negation—
FEC_TX_ER	O	Transmit error	
		State Meaning	Asserted/Negated—Assertion of this signal 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 affect when operating at 10Mbps or when FEC_TX_EN is negated.
		Timing	Assertion/Negation—This signal transitions synchronously with respect to FEC_TX_CLK.

21.5 Memory Map/Register Definition

The 10/100 controller uses a software model similar to that employed by the fast Ethernet function supported on the Motorola MPC8260 CPM FCC and in the 10/100 controller of the MPC860T.

The 10/100 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 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. Buffer descriptors are described in Section 21.6.3, “Buffer Descriptors.”

The MII registers are described in this section.

21.5.1 Top-Level Module Memory Map

The 10/100 controller implementation requires 4 Kbytes of memory-mapped space, of which more than 1Kbyte 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 21-3 defines the top-level memory map.

Table 21-3. Module Memory Map Summary

Address	Function
000–0FF	10/100 controller general control/status registers
100–2FF	10/100 controller transmit control/status registers
300–4FF	10/100 controller receive control/status registers
500–5FF	MAC registers
600–7FF	
800–8FF	HASH function registers
900–FFF	Future expansion space

21.5.2 Detailed Memory Map

Table 21-4 shows the address of each register, the name of the register, and a cross-reference to the complete description of the register.

Table 21-4. 10/100 Ethernet Module Memory Map

Offset	Name	Access ¹	Reset	Section/Page
10/100 Ethernet Controller Registers				21.5.3/21-15
10/100 Ethernet Controller General Control and Status Registers				21.5.3.1/21-15
0x2_6000–0x2_600C	Reserved	R	0x0000_0000	—
0x2_6010	Interrupt event register (IEVENT)	R/W	0x0000_0000	21.5.3.1.1/21-15

Table 21-4. 10/100 Ethernet Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_6014	Interrupt mask register (IMASK)	R/W	0x0000_0000	21.5.3.1.2/21-17
0x2_6018	Error disabled register (EDIS)	R/W	0x0000_0000	21.5.3.1.3/21-18
0x2_601C– 0x2_6020	Reserved	R	0x0000_0000	—
0x2_6024	Minimum frame length register (MINFLR)	R/W	0x0000_0040	21.5.3.1.4/21-19
0x2_6028	Pause time value register (PTV)	R/W	0x0000_0000	21.5.3.1.5/21-20
0x2_602C	DMA control register (DMACTRL)	R/W	0x0000_0000	21.5.3.1.6/21-20
0x2_6030– 0x2_603C	Reserved	R	0x0000_0000	
0x2_6040– 0x2_60FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Transmit Control and Status Registers				3.5.3.3/3-58
0x2_6100	Transmit control register (TCTRL)	R/W	0x0000_0000	21.5.3.1.7/21-22
0x2_6104	Transmit status register (TSTAT)	R/W	0x0000_0000	21.5.3.1.8/21-23
0x2_6108	Reserved	R	0x0000_0000	
0x2_610C	TxBD data length (TBDLEN)	R/W	0x0000_0000	21.5.3.1.9/21-24
0x2_6110– 0x2_6120	Reserved	R	0x0000_0000	
0x2_6124	Current TxBD pointer (CTBPTR)	R/W	0x0000_0000	21.5.3.1.10/21-24
0x2_6128– 0x2_6180	Reserved	R	0x0000_0000	
0x2_6184	TxBD pointer (TBPTR)	R/W	0x0000_0000	21.5.3.1.11/21-25
0x2_6188– 0x2_6200	Reserved	R	0x0000_0000	
0x2_6204	TxBD base address (TBASE)	R/W	0x0000_0000	21.5.3.1.12/21-25
0x2_6208– 0x2_62AC	Reserved for more Tx base addresses	R	0x0000_0000	
0x2_62B0	Out-of-sequence TxBD register (OSTBD)	R/W	0x0000_0000	21.5.3.1.13/21-26
0x2_62B4	Out-of-sequence TxBD pointer register (OSTBDP)	R/W	0x0000_0000	21.5.3.1.14/21-28
0x2_62B8– 0x2_62FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Receive Control and Status Registers				3.5.3.4/3-72
0x2_6300	Receive control register (RCTRL)	R/W	0x0000_0000	21.5.3.2.1/21-29
0x2_6304	Receive status register (RSTAT)	R/W	0x0000_0000	21.5.3.2.2/21-30
0x2_6308	Reserved	R	0x0000_0000	
0x2_630C	RxBD data length (RBDLEN)	R/W	0x0000_0000	21.5.3.2.3/21-31

Table 21-4. 10/100 Ethernet Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_6310– 0x2_6320	Reserved	R	0x0000_0000	
0x2_6324	Current RxBD pointer (CRBPTR)	R/W	0x0000_0000	21.5.3.2.4/21-31
0x2_6328– 0x2_633C	Reserved	R	0x0000_0000	
0x2_6340	Maximum receive buffer length register (MRBLR)	R/W	0x0000_0000	21.5.3.2.5/21-32
0x2_6344	Reserved	R	0x0000_0000	
0x2_6348– 0x2_635C	Reserved for more Rx buffer length registers	R	0x0000_0000	
0x2_6360– 0x2_6380	Reserved	R	0x0000_0000	
0x2_6384	RxBD pointer (RBPTR)	R/W	0x0000_0000	21.5.3.2.6/21-33
0x2_6388– 0x2_639C	Reserved	R	0x0000_0000	
0x2_63A0– 0x2_63FC	Reserved for more Rx pointers	R	0x0000_0000	
0x2_6400	Reserved	R	0x0000_0000	
0x2_6404	RxBD base address (RBASE)	R/W	0x0000_0000	21.5.3.2.7/21-33
0x2_6408– 0x2_641C	Reserved	R/W	0x0000_0000	
0x2_6420– 0x2_64FC	Reserved for more Rx base addresses	R	0x0000_0000	
10/100 Ethernet Controller MAC registers				21.5.3.3/21-34
0x2_6500	MAC configuration #1 (MACCFGF1)	R/W, R	0x0000_0000	21.5.3.4.1/21-37
0x2_6504	MAC configuration #2 (MACCFGF2)	R/W	0x0000_7000	21.5.3.4.2/21-38
0x2_6508	IPGIFG	R/W	0x4060_5060	21.5.3.4.3/21-40
0x2_650C	Half-duplex (HAFDUP)	R/W	0x00A0_F037	21.5.3.4.4/21-41
0x2_6510	Maximum frame (MAXFRM)	R/W	0x0000_0600	21.5.3.4.5/21-42
0x2_6514	Reserved	R	0x0000_0000	
0x2_6518	Reserved	R	0x0000_0000	
0x2_651C	Reserved	R	0x0000_0000	
0x2_6520	MII management: configuration (MIIMCFG)	R/W	0x0000_0000	21.5.3.4.6/21-42
0x2_6524	MII management: command (MIIMCOM)	R/W	0x0000_0000	21.5.3.4.7/21-43
0x2_6528	MII management: address (MIIMADD)	R/W	0x0000_0000	21.5.3.4.8/21-44
0x2_652C	MII management: control (MIIMCON)	WO	0x0000_0000	21.5.3.4.9/21-44
0x2_6530	MII management: status (MIIMSTAT)	R	0x0000_0000	21.5.3.4.10/21-45

Table 21-4. 10/100 Ethernet Module Memory Map (continued)

Offset	Name	Access ¹	Reset	Section/Page
0x2_6534	MII management: indicators (MIIMIND)	R	0x0000_0000	21.5.3.4.11/21-46
0x2_6538	Reserved	R	0x0000_0000	
0x2_653C	Interface status (IFSTAT)	R,R/LH	0x0000_0001	21.5.3.4.12/21-46
0x2_6540	Station Address, part 1 (MACSTNADDR1)	R/W	0x0000_0000	21.5.3.4.13/21-47
0x2_6544	Station Address, part 2 (MACSTNADDR2)	R/W	0x0000_0000	21.5.3.4.14/21-47
0x2_6548– 0x2_673C	Reserved	R	0x0000_0000	
0x2_6740– 0x2_67FC	Reserved	R	0x0000_0000	
10/100 Ethernet Controller Hash Function Registers				3.5.3.7/3-125
0x2_6800	Individual address register 0 (IADDR0)	R/W	0x0000_0000	21.5.3.5.1/21-48
0x2_6804	Individual address register 1 (IADDR1)	R/W	0x0000_0000	
0x2_6808	Individual address register 2 (IADDR2)	R/W	0x0000_0000	
0x2_680C	Individual address register 3 (IADDR3)	R/W	0x0000_0000	
0x2_6810	Individual address register 4 (IADDR4)	R/W	0x0000_0000	
0x2_6814	Individual address register 5 (IADDR5)	R/W	0x0000_0000	
0x2_6818	Individual address register 6 (IADDR6)	R/W	0x0000_0000	
0x2_681C	Individual address register 7 (IADDR7)	R/W	0x0000_0000	
0x2_6820– 0x2_687C	Reserved	R	0x0000_0000	
0x2_6880	Group address register 0 (GADDR0)	R/W	0x0000_0000	21.5.3.5.2/21-49
0x2_6884	Group address register 1 (GADDR1)	R/W	0x0000_0000	
0x2_6888	Group address register 2 (GADDR2)	R/W	0x0000_0000	
0x2_688C	Group address register 3 (GADDR3)	R/W	0x0000_0000	
0x2_6890	Group address register 4 (GADDR4)	R/W	0x0000_0000	
0x2_6894	Group address register 5 (GADDR5)	R/W	0x0000_0000	
0x2_6898	Group address register 6 (GADDR6)	R/W	0x0000_0000	
0x2_689C	Group address register 7 (GADDR7)	R/W	0x0000_0000	
0x2_68A0– 0x2_6BF4	Reserved	R	0x0000_0000	
0x2_6BF8	Attribute register (ATTR)	R	0x0000_0000	21.5.3.8.6/21-134
0x2_6BFC	Attribute EL & EI register (ATTRELI)	R	0x0000_0000	21.5.3.8.7/21-136
10/100 Ethernet Controller Future Expansion Space				
0x2_6C00– 0x2_6FFC	Reserved	R	0x0000_0000	

¹ R = Read Only, WO = Write Only, R/W = Read and Write, LH = Latches High, SC = Self-clearing

21.5.3 Memory Mapped Register Descriptions

This section provides a detail description of all the 10/100 controller registers. Because all of the 10/100 controller registers are 32-bits wide, only 32-bit register accesses are supported.

21.5.3.1 10/100 Controller 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.

21.5.3.1.1 Interrupt Event Register (IEVENT)

When an event occurs that sets a bit in the interrupt event register, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. The bit in the interrupt event register is cleared if a one is written to that bit position. A write of zero 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 MII.

Interrupts resulting from errors/problems detected in the network or transceiver are:

- BABR, BABT, LATE_COL and COL_RETRY_LIM.

Interrupts resulting from internal errors are:

- EBERR, XFIFO_UN and BSY.

The IEVENT register is shown in Figure 21-6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	0	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	0	LC	CRL	XFUN
W																
Reset	0000_0000_0000_0000															
	16	17					22	23	24	25						31
R	RXB	0	0	0	0	0	0	GRSC	RXF	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x2_6010															

Figure 21-6. IEVENT Register Definition

Table 21-5. IEVENT Field Descriptions

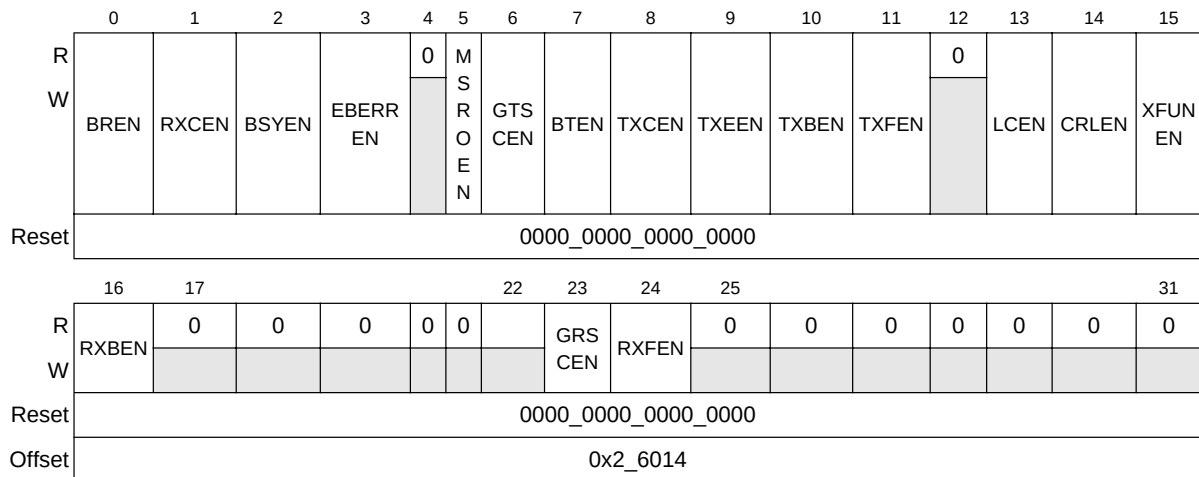
Bits	Name	Description
0	BABR	Babbling receive erro. This bit indicates a frame was received with length in excess of the MAC's maximum frame length register and MAC_CONF2[HFE] equals 1.
1	RXC	Receive control interrupt . A control frame has been received (MAC_CONF1[RFCE] must be set). As soon as the transmitter finishes sending the current frame, a pause operation is performed.
2	BSY	Busy condition interrupt. Set when a frame is received and discarded due to a lack of buffers.
3	EBERR	Ethernet bus error. This bit indicates that a system bus error occurred when a DMA transaction was underway. When the EBERR is set while transmission is in progress, the DMA will stop sending data to the Tx FIFO which eventually will cause an underrun error (XFUN). When the EBERR is set while receiving a frame, the DMA will discard the frame.
4	—	Reserved
5	MSRO	MSTAT register overflow . This interrupt will be asserted when the count for one of the MSTAT registers has exceeded its size.
6	GTSC	Graceful transmit stop complete. This interrupt will be 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 the setting of the DMACTRL[GTS] bit, is now complete. • A graceful stop, which was initiated by the setting of the TCTRL[TFC_PAUSE] bit is now complete.
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. This condition is usually caused by a frame that is too long being placed into the transmit data buffer(s). Truncation occurs if MACCFG2[HFE] equals 0, no truncation otherwise.
8	TXC	Transmit control interrupt. This bit indicates that a control frame has been transmitted.
9	TXE	Transmit error. An error occurred on the transmitted channel that has caused TSTAT[THLT] to be set by 10/100 controller. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL, XFUN).
10	TXB	Transmit buffe. This bit indicates that a transmit buffer descriptor, that had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame has been updated.
11	TXF	Transmit frame interrupt. This bit indicates that a frame has been transmitted and that the last corresponding Transmit Buffer Descriptor (TxBD) has been updated. Only occurs if the I (Interrupt) bit in the status word of the Buffer Descriptor is set.
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.
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. The frame is discarded without being transmitted and transmission of the next frame will commence. This only occurs while in half duplex mode.
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted.
16	RXB	Receive buffer. This bit indicates that a receive buffer descriptor, which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame, has been updated.

Table 21-5. IEVENT Field Descriptions (continued)

Bits	Name	Description
17–22	—	Reserved
23	GRSC	Graceful receive stop complete. This interrupt will be asserted when a graceful receive stop is completed. It allows the user to know when the system has completed the stop and it is safe to write to receive registers (i.e. status, control or configuration registers) that are used by the system during normal operation.
24	RXF	Receive frame interrupt. This bit indicates that a frame has been received and the last receive buffer descriptor (RxBD) in that frame, has been updated. This will occur only if the I (interrupt) bit in the buffer descriptor status word was set.
25–31	—	Reserved

21.5.3.1.2 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 will be generated. The interrupt signal will remain asserted until a 1 is written to the IEVENT bit (write 1 to clear) or a 0 is written to the IMASK bit. The IMASK register is shown in Figure 21-7.

**Figure 21-7. IMASK Register Definition****Table 21-6. 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

Table 21-6. IMASK Field Descriptions (continued)

Bits	Name	Description
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	CRLEN	Collision retry limit enable
15	XFUNEN	Transmit fifo underrun enable
16	RXBEN	Receive buffer interrupt enable
17–22	—	Reserved
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

21.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register (EDIS) allows the user to disable an error interruption. EDIS is shown in Figure 21-8.

	0	1	2	3	4	8	9	10	12	13	14	15
R	0	0			0	0	0	0		0	0	0
W			BSYDIS	EBERRDIS					TXEDIS			
Reset	0000_0000_0000_0000											
	16											31
R	0	0	0	0	0	0	0	0	0	0	0	0
W												
Reset	0000_0000_0000_0000											
Offset	0x2_6018											

Figure 21-8. EDIS Register Definition

Table 21-7. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disabled
3	EBERRDIS	Ethernet controller bus error disabled
4–8	—	Reserved
9	TXEDIS	Transmit error disable
10–12	—	Reserved
13	LCDIS	Late collision disabled
14	CRLDIS	Collision retry limit disable
15	XFUNDIS	Transmit fifo underrun disable
16–31	—	Reserved

21.5.3.1.4 Minimum Frame Length Register (MINFLR)

MINFLR, shown in Figure 21-9, is written by the user. It tells the 10/100 controller the smallest packet to accept and place in a receive buffer pointed to by the RxBD.

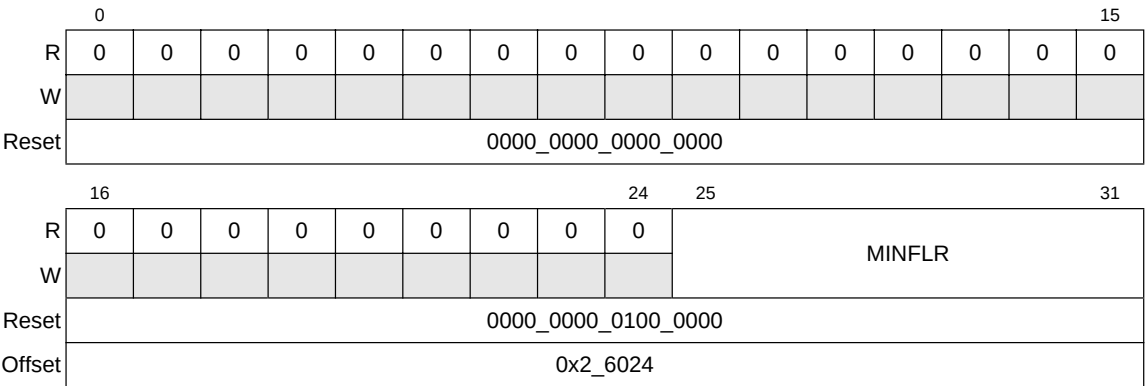


Figure 21-9. MINFLR Register Definition

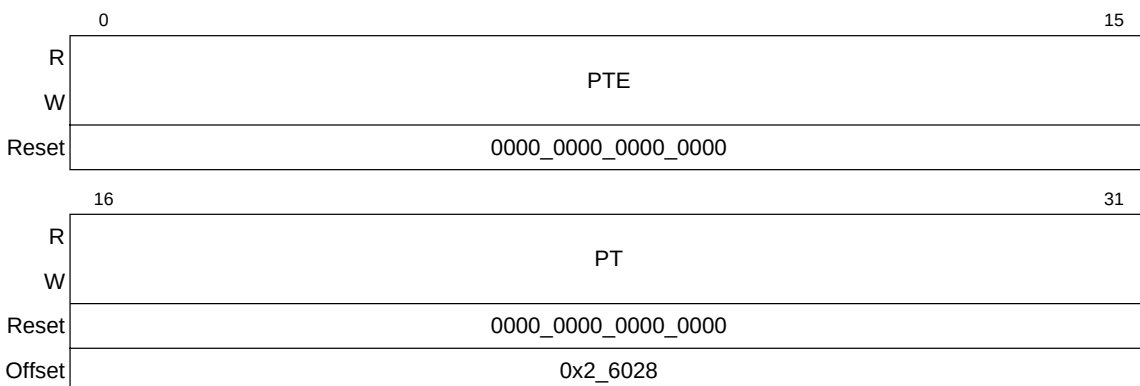
Table 21-8 describes the fields of the MINFLR register.

Table 21-8. 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 RxB[SH] (frame too short) is set in the last RxB. The largest allowable value for MINFLR is 64. Unlike the MPC8260 where PADS are added to make the transmit frame equal to MINFLR bytes, When padding is requested, 10/100 controller always PADS transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

21.5.3.1.5 Pause Time Value Register (PTV)

PTV is a 32 bit register written by the user to store the pause duration. The low-order 16 bits (PT) represent the pause value and the high-order 16 bits (PTE) represent the extended pause value. PTV is shown in Figure 21-10.

**Figure 21-10. PTV Register Definition****Table 21-9. PTV Field Descriptions**

Bits	Name	Description
0–15	PTE	Extended Pause time value
16–31	PT	Pause time value

21.5.3.1.6 DMA Control Register (DMACTRL)

DMACTRL, shown in Figure 21-11, is writable by the user to configure the DMA block.

	0																15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																	
Reset	0000_0000_0000_0000																

	16							23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	TDSSEN	TBDSSEN	0	GRS	GTS	TOD	WWR	WOP
W																
Reset	0000_0000_0000_0000															
Offset	0x2_602C															

Figure 21-11. DMACTRL Register Definition

Table 21-10 describes the fields of the DMACTRL register.

Table 21-10. DMACTRL Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TDSSEN	Tx Data snoop enable. 0 Disables snooping of all Transmit frames from memory. 1 Enables snooping of all Transmit frames from memory.
25	TBDSSEN	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. When this bit is set, the Ethernet controller will stop receiving frames after the frame that is currently being received is completed (i.e. a valid end of frame was received). The buffer of the receive frame associated with the EOF will be closed and the IEVENT[GRSC] will be asserted. Because the receive enable bit of the MAC may still be set, the MAC may continue to receive but the 10/100 controller ignores the Receive data until the GRS bit is cleared. When this bit is cleared the 10/100 controller, 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. When GRS is set the user needs to 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 the IEVENT[GRSC] bit and can write to receive registers that are accessible to both user and the 10/100 controller hardware without fear of conflict.
28	GTS	Graceful transmit stop. When this bit is set, the Ethernet controller will stop transmission after any frame that is currently being transmitted is completed and the GTSC interrupt in the IEVENT register will be asserted. If frame transmission is not currently underway, the GTSC interrupt will be asserted immediately. Once transmission has completed, a “restart” can be accomplished by clearing the GTS bit.
29	TOD	Transmit on demand. When this bit is asserted, the 10/100 controller gives high priority to the current TxBD and begins sending the frame without waiting the normal polling time to check the TxBD's R bit. TOD is cleared after transmission begins but TOD mode continues forever or until an unprepared BD (TxBD[R]=0) is reached or DMACTRL[GTS]=1 or TSTAT[THLT]=1. New TxBDs are processed in order. This bit is ignored if DMACTRL[WOP] =1.

Table 21-10. DMACTRL Field Descriptions (continued)

Bits	Name	Description
30	WWR	Write with response. This bit gives the user the assurance that a BD has been 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, RXBx, RxFx, 10/100 controller waits for acknowledgement from system that the transmit or receive BD being updated has been stored in memory.
31	WOP	Wait or poll. This bit, which is applicable only to the transmitter, is intended to give the user the option to have the 10/100 controller periodically poll TxBD or wait for software to tell 10/100 controller to fetch a buffer descriptor. While operating in the "Wait" mode, the 10/100 controller will allow two additional reads of a descriptor which is not ready before going into a halt state. No interrupt is driven. To resume transmission, software must clear the TSTAT[THLT] bit. 0 Poll TxBD every 512 clocks. 1 Beyond two re-reads of TxBD, do not poll but wait for a write to TSTAT[THLT].

21.5.3.1.7 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block.

	0															15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																	
Reset	0000_0000_0000_0000																
	16			19	20	21					26	27		28	29	31	
R	0	0	0	0	THDF	0	0	0	0	0	0	RFC_PAUSE		TFC_PAUSE	0	0	0
W																	
Reset	0000_0000_0000_0000																
Offset	0x2_6100																

Figure 21-12. TCTRL Register Definition**Table 21-11. TCTRL Field Descriptions**

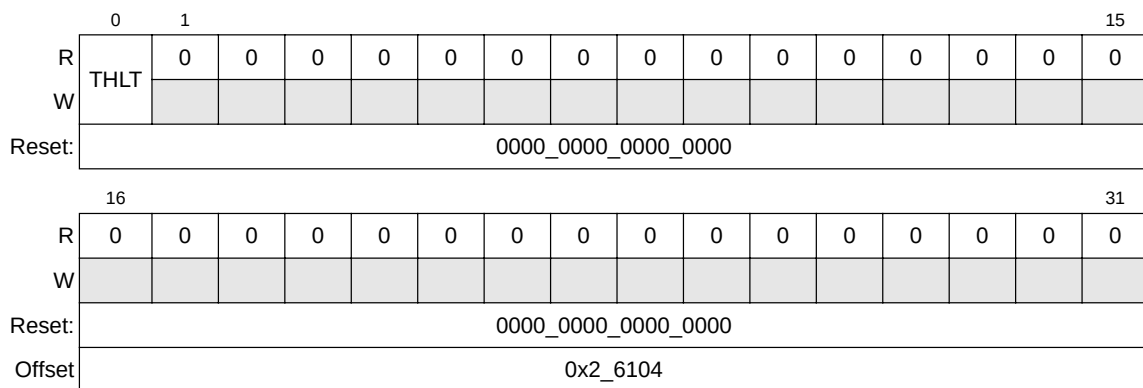
Bits	Name	Description
0–19	—	Reserved
20	THDF	Transmit Half-Duplex Flow Control (written by user) It is not self resetting. 0 Disable back pressure. 1 Back pressure should be applied to media.
21–26	—	Reserved

Table 21-11. TCTRL Field Descriptions (continued)

Bits	Name	Description
27	RFC_PAUSE	Receive flow control pause frame (written by 10/100 controller)—This read only status bit will be asserted when a flow control pause frame has been received and the transmitter is paused for the duration defined in the received pause frame This bit will automatically clear when the pause duration is complete.
28	TFC_PAUSE	Transmit flow control pause frame—Assert to transmit a PAUSE frame. When this bit is set, the MAC will stop transmission of data frames after the current transmission is complete. At this time, the GTSC interrupt in the IEVENT register will be asserted. With transmission of data frames stopped, the MAC will transmit 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 will clear the TFC_PAUSE bit and resume 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.
29–31	—	Reserved

21.5.3.1.8 Transmit Status Register (TSTAT)

This register is read/write and is written by 10/100 controller to convey DMA status information.

**Figure 21-13. TSTAT Register Definition****Table 21-12. TSTAT Field Descriptions**

Bit	Name	Description
0	THLT	Transmit Halt. 0 No hardware initiated transmission halt. 1 10/100 controller Transmission function halted. It is written by 10/100 controller to inform the user, it is no longer processing transmit frames and 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

21.5.3.1.9 TxBD Data Length Register (TBDLEN)

TBDLEN, shown in Figure 21-14, is a DMA register that contains the number of bytes remaining in the current transmit buffer.

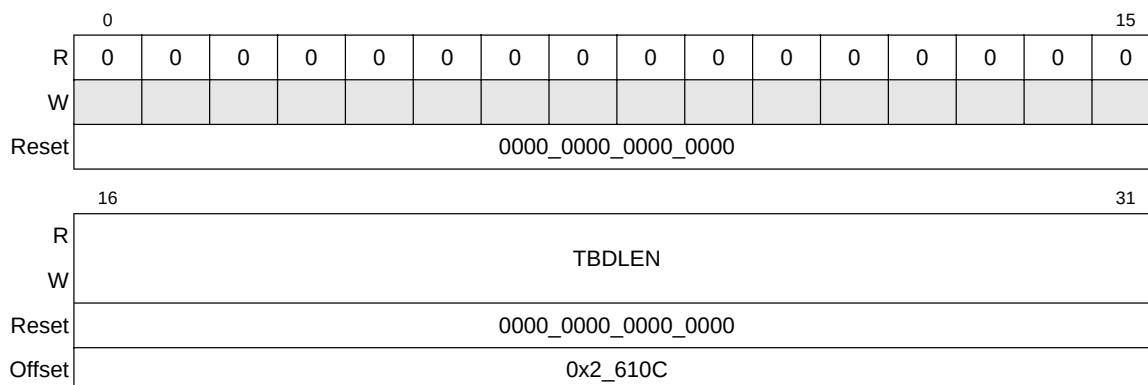


Figure 21-14. TBDLEN Register Definition

Table 21-13 describes the fields of the TBDLEN register.

Table 21-13. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	The TBDLEN is internally written by the DMA module. The transmit channel will stay active until TBDLEN is 0.

21.5.3.1.10 Current Transmit Buffer Descriptor Pointer (CTBPTR)

CTBPTR, shown in Figure 21-15, contains the address of the transmit buffer descriptor currently being worked on, or last worked on.

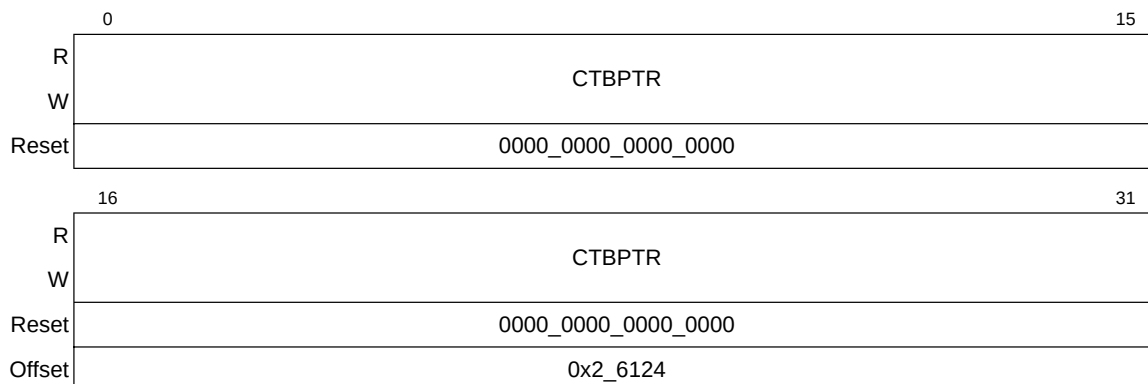


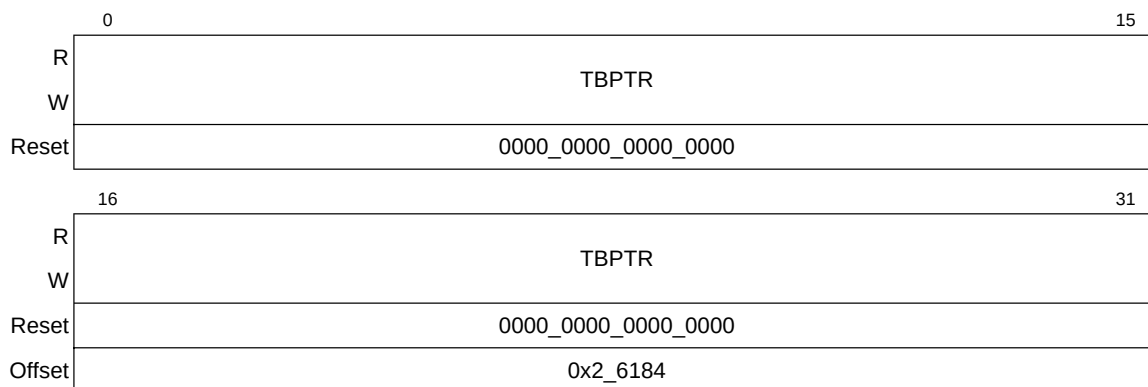
Figure 21-15. CTBPTR Register Definition

Table 21-14. CTBPTR Field Descriptions

Bits	Name	Description
0–31	CTBPTR	The CTBPTR register is internally written by the DMA module. The value will increment by eight (bytes) each time a descriptor is read from memory.

21.5.3.1.11 Transmit Buffer Descriptor Pointer (TBPTR)

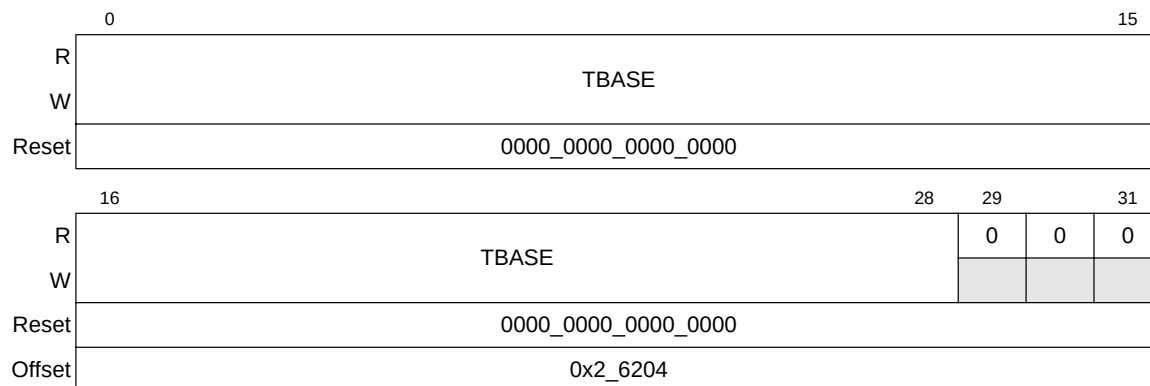
TBPTR, shown in Figure 21-16, contains the next transmit buffer descriptor address.

**Figure 21-16. TBPTR Register Definition****Table 21-15. TBPTR Field Descriptions**

Bit	Name	Description
0–31	TBPTR	The TBPTR register is internally written by the DMA module. The value will increment by eight (bytes) each time a descriptor is read from memory.

21.5.3.1.12 Transmit Descriptor Base Address (TBASE)

The TBASE register, shown in Figure 21-17, is written by the user with the TxBD base address; must be divisible by eight for the 8-byte data buffer descriptors.

**Figure 21-17. TBASE Register Definition****Table 21-16. 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 FEC transmit function.
29–31	—	Reserved

21.5.3.1.13 Out-of-Sequence TxBD Register (OSTBD)

OSTBD is the out-of-sequence TxBD register. This register includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. The OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. The user must set OSTBD[L] when preparing this BD. When a flow control frame is sent and OSTBD[I] is set, a TXC event is generated after frame transmission. This area should be cleared when not in use. Once the 10/100 controller is in paused mode the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC treats it like a regular TxBD.

OSTBD is shown in Figure 21-18.

	0	1	2	3	4	5	6	7	8	9	10	13	14	15	
R	R	PAD	W	I	L	TC	DEF	0	LC	RL	RC		UN	0	
W															
Reset	0000_0000_0000_0000														
	16														31
R	OSTBDLEN														
W															
Reset	0000_0000_0000_0000														
Offset	0x2_62B0														

Figure 21-18. OSTBD Register Definition

The out of sequence TxBD fields are described in detail below.

Table 21-17. OSTBD Field Descriptions

Bits	Name	Description
0	R	Ready. Written by both the 10/100 controller and the 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 10/100 controller clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
1	PAD	Padding for short frames. Valid only when one TxBD is used (L = 1) and MACCFG2 register's PAD/CRC Enable bit is zero, and it's CRC Enable bit is zero; otherwise pads are added to short frames 0 Do not add PADs to short frames unless TxBD[TC] is set. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which PADs up to MINFLR value, 10/100 controller PADs always up to the IEEE minimum frame length of 64 bytes.
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[TXF] is set after this buffer is serviced. Setting this bit can cause an interrupt if it is enabled (IEVENT[TXFEN]).
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.
5	TC	Tx CRC. Written by use. Valid only when one TxBD is used (L = 1), TxBD[PAD] is zero, the MACCFG2 register's PAD/CRC enable bit is zero, and it's CRC enable bit is zero; otherwise, a CRC will be added to all frames. 0 End transmission immediately after the last data byte, unless TxBD[PAD] is set. 1 Transmit the CRC sequence after the last data byte.

Table 21-17. OSTBD Field Descriptions (continued)

Bits	Name	Description
6	DEF	Hardware updates this bit after transmitting a frame when used as a “Defer indicator”. Software/user updates this bit while building a transmit buffer descriptor when used as a “Hardware Event indicator”. DEF—Defer indication, written by hardware. 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	LC	Late collision. Written by 10/100 controller. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The 10/100 controller terminates the transmission and updates LC.
9	RL	Retransmission limit. Written by 10/100 controller. 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 10/100 controller terminates the transmission and updates RL.
10–13	RC	Retry count. Written by 10/100 controller. 0 The frame is sent correctly the first time. 1 More than zero attempts were needed to send the transmit frame. When this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. Written by 10/100 controller. 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 10/100 controller 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 10/100 controller should transmit from this BD's data buffer. It is never modified by the 10/100 controller.

21.5.3.1.14 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

OSTBDP is the out-of-sequence Tx data buffer pointer register. This out-of-sequence transmit data buffer pointer register is written by the user and contains the data buffer pointer fields in the same format as a regular TxBD and, together with OSTBD, provides the complete 8-byte descriptor. This area should be cleared when not in use.

The OSTBDP is shown in Figure 21-19.

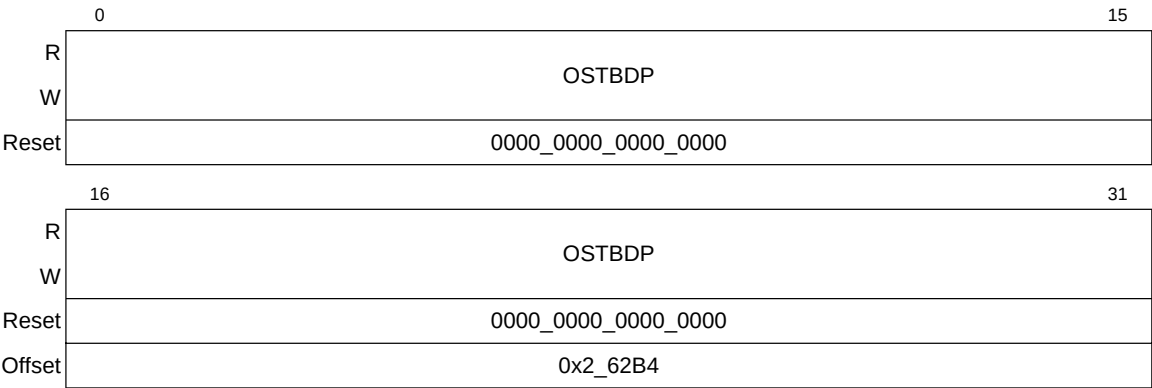


Figure 21-19. OSTBDP Register Definition

Table 21-18. OSTBDP Field Descriptions

Bit	Name	Description
0–31	OSTBDP	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.

21.5.3.2 10/100 Controller 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.

21.5.3.2.1 Receive Control Register (RCTRL)

The RCTRL register is programmed by the user and controls the operational mode of the receive block. It should be written only after a system reset (at initialization) or when DMACTRL[GRS] is set. RCTRL is shown in Figure 21-20.

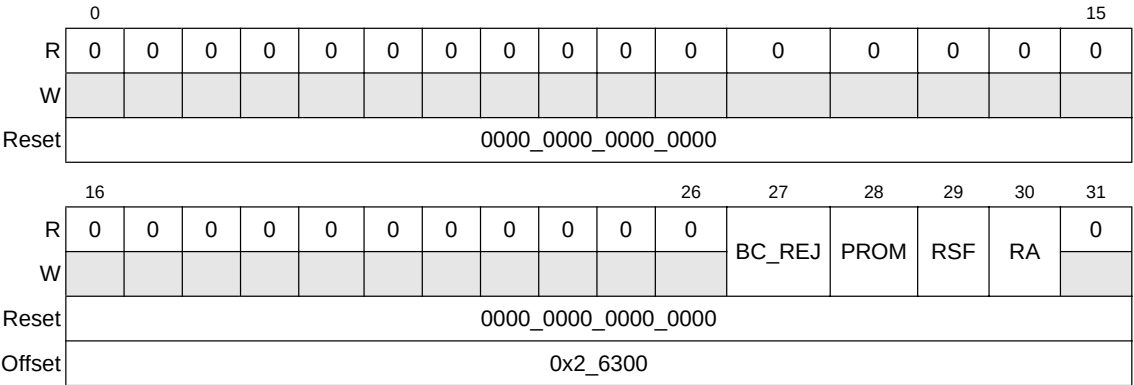


Figure 21-20. RCTRL Register Definition

Table 21-19. RCTRL Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) = FFFF_FFFF_FFFF will be rejected unless the PROM bit is set. If both BC_REJ and PROM are set then frames with broadcast DA will be accepted (unless RA is set) and the M (MISS) bit will be set in the receive buffer descriptor.
28	PROM	Promiscuous mode. All frames, regardless of the addresses, are accepted unless RA is set.
29	RSF	Receive short frame mode. Enable the reception of frames shorter than MINFLR bytes.
30	RA	Reject all mode. Do not accept any frames based on DA hit.
31	—	Reserved

21.5.3.2.2 Receive Status Register (RSTAT)

The 10/100 controller writes to the RSTAT register under the following conditions:

- The receiver runs out of descriptors.
- The receiver was halted because an error condition was encountered while receiving a frame.

In either of these cases, software should clear the appropriate bits to take the 10/100 controller's receiver function out of a halt state. RSAT is shown in Figure 21-21.

	0							7	8	9						15
R	0	0	0	0	0	0	0	0	QHLT	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	0x2_6304															

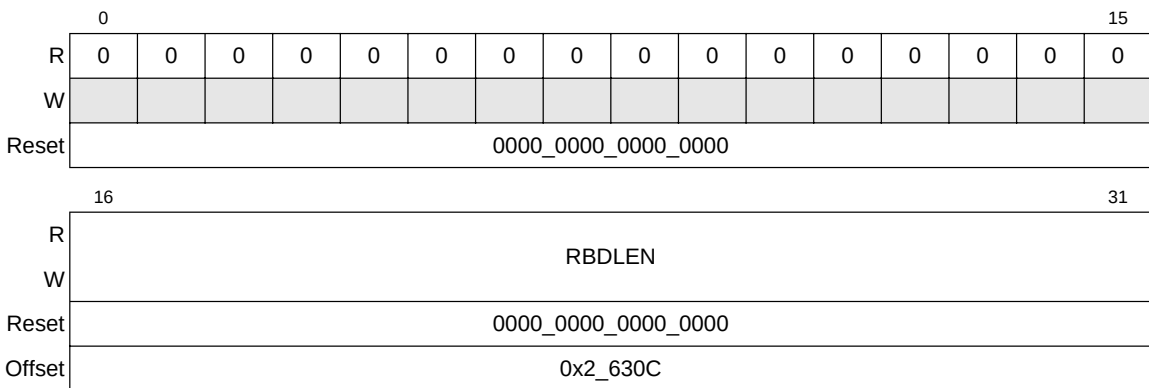
Figure 21-21. RSTAT Register Definition

Table 21-20. RSTATField Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. It is a hardware-initiated stop indication (DMACTRL[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

21.5.3.2.3 RxBD Data Length Register (RBDLEN)

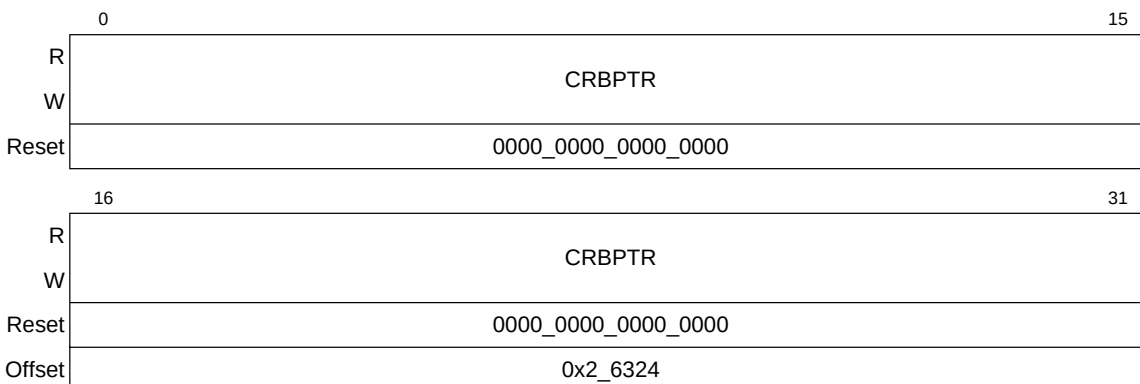
RBDLEN, shown in Figure 21-22, is a DMA register that contains the number of bytes remaining in the current receive buffer.

**Figure 21-22. RBDLEN Register Definition****Table 21-21. RBDLEN Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	RBDLEN	The RBDLEN is internally written by the DMA module. When RBDLEN is 0 all activity in the receive channel will stop. The user should avoid writing to this register.

21.5.3.2.4 Current Receive Buffer Descriptor Pointer (CRBPTR)

CRBPTR, shown in Figure 21-23, contains the address of the receive buffer descriptor currently being worked on, or last worked on.

**Figure 21-23. CRBPTR Register Definition****Table 21-22. CRBPTR Field Descriptions**

Bits	Name	Description
0–31	CRBPTR	The CRBPTR register is internally written by the DMA module. The value will increment by eight (bytes) each time a descriptor is read from memory.

21.5.3.2.5 Maximum Receive Buffer Length Register (MRBLR)

The MRBL register, shown in Figure 21-24, is written by the user. It tells the 10/100 controller how much space is in the receive buffer pointed to by the RxBD.

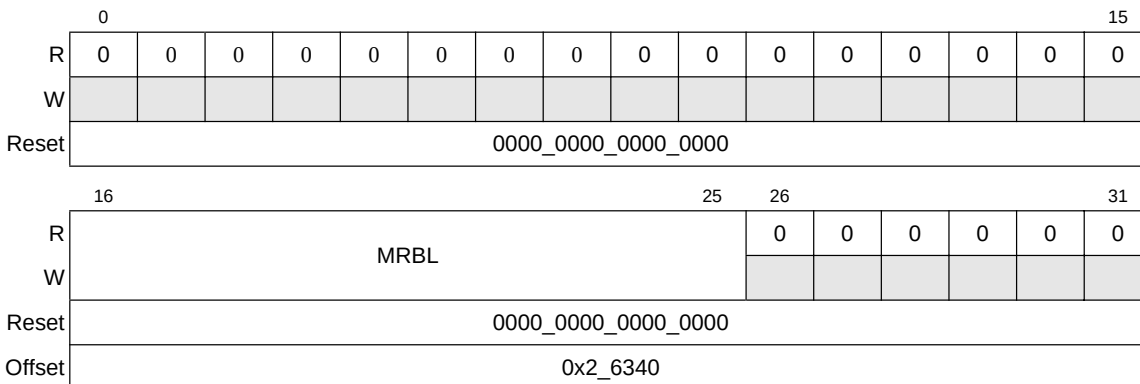
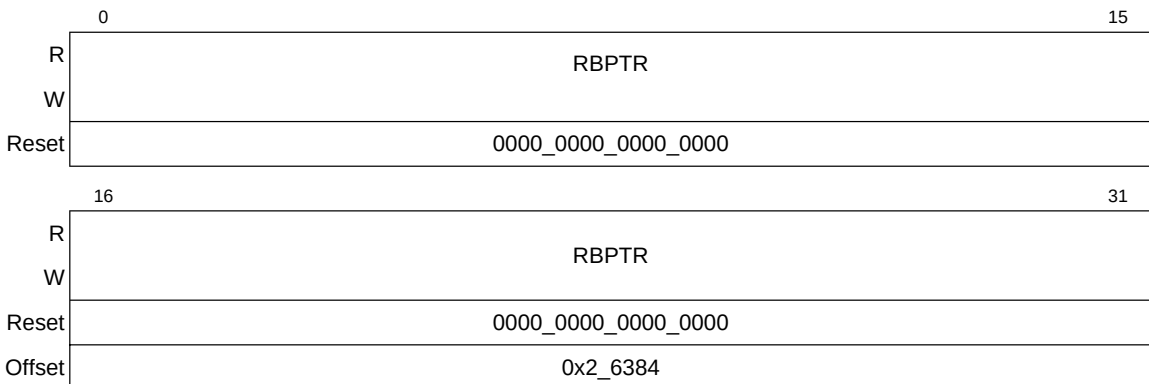
**Figure 21-24. MRBLR Register Definition**

Table 21-23. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	To ensure that MRBL is a multiple of 64 these bits are reserved and should be cleared.
16–25	MRBL	Maximum receive buffer length. MRBL is the number of bytes that the 10/100 controller receiver writes to receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. 10/100 controller 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 should be at least as large as the MRBLR. Note that transmit buffers can have varying lengths by programming TxBD[Data Length], as needed, and are not affected by the value in MRBLR. MRBLR is not intended to be changed dynamically while 10/100 controller is operating. Change MRBLR only when the 10/100 controller receive function is disabled.
26–31	—	To ensure that MRBL is a multiple of 64 these bits are reserved and should be cleared.

21.5.3.2.6 Receive Buffer Descriptor Pointer (RBPTR)

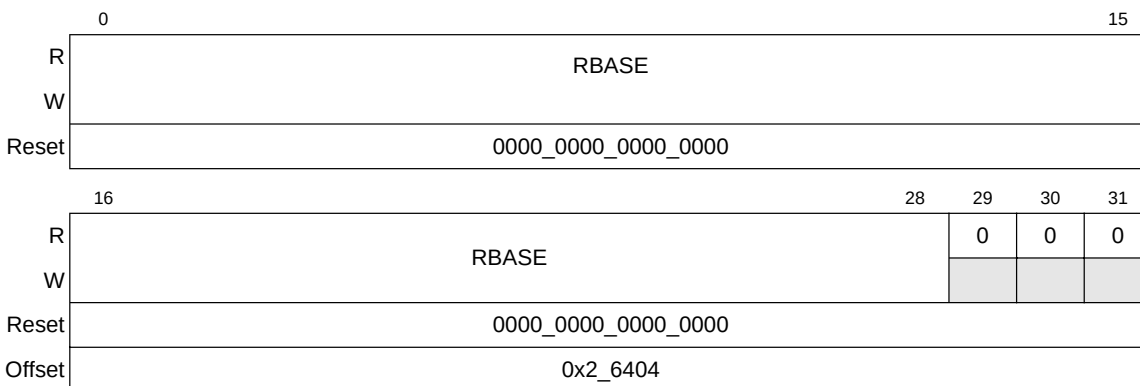
RBPTR, shown in Figure 21-25, contains the next receive buffer descriptor address.

**Figure 21-25. RBPTR Register Definition****Table 21-24. RBPTR Field Descriptions**

Bits	Name	Description
0–31	RBPTR	The RBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.

21.5.3.2.7 Receive Descriptor Base Address (RBASE)

The RBASE register, shown in Figure 21-26, is written by the user with the RxBD base address; must be divisible by eight for the 8-byte buffer descriptors.

**Figure 21-26. RBASE Register Definition****Table 21-25. RBASE Field Descriptions**

Bits	Name	Description
0–28	RBASE	Receive base. RBASE defines the starting location in the memory map for the FEC RxBD. 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.
29–31	—	Reserved

21.5.3.3 MAC Functionality

This section describes the MAC registers and provides a brief overview of some of the functionality that can be exercised through the use of these registers, especially those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

21.5.3.3.1 Configuring the MAC

The MAC configuration registers 1 and 2 provide the means of 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 value.
- Varying pad/CRC combinations— Two different pad/CRC combinations are provided to handle a variety of system requirements. These 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.

21.5.3.3.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the 10/100. Half-duplex is only supported

for 10 Mbps 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 (FEC_CRS) and collision (FEC_COL) indications from the PHY are ignored, but in half-duplex mode, the 10/100 defers to FEC_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 FEC_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 FEC_CRS is sensed. During the final one-third of the IPG, FEC_CRS is ignored and the transmission begins once IPG has been timed. The two-thirds/one-third ratio is the recommended value.

21.5.3.3.3 Handling Packet Collisions

When transmitting a packet in half-duplex mode, the 10/100 is sensitive to FEC_COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is made up of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating a collision occurred and the start of the frame is needed for retransmission. The 10/100 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 should 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 10/100 is deferring to a carrier, the 10/100 simply waits until the end of the carrier event and the timing of IPG before it honors the request.

When packet transmission attempts experience collisions, the 10/100 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:

$$0 < r \leq 2^k \text{ where } k = \min(n, 10).$$

So, after the first collision, 10/100 will back-off either 0 or 1 slot times. After the fifth collision, 10/100 will back-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 will be more aggressive after 7 collisions than other stations on the network.

21.5.3.3.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways with the 10/100 controller. 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. Since 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 10/100 implements the optional back pressure mechanism using the raise carrier method. When the system receive logic wants to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is asserted (TCTRL[THDF]). If the medium is idle, the 10/100 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, 10/100 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. When BPNB is asserted, the 10/100 will wait an inter-packet gap before resuming the transmission of preamble following the collision and will not defer. If de-asserted, the 10/100 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 should be set.

The 10/100 controller will drop carrier (cease transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If while applying back pressure, the 10/100 is requested to send a packet, it will stop sending preamble, and wait one IPG before sending the packet. BPNB will apply for any collision that occurs during the sending of this packet. Collisions for packets while THBP is asserted will be counted. The 10/100 will 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.

21.5.3.3.5 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.

When MIIMCOM[Read Cycle] is set, an MII management read cycle is performed using the pre-configured phy address at MIIMADD[PHY Address] and the register address at

MIIMADD[Register Address]. The 16-bit read data is returned in register MIIMCOM[PHY Status] for use by the host.

MII management write cycles are performed writing 16-bit data to MIIMCON, and using the pre-configured phy address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]).

The 10/100 MII's registers provide the ability to perform continuous read cycles (called a scan cycle).even though scan cycles are not explicitly defined in the Standard. When requested (by setting MIIMCOM[scan cycle]), the part performs read cycle after read cycle 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 to the host when each read of the scan cycle is available 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 the PHY (or PHYs) support preamble suppression, the host may so configure the 10/100. When enabled, the length of MII management frames are reduced from 64 clocks to 32 clocks. This effectively doubles the efficiency of the interface.

21.5.3.4 MAC Registers

This section contains descriptions of the MAC registers.

21.5.3.4.1 MAC Configuration 1 Register (MACCFG1)

The MACCFG1, shown in Figure 21-27, is written by the user.

	0	1	2								11	12	13	14	15	
R	Soft_Reset	0	0	0	0	0	0	0	0	0	0	Reset_Rx_MC	Reset_Tx_MC	Reset_Rx_Fun	Reset_Tx_Fun	
W																
Reset	0000_0000_0000_0000															
	16						22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	Loop_Back	0	0	Rx_Flow	Tx_Flow	Sync'd_Rx_EN	Rx_EN	Sync'd_Tx_EN	Tx_EN
W																
Reset	0000_0000_0000_0000															
Offset	0x2_6500															

Figure 21-27. MACCFG1 Register Definition

Table 21-26. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	0 Default 1 Setting this bit puts all modules within the MAC in reset.
1–11	–	Reserved
12	Reset_Rx_MC	0 Default 1 Setting this bit puts the receive MAC control block in reset. This block detects control frames and contains the pause timers.
13	Reset_Tx_MC	0 Default 1 Setting this bit puts the PETMC transmit MAC control block in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset_Rx_Fun	0 Default 1 Setting this bit puts the receive function block in reset. This block performs the receive frame protocol.
15	Reset_Tx_Fun	0 Default 1 Setting this bit puts the transmit function block in reset. This block performs the frame transmission protocol.
16–22	–	Reserved
23	Loop_Back	0 Default. Clearing this bit results in normal operation. 1 Setting this bit causes the MAC transmit outputs to be looped back to the MAC receive inputs.
24–25	–	Reserved
26	Rx_Flow	0 Default. Clearing this bit causes the receive MAC control to ignore PAUSE flow control frames. 1 Setting this bit causes the receive MAC control to detect and act on PAUSE flow control frames.
27	Tx_Flow	0 Default. Clearing this bit prevents the transmit MAC control from sending flow control frames. 1 Setting this bit allows the transmit MAC control to send PAUSE flow control frames when requested by the system.
28	Sync'd_Rx_EN	Receive enable synchronized to the receive stream. (Read only) 0 Frame reception has not been enabled. 1 Frame reception has been enabled.
29	Rx_EN	0 Default. Clearing this bit prevents the reception of frames. 1 Setting this bit allows the MAC to receive frames from the PHY.
30	Sync'd_Tx_EN	Transmit enable synchronized to the transmit stream. (Read only) 0 Frame transmission has not been enabled. 1 Frame transmission has been enabled.
31	Tx_EN	0 Default. Clearing this bit prevents the transmission of frames. 1 Setting this bit allows the MAC to transmit frames from the system.

21.5.3.4.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2, shown in Figure 21-28, is written by the user.

	0															15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																	
Reset	0000_0000_0000_0000																
	16	19			20	21	22	23	24	25	26	27	28	29	30	31	
R	Preamble_Length				0	0	I/F_Mode		0	0	Huge_Frame	Length_Check	0	PAD/CRC	CRC EN	Full_Duplex	
W																	
Reset	0111_0000_0000_0000																
Offset	0x2_6504																

Figure 21-28. MACCFG2 Register Definition

Table 21-27. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble_Length	This field determines the length of the preamble field of the packet, in bytes. Its default is '0x7'.
20–21	—	Reserved
22–23	I/F_Mode	This field determines the type of interface the MAC is connected to. Its default is '00'. 00 Reserved 01 Nibble mode (MII) 10 Reserved 11 Reserved
24–25	—	Reserved
26	Huge_Frame	0 Default. Clear this bit to have the MAC limit the length of frames at the MAXIMUM FRAME LENGTH value. 1 Allows frames longer than the MAXIMUM FRAME LENGTH to be transmitted and received.
27	Length_Check	0 Default. Clear this bit if no length field checking is desired. 1 Causes the MAC to check the frame's length field to ensure it matches the actual data field length.
28	—	Reserved
29	PAD/CRC	0 Default. Clear this bit if frames presented to the MAC have a valid length and contain a CRC. 1 Causes the MAC to pad all transmitted short frames and append a CRC to every frame whether or not padding was required.
30	CRC EN	CRC enable 0 Default. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC. 1 Causes the MAC to append a CRC on all frames. If the configuration bit PAD/CRC enable or the per-packet PAD/CRC enable is set, CRC enable is ignored.
31	Full_Duplex	0 Default. Clearing this bit configures the MAC to operate in half-duplex mode only. 1 Configures the MAC to operate in full-duplex mode.

21.5.3.4.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG, shown in Figure 21-29, is written by the user.

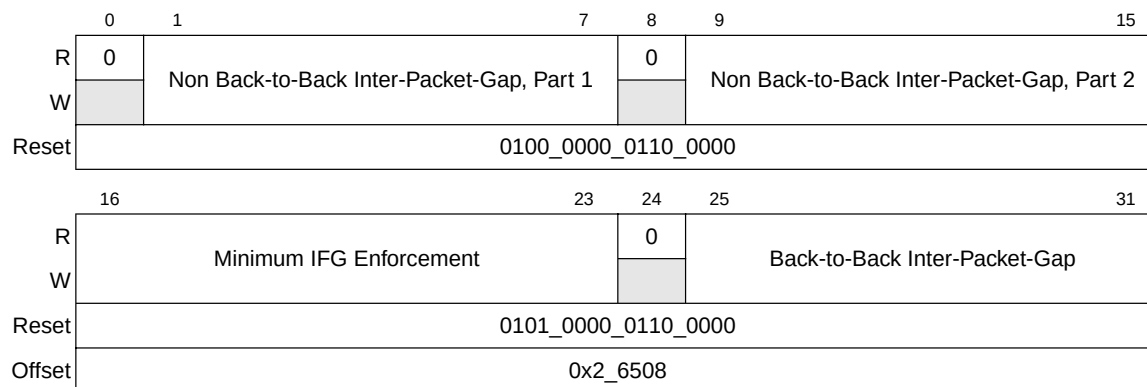


Figure 21-29. IPGIFG Register Definition

Table 21-28. 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-thirds guideline.
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 whose IFG is less than that programmed is dropped. 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 when 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.

21.5.3.4.4 Half Duplex Register (HAFDUP)

HAFDUP, shown in Figure 21-30, is written by the user.

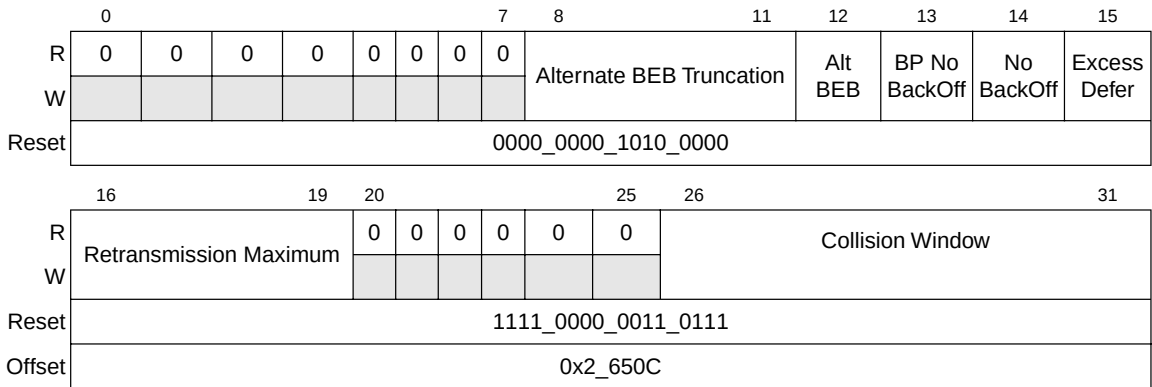


Figure 21-30. Half Duplex Register Definition

Table 21-29. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used when 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	Setting this bit configures the Tx MAC to use 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 210 – 1 as the maximum backoff time. Clearing this bit causes the Tx MAC to follow the standard binary exponential back off rule. Its default is '0'.
13	BP No BackOff	Setting this bit configures the Tx MAC to immediately re-transmit, following a collision, during back pressure operation. Clearing this bit causes the Tx MAC to follow the binary exponential back off rule. Its default is '0'.
14	No BackOff	Setting this bit configures the Tx MAC to immediately re-transmit following a collision. Clearing this bit causes the Tx MAC to follow the binary exponential back off rule. Its default is '0'.
15	Excess Defer	Setting this bit configures the Tx MAC to allow the transmission of a packet that has been excessively deferred. Clearing this bit causes the Tx MAC to abort the transmission of a packet that has been excessively deferred. Its default is '1'.
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 attemptLimit to be 0xF (15d). Its default 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.

21.5.3.4.5 Maximum Frame Length Register (MAXFRM)

This register is written by the user.

	0														15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
	16														31
R	Maximum Frame														
W															
Reset	0000_0110_0000_0000														
Offset	0x2_6510														

Figure 21-31. Maximum Frame Length Register Definition

Table 21-30. MAXFRM Descriptions

Bit	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].)

21.5.3.4.6 MII Management Configuration Register (MIIMCFG)

This register is written by the user.

	0	1													15			
R	Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
W	Mgmt																	
Reset	0000_0000_0000_0000																	
	16										26	27	28	29	31			
R	0	0	0	0	0	0	0	0	0	0	No Pre		Mgmt Clock Select					
W																		
Reset	0000_0000_0000_0000																	
Offset	0x2_6520																	

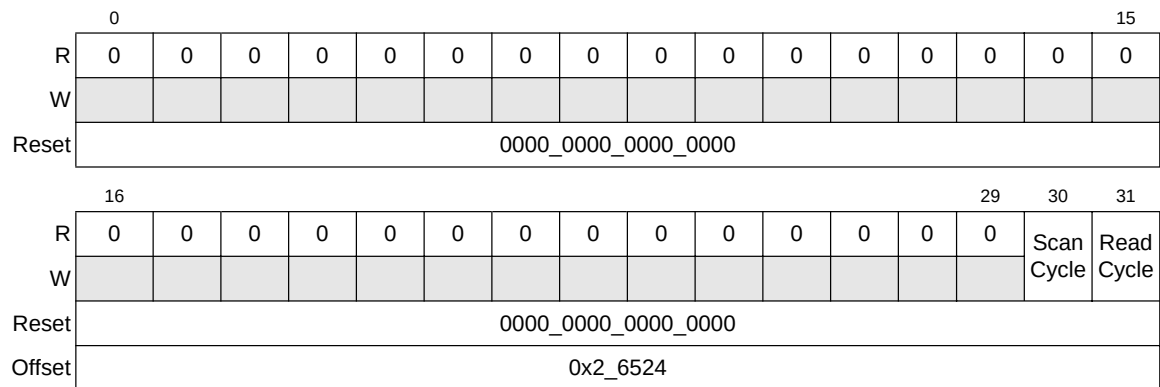
Figure 21-32. MII Management Configuration Register Definition

Table 21-31. MIIMCFG Field Descriptions

Bit	Name	Description
0	Reset Mgmt	Setting this bit resets the MII MGMT. Clearing this bit allows the MII MGMT to perform mgmt read/write cycles when requested . Its default is '0'.
1–26	—	Reserved
27	No Pre	Setting this bit causes the MII MGMT to suppress preamble generation and reduce the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2. Clearing this bit causes the MII MGMT to perform Mgmt read/write cycles with the 32 clocks of preamble. Its default is '0'.
28	—	Reserved
29–31	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (EC_MDC). Its default is '000'. The source clock frequency is equal to the core complex bus clock divided by eight. 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

21.5.3.4.7 MII Management Command (MIIMCOM)

This register is written by the user.

**Figure 21-33. MII Mgmt Command Register Definition****Table 21-32. MIIMCOM Descriptions**

Bits	Name	Description
0–29	—	Reserved

Table 21-32. MIIMCOM Descriptions (continued)

Bits	Name	Description
30	Scan Cycle	Causes the MII management to perform read cycles continuously. This is useful for monitoring link fail for example. Its default is '0'.
31	Read Cycle	Causes the MII management to perform a single read cycle. When set, an MII management read cycle is performed using the phy address (at MII Mgmt Address[PHY Address]) and the register address (at MII Mgmt Address[Register Address]). The read data is returned in register MII Mgmt Status[PHY Status]. Its default is '0'.

21.5.3.4.8 MII Management Address Register (MIIMADD)

This register is written by the user.

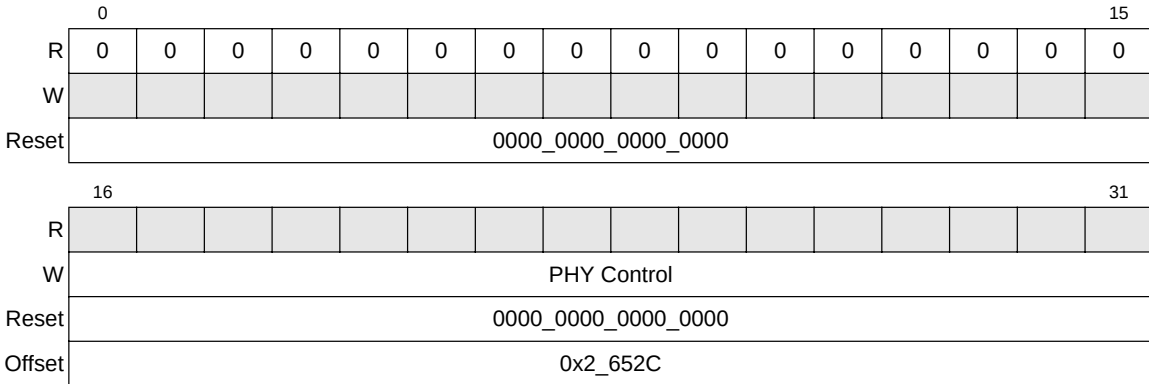
	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
	16		18	19			23	24		26	27					31
R	0	0	0					0	0	0						
W																
Reset	0000_0000_0000_0000															
Offset	0x2_6528															

Figure 21-34. MII Management Address Register Definition**Table 21-33. MIIMADD Field Descriptions**

Bit	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). Its default is '0x00'.
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 is '0x00'.

21.5.3.4.9 MII Management Control Register (MIIMCON)

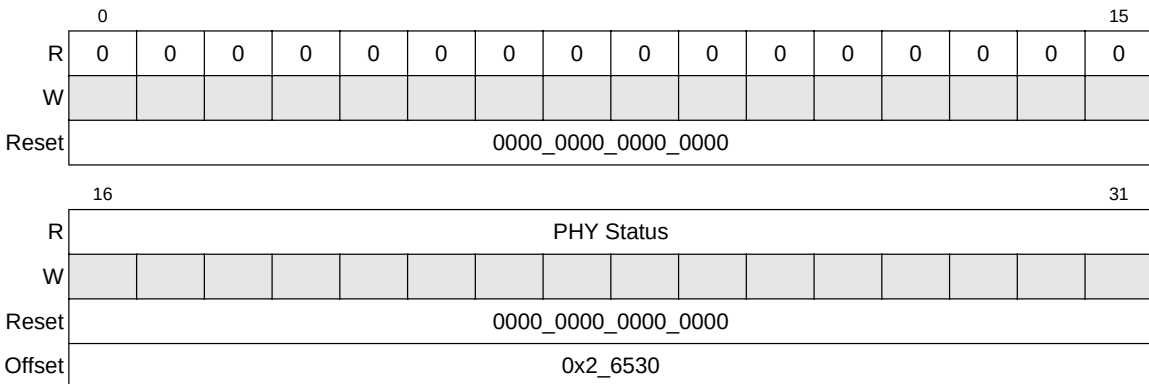
This register is written by the user and is not readable.

**Figure 21-35. MII Management Control Register Definition****Table 21-34. MIIMCON Field Descriptions**

Bit	Name	Description
0–15	—	Reserved
16–31	PHY Control	When written, an MII management write cycle is performed using this 16-bit data, the pre-configured phy address (at MII Mgmt Address[PHY Address]) and the register address (at MII Mgmt Address[Register Address]). Its default is '0x0000'.

21.5.3.4.10 MII Management Status Register (MIIMSTAT)

This register is read-only by the user.

**Figure 21-36. MII Management Status Register Definition****Table 21-35. MIIMSTAT Field Descriptions**

Bit	Name	Description
0–15	—	Reserved
16–31	PHY Status	Following an MII management read cycle, the 16-bit data can be read from this location. Its default is 0x0000.

21.5.3.4.11 MII Management Indicator Register (MIIMIND)

This register is written by the user.

	0															15	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																	
Reset	0000_0000_0000_0000																
	16												28		29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	Not Valid	Scan	Busy		
W																	
Reset	0000_0000_0000_0000																
Offset	0x2_6534																

Figure 21-37. MII Management Indicator Register Definition

Table 21-36. MIIMIND Field Descriptions

Bits	Name	Description
0–28	–	Reserved
29	Not Valid	1 indicates MII management read cycle has not completed and the read data is not yet valid.
30	Scan	When '1' is returned - indicates a scan operation (continuous MII management read cycles) is in progress.
31	Busy	When '1' is returned - indicates MII management block is currently performing an MII management read or write cycle.

21.5.3.4.12 Interface Status Register (IFSTAT)

This register is written by the user.

	0															15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																
Reset	0000_0000_0000_0000															
	16					21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	Excess Defer	0	0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0001															
Offset	0x2_653C															

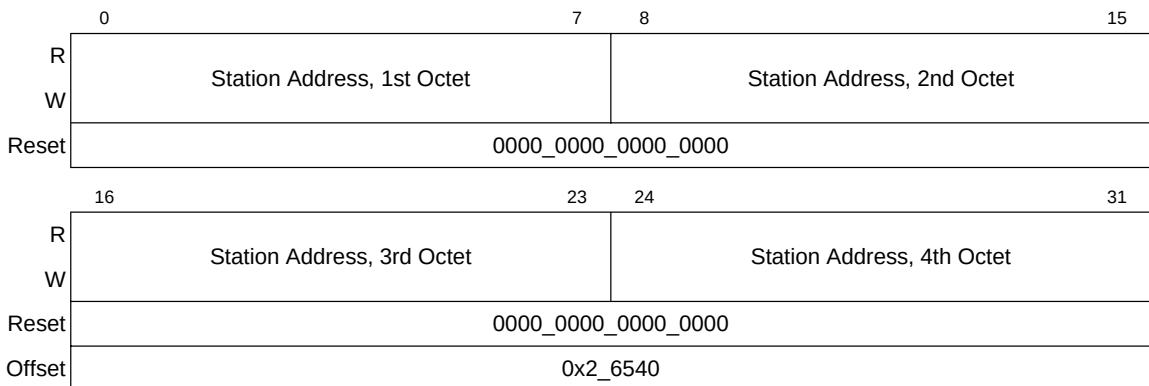
Figure 21-38. Interface Status Register Definition

Table 21-37. IFSTAT Field Descriptions

Bit	Name	Description
0–21	—	Reserved
22	Excess Defer	This bit sets when the MAC excessively defers a transmission. It clears when read. This bit latches high. Its default is '0'.
23–31	—	Reserved

21.5.3.4.13 Station Address Register Part 1 (MACSTNADDR1)

This register is written by the user.

**Figure 21-39. Station Address Register Part 1 Definition****Table 21-38. MACSTNADR1 Field Descriptions**

Bit	Name	Description
0–7	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet is stored in 0–7 and defaults to '0x00'.
8–15	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet is stored in 8–15 and defaults to '0x00'.
16–23	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet is stored in 16–23 and defaults to '0x00'.
24–31	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet is stored in 24–31 and defaults to '0x00'.

21.5.3.4.14 Station Address Register Part 2 (MACSTNADDR2)

This register is written by the user.

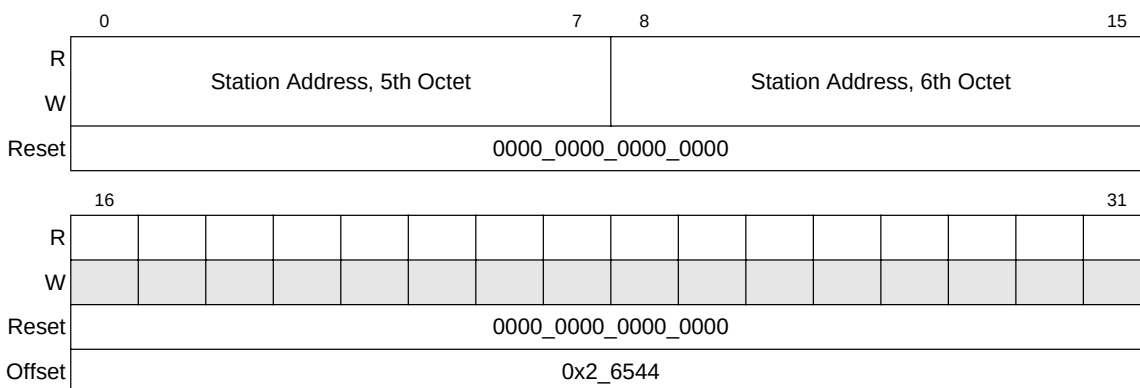


Figure 21-40. Station Address Register Part 2 Definition

Table 21-39. MACSTNADDR2 Field Descriptions

Bit	Name	Description
0–7	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet is stored in 0–7 and defaults to '0x00'.
8–15	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet is stored in 8–15 and defaults to '0x00'.
16–31	—	Reserved

21.5.3.5 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8-bit MSB 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 when the DA CRC result points to an enabled hash entry. The user should further filter the address. See Section 21.6.2.4.2, “Hash Table Algorithm” for more information on the hash algorithm.

21.5.3.5.1 Individual Address Registers 0–7 (IADDR_n)

The IADDR_n register is written by the user. Together 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-bit MSB 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 when the DA CRC result points to an enabled hash entry.

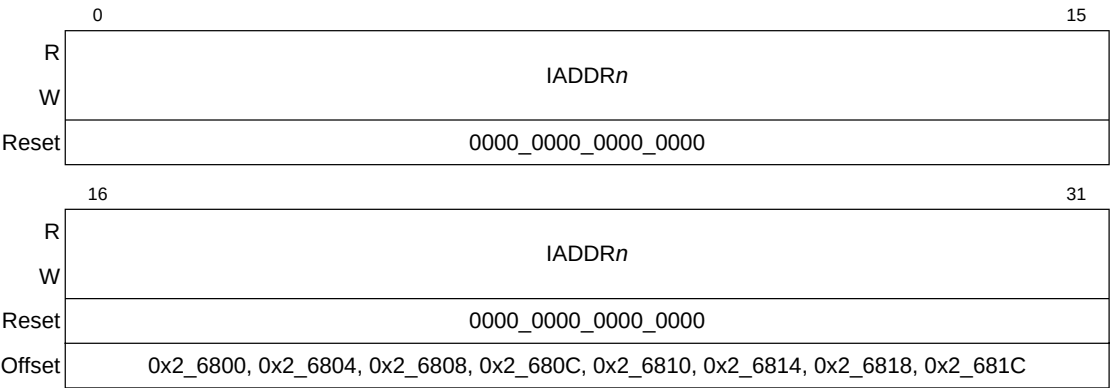


Figure 21-41. IADDR_n Register Definition

Table 21-40. IADDR_n Field Description

Bits	Name	Description
0–31	IADDR _n	Represents the 32 bit value associated with the corresponding register. IADDR ₀ contains the high-order 32 bits of the 256 entry hash table and IADDR ₇ represents the low-order 32 bits.

21.5.3.5.2 Group Address Registers 0–7 (GADDR_n)

The GADDR_n register is written by the user. Together these registers represent 256 entries of the group (multicast) 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-bit MSB 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 when the DA CRC result points to an enabled hash entry.

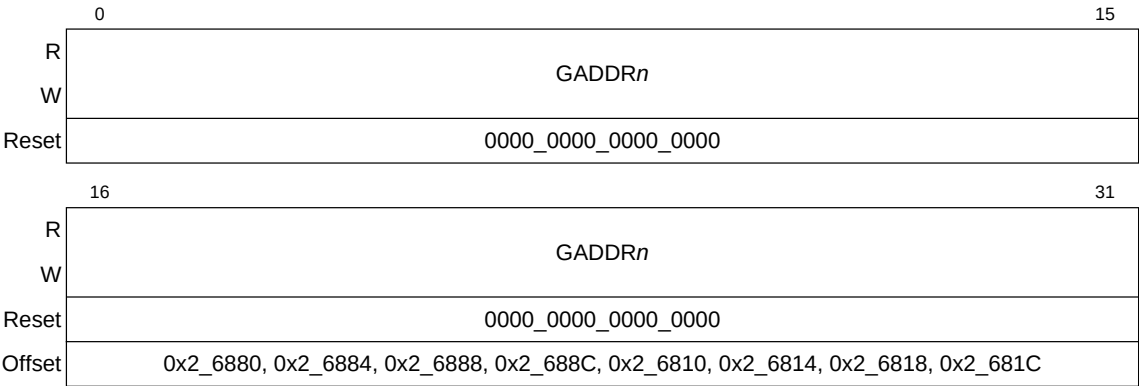


Figure 21-42. GADDR_n Register Definition

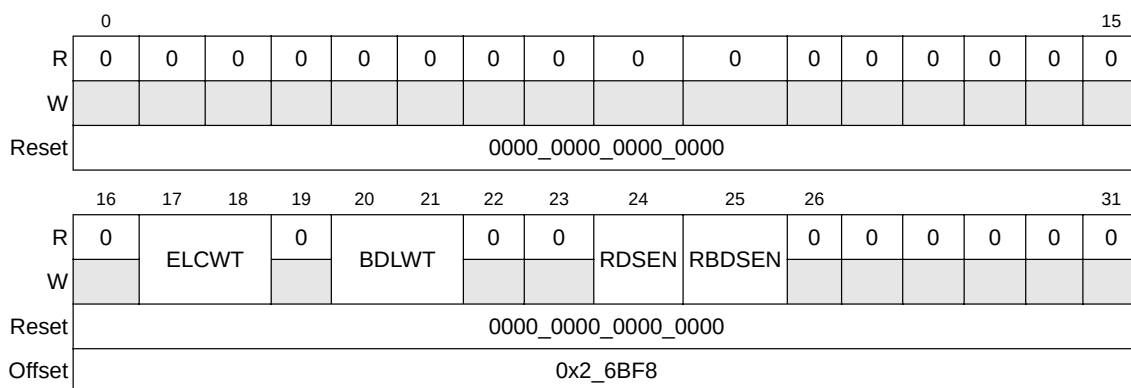
Table 21-41. GADDR_n Field Description

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.

21.5.3.5.3 Attribute Register (ATTR)

The ATTR is written by the user to determine where to file, when to extract, and how to store the received frames and their associated buffer descriptors.

Extraction is possible by setting the ATTR[EL] field to a non-zero value. The reset conditions of this register are: no extraction (ATTR[EL]=0), ATTR[BDLWT] (Buffer descriptor L2 cache write type), and ATTR[ELCWT] (Extracted L2 cache write type) are set to no allocation.

**Figure 21-43. ATTR Register Definition****Table 21-42. ATTR Field Descriptions**

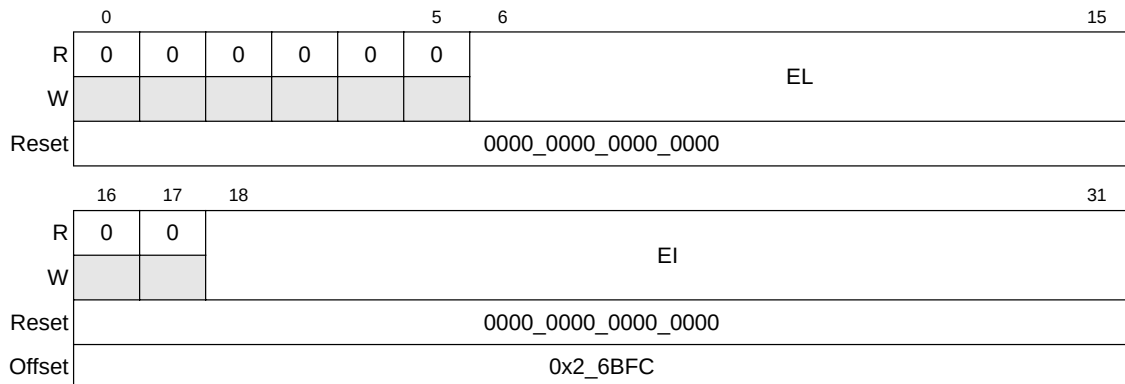
Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Contains the write transaction type to perform for the extracted data. This occurs only if ATTR[EL] is non-zero. For maximum performance, it is recommended that when ELCWT is set to allocate, BDLWT should also be set to allocate. Writes to cache are always done with snoop. 00 No allocation performed 01 Reserved 10 Allocate L2 cache line 11 Allocate and lock L2 cache line
19	—	Reserved

Table 21-42. ATTR Field Descriptions

Bits	Name	Description
20-21	BDLWT	Buffer descriptor L2 cache write type. Contains the write transaction type to perform for the buffer descriptor for a receive frame. Writes to cache are always done with snoop. 00 No allocation performed 01 Reserved 10 Allocate L2 cache line 11 Allocate and lock L2 cache line
22-23	—	Reserved
24	RDSSEN	Rx data snoop enable. 0 Disables snooping of all receive frames data to memory (except to cache). 1 Enables snooping of all receive frames data to memory.
25	RBDSEN	RxBD snoop enable. 0 Disables snooping of all receive BD memory accesses (except to cache). 1 Enables snooping of all receive BD memory accesses.
26-31	—	Reserved

21.5.3.5.4 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the default extract index and extract length to be used.

**Figure 21-44. ATTRELI Register Definition****Table 21-43. ATTRELI Field Descriptions**

Bits	Name	Description
0-5	—	Reserved
6-15	EL	Extracted length. Contains the number of bytes to extract from the receive frame. The DMA controller uses this field to perform extraction. When 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.

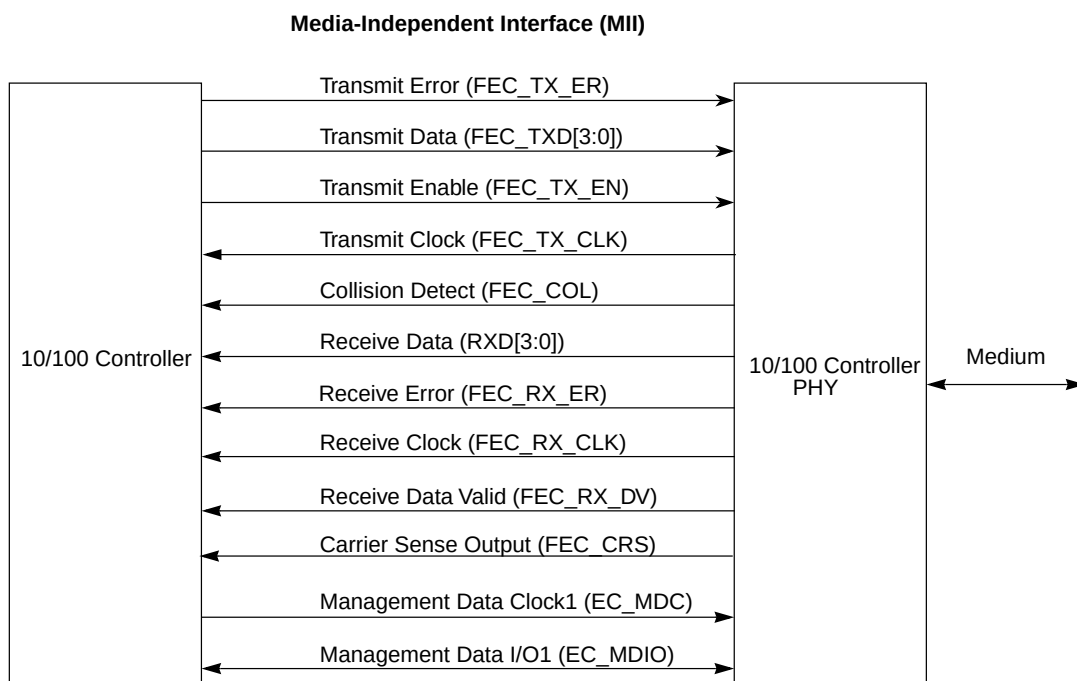
21.6 Functional Description

21.6.1 Connecting the 10/100 Controller to the Physical Interface

This section describes how to connect the 10/100 controller to the MII interface. The bus follows the bus conventions used in the IEEE 802.3 specification because the PHY follows the same conventions (that is, in the bus TXD[3:0], bit 3 is the MSB and bit 0 the LSB).

21.6.1.1 MII Interface

This section describes the media independent interface (MII) intended to be used between the PHYs and the 10/100 controller. Figure 3-154 shows the basic components of the MII including the signals required to make the 10/100 controller module connection with a PHY.



¹ The management signals (EC_MDC and EC_MDIO) can be common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 21-45. 10/100 Controller-MII Connection

An MII interface has 18 signals (including the EC_MDC and EC_MDIO signals), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

Table 21-44. Pins

MII INTERFACE		
Frequency [MHz] 25		
Voltage[V] 3.3		
Signals	I/O	No. of Pins
FEC_TX_CLK	I	1
FEC_TxD[0]	O	1
FEC_TxD[1]	O	1
FEC_TxD[2]	O	1
FEC_TxD[3]	O	1
FEC_TX_EN	O	1
FEC_TX_ER	O	1
FEC_RX_CLK	I	1
FEC_RxD[0]	I	1
FEC_RxD[1]	I	1
FEC_RxD[2]	I	1
FEC_RxD[3]	I	1
FEC_RX_DV	I	1
FEC_RX_ER	I	1
FEC_COL	I	1
FEC_CRs	I	1
Sum		16

Table 21-45. Shared Pins

Signals	I/O	No. of Pins	Function
EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	Management Interface Clock
Sum		2	

21.6.2 10/100 Controller Channel Operation

This section describes the operation of the 10/100 controller. First the software initialization sequence is described. Next the software (Ethernet driver) interface for transmitting and receiving frames is described. Address recognition and the hash table algorithm are described. Then the section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

21.6.2.1 Initialization Sequence

This section describes which registers are reset due to a hard or software reset and what registers the user must initialize before the 10/100 controller is enabled.

21.6.2.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All of the 10/100 controller's registers and control logic are reset to their default states after a hard reset has occurred.

21.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic 10/100 controller registers. Other registers can also be initialized during this time, but they are optional and should be determined based on the requirements of the system. See Table 21-3 for the register list.

The steps for minimum register initialization are as follows:

1. Clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII management interface
5. Clear IEVENT
6. Initialize IMASK
7. Initialize IADDR_n
8. Initialize GADDR_n
9. Initialize RCTRL
10. Initialize DMACTRL

After initializing registers, the user must to execute the following steps in the order described below to bring the 10/100 controller into a functional state (out of reset):

1. Write to the MACCFG1 register and set the appropriate bits. These need to include RX_EN = 1, and TX_EN = 1. To enable flow control, Rx_Flow and Tx_Flow should also be set.

2. For the transmission of ethernet frames, TxBDs have to be first 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.
3. Likewise for the reception of ethernet frames, the receive queue should be ready, with its RxBDs linked together as a ring and pointed to by the RBASE register. Both transmit and receive can be gracefully stopped after transmission and reception begins.
4. A write to the DMACTRL[GTS] bit with a value of 0 triggers the transmission of frame data if the transmitter had been previously stopped while the DMACTRL[GRS] must be cleared if the receiver had been previously stopped. Refer to the DMACTRL register section, and Section 21.6.3.1, “Data Buffer Descriptor,” for more information.

21.6.2.2 10/100 Ethernet Frame Transmission

The Ethernet transmitter requires almost no core intervention. After the software driver initializes the system, the 10/100 begins to poll the first transmit buffer descriptor (TxBD) in the TxBD ring every 512 transmit clocks. If TxBD[R] bit is set, the 10/100 will begin 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, setting DMACTRL[TOD] eliminates waiting for the next poll, and a DMA transfer of the transmit data buffers begins immediately. The transmission begins when all data for the frame is loaded into the Tx FIFO or sufficient transmit data is in the Tx FIFO. If the line is not busy, the MAC transmit logic asserts TX_EN and sends the preamble sequence, start frame delimiter, and frame information in that order; see Figure 21-3. If the line is busy, the controller waits for the carrier sense signal, FEC_CRS, to remain inactive for 60 bit times (60 clocks) and transmission begins after an additional 36 bit times (96 bit times after FEC_CRS became active). In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap (96 bit times) regardless of FEC_CRS.

There is one internal buffer for out-of-sequence flow control frames. When the 10/100 is between frames, this buffer is polled. The buffer must contain the whole frame. Once the 10/100 is in paused mode, the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC treats it like a regular TxBD.

In half duplex mode (MACCFG2[Full Duplex] = 0) the MAC defers transmission if the line is busy (FEC_CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if FEC_CRS remains negated for 60

Functional Description

clocks. If so, transmission begins after an additional 36 bit times (96 bit times after FEC_CRS originally became negated). If FEC_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 needs to be enforced.

The transmit block also implements full duplex flow control. If a flow control frame is received, the MAC will not service the transmitter's request to send data until the pause duration is over. If the MAC is currently sending data while a pause frame is received, the MAC will finish sending the current frame, then will suspend 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 will automatically append FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] = 1 in first Tx BD
- TxBD[TC] = 1 in first Tx BD
- MACCFG2[PAD/CRC] = 1
- MACCFG2[CRC] = 1

Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. When the end of the current buffer is reached and TxBD[L] = 0 (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] = 1, the Ethernet controller pads any frame shorter than 64 bytes.

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. When this bit is set the 10/100 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 when the graceful transmit stop operation is completed. When the DMACTRL[GTS] is cleared, the 10/100 resumes transmission with the next frame.

When the 10/100 is in 10/100Mbps mode it sends bytes least-significant nibble first and each nibble is sent least significant bit first.

21.6.2.3 Gigabit Ethernet Frame Reception

The 10/100 controller Ethernet receiver is designed to work with almost no 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 FEC_RX_DV is asserted and as long as FEC_COL remains negated (full-duplex mode ignores FEC_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 and the frame begins to be processed. If a valid header is not found, the frame is ignored.

When the receiver detects the first bytes of a frame, the 10/100 controller begins to perform the frame recognition function, using destination address (DA) recognition (see frame recognition section 3.3.5 - pg. 24). Based on this match the frame can be accepted or rejected. Once accepted, the 10/100 controller 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 not wanted by this station.

If a frame is accepted, the Ethernet controller fetches a receive buffer descriptor (RxBD) from the queue. If RxBD is not being used by software (RxBD[E] = 1), the 10/100 controller starts transferring the incoming frame. The RxBD[F] bit is set for the first RxBD used for any particular receive frame.

When the buffer has been filled, the 10/100 controller clears RxBD[E] and generates an interrupt if RxBD[I] is set. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table; if it is empty, it continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBD's are used. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems.

The RxBD length for the queue is determined by the MRBL field in the maximum receive buffer length registers (MRBLR). The smallest valid value is 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. When the frame ends (FEC_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.

Functional Description

Receive frames are not truncated if they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame]=1, however the babbling receiver error interrupt occurs (IEVENT[BABR]=1) and RxBD[LG] is set.

When 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 a maskable interrupt, indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

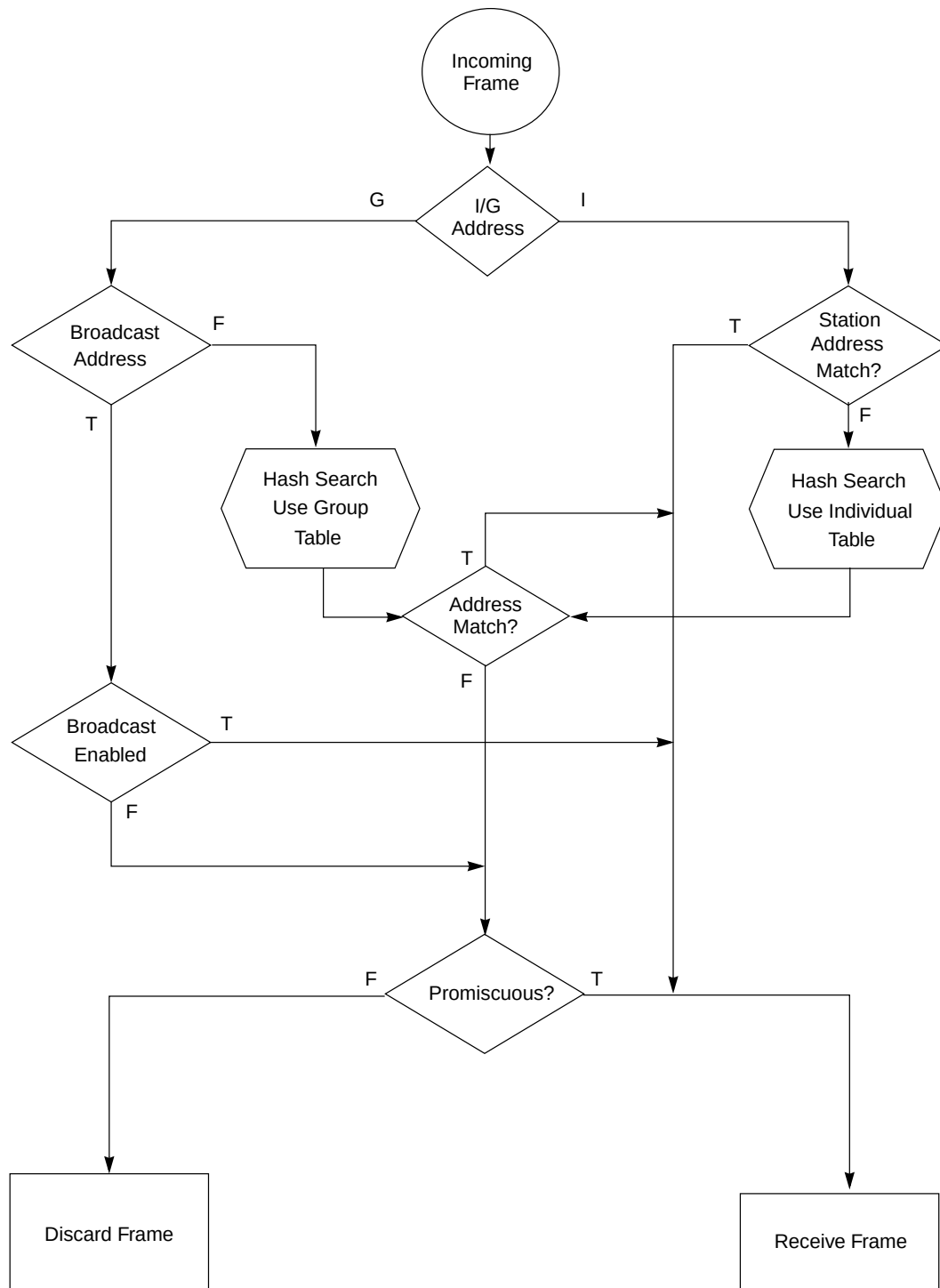
To interrupt reception, or rearrange the receive queue, DMACTRL[GRS] must be set. When this bit is set the 10/100 controller receiver does a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event will be signalled when 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 user and the 10/100 controller hardware without fear of conflict. When DMACTRL[GRS] is cleared, the 10/100 controller 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.

21.6.2.4 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome of the recognition.

21.6.2.4.1 Destination Address Recognition

The Ethernet controller performs the frame filtering using destination address (DA) recognition methods.

**Figure 21-46. Ethernet Address Recognition Flowchart**

The Ethernet controller compares the destination address field of the received frame 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, then 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 21.5.3.2.1, “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 conditionally 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. This allows the 10/100 controller to perform a “discriminating promiscuous” mode, for instance, by allowing all but a few specific addresses to be received.

The address recognition mode allows the 10/100 controller to perform the traditional DA selection criteria to determine which frames to keep and which to reject.

21.6.2.4.2 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 8 high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table. The higher order 3 bits of this 8-bit field are used to select one of the 8 registers in either the individual or group hash table. The low-order 5 bits select a bit within the 32-bit register. A value of 0 in the high-order 3 bits selects IADDR0/GADDR0.

The same process is used when 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 conditionally 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.

21.6.2.5 Flow Control

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. When 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 21-46 shows the flow control frame structure.

Table 21-46. 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 period measured in slot times, most-significant octet first with a two time-slot resolution.Note: Because the pause period has a resolution of two time slots, the value programmed in this field is rounded up to the nearest even number before being used, as follows: <table><thead><tr><th><u>MAC Parameter Value</u></th><th><u>Pause Period</u></th></tr></thead><tbody><tr><td>0</td><td>none</td></tr><tr><td>1 or 2</td><td>2 x slot time</td></tr><tr><td>3 or 4</td><td>4 x slot time</td></tr><tr><td>...</td><td>...</td></tr></tbody></table>	<u>MAC Parameter Value</u>	<u>Pause Period</u>	0	none	1 or 2	2 x slot time	3 or 4	4 x slot time	...	...
<u>MAC Parameter Value</u>	<u>Pause Period</u>												
0	none												
1 or 2	2 x slot time												
3 or 4	4 x slot time												
...	...												
42	Reserved	—											
4	FCS		Frame check sequence (CRC)										

When flow-control mode is enabled ($\text{MACCFG1}[\text{RFCE}] = 1$) 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 ($\text{TCTRL}[\text{TFC\_PAUSE}] = 1$). 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.

21.6.2.6 Interrupt Handling

The following describes what usually occurs within a 10/100 controller interrupt handler:

- When 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 TXE] 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 10/100 controller. 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 21-47.

- Obtain data from the RxBD if IEVENT[RXC,RXB0-3 or RXF0-3] is set. If the receive speed is fast or the interrupt delay is long, the 10/100 controller may have received more than one receive buffer. 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 10/100 controller pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 21-47.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 21-47. Non-Error Transmit Interrupts

Interrupt	Description	Action taken by 10/100 controller
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 has been sent.	None
TXB	Transmit buffer: A transmit buffer descriptor, that is not the last one in the frame, has been updated.	Programmable “write with response” TxBD to memory before setting IEVENT[TXB].
TXF	Transmit frame: A frame has been transmitted and the last transmit buffer descriptor (TxBD) of that frame has been updated.	Programmable “write with response” to memory on the last TxBD before setting IEVENT[TXF].

Table 21-48. Non-Error Receive Interrupts

Interrupt	Description	Action taken by 10/100 controller
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 has been 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, has been updated.	Programmable “write with response” RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame has been received and the last receive buffer descriptor (RxBD) of that frame has been updated.	Programmable “write with response” to memory on the last RxBD before setting IEVENT[RXF].

21.6.2.7 Inter-Packet Gap Time

When a station needs to transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). When a station starts 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 (backoff) before attempting to send again. When the backoff 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 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.

21.6.2.8 Internal and External Loop Back

Setting the MACCFG1[Loop Back] bit causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. Default is '0'. Clearing this bit results in normal operation. Its default is '0'.

21.6.2.9 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in Table 21-49.

Table 21-49. 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, IEVENT[XFUN], and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] has been cleared (and DMACTRL[GTS] = 0).
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] has been cleared (and DMACTRL[GTS] = 0).
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] has been cleared (and DMACTRL[GTS] = 0).
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 will cause an underrun error but IEVENT[XFUN] is not set. The TSTAT[THLT] will be set. Transmits are continued once TSTAT[THLT] is cleared.

Table 21-49. Transmission Errors

Error	Response
Babbling Transmit Error	A frame is transmitted which exceeds the MAC's Maximum Frame Length. The controller sets IEVENT[BABT] and continues without interruption.

Reception errors are described in Table 21-50.

Table 21-50. 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 RxB[OV], sets RxB[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 proper RSTAT halt bit will be set. The halted queue resumes reception once the RSTAT halt bit is cleared.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits when 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 (RxB[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The 10/100 controller relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.
CRC error	When a CRC error occurs, the controller sets RxB[CR], closes the buffer, and sets IEVENT[RXF]. This 10/100 controller 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 proper RSTAT halt bit will be set. The halted queue will resume reception once the RSTAT halt 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

21.6.3 Buffer Descriptors

The 10/100 controller buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller (FEC) BD programming model for ease of reuse, with descriptor base registers pointing to the beginning of BD rings. The 10/100 controller BD also expands upon the MPC8260 BD model to accommodate 10/100 controller's unique features, and to provide an expansion path for future requirements. There is an 8-byte data BD format designed to be similar to the existing MPC8260 BD model.

The 10/100 controller is capable of placing extracted 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. This results in substantial improvement in throughput and hence reduction in latency.

21.6.3.1 Data Buffer Descriptor

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 21-47). Data BD's encapsulates all information necessary for the 10/100 controller 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 is 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 10/100 controller and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see Figure 21-48), all BDs must reside sequentially in memory. The 10/100 controller 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 10/100 controller 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 10/100 controller.

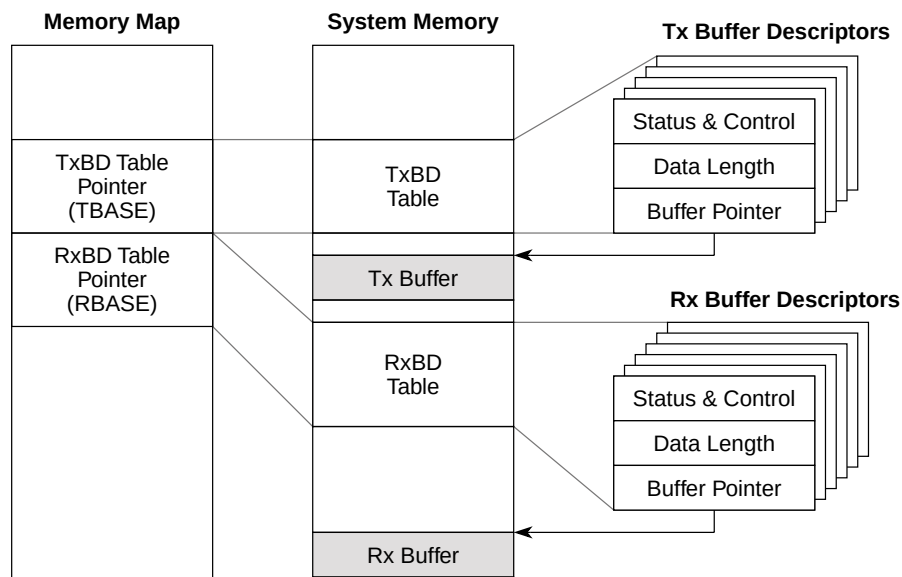


Figure 21-47. Example of 10/100 controller Memory Structure for 8-byte BD

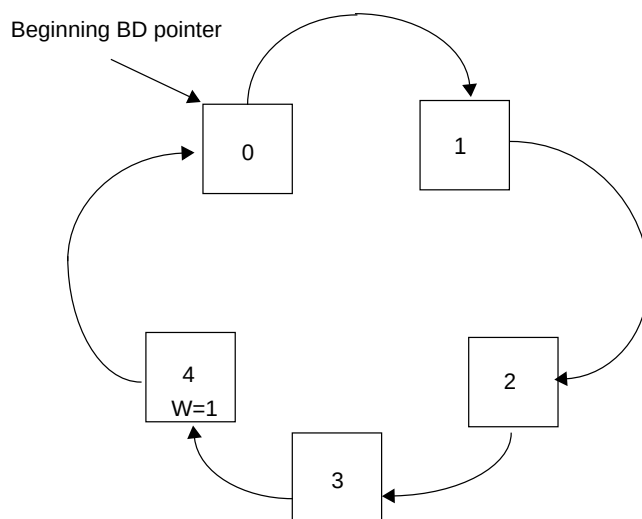


Figure 21-48. Buffer Descriptor Ring

21.6.3.2 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the 10/100 controller for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The 10/100 controller 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.

	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	DEF	0	HFE/ LC	RL	RC				UN		0
Offset + 2	DATA LENGTH																
Offset + 4	TX DATA BUFFER POINTER																
Offset + 6																	

Figure 21-49. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 21-51.

Table 21-51. Transmit Data Buffer Descriptor (TxBd) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	R	Ready, written by 10/100 controller 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 10/100 controller clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, which has been prepared for transmission by the user, has not been 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 when it is set in the first BD and MAC_CONFIG2[PAD/CRC enable] = 0). If MAC_CONFIG2[PAD/CRC enable] = 1, 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. Unlike the MPC8260 which PADs up to MINFLR value, 10/100 controller 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 EVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bit can cause an interrupt if they are enabled (IEVENT[TXBEN] or IEVENT[TXFEN]).
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 it is set in first BD and TxBd[PAD/CRC] = 0 and the MACCFG2[PAD/CRC enable] = 0 and MACCFG2[CRC enable] = 0.) If MACCFG2[PAD/CRC enable] = 1 or MACCFG[CRC enable] = 1, 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.
Offset + 0	6	DEF	Hardware updates this bit after transmitting a frame when used as a “Defer indicator”. Software/user updates this bit while building a transmit buffer descriptor when used as a “Hardware Event indicator”. DEF—Defer indication, written by hardware. 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.
Offset + 0	7	—	Reserved

Table 21-51. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	8	HFE/LC	Huge Frame Enable(written by user)/Late collision (written by 10/100 controller) Huge Frame Enable (written by user). Valid only when it is set in first BD and the MACCFG2[Huge Frame] = 0. If MACCFG2[Huge Frame] = 1, 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 10/100 controller). 0 No late collision. 1 A collision occurred after 64 bytes are sent. The 10/100 controller terminates the transmission and updates LC
Offset + 0	9	RL	Retransmission Limit, written by 10/100 controller. 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 10/100 controller terminates the transmission and updates RL.
Offset + 0	10–13	RC	Retry Count, written by 10/100 controller. 0 The frame is sent correctly the first time. x More than zero attempts where needed to send the transmit frame. When this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
Offset + 0	14	UN	Underrun, written by 10/100 controller. 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 10/100 controller terminates the transmission and updates UN.
Offset + 0	15	—	Reserved
Offset + 2	0–15	Data Length	Data length is the number of octets the 10/100 controller will transmit from this BD's data buffer. It is never modified by the 10/100 controller. 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.

21.6.3.3 Receive Buffer Descriptor (RxBD)

In the RxBD the user initializes the E, I and W bits in the first word and the pointer in the second word. When the data buffer has been used, the 10/100 controller 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 10/100 controller when the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed below.

	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 21-50. Receive Buffer Descriptor

Table 21-52. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty, written by the 10/100 controller (=0) and user (=1). 0 The data buffer associated with this BD has been filled with received data, or data reception has been 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 This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will 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] or IMASK[RXFEN]). If the user wants to be interrupted only when RXF occurs then the user should disable RXB0 (IMASK[RXBEN]=0 and enable RXF0 IMASK[RXFEN]=1.
Offset + 0	4	L	Last in frame, written by the 10/100 controller. 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 the 10/100 controller. 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 the 10/100 controller. (This bit is valid only if the L-bit is set and 10/100 controller is in promiscuous mode.) This bit is set by the 10/100 controller 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 10/100 controller. (Only valid if L = 1.) Will be set if the DA is broadcast (FF-FF-FF-FF-FF-FF).

Table 21-52. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	9	MC	Multicast. Written by 10/100 controller. (Only valid if L = 1.) Will be set if the DA is multicast and not BC.
Offset + 0	10	LG	Rx frame length violation, written by the 10/100 controller (only valid if L = 1). A frame length greater than maximum frame length was recognized. If MACCFG2[HUGE FRAME ENABLE] equals 0, 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.
Offset + 0	11	NO	Rx non-octet aligned frame, written by the 10/100 controller (only valid if L = 1). A frame that contained a number of bits not divisible by 8 was received.
Offset + 0	12	SH	Short frame, written by the 10/100 controller (only valid if L = 1). A frame length that was less than the minimum length defined for this channel (MINFLR) was recognized provided RCNTRL[RSF]=1.
Offset + 0	13	CR	Rx CRC error, written by the 10/100 controller (only valid if L = 1). This frame contains a CRC error and is an integral number of octets in length.
Offset + 0	14	OV	Overflow, written by the 10/100 controller (only valid if L = 1). A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and will be zero.
Offset + 0	15	TR	Truncation, written by the 10/100 controller (only valid if L = 1). Will be set if the receive frame is truncated. This can happen if the frame length greater than maximum frame length was received and the MACCFG2[Huge Frame] = 0. If the TR bit is set, the frame should be discarded and the other error bits should be ignored as they may be incorrect.
Offset + 2	0–15	Data Length	Data length (written by the 10/100 controller). Data length is the number of octets written by the 10/100 controller into this BD's data buffer if L = 0 (the value will be equal to the MRBLR), or the length of the frame including CRC if L = 1.
Offset + 4	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 64-byte aligned. The buffer must reside in memory external to the 10/100 controller.

NOTE

Software can only reuse an RxBD if the E bit is zero.

Index

Numerics

- 60x bus
 - data transfers
 - port size, 13-92
 - port size device interfaces, 13-93

A

- Accumulator, 6-37
- Address maps
 - addressing on PCI bus, 16-54
- Address translation
 - overview, 5-26
- Addressing
 - PCI bus
 - addressing, 16-56
 - configuration space, 16-56
 - I/O space, 16-56
 - memory space, 16-55
- AD_n (PCI address/data bus) signals, 16-55
- Alignment, byte, 16-57
- Applications
 - control processing, 1-20
 - examples, 1-20
- APUs, accumulator, 6-37
- Arbitration
 - I²C arbitration procedure, 11-14
 - PCI bus, 16-49
- Architecture
 - overview, 1-7
- Atomic update memory references, 5-28

B

- Block diagram, 11-1, 13-1
- Block diagrams
 - DMA controller, 15-1
 - RapidIO interface, 17-3
- Book E
 - 32-bit implementations, 6-3
 - interrupt registers, 6-16
- Boot page translation, 4-20
- Boot sequencer
 - general, 4-21

- overview, 1-15
- Burst operations
 - ordering
 - PCI cache wrap mode, 16-55
 - PCI linear incrementing, 16-55
 - PCI bus transfer, 16-52
- Bus operations
 - PCI bus transactions, *see* PCI interface
- Byte
 - enable signals, 16-8, 16-56, 16-57
 - ordering, PCI bus, 16-3
 - PCI alignment, 16-57
- Byte, alignment, 16-57

C

- $\overline{C}/\overline{BEn}$ (command/byte enable) signals, 16-8, 16-57, 16-75
- Cache
 - chance wrap mode, PCI, 16-55
 - control instructions, overview, 5-28
 - data
 - configuration, 7-2
 - organization, 7-3
 - on-chip implementation, 5-19
- CCSRBAR, updating, 4-17
- Clocking
 - boot page translation register (BPTR), 4-21
 - Ethernet, 4-16
 - introduction, 4-14
 - overview, 1-18
 - RapidIO, 4-15
 - real time, 4-16
 - system clock/PCI clock, 4-14
- Clocks
 - clock stretching, 11-15
 - clock synchronization, 11-14
 - signal description, 4-2
 - signals *see* Signals, clock
- Commands
 - PCI
 - $\overline{C}/\overline{BEn}$ signals, 16-53
 - encodings, 16-53
 - interrupt-acknowledge transaction, 16-73

- special-cycle, 16-73
- Communications processor module (CPM)
 - fast communications controllers (FCCs)
 - fast Ethernet mode
 - hash table effectiveness, 14-125
- Completion, PCI, 16-61
- Configuration
 - accessing registers
 - CONFIG_ADDR and CONFIG_DATA, 16-68
 - header registers
 - PCI registers, *see* Registers, PCI interface, 16-67
 - PCI configuration cycles
 - configuration space header, 16-66
 - type 0 and 1 accesses, 16-68
 - PCI space addressing, 16-56
 - status registers
 - processor bus error status registers, 16-74—16-76
- Configuration space, accessing alternate, 4-18
- Context ID registers, 20-25—20-26
- Core complex bus (CCB)
 - overview, 5-29
- Critical save/restore register 0, 6-16
- Critical save/store register 1, 6-16
- CSRR0 (critical save/restore register 0), 6-16
- CSRR1 (critical save/restore register 1), 6-16
- Current context ID register (CCIDR), 20-26

D

- Data
 - exception address register (DEAR), 6-17
- Data cache
 - configuration, 7-2
 - organization, 7-3
- DDR
 - chip select configuration register, 9-12
 - chip select memory bounds registers, 9-11
 - clock distribution, 9-43
 - data beat ordering, 9-50
 - debug configuration, 4-13
 - error checking and correcting (ECC), 9-51
 - error management, 9-53
 - external signal description, 9-3
 - features, 9-2
 - functional description, 9-30
 - initialization and application information, 9-54
 - memory data path
 - error injection mask
 - ECC register, 9-21
 - high register, 9-20
 - low register, 9-21
 - read capture
 - ECC register, 9-23
 - high register, 9-22

- low register, 9-23
- memory error
 - address capture register, 9-28
 - attributes capture register, 9-26
 - detect register, 9-24
 - disable register, 9-25
 - interrupt enable register, 9-25
 - interrupt enable register (ERR_INT_EN), 9-25
- memory map, 9-10
- modes of operation, 9-3
- page mode and logical bank retention, 9-50
- register descriptions, 9-11
- self-refresh in sleep modes, 9-48
- signals
 - clock interface, 9-9
 - debug, 9-10
 - detailed descriptions, 9-5
 - memory interface, 9-5
 - overview, 9-3
- single-bit ECC memory error management register, 9-29

DDR SDRAM

- address multiplexing, 9-37
- control configuration register, 9-17
- initialization sequence, 9-55
- interface operation, 9-35
- interval configuration register, 9-19
- JEDEC standard interface commands, 9-38
- mode configuration register, 9-18
- mode-set command timing, 9-43
- refresh, 9-46
- refresh and power-saving modes, 9-48
- refresh timing, 9-47
- registered DIMM mode, 9-44
- supported organizations, 9-36
- timing configuration register 2, 9-16
- timing configuration register1, 9-13
- write timing adjustments, 9-45

DDR SDRAM controller

- overview, 1-13
- DDR SDRAM interface debug, 20-29
- DEAR, 6-17
- Debug mode
 - block diagram, 20-1
 - DDR SDRAM interface, 20-4, 20-29
 - detailed signal descriptions, 20-6
 - external signal description, 20-5
 - features, 20-3
 - functional description, 20-27
 - information on debug Pins, 20-29
 - information on ECC pins, 20-30
 - L1_TSTCLK input signal, 20-10
 - L2_TSTCLK input signal, 20-10
 - local bus interface, 20-4, 20-30

- LSSD_MODE input signal, 20-10
- memory data valid (MDVAL) output, 20-6
- memory ECC (MECC{0-7}) output, 20-6
- memory map and register definition, 20-10
- memory source ID (MSRCID{0-4}) output, 20-7
- operation, 20-3
- overview, 20-2
- PCI address (PCI_AD{63-59}) output signal, 20-8
- PCI/PCI-X
 - interface, 20-29
- PCI/PCI-X interface, 20-4
- TCK (JTAG test clock) signal, 20-8
- TDI (JTAG test data input) signal, 20-9
- TDO (JTAG test data output) signal, 20-9
- test and configuration signals, 20-8
- TEST_SEL input signal, 20-10
- TMS (JTAG test mode select) signal, 20-9
- trace buffer, 20-5
- trigger in (TRIG_IN) input signal, 20-7
- trigger out (TRIG_OUT) output signals, 20-8
- watchpoint monitor, 20-4
- watchpoint trigger signals, 20-7
- Decrementer auto-reload register (DECAR), 6-15
- descriptor queue/enqueue pointer address register (ODQEPAR), 17-74
- DEVSEL (device select) signal, 16-8, 16-56
- DHn/DLn (data bus) signals, 9-6
- Diagram
 - block, 11-1, 13-1
- Diagrams
 - debug block, 20-1
- DMA controller
 - alternate mode transfer, 15-34
 - bandwidth control, 15-37
 - basic chaining mode, 15-31
 - basic chaining single-write start mode, 15-32
 - basic channel continue mode, 15-36
 - basic direct mode, 15-30
 - basic direct single-write start mode, 15-31
 - basic mode transfer, 15-30
 - block diagram, 15-1
 - byte count registers (BCRn), 15-23
 - chaining mode, extended, 15-33
 - chaining single-write start mode, extended, 15-34
 - channel abort, 15-36
 - channel continue mode, cascading transfer chains, 15-35
 - channel continue mode, extended, 15-36
 - channel operation, 15-29
 - channel state, 15-37
 - control mode transfer, external, 15-34
 - current link descriptor address registers (CLNDARn), 15-14
 - current list-alternate base descriptor address registers (CJABDARn), 15-24
 - descriptors, 15-38
 - destination
 - address registers (DARn), 15-21
 - address registers for RapidIO maintenance writess (DARMn), 15-22
 - attributes registers (DATRn), 15-19
 - stride registers (DSRn), 15-26
 - direct mode, extended, 15-33
 - direct single-write start mode, extended, 15-33
 - errors, 15-38
 - features, 15-2
 - functional description, 15-29
 - general status register (DGSR), 15-27
 - limitations and restrictions, 15-41
 - memory map/register definition, 15-7
 - mode registers (MRn), 15-10
 - mode transfer, extended, 15-33
 - modes of operation, 15-2
 - next link descriptor address registers (NLNDARn), 15-23
 - next list descriptor address register (NLSDARn), 15-25
 - overview, 15-2
 - register descriptions, 15-10—15-29
 - register summary, 15-8
 - signal description, external, 15-5
 - source address registers (SARn), 15-18
 - source address registers for RapidIO maintenance reads (SARMn), 15-18
 - source attributes registers (SATRn), 15-16
 - source stride registers (SSRn), 15-26
 - status registers (SRn), 15-13
 - stride size and distance, 15-37
 - system considerations, 15-42
 - transfer interfaces, 15-38
 - unusual scenarios, 15-44
- Doze mode, 1-18
- Dual universal asynchronous receiver/transmitter (DUART)
 - overview, 1-15
- DUART
 - alternate function registers (UAFR0 and UAFR1), 12-17
 - baud-rate generator logic, 12-21
 - data transfer, 12-20
 - detailed signal descriptions, 12-3
 - divisor most and least significant byte registers (UDMB, UDLB), 12-7
 - DMA mode select, 12-23
 - DMA status registers (UDSR0 and UDSR1), 12-17
 - errors, 12-22
 - external signal description, 12-3

- features, 12-2
- FIFO
 - control registers (UFCR0 and UFCR1), 12-11
 - interrupts, 12-23
 - mode, 12-22
- framing error, 12-22
- functional description, 12-19
- initialization sequence, 12-24
- initialization/application information, 12-24
- interrupt control logic, 12-23
- interrupt enable register (UIER), 12-8
- interrupt ID registers (UIIR0 and UIIR1), 12-9
- line control registers (ULCR0 and ULCR1), 12-12
- line status registers (ULSR0 and ULSR1), 12-14
- local loop-back mode, 12-21
- memory map/register definition, 12-4
- MODEM control registers (UMCR0 and UMCR1), 12-13
- MODEM status registers (UMSR0 and UMSR1), 12-16
- modes of operation, 12-3
- overflow error, 12-22
- overview, 12-1
- parity bit, 12-20
- parity error, 12-22
- receiver buffer registers (URBR0 and URBR1), 12-6
- register descriptions, 12-5—12-19
- scratch registers (USCR0 and USCR1), 12-16
- serial interface, 12-19
- START bit, 12-20
- STOP bit, 12-21
- transmitter holding registers (UTHR0 and UTHR1), 12-6

E

- e500 coherency module (ECM)
 - overview, 1-13
- e500 core overview, 1-7
- ECM
 - CCB arbiter, 8-9
 - CCB interface, 8-9
 - error address capture register (EEADR), 8-8
 - error attributes capture register (EEATR), 8-6
 - error detect register (EEDR), 8-5
 - error enable register (EEER), 8-5
 - features, 8-2
 - global data multiplexor, 8-9
 - I/O arbiter, 8-9
 - initialization/application information, 8-10
 - memory map/register definition, 8-2
 - overview, 8-1
 - register descriptions, 8-3—8-8

- transaction queue, 8-9
- Embedded floating-point instructions, 5-11
- Endianness
 - differences, 5-32
- Errors
 - PCI interface
 - address/data parity, 16-76
 - transactions, 16-74
 - reporting
 - address/data parity, 16-62
 - PCI bus, 16-74
 - $\overline{\text{PERR}}$ and $\overline{\text{SERR}}$ signals, 16-76
 - processor bus status register, 16-74
 - target-initiated termination, 16-61
 - retry transactions, 16-61
 - target-abort, 16-61
 - target-disconnect, 16-61
- ESR, 6-18
- Ethernet clocks, 4-16
- Ethernet controller
 - 8-byte receive buffer descriptor (RxBD), 21-68
 - 8-byte transmit data buffer descriptor (TxBD), 21-66
 - buffer descriptors, 21-64
 - channel operation, 21-54
 - connecting the 10/100 to the physical interface, 21-52
 - current receive buffer descriptor pointer low (CRBPTL), 21-31
 - current transmit buffer descriptor pointer low (CTBPTL), 21-24
 - data buffer descriptor, 21-65
 - destination address recognition, 21-58
 - detailed signal descriptions, 21-8
 - DMA control register (DMACTRL), 21-20
 - error disabled register (EDIS), 21-18
 - external signals description, 21-7
 - features, 21-6
 - flow control, 21-61
 - frame recognition, 21-58
 - frame transmission, 21-55
 - functional description, 21-52
 - gigabit frame reception, 21-57
 - group address registers 0–7 (GADDRx), 21-49
 - half duplex register (HAFDUP), 21-41
 - hardware controlled initialization, 21-54
 - hash function registers, 21-48
 - hash table algorithm, 21-60
 - individual address registers 0–7 (IADDRx), 21-48
 - initialization sequence, 21-54
 - interface status register (IFSTAT), 21-46
 - internal and external loop back, 21-63
 - inter-packet gap time, 21-63

- inter-packet gap/inter-frame gap register (IPGIFG), 21-40
- interrupt event register (IEVENT), 21-15
- interrupt handling, 21-61
- interrupt mask register (IMASK), 21-17
- introduction, 21-1
- MAC configuration 1 register (MACCFG1), 21-37
- MAC configuration 2 register (MACCFG2), 21-38
- MAC functionality, 21-34—21-37
- maximum frame length register (MAXFRM), 21-42
- maximum receive buffer length R0R1 register (MRBLR0R1), 21-32
- media independent interface (MII), 21-52
- memory map/register definition, 21-10
- MII management
 - address register, 21-44
 - command (MIIMCOM), 21-43
 - configuration register (MIIMCFG), 21-42
 - control register (MIIMCON), 21-44
 - indicator register (MIIMIND), 21-46
 - status register (MIIMSTAT), 21-45
- minimum frame length register (MINFLR), 21-19
- modes of operation, 21-6
- out-of-sequence Tx data buffer pointer register (OSTBDP), 21-28
- out-of-sequence TxBD register (OSTBD), 21-26
- overview, 21-5
- pause time value register (PTV), 21-20
- receive buffer descriptor pointer
 - low 0-3 (RBPTRLx), 21-33
- receive control register (RCTRL), 21-29
- receive status register (RSTAT), 21-30
- register descriptions, 21-15—21-51
- RxBD data length register (RBDLEN), 21-31
- signals
 - overview, 21-7
- station address register
 - part 1 (MACSTNADDR1), 21-47
 - part 2 (MACSTNADDR2), 21-47
- top-level module memory map, 21-11
- transmit
 - buffer descriptor pointer low (TBPTRL), 21-25
 - control register (TCTRL), 21-22
 - descriptor base address low (TBASEL), 21-25
 - status register (TSTAT), 21-23
- TxBD data length register (TBDLEN), 21-24
- user initialization, 21-54
- Ethernet mode
 - fast communications controller (FCC)
 - hash table effectiveness, 14-125
- Exception handling
 - extended model, 5-19
 - overview, 5-31

- Exceptions
 - syndrome register, 6-18

F

- Fast communications controllers (FCCs)
 - fast ethernet mode
 - hash table effectiveness, 14-125
- Features list, 5-5
- FRAME signal, 16-52
- Full-on mode, 1-18

G

- General-purpose registers (GPRs), 6-8
- Global control register 0, 5-29
- Global control register 0 (PMGC0), 6-38
- Global utilities
 - alternate function signal multiplex control register (PMUXCR), 18-14
 - clock control, 18-2
 - clock out control register (CLKOCR), 18-21
 - DDR DLL control register (DDRDLPCR), 18-21
 - detailed signal descriptions, 18-3
 - device disable register (DEVDIR), 18-15
 - doze mode, 18-24
 - dynamic power management, 18-24
 - external signal description, 18-2
 - features, 18-1
 - functional description, 18-23
 - general-purpose I/O signals, 18-2
 - I/O control register, general-purpose (GPIOCR), 18-11
 - I/O signals, general-purpose, 18-27
 - input data register, general-purpose (GPINDR), 18-13
 - interrupt and local bus signal multiplexing, 18-2
 - interrupts and power management, 18-26
 - local bus DLL control register (LBDLLCR), 18-22
 - machine check summary register (MCPSUMR), 18-19
 - memory map/register definition, 18-4
 - nap mode, 18-25
 - output data register, general-purpose (GPOUTDR), 18-12
 - overview, 18-1
- POR
 - boot mode status register (PORBMSR), 18-6
 - configuration register, general-purpose (GPPORCR), 18-11
 - debug mode status register (PORDBGMSR), 18-10
 - device status register (PORDEVSR), 18-9
 - I/O impedance status and control register (PORIMPSCR), 18-8

- PLL status register (PORPLLSR), 18-5
- power management
 - and device disables, 18-1
 - control and status register (POWMGTCSR), 18-17
 - control bits, 18-25
 - functional description, 18-23
- processor version register (PVR), 18-20
- register descriptions, 18-5—18-23
- shutting down unused blocks, 18-24
- signals overview, 18-2
- sleep mode, 18-25
- snooping, power-down state, 18-26
- software access to current POR configuration
 - settings, 18-1
- software considerations for power management, 18-26
- software-controlled power-down states, 18-24
- system version register (SVR), 18-20
- $\overline{\text{GNT}}$ (PCI bus grant) signal, 16-49

I

- I²C interface
 - arbitration procedure, 11-14
 - clock stretching, 11-15
 - clock synchronization, 11-14
 - data transfer, 11-13
 - features, 11-2
 - handshaking, 11-15
 - signals, 11-4
 - slave address transmission, 11-12
 - start condition, 11-12
 - start generation, 11-22
 - stop condition, 11-13
- Interrupt and local bus signal multiplexing, 18-27
- Initialization/application information, 9-54, 10-49, 13-88
- Instruction fetch
 - stages, 5-15
- Instruction flow, 5-13
- Instruction pipeline stages
 - complete, 5-17
 - decode/dispatch, 5-16
 - execute, 5-17
 - instruction fetch, 5-15
 - issue queues (FIQ, VIQ, GIQ), 5-16
 - write-back, 5-17
- Instruction set
 - compatibility, 5-31
 - overview, 5-11
- Instructions
 - cache control, e500 core, 5-28
 - scalar floating-point, 5-11

- Integer
 - exception registers, 6-8
- Intel PC133
 - SDRAM interface commands, 13-57
- Interrupt classes
 - categories, 5-20
- Interrupt registers
 - defined by Book E, 6-16
 - overview, 5-22
- Interrupts
 - and exception handling registers
 - ESR, 6-18
 - MSR, 6-10
 - PIC
 - messaging, 10-48
 - nesting, 10-47
 - vector offset registers, 6-17
 - vector prefix register, 6-17
- $\overline{\text{IRDY}}$ (initiator ready) signal, 16-52
- IVORs, 6-17
- IVPR, 6-17

J

- JEDEC-standard SDRAM interface commands, 13-57
- JTAG test reset ($\overline{\text{TRST}}$) signal, 20-9

K

- Key features, 1-2

L

- L2 cache
 - allocation of lines, 7-27
 - coherency rules, 7-22
 - control register (L2CTL), 7-6
 - error control and capture registers, 7-14—7-21
 - error injection registers, 7-12
 - external write address registers (L2CEWAR_n), 7-9
 - external write control registers
 - (L2CEWCRO—L2CEWCRO3), 7-10
 - flash clearing, instruction and data locks, 7-25
 - locking
 - clearing on selected lines, 7-24
 - entire, 7-23
 - programmed memory ranges, 7-24
 - selected lines, 7-24
 - with stale data, 7-25
 - memory map/register definition, 7-5
 - memory-mapped SRAM
 - base address registers (L2SRBAR_n), 7-11
 - coherency rules, 7-23
 - operation, 7-27
 - organization, 7-2

- overview, 7-1
- PLRU bit update considerations, 7-26
- register descriptions, 7-6—7-21
- replacement policy, 7-25
- SRAM features, 7-2
- states, 7-27
- timing, 7-21
- transitions, 7-28
- LBC**
 - activate to read/write interval, 13-60
 - address and address space checking, 13-38
 - atomic operation, 13-41
 - base registers (BR0–7), 13-11
 - boot chip-select operation, 13-54
 - bus clock and clock ratios, 13-4
 - bus monitor, 13-42
 - chip-select and write enable negation timing, 13-48
 - chip-select assertion timing, 13-47
 - clock ratio register (LCRR), 13-35
 - data buffer control (LBCTL), 13-41
 - data transfer acknowledge (TA), 13-40
 - extended hold Time on read accesses, 13-52
 - external access termination, 13-54
 - external signal description, 13-5
 - features, 13-2
 - general-purpose chip-select machine (GPCM), 13-42
 - local bus configuration register (LBCR), 13-34
 - memory data register (MDR), 13-24
 - memory map/register definition, 13-10
 - memory refresh timer prescaler register (MRTPR), 13-23
 - modes of operation, 13-3
 - option registers (ORx), 13-13
 - output enable timing, 13-52
 - overview, 13-2
 - page hit checking, 13-58
 - page management, 13-58
 - parity generation and checking (LDP), 13-42
 - power-down mode, 13-4
 - precharge-to-activate interval, 13-60
 - programmable wait state configuration, 13-48
 - register descriptions, 13-11
 - relaxed timing, 13-49
 - SDRAM machine, 13-55
 - SDRAM machine mode register (LSDMR), 13-25
 - SDRAM refresh timer (LSRT), 13-28
 - signal descriptions, 13-5
 - signals
 - external address latch enable (LALE), 13-38
 - source ID debug mode, 13-4
 - timing configuration, 13-43
 - transfer error address register (LTEAR), 13-34
 - transfer error attributes register (LTEATR), 13-33
 - transfer error check disable register (LTEDR), 13-31
 - transfer error interrupt enable register (LTEIR), 13-32
 - transfer error status register (LTESR), 13-29
 - unctional description, 13-37
 - UPM memory address register (MAR), 13-20
 - UPM mode registers (MxMR), 13-20
 - UPM refresh timer (LURT), 13-27
- Local bus interface debug, 20-30
- Local control A registers (PMLCa0–PMLCa3), 6-39
- Local control B registers (PMLCb0–PMLCb3), 6-40
- Local control registers
 - A 0 – 3 (PMLCa0–PMLCa3), 6-39
 - B 0 – 3 (PMLCb0–PMLCb3), 6-40
 - overview, 5-30
- LR, 6-10
- M**
 - MAA (memory address attribute) signals, 20-7
 - Machine check interrupt, 5-21
 - Machine state register (MSR), 6-10
 - Master-abort termination, PCI, 16-61
 - Memory access
 - ordering overview, 5-28
 - Memory coherency
 - e500 core, 5-27
 - Memory management
 - overview, 5-24, 5-32
 - Memory map
 - accessing CCSR memory from external masters, 2-11
 - address translation and mapping, 2-3
 - CCSR map, 2-16
 - CCSR organization, 2-12
 - configuration, control, and status registers, 2-10—2-16
 - configuring local access windows, 2-8
 - device-specific utilities registers, 2-15
 - distinguishing local access windows from other mapping functions, 2-8
 - general utilities registers, 2-12
 - illegal interaction between inbound ATMUs and local access windows, 2-10
 - illegal interaction between local access windows and DDR SDRAM chip selects, 2-9
 - inbound address translation and mapping windows, 2-9
 - interrupt controller and CCSR, 2-14
 - local access register, 2-5
 - local access window *x* attributes registers (LAWAR0–LAWAR7), 2-7
 - local access window *x* base address registers (LAWBAR0–LAWBAR7), 2-6

- local access windows
 - introduction, 2-4
 - precedence, 2-8
- outbound address translation and mapping
 - windows, 2-9
- overview and example, 2-1
- PCI/PCI-X inbound ATMU, 2-10
- RapidIO inbound ATMU, 2-9
- SRAM windows, 2-4
- window into configuration space, 2-4
- Memory map/register definitions
 - L2 cache, 7-5
- Memory maps
 - Ethernet controller, 21-10
 - LBC, 13-10
- Memory references, atomic update, 5-28
- Memory subsystem
 - overview, 5-31
- MMU
 - assist registers (MAS0–MAS4 and MAS6), 6-23
 - assist registers (MAS1–MAS4 and MAS6), 5-26
 - registers, 6-20
 - TLB
 - coherency, 5-27
- Modes
 - basic
 - chaining, 15-31
 - chaining single-write start, 15-32
 - direct, 15-30
 - direct single-write start, 15-31
 - transfer, 15-30
 - cache wrap, 16-55
 - doze, 18-24
 - nap, 18-25
 - sleep, 18-25
- MPC8240
 - possible applications, 1-20
 - uses, 1-20
- MPC8245
 - as PCI bus master, 16-4
 - as PCI target, 16-4

N

- Nap mode description, 1-18

O

- On-chip cache implementation, 5-19
- On-chip memory
 - as L2 Cache, 1-12
 - as mapped SRAM, 1-12
 - unit introduction, 1-12
- Overflow (OV), 6-8

P

- Page characteristics
 - e500 core, 5-28
- PAR (PCI parity) signal, 16-75
- PCI
 - arbiter configuration, 4-11
 - debug configuration, 4-12
 - I/O impedance, 4-11
- PCI bus
 - device as initiator, 16-4
 - device as target, 16-4
 - external signal descriptions, 16-6
 - features, 16-4
 - host/agent/agent lock mode, 16-5
 - introduction, 16-1
 - modes of operation, 16-5
 - overview, 16-2
 - PCI-X signal output hold timing, 16-6
- PCI interface
 - accessing registers, see Registers, 16-69
 - address bus decoding, 16-55
 - address/data parity errors, 16-62
 - agent config lock mode, 16-77
 - burst operation, 16-52
 - bus arbitration, 16-49
 - bus commands, 16-53
 - bus protocol, 16-52
 - bus transactions
 - interrupt-acknowledge transaction, 16-73
 - read transactions, 16-58
 - special-cycle transaction, 16-73
 - timing diagrams, 16-57
 - transaction termination, 16-60
 - master-initiated, 16-60
 - target-initiated, 16-61
 - write transactions, 16-59
 - bus transactions, see PCI interface
 - byte alignment, 16-57
 - byte ordering, 16-3
 - $\overline{C}/\overline{BE}$ signals, 16-75
 - $\overline{C}/\overline{BE}n$ signals, 16-75
 - cache wrap mode, 16-55
 - command register, 16-67
 - configuration cycles
 - configuration header, 16-67
 - type 0 and 1 accesses, 16-68
 - configuration space addressing, 16-56
 - data transfers, 16-52, 16-57
 - error detection and reporting, 16-74
 - error transactions, 16-74
 - fast back-to-back transactions, 16-64
 - host mode, 16-77
 - I/O space addressing, 16-56

- linear incrementing, 16-55
- master-initiated transaction termination, 16-61
- memory space addressing, 16-55
- MPC8245 as PCI bus master, 16-4
- MPC8245 as PCI target, 16-4
- PCI commands
 - $\overline{C}/\overline{BEN}$ signals, 16-53
 - command summary, 16-53
 - encodings, 16-53
 - interrupt-acknowledge transaction, 16-73
 - special-cycle, 16-73
- power-on reset configuration modes, 16-76
- read transactions, 16-58
- registers, *see* Registers, PCI interface, 16-67
- retry PCI transactions, 16-62
- signals
 - $\overline{DEVSEL}$, 16-8
 - $\overline{TRDY}$, 16-13
- signals, *see* Signals, PCI interface
- special cycle command, 16-74
- special-cycle operations, 16-73, 16-74
- target-abort error, 16-62
- target-disconnect, 16-4, 16-61
- target-disconnect termination, 16-61
- target-initiated termination, 16-62
- transaction termination, 16-57, 16-60
- turnaround cycle, 16-57
- write transactions, 16-59
- PCI/PCI-X
 - and RapidIO, bridging between, 16-78
 - inbound ATMU windows, 2-10
 - output hold configuration, 4-13
- PCI/PCI-X interface debug, 20-29
- PCI-X
 - configuration, 4-12
 - nonposted writes, 16-77
- Performance monitor
 - burstiness, 19-15
 - control registers, 19-4—19-10
 - counter
 - chaining, 19-14
 - registers, 19-10
 - triggering, 19-14
 - counters (PMC_n), 19-10
 - event counting, 19-12
 - events, 19-17—19-37
 - examples, 19-38
 - external signals, 19-3
 - features, 19-2
 - functional description, 19-12
 - global control register (PMGC0), 19-5
 - interrupt, 19-12
 - local control registers (PMLC_{An} and PMLC_{Bn}), 19-5
 - mask registers
 - even (PMMR0,2,4,6), 10-29
 - odd (PMMR1,3,5,7), 10-30
 - memory map and register definition, 19-3
 - overview, 19-1
 - purposes, 5-5
 - threshold events, 19-13
- Performance monitor APU
 - counter registers, 5-30, 6-41
- Performance monitoring, overview, 5-29
- PERR (PCI parity error) signal, 16-76
- PIC
 - changing interrupt source configuration, 10-51
 - feature reporting register (FRR), 10-15
 - features, 10-4
 - functional description, 10-45
 - global registers, 10-15—10-19
 - configuration (GCR), 10-16
 - processor initialization (PIR), 10-17
 - spurious vector (SVR), 10-19
 - vendor identification (VIR), 10-16
 - global timer registers, 10-20—10-25, ??—10-26
 - base count (GTBCR_n), 10-21
 - control (TCR), 10-24
 - current count (GTCCR_n), 10-20
 - destination (GTDR), 10-23
 - frequency reporting (TFRR), 10-20
 - vector/priority (GTVPR_n), 10-22
 - initialization/application information, 10-49
 - internal interrupt sources, 10-6
 - interrupt acknowledge, 10-47
 - interrupt source configuration registers, 10-33—10-39
 - external interrupt destination (EIDR0 to EIDR11), 10-34
 - external interrupt vector/priority (EIVPR0 to EIVPR11), 10-33
 - internal interrupt destination (IIDR0 to IIDR31), 10-36
 - internal interrupt vector/priority (IIVPR0 to IIVPR31), 10-35
 - messaging interrupt destination (MIDR0 to MIDR3), 10-38
 - messaging interrupt vector/priority (MIVPR0 to MIVPR3), 10-37
 - interrupt source priority, 10-46
 - introduction, 10-1
 - IPI vector/priority registers (IPIVPR_n), 10-18
 - memory map and register definition, 10-9—10-45
 - message registers
 - enable(MER), 10-31
 - status (MSR), 10-32
 - message registers (MSGR0 to MSGR3), 10-31
 - messaging interrupts, 10-48

- modes
 - introduction, 10-5
 - mixed (GCR{M} set), 10-5
 - pass-through (GCR{M} cleared), 10-5
- MSG, IPI, and timer priority levels, 10-51
- nesting of interrupts, 10-47
- overview, 10-4
- per CPU registers, 10-39—10-45
 - interprocessor interrupt dispatch (IPIDR0–IPIDR3), 10-41
 - processor current task priority (CTPR), 10-42
 - processor end of interrupt (EOI), 10-45
 - processor interrupt acknowledge (IACK), 10-44
 - who am I (WHOAMI), 10-43
- processor current task priority, 10-47
- processor initialization, 10-47
- programming guidelines, 10-49
- registers, programming, 10-49
- reset, 10-49
- signal descriptions, 10-7—10-9
- signals overview, 10-7
- spurious vector generation, 10-48
- summary registers, 10-26—10-29
 - critical interrupt 0, 10-27
 - critical interrupt 1, 10-28
 - IRQ_OUT 0 (IRQSR0), 10-26
 - IRQ_OUT 1 (IRQSR1), 10-27
- timers, 10-48
- PIR, 6-12, 6-20
- Power management
 - global utilities, 18-23
 - nap mode, 16-74
 - overview, 1-18
 - PCI special-cycle operations, 16-73
- Power-on reset
 - output signal state, 3-16
- PowerPC architecture legacy support overview, 5-30
- Process ID registers (PID)
 - overview, 5-27
- Processor ID register (PIR), 6-20
- Programmable interrupt controller (PIC)
 - overview, 1-14
- Programmable page characteristics, 5-28
- Programmed context ID register (PCIDR), 20-25
- Programming model
 - e500 core, 5-18, 6-2
 - overview, 5-17
- PVR, 6-12

R

- RapidIO
 - and CCSR, 2-14
 - and PCI/PCI-X, bridging between, 16-78
 - clocks, 4-15
 - device ID, 4-10
 - inbound ATMU windows, 2-9
 - transmit clock source, 4-10
- RapidIO interconnect
 - overview, 1-18
- RapidIO interface
 - 8/16 LP-LVDS layer functionality lists, 17-95
 - 8/16 LP-LVDS port maintenance block header 0
 - command and status register (PMBH0CSR), 17-26
 - accept-all mode, 17-6
 - architectural registers, 17-14—17-33
 - assembly identity capability register (AIDCAR), 17-15
 - assembly information capability register (AICAR), 17-16
 - ATMU (address translation and mapping unit), 17-112
 - ATMU registers, 17-36—17-44
 - base device ID command and status register (BDIDCSR), 17-24
 - basic functionality list, 17-95
 - block diagram, 17-3
 - common transport layer
 - basic fFunctionality list, 17-103
 - detectable errors list, 17-104
 - packet transmission list, 17-103
 - common transport layer packet reception list, 17-103
 - component tag command and status register (CTCSR), 17-25
 - configuration register (CR), 17-33
 - CRC checking modes, 17-5
 - data message controller, 17-117
 - data message controller limitations and restrictions, 17-125
 - destination operations capability register (DOCAR), 17-19
 - detailed signal descriptions, 17-8
 - device identity capability register (DIDCAR), 17-14
 - device information capability register (DICAR), 17-14
- doorbell
 - message controller, 17-125
 - mode register (DMR), 17-80
 - queue/dequeue pointer address registers (DQDPAR), 17-83
 - queue/enqueue pointer address registers (DQEPAR), 17-84
 - registers, 17-80—17-85
 - status register (DSR), 17-82
- error checking disable mode, 17-6
- error management registers, 17-44—17-66

- errors, 17-106
- external signal description, 17-6
- features, 17-4
- features not implemented, 17-5
- functional description, 17-88
- functionality, 17-94
- general device
 - link maintenance list, 17-97
 - nonrecoverable errors list, 17-102
 - packet reception list, 17-98
 - packet transmission list, 17-97
 - recoverable errors list, 17-100
- general specification functionality list, 17-94
- host base device ID lock command and status register (HBDIDLCSR), 17-25
- inbound
 - frame queue/dequeue pointer address register (IFQDPAR), 17-79
 - frame queue/enqueue pointer address register (IFQEPAR), 17-79
 - message registers, 17-75—17-85
 - mode register (IMR), 17-75
 - status register (ISR), 17-77
 - window attributes registers 0–4 (RIWAR_n), 17-42
 - window base address registers 1–4 (RIWBAR_n), 17-42
 - window translation address registers 0–4 (RIWTAR_n), 17-41
- inbound ATMU
 - bypass mode, 17-114
 - LCSRBAR window, 17-114
 - translation, 17-114
- inbound special transactions and requirements, 17-115
- initialization and application information, 17-129
- introduction, 17-1
- link response time-out disable mode, 17-6
- local configuration space base address 1 command and status register (LCSBA1CSR), 17-23
- logical layer
 - basic functionality list, 17-104
 - detectable errors list, 17-106
 - source transaction support list, 17-106
 - target transaction support list, 17-105
- mailbox command and status register (MSR), 17-21
- memory map/register definition, 17-9
- message unit, 17-116
 - chaining mode operation, 17-118
 - description, 17-117
 - descriptor format, 17-122
 - direct mode operation, 17-118
 - doorbell controller interrupts, 17-127
 - doorbell queue entry format, 17-127
 - error response conditions, 17-125
 - features, 17-116
 - inbound doorbell reception, 17-126
 - inbox controller interrupts, 17-125
 - inbox controller operation, 17-123
 - operational modes, 17-117
 - outbox controller operation, 17-118
 - port-write controller structure, 17-128
 - response conditions, 17-127
 - retry response conditions, 17-125
 - special error case condition, 17-123
- message unit registers, 17-67
- operational modes, 17-5
- outbound, 17-74
 - descriptor queue/dequeue pointer address register (ODQDPAR), 17-70
 - destination attributes register (ODATR), 17-72
 - destination port register (ODPR), 17-72
 - double-word count register (ODCR), 17-73
 - message registers, 17-67—17-75
 - mode register (OMR), 17-67
 - special transactions and requirements, 17-113
 - status register (OSR), 17-69
 - unit source address register (OSAR), 17-71
- outbound ATMU bypass mode, 17-113
- outbound ATMU translation, 17-113
- outbound window
 - attributes registers 0–8 (ROWAR_n), 17-39
 - base address registers 1–8 (ROWBAR_n), 17-38
 - translation address registers 0–8 (ROWTAR_n), 17-36
 - translation address registers 0–8 for maintenance transactions (ROWTAR_n), 17-38
- output port driver disable mode, 17-6
- overview, 17-7
- packet error capture register 1
 - type 1 (PECR1T1), 17-52
 - type 10 request (PECR1T10), 17-56
 - type 11 (PECR1T11), 17-57
 - type 13 (PECR1T13), 17-57
 - type 2 (PECR1T2), 17-53
 - type 5 (PECR1T5), 17-53
 - type 6 (PECR1T6), 17-54
 - type 8 request (PECR1T8RQ), 17-55
 - type 8 response (PECR1T8R), 17-55
- packet error capture register 2
 - type 10 (PECR2T10), 17-61
 - type 6 (PECR2T6), 17-59
 - type 8 (PECR2T8), 17-60
 - type 8 response (PECR2T8R), 17-60
 - types 1, 2, and 5 (PECR1T125), 17-58
 - types 11 and 13 (PECR2T11), 17-62
- packet response time-out disable mode, 17-6
- port configuration register (PCR), 17-34

- port control command and status register (PCCSR), 17-32
- port error
 - and status command and status register (PESCSR), 17-31
 - control register (PECR), 17-50
 - injection register (PEIR), 17-35
 - packet register 1 (PEPR1), 17-52
 - packet register 2 (PEPR2), 17-58
 - packet/control symbol register 0 (PEPCSR0), 17-51
 - recovery threshold register (PERTR), 17-65
- port general control command and status register (PGCCSR), 17-28
- port link
 - maintenance request command and status register (PLMREQCSR), 17-29
 - maintenance response command and status register (PLMRESPCSR), 17-29
 - time-out control command and status register (PLTOCCSR), 17-27
- port local ackID status command and status register (PLASCSR), 17-30
- port notification/fatal error
 - detect register (PNFEDR), 17-44
 - disable register (PNFEDiR), 17-47
 - interrupt enable register (PNFEIER), 17-49
- port recoverable error detect register (PREDR), 17-62
- port response time-out control command and status register (PRTOCCSR), 17-27
- port retry threshold register (PRTR), 17-66
- port-write
 - and doorbell command and status register (PWDCSR), 17-22
 - mode register (PWMR), 17-85
 - queue base address register (PWQBAR), 17-87
 - registers, 17-85—17-88
 - status register (PWSR), 17-86
- processing element features capability Register (PEFCAR), 17-16
- processing element logic layer control command and status register (PELLCCSR), 17-23
- source operations capability register (SOCAR), 17-18
- switch port information capability register (SPICAR), 17-17
- terminology, 17-2
- transaction, packet, and control symbols, 17-88
- transmit clock-select mode, 17-5
- RapidIO registers, 17-9—17-14
- Real time clock, 4-16
- Register model
 - 32-bit Book E implementations, 6-3
 - branch operations, 6-8
 - condition register (CR), 6-9
 - count register (CTR), 6-10
 - time base, 6-15
- Registers
 - accessing configuration, control, and status, 4-17
 - alternate configuration attribute (ALTCAR), 4-20
 - alternate configuration base address (ALTCBAR), 4-19
 - alternate signal multiplex control, 18-14
 - branch operations, 6-8
 - byte count, 15-23
 - chip select configuration, 9-12
 - chip select memory bounds, 9-11
 - clock out control, 18-21
 - condition, 6-9
 - configuration, 16-67
 - configuration control and status register base address (CCSRBAR), 4-18
 - configuration header summary, 16-67
 - count (CTR), 6-10
 - critical save/restore 0 (CSRR0), 6-16
 - critical save/restore1 (CSRR1), 6-16
 - current context ID, 20-26
 - current link descriptor address, 15-14
 - current list-alternate base descriptor address, 15-24
 - data exception address (DEAR), 6-17
 - DDR DLL control, 18-21
 - DDR SDRAM control configuration, 9-17
 - DDR SDRAM interval configuration, 9-19
 - DDR SDRAM mode configuration, 9-18
 - DDR SDRAM timing configuration 1, 9-13
 - DDR SDRAM timing configuration 2, 9-16
 - decrementer (DEC), 6-15
 - decrementer auto-reload (DECAR), 6-15
 - destination address for RapidIO maintenance writes, 15-22
 - destination attributes, 15-19
 - destination stride, 15-26
 - device disable, 18-15
 - DMA destination address, 15-21
 - DMA register summary, 15-8
 - DUART
 - alternate function, 12-17
 - divisor most, least significant byte, 12-7
 - DMA status, 12-17
 - FIFO control, 12-11
 - interrupt enable, 12-8
 - interrupt ID, 12-9
 - line control, 12-12
 - line status, 12-14
 - MODEM control, 12-13
 - MODEM status, 12-16
 - receive buffer, 12-6

- receiver buffer, 12-6
- scratch, 12-16
- transmitter holding, 12-6
- e500 core assist, 5-26
- ECM
 - error address capture, 8-8
 - error attributes capture, 8-6
 - error detect, 8-5
 - error enable, 8-5
- Ethernet controller
 - current
 - receive buffer descriptor pointer low, 21-31
 - transmit buffer descriptor pointer low, 21-24
 - DMA control, 21-20
 - error disabled, 21-18
 - group address, 21-49
 - half duplex, 21-41
 - hash function, 21-48
 - individual address, 21-48
 - interface status, 21-46
 - inter-packet gap/inter-frame gap, 21-40
 - interrupt event, 21-15
 - interrupt mask, 21-17
 - MAC configuration 1, 21-37
 - MAC configuration 2, 21-38
 - management status, 21-45
 - maximum frame length, 21-42
 - maximum receiver buffer length R0R1, 21-32
 - MII management
 - address, 21-44
 - command, 21-43
 - configuration, 21-42
 - control, 21-44
 - indicator, 21-46
 - minimum frame length, 21-19
 - out-of sequence Tx data buffer pointer, 21-28
 - out-of-sequence TxBD, 21-26
 - pause time value, 21-20
 - receive buffer descriptor pointer
 - low 0-3, 21-33
 - receive control, 21-29
 - receive status, 21-30
 - RxBD data length, 21-31
 - station address part 1, 21-47
 - station address part 2, 21-47
 - transmit buffer descriptor pointer
 - low, 21-25
 - transmit control, 21-22
 - transmit descriptor base address
 - low, 21-25
 - transmit status, 21-23
 - TxBD data length, 21-24
- exception syndrome (ESR), 6-18
- external interrupt vector/priority, 10-33

- general status, 15-27
- general-purpose, 6-8
 - I/O control, 18-11
 - input data, 18-13
 - output data, 18-12
 - POR configuration, 18-11
- global control, 5-29
- integer exception (XER), 6-8
- interrupt
 - defined by Book E, 6-16
 - general, 6-16
 - vector offset (IVORs), 6-17
 - vector prefix (IVPR), 6-17
- interrupt destination, 10-34
- L2 cache
 - control, 7-6
 - error control and capture, 7-14—7-21
 - external write address, 7-9
 - external write control, 7-10
 - injection, 7-12
 - memory-mapped SRAM base address, 7-11
- LBC
 - base, 13-11
 - clock ratio, 13-35
 - local bus configuration, 13-34
 - memory data, 13-24
 - memory refresh timer prescaler, 13-23
 - option, 13-13
 - SDRAM machine mode register, 13-25
 - transfer error address, 13-34
 - transfer error attributes, 13-33
 - transfer error check disable, 13-31
 - transfer error interrupt enable, 13-32
 - transfer error status, 13-29
 - UPM memory address, 13-20
 - UPM mode, 13-20
- link (LR), 6-10
- local bus control, 18-22
- local control, 5-30
- local control A (PMLCa0–PMLCa3), 6-39
- local control B (PMLCb0–PMLCb3), 6-40
- machine check summary, 18-19
- machine state (MSR), 6-10
- memory data path
 - error injection mask ECC, 9-21
 - error injection mask high, 9-20
 - error injection mask low, 9-21
 - read capture high, 9-22
 - read capture low, 9-23
- memory data path read capture low, 9-23
- memory error
 - address capture, 9-28
 - attributes capture, 9-26
 - detect, 9-24

- disable, 9-25
- memory error interrupt enable, 9-25
- memory map
 - device-specific utilities, 2-15
 - general utility, 2-12
 - interrupt controller and CCSR, 2-14
 - local access window x attributes, 2-7
 - local access window x base address, 2-6
- MMU overview, 6-20
- mode, 15-10
- next link descriptor address, 15-23
- next list descriptor address, 15-25
- PCI interface
 - CONFIG_DATA and CONFIG_ADDR, 16-68
 - status, 16-67—16-75
- performance monitor
 - counters, 19-10
 - global control, 19-5
 - local control, 19-5
- performance monitor counter, 5-30, 6-41
- performance monitor mask even, 10-29
- performance monitor mask odd, 10-30
- PIC
 - critical interrupt summary 0, 10-27
 - critical interrupt summary 1, 10-28
 - feature reporting, 10-15
 - global
 - configuration, 10-16
 - global timer
 - base count, 10-21
 - current count, 10-20
 - destination, 10-23
 - vector/priority, 10-22
 - internal interrupt destination, 10-36
 - internal interrupt vector/priority, 10-35
 - IPI vector/priority, 10-18
 - message enable, 10-31
 - messaging interrupt destination, 10-38
 - messaging interrupt vector/priority, 10-37
 - per CPU
 - interprocessor interrupt dispatch, 10-41
 - processor current task priority, 10-42
 - processor end of interrupt, 10-45
 - processor interrupt acknowledge, 10-44
 - who am I, 10-43
 - processor initialization, 10-17
 - spurious vector, 10-19
 - timer control, 10-24
 - timer frequency reporting, 10-20
 - vendor identification, 10-16
- PIC message, 10-31
- POR
 - boot mode status, 18-6
 - debug mode status, 18-10
 - device status, 18-9
 - I/O impedance status and control, 18-8
 - PLL status, 18-5
- power management control and status, 18-17
- process ID, 5-27
- processor
 - control, 6-10
 - ID (PIR), 6-12, 6-20
 - version (PVR), 6-12
- processor version, 18-20
- programmed context ID, 20-25
- programming PIC, 10-49
- RapidIO interface
 - 8/16 LP-LVDS port maintenance block header 0
 - command and status, 17-26
 - assembly identity capability, 17-15
 - assembly information capability, 17-16
 - base ID command and status, 17-24
 - component tag command and status, 17-25
 - configuration, 17-33
 - destination operations capability, 17-19
 - device identity capability, 17-14
 - device information capability, 17-14
 - doorbell
 - mode, 17-80
 - queue/dequeue pointer address, 17-83
 - queue/enqueue pointer address, 17-84
 - status, 17-82
 - host base ID lock command and status, 17-25
 - inbound
 - frame queue/dequeue pointer address, 17-79
 - frame queue/enqueue pointer address, 17-79
 - mode, 17-75
 - status, 17-77
 - window attributes 0-4, 17-42
 - window base address 1-4, 17-42
 - window translation address 0-4, 17-41
 - local configuration space base address 1
 - command and status, 17-23
 - mailbox command and status, 17-21
 - outbound
 - descriptor queue/dequeue pointer address, 17-70
 - descriptor queue/enqueue pointer address, 17-74
 - destination attributes, 17-72
 - destination port, 17-72
 - double-word count, 17-73
 - mode, 17-67
 - status, 17-69
 - unit source address, 17-71
 - outbound window
 - attributes 0-8, 17-39
 - base address 1-8, 17-38

- translation address, 17-36
- translation address 0-8, 17-38
- packet error capture 1
 - type 1, 17-52
 - type 10, 17-56
 - type 11, 17-57
 - type 13, 17-57
 - type 2, 17-53
 - type 5, 17-53
 - type 6, 17-54
 - type 8 request, 17-55
 - type 8 response, 17-55
- packet error capture 2
 - type 10, 17-61
 - type 6, 17-59
 - type 8, 17-60
 - type 8 response, 17-60
 - types 1, 2, and 5, 17-58
 - types 11 and 13, 17-62
- port configuration, 17-34
- port control command and status, 17-32
- port error
 - and status command, 17-31
 - control, 17-50
 - injection, 17-35
 - packet 1, 17-52
 - packet 2, 17-58
 - packet/control symbol 0, 17-51
 - recovery threshold, 17-65
- port general control command and status register, 17-28
- port link
 - maintenance request command and status, 17-29
 - maintenance response command and status, 17-29
 - time-out control command and status, 17-27
- port local ackID status command and status, 17-30
- port notification/fatal error
 - detect, 17-44
 - disable, 17-47
 - interrupt enable, 17-49
- port recoverable error detect, 17-62
- port response time-out control command and status, 17-27
- port retry threshold, 17-66
- port-write
 - and doorbell command and status, 17-22
 - mode, 17-85
 - queue base address, 17-87
 - status, 17-86
- processing element features capability, 17-16
- processing element logic layer control command and status, 17-23
- source operations capability, 17-18
- switch port information capability, 17-17
- save/restore 0 (SRR0), 6-16
- save/restore 1 (SRR1), 6-16
- single-bit ECC memory error management, 9-29
- software-use SPRs, USPRG0, 6-19
- source address, 15-18
- source address for RapidIO maintenance reads, 15-18
- source attribute, 15-16
- source stride, 15-26
- special-purpose (SPRs)
 - general, 6-4
- SPRG0-SPRG7, 6-19
- status, 15-13
- system version, 18-20
- three-speed Ethernet controller
 - transmit
 - FCS error counter, 14-86
- three-speed Ethernet controller
 - jabber frame counter, 14-86
- three-speed Ethernet controller
 - hash function, 14-95
- Three-speed Ethernet controller
 - MII
 - management status, 14-57
 - transmit
 - control, 14-34
 - transmit and receive 65- and 127-byte frame counter, 14-62
- three-speed Ethernet controller
 - AN advertisement, 14-101
 - AN link partner ability next page, 14-106
 - AN link partner base page ability, 14-103
 - AN next page transmit, 14-105
 - attribute, 14-96
 - attribute extract length and extract index, 14-97
 - carry mask register one, 14-92
 - carry mask register two, 14-93
 - carry register one, 14-90
 - carry register two, 14-91
 - control, 14-99
 - current receive buffer descriptor pointer, 14-43
 - current transmit buffer descriptor, 14-36
 - DMA control, 14-29
 - drop frame counter, 14-85
 - error disabled, 14-25
 - Ethernet control, 14-26
 - excessive collision packet counter, 14-84
 - extended status, 14-107
 - FIFO transmit starve, 14-33
 - FIFO transmit starve shutoff, 14-33

- FIFO transmit threshold, 14-32
- group address, 14-95
- half-duplex, 14-52
- individual address, 14-95
- interface status, 14-58
- inter-packet gap/inter-frame gap, 14-51
- interrupt event, 14-21
- interrupt mask, 14-24
- jitter diagnostics, 14-108
- MAC configuration 1, 14-48
- MAC configuration 2, 14-50
- maximum frame length, 14-54
- maximum receive buffer length, 14-43
- MII
 - management address, 14-56
 - management control, 14-57
 - management indicator, 14-58
- MII management command, 14-55
- MII management configuration, 14-54
- minimum frame length, 14-28
- pause time value, 14-28
- receive
 - alignment error counter, 14-71
 - broadcast packet counter, 14-68
 - byte counter, 14-66
 - carrier sense error counter, 14-73
 - code error counter, 14-72
 - control, 14-41
 - control frame packet counter, 14-69
 - descriptor base address, 14-45
 - dropped packet counter, 14-76
 - fragments counter, 14-75
 - frame length error counter, 14-72
 - jabber counter, 14-76
 - multicast packet counter, 14-68
 - oversize packet counter, 14-74
 - packet counter, 14-66
 - pause frame packet counter, 14-70
 - status, 14-41
 - undersize packet counter, 14-74
 - unknown OPcode packet counter, 14-70
- RxBD data length, 14-42
- station address part 1, 14-59
- station address part 2, 14-60
- status, 14-100
- TBI control, 14-109
- TBI physical address, 14-31
- transmit
 - broadcast packet counter, 14-79
 - byte counter, 14-77
 - control frame counter, 14-87
 - deferral packet counter, 14-80
 - descriptor base, 14-37
 - excessive deferral packet counter, 14-81
 - fragment counter, 14-89
 - multicast packet counter, 14-78
 - oversize frame counter, 14-88
 - packet counter, 14-78
 - pause control frame counter, 14-80
 - single collision packet counter, 14-82
 - status, 14-35
 - undersize frame counter, 14-88
- transmit and receive
 - 1024- to 1518-byte frame counter, 14-64
 - 128- to 255-byte frame counter, 14-62
 - 1519- to 1522-byte VLAN frame counter, 14-65
 - 256- to 511-byte frame counter, 14-63
 - 512- to 1023-byte frame counter, 14-64
 - 64-byte frame counter, 14-61
- transmit total
 - collision counter, 14-84
- TSEC FIFO control and status, 14-31
- TxBD data length, 14-36
- timer
 - control (TCR), 6-13
 - general, 6-13
 - status (TSR), 6-14
- trigger out source, 20-27
- user global control 0 (UPMG0), 6-39
- user local control A (UPMLCa0–UPMLCa3), 6-40
- user local control B (UPMLCb0–UPMLCb3), 6-41
- watchpoint address, 20-13
- watchpoint control 1, 20-12
- watchpoint monitor
 - address mask, 20-14
 - control 0, 20-11
 - status, 20-17
 - trace buffer control 0, 20-18
 - transaction mask, 20-15
- watchpoint monitor trace buffer
 - access control, 20-23
 - access data, 20-25
 - access data high, 20-24
 - address, 20-20
 - control 1, 20-20
 - mask, 20-21
 - status, 20-22
 - transaction, 20-22
- $\overline{REQ}$ (PCI bus request) signal, 16-49
- Reset
 - common vector, 5-32
 - DDR debug configuration, 4-13
 - functional device description, 4-3
 - hard, 4-3
 - local bus output hold configuration, 4-14
 - memory debug configuration, 4-12
 - PCI arbiter configuration, 4-11

- PCI debug configuration, 4-12
- PCI I/O impedance, 4-11
- PCI width configuration, 4-11
- PCI/PCI-X output hold configuration, 4-13
- PCI-X configuration, 4-12
- power-on configuration (POR), 4-5
- power-on sequence, 4-3
- soft, 4-3
- Reset and clocking blocks
 - boot ROM location, 4-7
 - boot sequencer configuration, 4-8
 - clock signals, 4-2
 - CPU boot configuration, 4-8
 - e500 core PLL ratio, 4-6
 - external signal description, 4-1
 - functional description, 4-3
 - host/agent configuration, 4-7
 - local configuration control, 4-16
 - memory map/register definition, 4-3
 - operations, 4-3
 - RapidIO device ID, 4-10
 - RapidIO transmit clock source, 4-10
 - system control signals, 4-1
 - system PLL ratio, 4-6
 - TSEC width, 4-9
 - TSEC1 protocol, 4-9
 - TSEC2 protocol, 4-10

S

- Save/restore register 1 (SRR1), 6-16
- SDRAM
 - address multiplexing, 13-58
 - device-specific parameters, 13-59
 - machine, 13-55
 - power-on initialization, 13-56
 - supported configurations, 13-55
- SDRAM interface
 - registered DIMM mode, 9-44
 - timing, 9-39
- $\overline{\text{SERR}}$ (system error) signal, 16-76
- Signals
 - 60x processor interface DHn/DLn, 9-6
 - BIDI, 14-12
 - byte enable signals, 16-56
 - $\overline{\text{C/BE}}_n$, 16-8, 16-57
 - clock, 4-2
 - configuration, sampled at reset, 3-15
 - control, description, 4-1
 - description, external, 15-5
 - detailed global utilities descriptions, 18-3
 - DEVSEL, 16-8
 - DUART
 - descriptions, 12-3
 - EC_GTX_CLK125, 14-12
 - EC_MDC, 14-12
 - EC_MDIO, 14-12
 - eset and clocking blocks, 4-1
 - Ethernet controller
 - overview, 21-7
 - general-purpose I/O, 18-2, 18-27
 - global utilities external summary, 18-2
 - I²C interface
 - summary, 11-4
 - interrupt and local bus multiplexing, 18-2
 - introduction, 3-1
 - L1_TSTCLK input, 20-10
 - L2_TSTCLK input, 20-10
 - LBC external address latch enable, 13-38
 - local bus multiplexing, 18-27
 - LSSD_MODE, 20-10
 - MAA, 20-7
 - output signal states at power-on reset, 3-16
 - overview, 3-1
 - PCI address output, 20-8
 - PCI interface
 - ADn, 16-55
 - $\overline{\text{C/BE}}_n$, 16-75
 - $\overline{\text{DEVSEL}}$, 16-56
 - $\overline{\text{FRAME}}$, 16-52
 - $\overline{\text{GNT}}$, 16-49
 - $\overline{\text{IRDY}}$, 16-52
 - PAR (PCI parity), 16-75
 - $\overline{\text{PERR}}$, 16-76
 - REQ, 16-49
 - $\overline{\text{SERR}}$, 16-76
 - $\overline{\text{STOP}}$, 16-57
 - $\overline{\text{TRDY}}$, 16-53
 - turnaround cycle, 16-57
 - PIC overview, 10-7
 - RapidIO interface, detailed descriptions, 17-8
 - reset and clocking blocks, system, 4-1
 - system control, description, 4-2
 - test and configuration, 20-8
 - description, 20-8
 - TCK (JTAG test clock), 20-8
 - TDI (JTAG test data input), 20-9
 - TDO (JTAG test data output), 20-9
 - TMS (JTAG test mode select), 20-9
 - $\overline{\text{TRST}}$ (JTAG test reset), 20-9
 - $\overline{\text{TRDY}}$, 16-13
 - trigger in, 20-7
 - trigger out, 20-8
 - TSECn_CRs, 14-11
 - TSECn_GTX_CLK, 14-11
 - TSECn_RX_CLK, 14-12
 - TSECn_RX_DV, 14-12
 - TSECn_RX_ER, 14-13

- TSECN_RXD, 14-12
- TSECN_TX_CLK, 14-13
- TSECN_TX_EN, 14-14
- TSECN_TXD, 14-13
- TSECN_TX_ER, 14-14
- Sleep mode description, 1-18
- SPRG0-SPRG7, 6-19
- SPRs, 6-4
- SRAM
 - coherency, 7-22
 - features, 7-2
 - memory-mapped windows, 2-4
- SRR0, 6-16
- SRR1, 6-16
- Status register, PCI, 16-57, 16-67, 16-75
- STOP signal, 16-57
- Summary overflow (SO), 6-8

T

- Target-disconnect, *see* PCI interface
- Target-initiated termination
 - description, 16-4, 16-61
 - PCI status register, 16-62
- TCR, 6-13
- Termination
 - PCI
 - completion, 16-61
 - master-abort termination, 16-61
 - target-disconnect, 16-4, 16-61
 - target-initiated, 16-62
 - termination of PCI transaction, 16-60
 - timeout, 16-61
- Test and configuration signals, *see* Signals, 20-8
- Three-speed Ethernet controller
 - AN advertisement register (ANA), 14-101
 - AN link partner ability next page register (ANLPANP), 14-106
 - AN link partner base page ability register (ANLPBPA), 14-103
 - AN next page transmit register (ANNPT), 14-105
 - application information, 14-135
 - attribute extract length and extract index register (ATTRELI), 14-97
 - attribute register (ATTR), 14-96
 - buffer descriptors, 14-129
 - carry mask register one (CAM1), 14-92
 - carry mask register two (CAM2), 14-93
 - carry register one (CAR1), 14-90
 - carry register two (CAR2), 14-91
 - connecting to physical interfaces, 14-110
 - control register (CR), 14-99
 - control/status registers, 2-51, 14-21

- current receive buffer descriptor pointer (CRBPTR), 14-43
- current transmit buffer descriptor pointer (CTBPTR), 14-36
- data buffer descriptor, 14-129
- destination address recognition, 14-123
- detailed memory map—control/status registers, 14-15
- DMA control register (DMACTRL), 14-29
- drop frame counter (TDRP), 14-85
- error disabled register (EDIS), 14-25
- error-handling procedure, 14-128
- ethernet control register (ECNTRL), 14-26
- excessive collision packet counter (TXCL), 14-84
- extended status register (EXST), 14-107
- FIFO transmit starve register (FIFO_TX_STARVE), 14-33
- FIFO transmit starve shutoff register (FIFO_TX_STARVE_SHUTOFF), 14-33
- FIFO transmit threshold register (FIFO_TX_THR), 14-32
- flow control, 14-126
- frame recognition, 14-123
- functional description, 14-110
- gigabit Ethernet channel operation, 14-118
- gigabit Ethernet frame reception, 14-122
- gigabit ethernet frame transmission, 14-120
- gigabit media-independent interface (GMII), 14-111
- GMII interface mode, 14-139
- group address registers 0–7 (GADDRn), 14-95
- half-duplex register (HAFDUP), 14-52
- hardware controlled initialization, 14-118
- hash function registers, 14-95
- hash table algorithm, 14-125
- individual address registers 0–7 (IADDRn), 14-95
- initialization sequence, 14-118
- interface mode configuration, 14-135
- interface status register (IFSTAT), 14-58
- internal and external loop back, 14-128
- inter-packet gap time, 14-127
- inter-packet gap/inter-frame gap register (IPGIFG), 14-51
- interrupt event register (IEVENT), 14-21
- interrupt handling, 14-126
- interrupt mask register (IMASK), 14-24
- introduction, 14-1
- jabber frame counter (TJBR), 14-86
- jitter diagnostics register (JD), 14-108
- MAC
 - configuration 1 register (MACCFG1), 14-48
 - configuration 2 register (MACCFG2), 14-50
 - configuring, 14-46
 - controlling packet flow, 14-47
 - controlling PHY links, 14-48

- CSMA/CD controlling, 14-46
- functionality, 14-45
- handling packet collisions, 14-46
- MAC registers, 14-48—14-61
- maximum frame length register (MAXFRM), 14-54
- maximum receive buffer length register (MRBL), 14-43
- media-independent interface (MII), 14-111
- memory map/register definition, 14-14
- memory-mapped register descriptions, 14-21
- MIB registers, 14-61—14-94
- MII
 - interface mode, 14-135
 - management address register (MIIMADD), 14-56
 - management command (MIIMCOM), 14-55
 - management configuration register (MIIMCFG), 14-54
 - management control register (MIIMCON), 14-57
 - management indicator register (MIIMIND), 14-58
- MII management status register (MIIMSTAT), 14-57
- minimum frame length register (MINFLR), 14-28
- modes of operation, 14-8
- op-level module memory map, 14-15
- overview, 1-14, 14-7
- pause time value Register (PTV), 14-28
- receive
 - alignment error counter (RALN), 14-71
 - broadcast packet counter (RBCA), 14-68
 - byte counter (RBYT), 14-66
 - carrier sense error counter (RCSE), 14-73
 - code error counter (RCDE), 14-72
 - control frame packet counter (RXCF), 14-69
 - control register (RCTRL), 14-41
 - descriptor base address register (RBASE), 14-45
 - dropped packet counter (RDRP), 14-76
 - fragments counter (RFRG), 14-75
 - frame length error counter (RFLR), 14-72
 - jabber counter (RJBR), 14-76
 - multicast packet counter (RMCA), 14-68
 - oversize packet counter (ROVR), 14-74
 - packet counter (RPKT), 14-66
 - pause frame packet counter (RXPF), 14-70
 - status register (RSTAT), 14-41
 - undersize packet counter (RUND), 14-74
 - unknown OPcode packet counter (RXUO), 14-70
- receive buffer descriptor (RxBD), 14-133
- receive control and status registers, 14-40—14-45
- reduced gigabit media-independent interface (RGMII), 14-112
- reduced ten-bit interface (RTBI), 14-114
- RMON support, 14-123
- RxBD data length register (RBDLEN), 14-42

- signals, 14-9—14-14
- BIDI (input/output), 14-12
- detailed descriptions, 14-11—14-15
- EC_GTX_CLK125—Gigabit Transmit 125 MHz Source (input), 14-12
- EC_MDC—Management Data Clock (output), 14-12
- EC_MDIO—Management Data input/output, 14-12
- TSEC_CRS—Carrier Sense (input), 14-11
- TSECn_COL—Collision (input), 14-11
- TSECn_GTX_CLK—Gigabit Transmit Clock (output), 14-11
- TSECn_RX_CLK—Receive Clock (input), 14-12
- TSECn_RX_DV—Receive Data Valid (input), 14-12
- TSECn_RX_ER—Receive Error (input), 14-13
- TSECn_RXD—Receive Data (input), 14-12
- TSECn_TX_CLK—Transmit Clock (input), 14-13
- TSECn_TX_EN—Transmit Data Valid (output), 14-14
- TSECn_TX_ER—Transmit Error (output), 14-14
- TSECn_TXD—Transmit Data (output), 14-13
- soft reset and reconfiguring procedure, 14-119
- station address part 1 register (MACSTNADDR1), 14-59
- station address part 2 register (MACSTNADDR2), 14-60
- status register (SR), 14-100
- transmit and receive
 - 256- to 511-byte frame counter (TR511), 14-63
- TBI control register (TBICON), 14-109
- TBI MII set register descriptions, 14-98
- TBI physical address register (TBIPA), 14-31
- ten-bit interface (TBI), 14-98, 14-113
- transmit
 - broadcast packet counter (TBICA), 14-79
 - byte counter (TBYT), 14-77
 - control frame counter (TXCF), 14-87
 - deferral packet counter (TDFR), 14-80
 - descriptor base address (TBASE), 14-37
 - excessive deferral packet counter (TEDF), 14-81
 - fragment counter (TFRG), 14-89
 - multicast packet counter (TMCA), 14-78
 - oversize frame counter (TOVR), 14-88
 - packet counter (TPKT), 14-78
 - pause control frame counter (TXPF), 14-80
 - single collision packet counter (TSCL), 14-82
 - status register (TSTAT), 14-35
 - total collision counter (TNCL), 14-84
 - undersize frame counter (TUND), 14-88
- transmit and receive

- 1024- to 1518-byte frame counter (TRMAX), 14-64
- 128- to 255-byte fFrame counter (TR255), 14-62
- 1519- to 1522-byte VLAN frame counter (TRMGV), 14-65
- 512- to 1023-byte frame counter (TR1K), 14-64
- 64-Byte frame counter (TR64), 14-61
- transmit and receive 65- to 127-byte frame counter (TR127), 14-62
- Transmit Control Register (TCTRL), 14-34
- transmit data buffer descriptor (TxBD), 14-131
- transmit FCS error counter (TFCS), 14-86
- TSEC FIFO control and status registers, 14-31
- TSEC general control and status registers, 14-21
- TSEC transmit control and status registers, 14-34
- TxBD data length register (TBDLEN), 14-36
- user initialization, 14-118
- Timeout, PCI transaction, 16-61
- Timing diagrams
 - DRAM/EDO, self-refresh in sleep and suspend modes, 9-49
- TLBs (translation lookaside buffers) coherency, 5-27
- Trace buffer
 - overview, 20-31
 - register descriptions, 20-18—20-25
- Traced data (TBCR{TRSEL}), 20-31
- Transactions
 - error transactions, 16-74
 - PCI bus
 - fast back-to-back transactions, 16-64
 - read transactions, 16-58
 - write transactions, 16-57
 - PCI transaction termination, 16-60
 - retry PCI transactions, 16-62
- $\overline{\text{TRDY}}$ (target ready) signal, 16-13, 16-53, 16-60
- Trigger out source register (TOSR), 20-27
- TSR, 6-14
- Turnaround cycle and PCI bus, 16-57

U

- User global control register 0 (UPMGC0), 6-39
- User local control A registers
 - (UPMLCa0–UPMLCa3), 6-40
- User local control B registers
 - (UPMLCb0–UPMLCb3), 6-41
- USPRG0, 6-19

W

- Watchpoint monitor
 - address mask register (WMAMR), 20-14
 - address register (WMAR), 20-13
 - control register 0 (WMCR0), 20-11
 - control register 1 (WMCR1), 20-12

- general, 20-30
- performance monitor events, 20-30
- register descriptions, 20-11
- status register (WMSR), 20-17
- trace buffer
 - address mask register (TBAMR), 20-21
 - address register (TBAR), 20-20
 - control register 0 (TBCR0), 20-18
 - control register 1 (TBCR1), 20-20
 - status register (TBSR), 20-22
 - transaction mask register (TBTMR), 20-22
- trace buffer access
 - control register (TBACR), 20-23
 - data high register (TBADHR), 20-24
 - data register (TBADR), 20-25
 - transaction mask register (WMTMR), 20-15

X

- XER, 6-3, 6-8